



语义网与知识图谱

上海大学计算机学院

主讲：刘 炜





描述逻辑及语义

上海大学计算机学院 刘炜

2020年9月

一、描述逻辑


1. 描述逻辑简介
2. 描述逻辑基础
3. 描述逻辑的体系结构
4. Description Logic SROIQ(D)

二、描述逻辑语义

描述逻辑定义

描述逻辑 (Description Logic) 是基于对象的知识表示的形式化, 也叫概念表示语言或术语逻辑。它吸取了KL-ONE的主要思想, 是一阶谓词逻辑的一个可判定子集。

- 源于语义网络和KL-ONE
- 是一阶逻辑FOL的一个可判定的子集
- 建立在概念和关系(Role)之上
- 概念解释为对象的集合
- 关系解释为对象之间的二元关系
- 具有合适定义的语义(基于逻辑)

A yellow thought bubble with a black outline, containing text. It is connected to the list item '是一阶逻辑FOL的一个可判定的子集' by a line of three small circles.

总能保证
推理算法
的终止

描述逻辑特性

- 是以往表示工具的逻辑重构和统一形式化
 - ✓ 语义网络 (Semantic Networks)
 - ✓ 框架系统 (Frame-based systems)
 - ✓ 面向对象表示 (OO representation)
 - ✓ 语义数据模型 (Semantic data models)
 - ✓ 类型系统 (Type systems)
 - ✓ 特征逻辑 (Feature Logics)
 - 具有很强的表达能力
 - 是可判定的，总能保证推理算法终止
-



为什么用描述逻辑

若直接使用一阶逻辑，而不附加任何约束，则：

- 知识的结构将被破坏，这样就不能用来驱动推理
- 对获得可判定性和有效的推理问题来说，其表达能力太高，（也许是太抽象了）
- 对兴趣表达，但仍然可判定的理论，其推理能力太低。

DL的重要特征是：

- 它有清晰的模型-理论机制，具有很强的表达能力；
 - 适合于通过概念分类学来表示应用领域；
 - 提供了可判定的推理服务，能保证推理算法总能停止，并返回正确的结果。
-

一阶谓词逻辑:

- Mary is a female: $\text{female}(\text{Mary})$
- John is a male: $\text{male}(\text{John})$
- Everybody is male or female: $x: \text{male}(x) \vee \text{female}(x)$

简单的推理：

- $\neg (\text{male}(\text{Mary}))$

$x: \text{male}(x) \rightarrow \neg \text{female}(x)$: $p \vee q = p \rightarrow \neg q$

$x: \text{female}(x) \rightarrow \neg \text{male}(x)$: $p \rightarrow \neg q = q \rightarrow \neg p$

$\text{female}(\text{Mary}) \rightarrow \neg \text{male}(\text{Mary})$: if $x = \text{Mary}$

一、描述逻辑

1. 描述逻辑简介
2. 描述逻辑基础
3. 描述逻辑的体系结构
4. Description Logic SROIQ(D)

二、描述逻辑语义

- **个体individuals (记作 URIs)**
 - 也称: 常量(一阶谓词逻辑), 资源(RDF), 实例
 - `http://example.org/sebastianRudolph`
 - `http://www.semantic-web-book.org/`
 - 我们用小写或缩写表示, e.g. "sebastianRudolph"
 - **类(记作URIs!)**
 - 也称: 概念, 一元谓词 (FOL)
 - 我们采用大写开头单词表示, e.g. "Father"
 - **属性 (也记作 URIs!)**
 - 也称: 角色 (Role), 二元谓词 (一阶谓词逻辑)
 - 采用小写开头的单词, e.g. "hasDaughter"
-

描述逻辑语法

RDF语法

- `Person(mary)`
 - `:mary rdf:type :Person .`
 - `Woman \sqsubseteq Person`
 - `Person \equiv HumanBeing` (class equivalence):
`Person \sqsubseteq HumanBeing` `AND`
`HumanBeing \sqsubseteq Person`
 - `:Woman rdfs:subClassOf :Person .`
 - `hasWife(john,mary)`
 - `:john :hasWife :mary .`
 - `hasWife \sqsubseteq hasSpouse`
 - `hasSpouse \equiv marriedWith` (property equivalence)
 - `:hasWife rdfs:subPropertyOf :hasSpouse`
-

描述逻辑语法

FOL语法

ABox statements

- $\text{Person}(\text{mary})$
- $\text{Person}(\text{mary})$
- $\text{Woman} \sqsubseteq \text{Person}$
 - $\text{Person} \equiv \text{HumanBeing}$ (class equivalence)
- $\forall x (\text{Woman}(x) \rightarrow \text{Person}(x))$
- $\text{hasWife}(\text{john}, \text{mary})$
- $\text{hasWife}(\text{john}, \text{mary})$
- $\text{hasWife} \sqsubseteq \text{hasSpouse}$
 - $\text{hasSpouse} \equiv \text{marriedWith}$ (property equivalence)
- $\forall x \forall y (\text{hasWife}(x, y) \rightarrow \text{hasSpouse}(x, y))$

TBox statements



DL基础—特殊类和属性

- **owl:Thing** (RDF syntax) 概念全集
 - DL-syntax: \top
 - contains everything
 - **owl:Nothing** (RDF syntax) 概念空集
 - DL-syntax: \perp
 - empty class
 - **owl:topProperty** (RDF syntax) 属性全集
 - DL-syntax: U
 - every pair is in U
 - **owl:bottomProperty** (RDF syntax) 属性空集
 - empty property
-

