语义网与知识图谱

上海大学计算机学院

主讲： 刘 炜

知识图谱推理

上海大学计算机学院 刘炜

2020年11月

内容概览

一、推理概述

二、基于演绎的知识图谱推理

三、基于归纳的知识图谱推理

内容概览

什么是推理？

什么是推理


王健林老婆的儿子
百度一下
网页 资讯 视频 图片 知道 文库 贴吧 采购 地图 更多»
百度为您找到相关结果约907,000个
搜索工具
林宁儿子：
王思聪
王思聪，1988年1月3日出生于辽宁省大连市，毕业于伦敦大学学院哲学系，万达集团董事长王健林的独子，北京普思投资有限公司董事长、IG电子竞技俱乐部创始人、万达集团董事... 详情>>
来自百度百科


Known Facts
演绎 归纳 溯因
预测 推断 类比
New Facts
New Knowledge

演绎推理

Top-down logic

肯定前件假言推理

1. P→Q (条件声明)
2. P (假设声明)
3. Q (演绎结论)

1. 如果今天是周二，小明会去上班.
2. 今天是周二.
3. 因此小明会去上班.

否定后件假言推理

1. P→Q
2. ¬Q
3. 因此，可以推出¬P

1.如果会下雨，天上会有云.
2. 天上没有云.
3. 因此今天不会下雨.