DL基础—Class Constructor类构造子



- **conjunction 合取** $\forall x (Mother(x) \leftrightarrow Woman(x) \wedge Parent(x))$
 - **$Mother \equiv Woman \sqcap Parent$**
„Mothers are exactly those who are women and parents.“
 - **disjunction 析取** $\forall x (Parent(x) \leftrightarrow Mother(x) \vee Father(x))$
 - **$Parent \equiv Mother \sqcup Father$**
„Parents are exactly those who are mothers or fathers.“
 - **negation 非** $\forall x (ChildlessPerson(x) \leftrightarrow Person(x) \wedge \neg Parent(x))$
 - **$ChildlessPerson \equiv Person \sqcap \neg Parent$**
„ChildlessPersons are exactly those who are persons and who are not parents.“
-

DL基础—Class Constructor类构造子



- existential quantification 存在量词

- 只用于一个角色（属性）——也称为属性限制

- $\text{Parent} \equiv \exists \text{hasChild. Person}$
„Parents are exactly those who have at least one child which is a Person.“

$$\forall x (\text{Parent}(x) \leftrightarrow \exists y (\text{hasChild}(x,y) \wedge \text{Person}(y)))$$

- universal quantification 全称量词

- 只用于一个角色（属性）——也称为属性限制

- $\text{Person} \sqcap \text{Happy} \equiv \forall \text{hasChild. Happy}$
„A (person which is also happy) is exactly (something all children of which are happy).“

$$\forall x (\text{Person}(x) \wedge \text{Happy}(x) \leftrightarrow \forall y (\text{hasChild}(x,y) \rightarrow \text{Happy}(y)))$$

- 类的构造子可以任意嵌套



描述逻辑基础语言--ALC

- 交 (\wedge) , 并 (\vee) , 非 (\neg) , 存在量词 (\exists) 和全称量词 (\forall)
- 上述5个构造子构成了最基本的描述逻辑形式语言ALC
Atttribute **c**oncept **d**escription **L**anguage
Complement)

例如, ALC中概念Happy-father定义为:

$\text{Man} \sqcap \exists \text{ has-child.Male}$

$\sqcap \exists \text{ has-child.Female}$

$\sqcap \forall \text{ has-child.}(\text{Doctor} \sqcup \text{Lawyer})$

一、描述逻辑

1. 描述逻辑简介
2. 描述逻辑基础
3. 描述逻辑的体系结构
4. Description Logic SROIQ(D)

二、描述逻辑语义

描述逻辑的体系结构

- 一个描述逻辑系统包括四个基本的组成部分：
 - ✓ (1) 表示概念和关系的构造集；
 - ✓ (2) Tbox术语集（概念术语的断言集合）；
 - ✓ (3) Abox断言集（个体的断言集合）；
 - ✓ (4) Tbox和Abox上的推理机制。
 - 不同的描述逻辑系统的表示能力与推理机制由于对这四个组成部分的不同选择而不同。
-



描述逻辑的体系结构

概念和关系

- 概念——解释为一个领域的子集

示例：学生，已婚者：

$\{x \mid \text{Student}(x)\}$, $\{x \mid \text{Married}(x)\}$

- 关系——解释为指该领域上的二元关系（笛卡尔乘积）

示例：朋友，爱人：

$\{ \langle x, y \rangle \mid \text{Friend}(x, y) \}$, $\{ \langle x, y \rangle \mid \text{Loves}(x, y) \}$



描述逻辑的体系结构

描述逻辑的知识库 $K = \langle T, A \rangle$ ， T 即Tbox， A 即Abox。

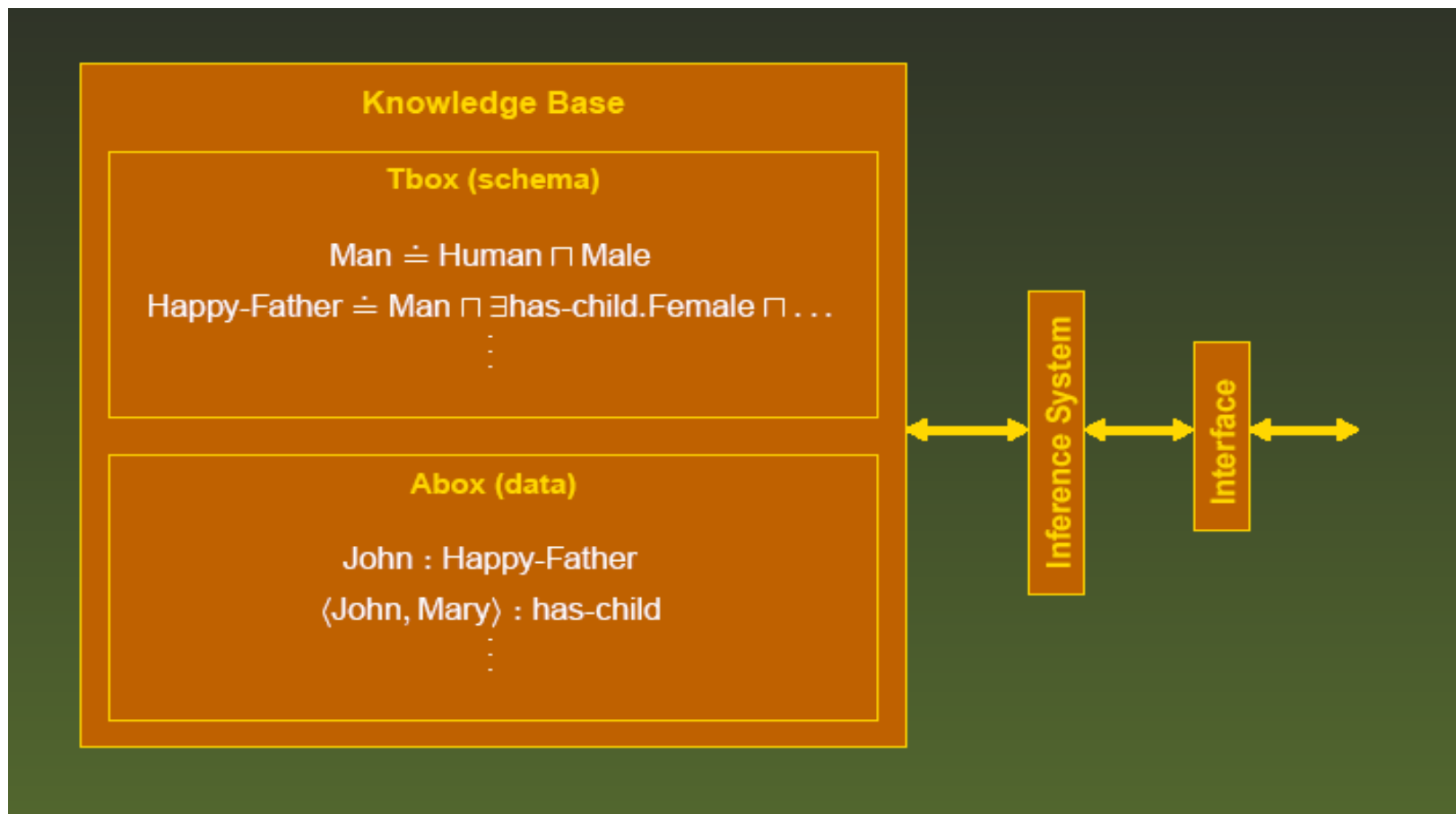
➤ Tbox

Tbox包含内涵知识，描述概念的一般性质。由于概念之间存在包含关系，Tbox知识形成类似格的结构，这种数学结构是由包含关系决定的，与具体实现无关；

➤ Abox

Abox包含外延知识（又称断言知识），描述论域中的特定个体。

描述逻辑的体系结构



描述逻辑的体系结构

TBox : 是描述领域结构的公理的集合

- **定义:** 引入概念的名称

$$A \equiv C, A \sqsubseteq C$$

$$\text{Father} \equiv \text{Man} \sqcap \exists \text{ has-child.Human}$$

$$\text{Human} \sqsubseteq \text{Animal} \sqcap \text{Biped}$$

- **包含:** 声明包含关系的公理

$$C \sqsubseteq D \quad (C \equiv D \Leftrightarrow C \sqsubseteq D, D \sqsubseteq C)$$

$$\exists \text{ has-degree.Masters} \sqsubseteq \exists \text{ has-degree.Bachelors}$$



描述逻辑的体系结构

Abox:是描述具体情形的公理的集合

- ◆ **概念断言** ——表示一个对象是否属于某个概念

$a:C$

例如：Tom是个学生，表示为

$Tom : Student$ 或者 $Student(Tom)$

$John : Man \sqcap \exists \text{ has-child.Female}$

- ◆ **关系断言** ——表示两个对象是否满足一定的关系

$\langle a, b \rangle : R$

例如：John有个孩子叫Mary

$\langle John, Mary \rangle : \text{has-child}$ 或者 $\text{has-child}(John, Mary)$

一、描述逻辑

1. 描述逻辑简介
2. 描述逻辑基础
3. 描述逻辑的体系结构
4. **Description Logic SROIQ(D)**

二、描述逻辑语义

描述逻辑ALC

- Set of individuals a, b, c, \dots
- Set of atomic classes (class names) A, B, \dots
- Set of role names R, S, \dots

- 复杂类可以通过以下构造方式进行表示

$$C, D ::= A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

$\text{Human} \sqsubseteq \exists \text{hasParent}.\text{Human}$

$\text{Orphan} \sqsubseteq \text{Human} \sqcap \forall \text{hasParent}.\neg \text{Alive}$

$\text{Orphan}(\text{harrypotter})$

$\text{hasParent}(\text{harrypotter}, \text{jamespotter})$

理解SROIQ(D)

ALC + role chains = SR

$\text{hasParent} \circ \text{hasBrother} \sqsubseteq \text{hasUncle}.$

$\forall x \forall y (\exists z ((\text{hasParent}(x,z) \wedge \text{hasBrother}(z,y)) \rightarrow \text{hasUncle}(x,y)))$

—包括属性全集和属性空集

- **includes $S = \text{ALC} + \text{transitivity}$**
 - **$\text{hasAncestor} \circ \text{hasAncestor} \sqsubseteq \text{hasAncestor}$**
 - **includes $\text{SH} = S + \text{role hierarchies}$**
 - **$\text{hasFather} \sqsubseteq \text{hasParent}$**
-



理解SROIQ(D)

- **O – nominals (名词性词，封闭类)**
 - **MyBirthdayGuests \equiv {bill,john,mary}**
 - **注意与以下的区别**
 - MyBirthdayGuests(bill)**
 - MyBirthdayGuests(john)**
 - MyBirthdayGuests(mary)**
 - **个体相等和不相等 (没有唯一的名字假设!)**
 - **bill = john**
 - **{bill} \equiv {john}**
 - **bill \neq john**
 - **{bill} \sqcap {john} $\equiv \perp$**
-



理解SROIQ(D)

- **I – inverse roles (逆角色)**
 - $\text{hasParent} \equiv \text{hasChild}^{-}$
 - $\text{Orphan} \equiv \forall \text{hasChild}^{-} . \text{Dead}$
 - **Q – qualified cardinality restrictions (限定基数限制)**
 - $\leq 4 \text{ hasChild.Parent}(\text{john})$
 - $\text{HappyFather} \equiv \geq 2 \text{ hasChild.Female}$
 - $\text{Car} \sqsubseteq = 4 \text{ hasTyre.T}$
-

理解SROIQ(D)