假言三段论

1. P→Q
2. Q →R
3. 因此，P →R

1. 如果小明病了，他就要请假.
2. 如果小明请假了，他就会错过他的讨论课。
3. 因此，如果小明病了，他就会错过他的讨论课。

演绎三段论

1. P→Q
2. O⊑P或O∈P
3. 因此，O →Q

1. 人都要喝水.
2. 男人也是人。
3. 男人都要喝水。

归纳推理

Bottom-up logic

归纳泛化

占样本比例Q的集合具有属性A

因此：

占所有人群比例Q的集合具有属性A

一个瓮里有20个黑色或白色的球。为了估计它们各自的数目，你抽取四个球的样本，发现三个是黑色的，一个是白色的。一个很好的归纳概括是瓮中有15个黑球和5个白球。

统计推理

占比Q的人群P具有属性A

个体 X是人群P的成员

因此：

X具有属性A相应的概率是Q

90%就读于某高中的同学都上了大学，如果小明是这所高中的同学，那么可以由统计推理得出小明有90%的概率会上大学。

溯因推理

Inference to the best explanation

是在给定一个或多个已有观察事实O（Observation），并根据已有的知识 T（Theory）推断出对已有观察最简单且最有可能的解释的过程

例如，当一个病人显示出某种病症，而造成这种病症的原因可能有很多，寻找这个病人例子里最可能的原因的过程就是溯因推理。

最佳解释推理

在溯因推理中，要使基于知识T而生成的对观察O的解释E是合理的，需要两个条件：

1. **E可以由T和O经过推理得出。**
2. **E和T是相关相容的。**

例如，我们知道下雨了马路一定会湿（T），如果观察到马路湿了（O）,可以通过溯因推理得到很大概率是因为下雨了（E）。