属性可以申明为以下特性：

- | | | |
|---------------------|-------------|---|
| • Transitive | hasAncestor | $R(a,b) \text{ and } R(b,c) \rightarrow R(a,c)$ |
| • Symmetric | hasSpouse | $R(a,b) \rightarrow R(b,a)$ |
| • Asymmetric | hasChild | $R(a,b) \rightarrow \text{not } R(b,a)$ |
| • Reflexive | hasRelative | $R(a,a) \text{ for all } a$ |
| • Irreflexive | parentOf | $\text{not } R(a,a) \text{ for any } a$ |
| • Functional | hasHusband | $R(a,b) \text{ and } R(a,c) \rightarrow b=c$ |
| • InverseFunctional | hasHusband | $R(a,b) \text{ and } R(c,b) \rightarrow a=c$ |

理解SROIQ(D)



• (D) – datatypes

- ✓ 到目前为止，我们只看到个体作为属性的第二参数，称为对象属性 (object properties)或抽象角色(abstract roles)。
- ✓ 属性在第二参数用数据类型字面量称为数据属性或者实角色 concrete roles.
- ✓ 在OWL中可以使用很多XML Schema中的datatypes, 包括 xsd:integer, xsd:string, xsd:float, xsd:boolean, xsd:anyURI, xsd:dateTime , 还有比如 owl:real.

理解SROIQ(D)



- (D) – datatypes

- ✓ hasAge(john, "51"^^xsd:integer)

- ✓ 约束的额外用法(from XML Schema)

- e.g. Teenager \equiv Person $\sqcap \exists \text{hasAge.}(\text{xsd:integer:} \geq 12 \text{ and } \leq 19)$

- ✓ 注意: 上述不是标准的 DL notation! 但在 OWL可以使用.

理解SROIQ(D)



- 更多表示特性

- ✓ Self

- `PersonCommittingSuicide` $\equiv \exists \text{kills.Self}$

- ✓ Keys (not really in SROIQ(D), but in OWL)

- 一组（对象或数据）属性，其值唯一标识对象 set of (object or data) properties whose values uniquely identify an object

- ✓ disjoint properties 不相交属性

- `Disjoint(hasParent, hasChild)`

- ✓ explicit anonymous individuals 显式匿名个体

- 像在RDF中: 可以用来代替命名的个体。

Overview—SROIQ(D)构造算子



- ABox assignments of individuals to classes or properties
 - ALC: \sqsubseteq, \equiv for classes
 $\sqcap, \sqcup, \neg, \exists, \forall$
 \top, \perp
 - SR: + **property chains, property characteristics, role hierarchies** \sqsubseteq
 - SRO: + nominals $\{o\}$
 - SROI: + inverse properties
 - SROIQ: + **qualified** cardinality constraints
 - SROIQ(D): + datatypes (including **facets**)
 - + **top and bottom roles** (for objects and datatypes)
 - + **disjoint properties**
 - + **Self**
 - + **Keys** (not in SROIQ(D), but in OWL)
-

OWL中的一些语法糖(Syntactic Sugar)



- SROIQ(D) 本质上 (语义上) 和OWL是一致的.
 - 在owl (见下文) 中可用作dl公理的语法糖 :
 - ✓ 不相交disjoint classes
 - $\text{Apple} \sqcap \text{Pear} \sqsubseteq \perp$
 - ✓ 不相交并disjoint union
 - $\text{Parent} \equiv \text{Mother} \sqcup \text{Father}$
 - $\text{Mother} \sqcap \text{Father} \sqsubseteq \perp$
 - ✓ 属性非 negative property assignments (also for datatypes)
 - $\neg \text{hasAge}(\text{jack}, "53" \wedge \wedge \text{xsd:integer})$
-



两种全局限制(Global Restriction)

- 任意属性链（角色链）公理会导致不可判定性 arbitrary property chain axioms lead to undecidability
- 限制: 属性链（角色链）集合必须是规则的 set of property chain axioms has to be regular
 - ✓ 属性必须是按严格顺序的。
 - ✓ 每个属性链公理需要按以下形式之一：

$$R \circ R \sqsubseteq R$$

$$S^- \sqsubseteq R$$

$$S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$$

$$R \circ S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$$

$$S_1 \circ S_2 \circ \dots \circ S_n \circ R \sqsubseteq R$$

从而， $S_i < R$ for all $i = 1, 2, \dots, n$.

Example 1: $R \circ S \sqsubseteq R$

$S \circ S \sqsubseteq S$

$R \circ S \circ R \sqsubseteq T$

→ regular with order $S < R < T$

Example 2: $R \circ T \circ S \sqsubseteq T$

→ not regular because form not admissible

Example 3: $R \circ S \sqsubseteq S$

$S \circ R \sqsubseteq R$

→ not regular because no adequate order exists



两种全局限制(Global Restriction)

- 集成属性链和基数限制可能导致不可判定性
- 限制：在基数表达式中只使用简单属性
- 技术处理：
 - ✓ 如果一个角色不出现在一个角色包含公理的右边，那它是简单的。
 - ✓ 简单角色的逆是简单的。
 - ✓ 如果角色R只出现在形为 $S \sqsubseteq R$ 的角色包含公理的右边，而且S是简单的，那么R是简单的。

Example: $Q \circ P \sqsubseteq R$ $R \circ P \sqsubseteq R$ $R \sqsubseteq S$ $P \sqsubseteq R$ $Q \sqsubseteq S$

non-simple: R, S simple: P, Q

一、描述逻辑

1. 描述逻辑简介
2. 描述逻辑基础
3. 描述逻辑的体系结构
4. Description Logic SROIQ(D)

二、描述逻辑语义

OWL有两种语义：

■ **1. 描述逻辑语义**

也称: 直接语义; 一阶谓词逻辑语义

可以通过将描述逻辑翻译成一阶谓词逻辑获得.

应用一些全局限制！

■ **2. RDF-based 语义 (requires RDF/XML syntax: done later)**

没有语义限制.

利用RDFS-推理特性扩展直接语义.

模型理论语义



为了获得可判定的语义，应用以下相应的语法限制：

- 禁止类型双关 Type separation / punning
- 禁止属性链有环 No cycles in property chains.

(See global restrictions mentioned earlier.)

- 禁止在基数限制中有传递属性 No transitive properties in cardinality restrictions.

(See global restrictions mentioned earlier.)



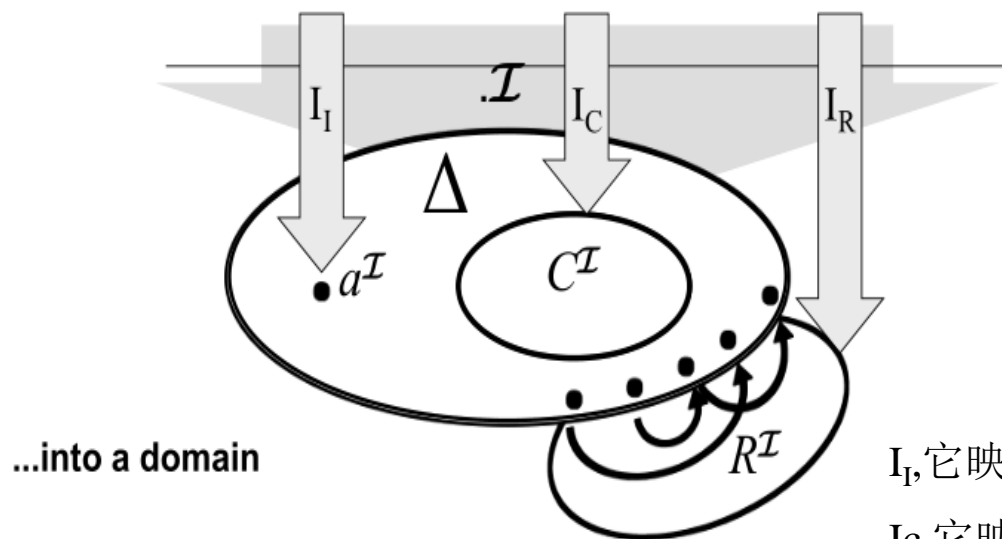
可判定性

- 如果一个问题总是存在终止算法，不管该问题是否可解，则这个问题是可判定的。 A problem is decidable if there exists an always terminating algorithm which determines, whether or not a solution exists.
 - 如果一个问题存在一个算法，能在有限时间可解，这这个问题是半判定问题。
 - 一个问题如果不是可判定的，则属于非判定问题。 A problem is undecidable if it is not decidable.
 - 存在同时是半判定和非判定的问题。
 - 一种描述逻辑如果如果其“蕴含公理” (entailment axioms)是可判定的则这种描述逻辑是可判定的。
 - 大多数的描述逻辑是可判定的，可判定性是判断一种“好的”描述逻辑的标准。
-

直接语义



- 模型理论语义
- 从解释器 (interpretations) 理解
- 一个解释器将个体名字, 类名字, 属性名字映射到一个领域



I_I , 它映射个体到域中的元素: $I_I: I \rightarrow \Delta$

I_C , 它映射类名字到域中的子集: $I_C: C \rightarrow 2^\Delta$ (类扩展)

I_R , 它映射角色到域中的二元关系, 即到域中元素对的集合: $I_R: R \rightarrow 2^{\Delta \times \Delta}$ (属性扩展)

OWL直接语义



- 映射扩展到复杂类

- $\top^I = \Delta^I$ $\perp^I = \emptyset$
- $(C \sqcap D)^I = C^I \cap D^I$ $(C \sqcup D)^I = C^I \cup D^I$ $(\neg C)^I = \Delta^I \setminus C^I$
- $(\forall R.C)^I = \{ x \mid \text{for all } (x,y) \in R^I \text{ we have } y \in C^I \}$
 $(\exists R.C)^I = \{ x \mid \text{there is } (x,y) \in R^I \text{ with } y \in C^I \}$
- $(\geq n R.C)^I = \{ x \mid \#\{ y \mid (x,y) \in R^I \text{ and } y \in C^I \} \geq n \}$
- $(\leq n R.C)^I = \{ x \mid \#\{ y \mid (x,y) \in R^I \text{ and } y \in C^I \} \leq n \}$

- ...and to role expressions:

- $U^I = \Delta^I \times \Delta^I$ $(R^-)^I = \{ (y,x) \mid (x,y) \in R^I \}$

- ...and to axioms:

- $C(a)$ holds, if $a^I \in C^I$ $R(a,b)$ holds, if $(a^I, b^I) \in R^I$
- $C \sqsubseteq D$ holds, if $C^I \subseteq D^I$ $R \sqsubseteq S$ holds, if $R^I \subseteq S^I$
- $\text{Disjoint}(R,S)$ holds if $R^I \cap S^I = \emptyset$
- $S_1 \circ S_2 \circ \dots \circ S_n \sqsubseteq R$ holds if $S_1^I \circ S_2^I \circ \dots \circ S_n^I \subseteq R^I$

一个解释就是一个公理集合的模型，如果所有的公理在解释下成立。

解释器实例



If we consider, for example, the knowledge base consisting of the axioms

```
Professor  $\sqsubseteq$  FacultyMember  
Professor(rudiStuder)  
hasAffiliation(rudiStuder, aifb)
```

then we could set

$$\Delta = \{a, b, \text{Ian}\}$$
$$I_I(\text{rudiStuder}) = \text{Ian}$$
$$I_I(\text{aifb}) = b$$
$$I_C(\text{Professor}) = \{a\}$$
$$I_C(\text{FacultyMember}) = \{a, b\}$$
$$I_R(\text{hasAffiliation}) = \{(a, b), (b, \text{Ian})\}$$

Intuitively, these settings are nonsense, but they nevertheless determine a valid interpretation.