类比推理

Inference via Analogy

可以看做只基于对一个事物的观察而进行的对另一个事物的归纳推理，是通过寻找两者之间可以类比的信息，将已知事物上的结论迁移到新的事物上的过程。

例如，小明和小红是同龄人，他们都喜欢歌手A和B，且小明还喜欢歌手C，通过类比推理可以得出小红也喜欢C。

由于被类比的两个事物虽然有可类比的信息，却不一定同源，而且有可能新推理出的信息和已知的可类比信息没有关系，所以类比推理常常会导致错误的结论，称为不当类比。

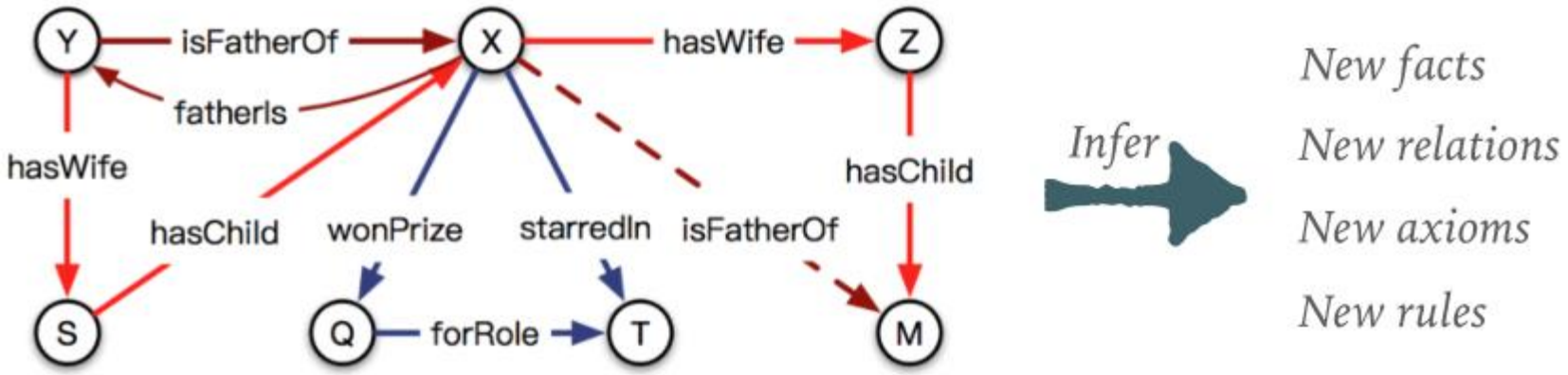
例如在上例中，如果歌手 C 和歌手 A、歌手 B 完全不是一种类型或一个领域的歌手，那么小明喜欢歌手C与他喜欢歌手A和歌手B是完全无关的，所以将“喜欢歌手 C”的结论应用到小红身上不合适。

其他推理

- 根据不确定的观察信息以及不确定性的知识进行推理的不确定性推理，不确定性推理与前述四种推理方式的最大区别是其所能利用的推理信息都具有很大的不确定性。
- 在知识演变的过程中，根据原有的推论可否被推翻可以分为不会被推翻的单调推理以及可能会被推翻的非单调推理。
- 从推理过程精确性来看，又可分为精确推理和模糊推理。
- 不同的研究领域也有各自的推理问题，如自然语言推理和知识图谱推理。

知识图谱推理

主要关注围绕关系的推理，即：基于图谱中已有的事实或关系和推断未知的事实或关系，一般着重考察实体、关系和图谱结构三个方面面的特征信息。



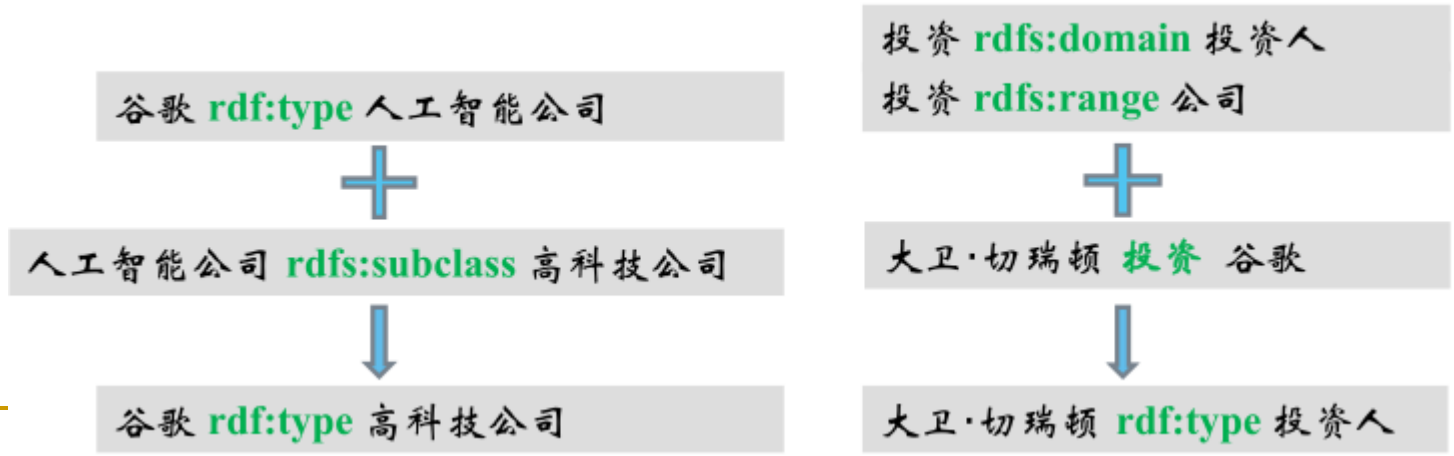
从知识图谱的生命周期来看，不同的阶段都涉及不同的推理任务，包括知识图谱补全 、不一致性检测、查询扩展等。