直观地，这些设定没有意义的，但是他们仍然定义了合法的解释。



实例中的解释是否为一个模型？

If we consider, for example, the knowledge base consisting of the axioms

```
Professor  $\sqsubseteq$  FacultyMember  
Professor(rudiStuder)  
hasAffiliation(rudiStuder, aifb)
```

then we could set

$$\begin{aligned}\Delta &= \{a, b, \text{Ian}\} \\ I_I(\text{rudiStuder}) &= \text{Ian} \\ I_I(\text{aifb}) &= b \\ I_C(\text{Professor}) &= \{a\} \\ I_C(\text{FacultyMember}) &= \{a, b\} \\ I_R(\text{hasAffiliation}) &= \{(a, b), (b, \text{Ian})\}\end{aligned}$$

Intuitively, these settings are nonsense, but they nevertheless determine a valid interpretation.

不是，因为如果是一个模型，我们需要 $(\text{rudiStuder}^I, \text{aifb}^I) \in \text{hasAffiliation}^I$ ，即， $(\text{Ian}, b) \in \{(a, b), (b, \text{Ian}), (\text{Ian}, b)\}$ ，而且 Ian 不包含在 $I_C(\text{Professor})$ 中

以下解释是一个模型

```
Professor  $\sqsubseteq$  FacultyMember  
Professor(rudiStuder)  
hasAffiliation(rudiStuder, aifb)
```

```
 $\Delta = \{a, r, s\}$   
 $I_I(\text{rudiStuder}) = r$   
 $I_I(\text{aifb}) = a$   
 $I_C(\text{Professor}) = \{r\}$   
 $I_C(\text{FacultyMember}) = \{r, s\}$   
 $I_R(\text{hasAffiliation}) = \{(r, a)\}$ 
```

如果基于前面描述的结构化方法的解释对于知识库而言是有意义的，那么这种解释就被称为知识库的模型（**model**）。

知识库的模型



```
Professor  $\sqsubseteq$  FacultyMember  
Professor(rudiStuder)  
hasAffiliation(rudiStuder, aifb)
```

	Model 1	Model 2	Model 3
Δ	$\{a, r, s\}$	$\{1, 2\}$	$\{\spadesuit\}$
$I_I(\text{rudiStuder})$	r	1	\spadesuit
$I_I(\text{aifb})$	a	2	\spadesuit
$I_C(\text{Professor})$	$\{r\}$	$\{1\}$	$\{\spadesuit\}$
$I_C(\text{FacultyMember})$	$\{a, r, s\}$	$\{1, 2\}$	$\{\spadesuit\}$
$I_R(\text{hasAffiliation})$	$\{(r, a)\}$	$\{(1, 1), (1, 2)\}$	$\{(\spadesuit, \spadesuit)\}$

模型使用集合论的术语表达了知识库的结构，一个知识库可以有多个模型，上面三个模型都描述了知识库的一个有意义的解释。

上述三个模型都可以得到我们并不想要的逻辑推理 **$\text{aifb} \in \text{FacultyMember}$** !

知识库的模型



- 知识库的每个模型提供了知识库的一个可能的视图(view)或实现(realization)。模型获得了必须的知识库结构，但也可能加入额外的关系，这通常是我们不希望的。为了剔除这些额外的关系，我们在定义逻辑结论的时候需要考虑到所有的模型。
 - 如果模型表达了知识库的所有可能的视图或实现，那么所有模型共有的东西必然是对于知识库全局有效的逻辑结论。
-

知识库的模型



- 令 K 为SROIQ的知识库， α 是一个一般包含公理或者个体的赋值，那么 α 是 K 的逻辑结论，记为如果 α^I 则 $K \models \alpha$ 。

$K \models C \sqsubseteq D$	当且仅当 $(C \sqsubseteq D)^I \text{ f. a. } I \models K$	当且仅当 $C^I \subseteq D^I \text{ f. a. } I \sqsubseteq K$
$K \models C(a)$	当且仅当 $(C(a))^I \text{ f. a. } I \models K$	当且仅当 $a^I \in C^I \text{ f. a. } I \models K$
$K \models R(a, b)$	当且仅当 $(R(a, b))^I \text{ f. a. } I \models K$	当且仅当 $(a^I, b^I) \in R^I \text{ f. a. } I \models K$
$K \models \neg R(a, b)$	当且仅当 $(\neg R(a, b))^I \text{ f. a. } I \models K$	当且仅当 $(a^I, b^I) \notin R^I \text{ f. a. } I \models K$

- 第一行陈述：如果 $C \sqsubseteq D$ 是 K 的逻辑结论，则对于 K 所有的模型 I 有 $(C \sqsubseteq D)^I$,当且仅当对于 K 的所有模型 I 有 $C^I \subseteq D^I$

反例



Returning to our running example knowledge base, let us show formally that `FacultyMember(aifb)` is not a logical consequence. This can be done by giving a model M of the knowledge base where $\text{aifb}^M \notin \text{FacultyMember}^M$. The following determines such a model.

$$\Delta = \{a, r\}$$

$$I_I(\text{rudiStuder}) = r$$

$$I_I(\text{aifb}) = a$$

$$I_C(\text{Professor}) = \{r\}$$

$$I_C(\text{FacultyMember}) = \{r\}$$

$$I_R(\text{hasAffiliation}) = \{(r, a)\}$$

通过谓词逻辑定义直接语义

□ OWL 2 DL 被认为是一阶谓词逻辑的片段

□ 可以把SROIQ知识库转换成一阶谓词逻辑

- 它表明OWL的形式语义是建立在经久不衰的数理逻辑基础上。
- 有助于把OWL的形式语义表达给已经一些形式逻辑背景的读者。

$$\pi(C \sqsubseteq D) = (\forall x)(\pi_x(C) \rightarrow \pi_x(D))$$

$$\pi_x(A) = A(x)$$

$$\pi_x(\neg C) = \neg \pi_x(C)$$

$$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$$

$$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$$

$$\pi_x(\forall R.C) = (\forall x_1)(R(x, x_1) \rightarrow \pi_{x_1}(C))$$

$$\pi_x(\exists R.C) = (\exists x_1)(R(x, x_1) \wedge \pi_{x_1}(C))$$

$$\pi(R_1 \sqsubseteq R_2) = (\forall x)(\forall y)(\pi_{x,y}(R_1) \rightarrow \pi_{x,y}(R_2))$$

$$\pi_{x,y}(S) = S(x, y)$$

$$\pi_{x,y}(R^-) = \pi_{y,x}(R)$$

$$\pi_{x,y}(R_1 \circ \dots \circ R_n) = (\exists x_1) \dots (\exists x_{n-1})$$

$$\left(\pi_{x,x_1}(R_1) \wedge \bigwedge_{i=1}^{n-2} \pi_{x_i, x_{i+1}}(R_{i+1}) \wedge \pi_{x_{n-1}, y}(R_n) \right)$$

$$\pi(\text{Ref}(R)) = (\forall x) \pi_{x,x}(R)$$

$$\pi(\text{Asy}(R)) = (\forall x)(\forall y)(\pi_{x,y}(R) \rightarrow \neg \pi_{y,x}(R))$$

$$\pi(\text{Dis}(R_1, R_2)) = \neg (\exists x)(\exists y)(\pi_{x,y}(R_1) \wedge \pi_{x,y}(R_2))$$

$$\pi_x(\geq n S.C) = (\exists x_1) \dots (\exists x_n) \left(\bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right)$$

$$\pi_x(\leq n S.C) = \neg (\exists x_1) \dots (\exists x_{n+1}) \left(\bigwedge_{i \neq j} (x_i \neq x_j) \wedge \bigwedge_i (S(x, x_i) \wedge \pi_{x_i}(C)) \right)$$

$$\pi_x(\{a\}) = (x = a)$$

$$\pi_x(\exists S.\text{Self}) = S(x, x)$$



不一致性和满足性问题

- 如果一个知识库至少有一个模型，则其称为可满足的satisfiable或者一致(consistent)的。否则称其为不可满足unsatisfiable，或者是不一致的(inconsistent)，矛盾的。
- 对于一个表达式C，如果存在一个模型使得 $C^I \neq \Phi$ ，则这个表达式称为是可满足的，否则称为不可满足。
- 知识库或者一个具名类的不可满足性通常可以指出模型错误。

Unicorn(beauty)

Unicorn \sqsubseteq Fictitious

Unicorn \sqsubseteq Animal

Fictitious \sqcap Animal $\sqsubseteq \perp$

这个知识库是不一致的，因为beauty会是一个Fictitious和Animal,这是最后一个公理不允许的，去掉第一条个体赋值，知识库是一致的，但是Unicorn是不可满足的（即是空集），因为Unicorn的存在会导致前后矛盾。



描述逻辑常见构造算子语义

构造算子	语法	语义	例子
原子概念	A	$A^I \subseteq \Delta^I$	
原子关系	R	$R^I \subseteq \Delta^I \times \Delta^I$	has-child
对概念C,D和关系(role)R			
合取	$C \sqcap D$	$C^I \cap D^I$	Human \sqcap Male
析取	$C \sqcup D$	$C^I \cup D^I$	Doctor \sqcup Lawyer
非	$\neg C$	$\Delta^I \setminus C^I$	\neg Male
存在量词	$\exists R.C$	$\{x / \exists y. \langle x, y \rangle \in R^I \wedge y \in C^I\}$	\exists has-child.Male
全称量词	$\forall R.C$	$\{x / \forall y. \langle x, y \rangle \in R^I \Rightarrow y \in C^I\}$	\forall has-child.Doctor



描述逻辑常见构造算子语义

构造算子	语法	语义	例子
数量约束	$\geq n R . C$	$\{x / \mid \{y / \langle x, y \rangle \in R^I, y \in C^I\} / \geq n\}$	≥ 3 has-child .Male
	$\leq n R . C$	$\{x / \mid \{y / \langle x, y \rangle \in R^I, y \in C^I\} / \leq n\}$	≤ 3 has-child .Male
逆	R^-	$\{\langle y, x \rangle / \langle x, y \rangle \in R^I\}$	has-child ⁻
传递闭包	R^*	$(R^I)^*$	has-child [*]

另外，有两个类似于FOL中的全集(true)和空集(false)的算子

top	\top	Δ^I	Male \sqcup \neg Male
Bottom	\perp	\emptyset	Man \sqcap \neg Man



附录：描述逻辑（OWL）的Turtle语法(个体)

```
:Mary rdf:type :Woman .
```

```
:John :hasWife :Mary .
```

```
:John owl:differentFrom :Bill .
```

$$\{John\} \sqcap \{Bill\} \sqsubseteq \perp$$

```
:James owl:sameAs :Jim.
```

$$\{John\} \equiv \{Jim\}$$

```
:John :hasAge "51"^^xsd:nonNegativeInteger .
```

```
[] rdf:type owl:NegativePropertyAssertion ;  
   owl:sourceIndividual :Bill ;  
   owl:assertionProperty :hasWife ;  
   owl:targetIndividual :Mary .
```

$$\neg \text{hasWife}(Bill, Mary)$$

```
[] rdf:type owl:NegativePropertyAssertion ;  
   owl:sourceIndividual :Jack ;  
   owl:assertionProperty :hasAge ;  
   owl:targetValue 53 .
```



附录：描述逻辑（OWL）的Turtle语法（类+属性）

```
:Woman rdfs:subClassOf :Person .
```

```
:Person owl:equivalentClass :Human .
```

```
[  
  rdf:type      owl:AllDisjointClasses ;  
  owl:members ( :Woman :Man ) .  
]
```

Woman \sqcap Man $\sqsubseteq \perp$

```
:hasWife rdfs:subPropertyOf :hasSpouse .
```

```
:hasWife rdfs:domain :Man ;  
        rdfs:range  :Woman .
```

附录：描述逻辑（OWL）的Turtle语法（复杂类）



```
:Mother owl:equivalentClass [  
  rdf:type owl:Class ;  
  owl:intersectionOf ( :Woman :Parent )  
] .
```

Mother \equiv Woman \sqcap Parent

```
:Parent owl:equivalentClass [  
  rdf:type owl:Class ;  
  owl:unionOf ( :Mother :Father )  
] .
```

Parent \equiv Mother \sqcup Father

```
:ChildlessPerson owl:equivalentClass [  
  rdf:type owl:Class ;  
  owl:intersectionOf ( :Person [ owl:complementOf :Parent ] )  
] .
```

ChildlessPerson \equiv Person $\sqcap \neg$ Parent

```
:Grandfather rdfs:subClassOf [  
  rdf:type owl:Class ;  
  owl:intersectionOf ( :Man :Parent )  
] .
```

附录：描述逻辑（OWL）的Turtle语法（复 杂类）



```
:Jack rdf:type [
  rdf:type          owl:Class ;
  owl:intersectionOf ( :Person
                        [ rdf:type          owl:Class ;
                          owl:complementOf :Parent      ]
                        )
] .
```

Person \sqcap \neg Parent (Jack)



附录：描述逻辑（OWL）的Turtle语法（限制）

```
:Parent owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty :hasChild ;  
  owl:someValuesFrom :Person  
] .
```

Parent $\equiv \exists \text{hasChild}.\text{Person}$

```
:Orphan owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty [ owl:inverseOf :hasChild ] ;  
  owl:allValuesFrom :Dead  
] .
```

Orphan $\equiv \forall \text{hasChild}^{\neg}.\text{Dead}$

附录：描述逻辑（OWL）的Turtle语法（限制）



```
:JohnsChildren owl:equivalentClass [  
  rdf:type          owl:Restriction ;  
  owl:onProperty   :hasParent ;  
  owl:hasValue     :John  
] .
```

JohnsChildren $\equiv \exists \text{hasParent}.\{\text{John}\}$

```
:NarcisticPerson owl:equivalentClass [  
  rdf:type          owl:Restriction ;  
  owl:onProperty   :loves ;  
  owl:hasSelf      "true"^^xsd:boolean .  
] .
```

NarcisticPerson $\equiv \exists \text{loves}.\text{Self}$



附录：描述逻辑（OWL）的Turtle语法（限制）

≤ 4 hasChild.Parent (John)

```
:John rdf:type [  
  rdf:type                owl:Restriction ;  
  owl:maxQualifiedCardinality "4"^^xsd:nonNegativeInteger ;  
  owl:onProperty          :hasChild ;  
  owl:onClass              :Parent  
] .
```

```
:John rdf:type [  
  rdf:type                owl:Restriction ;  
  owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;  
  owl:onProperty          :hasChild ;  
  owl:onClass              :Parent  
] .
```

≥ 2 hasChild.Parent (John)

```
:John rdf:type [  
  rdf:type                owl:Restriction ;  
  owl:qualifiedCardinality "3"^^xsd:nonNegativeInteger ;  
  owl:onProperty          :hasChild ;  
  owl:onClass              :Parent  
] .
```

$= 3$ hasChild.Parent (John)

附录：描述逻辑（OWL）的Turtle语法（限制）



```
:John  rdf:type  [
  rdf:type      owl:Restriction ;
  owl:cardinality  "5"^^xsd:nonNegativeInteger ;
  owl:onProperty   :hasChild
] .
```

=5 hasChild.⊤ (John)

```
:MyBirthdayGuests  owl:equivalentClass  [
  rdf:type      owl:Class ;
  owl:oneOf    ( :Bill   :John   :Mary )
] .
```

MyBirthdayGuests ≡ {Bill, John, Mary}



附录：描述逻辑（OWL）的Turtle语法（属性）

```
:hasParent owl:inverseOf :hasChild .
```

```
:Orphan owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty [ owl:complementOf :hasChild ] ;  
  owl:allValuesFrom :Dead  
] .
```

Orphan $\equiv \forall \text{hasChild}^c . \text{Dead}$

```
:hasSpouse rdf:type owl:SymmetricProperty .
```

```
:hasChild rdf:type owl:AsymmetricProperty .
```

```
:hasParent owl:propertyDisjointWith :hasSpouse .
```

```
:hasRelative rdf:type owl:ReflexiveProperty .
```

```
:parentOf rdf:type owl:IrreflexiveProperty .
```

```
:hasHusband rdf:type owl:FunctionalProperty .
```

```
:hasHusband rdf:type owl:InverseFunctionalProperty .
```

```
:hasAncestor rdf:type owl:TransitiveProperty .
```

附录：描述逻辑（OWL）的Turtle语法（属性）



```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ).
```

hasParent ◦ hasParent \sqsubseteq hasGrandParent

附录：描述逻辑（OWL）的Turtle语法（数据类型）



```
:personAge owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:onDatatype xsd:integer;
  owl:withRestrictions (
    [ xsd:minInclusive "0"^^xsd:integer ]
    [ xsd:maxInclusive "150"^^xsd:integer ]
  )
] .
```

Datatype facets

```
:majorAge owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:intersectionOf (
    :personAge
    [ rdf:type rdfs:Datatype;
      owl:datatypeComplementOf :minorAge ]
  )
] .
```

```
:toddlerAge owl:equivalentClass
[ rdf:type rdfs:Datatype;
  owl:oneOf ( "1"^^xsd:integer "2"^^xsd:integer )
] .
```