知识图谱推理

- 知识图谱推理的主要方法
  - ✓ 基于演绎的知识图谱推理，如基于描述逻辑 、Datalog、产生式规则等；
  - ✓ 基于归纳的知识图谱推理，如路径推理 、表示学习 、规则学习 、基于强化学习的推理等。

基于符号的推理——本体推理

- 传统的符号逻辑推理中主要与知识图谱有关的推理手段是基于描述逻辑的本体推理。
- 描述逻辑主要被用用来对事物的本体进行建模和推理，用来描述和推断概念分类及其概念之间的关系。
- 主要方法：
  - ✓ 基于描述逻辑表运算（Tableaux）及改进的方法：FaCT++、Racer、Pellet Hermit等
  - ✓ 基于逻辑编程与Datalog的方法如KAON、RDFox等
  - ✓ 基于产生式规则的算法（如Rete）：Jena 、Drools、GraphDB等



内容概览

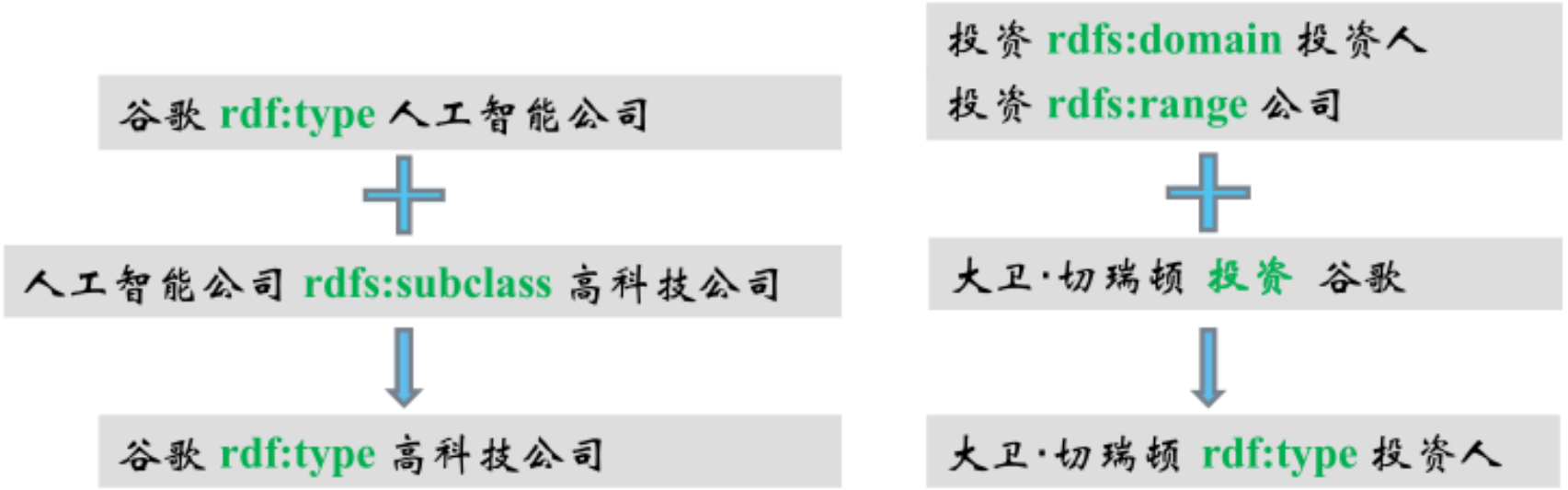
一、推理概述

二、基于演绎的知识图谱推理

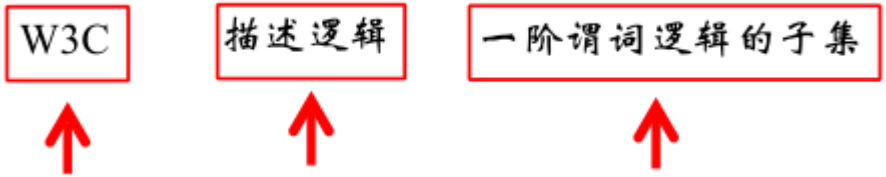
- ✓ 本体概念推理
- ✓ 基于Datalog的推理
- ✓ 产生式规则系统中的推理

三、基于归纳的知识图谱推理

基于RDFS的简单推理



OWL本体语言



OWL本体语言

- 是知识图谱语言中**最规范，最严谨，表达能力最强**的语言
- 基于RDF语法，使表示出来的文档具有语义理解的结构基础
- 促进了统一词汇表的使用，定义了丰富的语义词汇
- 允许逻辑推理

OWL本体语言

描述逻辑系统

- 一个描述逻辑系统包括四个基本的组成部分
  1) 最基本的元素：概念、关系和个体
  2) TBox术语集 (概念术语的公理集合)
  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。

OWL本体语言

- TBox——泛化的知识
  - ✓ 描述概念和关系的知识，被称之为公理 (Axiom)
  - ✓ 由于概念之间存在包含关系，TBox知识形成类似格(Lattice)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——具体个体的信息
  - ✓ ABox包含外延知识 (又称断言 (Assertion))，描述论域中的特定个体。

描述逻辑的知识库， 即TBox， 即ABox

OWL本体语言

- Tbox语言
  - 定义：引入概念以及关系的名称
    - 例如，Mother，Person，has_child
  - 包含：声明包含关系的公理
    - 例如，$\text{Mother} \sqsubseteq \exists \text{has\_child}.\text{Person}$

概念和概念，关系和概念之间可以组成更复杂的概念。

- Abox语言
  - 概念断言——表示一个对象是否属于某个概念
    - 例如，Mother(Alice)，Person(Bob)
  - 关系断言——表示两个对象是否满足特定关系
    - 例如，has_child(Alice, Bob)

OWL本体语言

- 描述逻辑的语义

构造算子	语法	语义	例子
原子概念	A	$A^I \subseteq \Delta^I$	Human
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 ⊓ Male
析取	$C \sqcup D$	$C^I \cup D^I$	Doctor ⊔ Lawyer
非	$\neg C$	$\Delta^I \backslash C$	¬Male
存在量词	$\exists R.C$	$\{x \mid \exists y.\langle x,y\rangle \in R^I \wedge y \in C^I\}$	∃has_child.Male
全称量词	$\forall R.C$	$\{x \mid \forall y.\langle x,y\rangle \in R^I \Rightarrow y \in C^I\}$	∀has_child.Doctor

有了语义之后，我们可以进行推理。通过语义来保证推理的正确和完备性。

OWL本体语言

- 描述逻辑与OWL词汇的对应

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{Prsident_Bush} $\equiv$ {G_W_Bush}
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{John} $\sqsubseteq \neg${Peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
uniqueProperty	$T \sqsubseteq \leq 1P$	T $\sqsubseteq \leq$ 1hasMother
unambiguousProperty	$T \sqsubseteq \leq 1P^-$	T $\sqsubseteq \leq$ 1isMotherOf$^-$

描述逻辑推理任务

- 可满足性 (satisfiability)
- 分类 (classification)
- 实例化 (materialization)

推理就是通过各种方法获取新的知识或者结论，这些知识和结论满足语义。

OWL本体推理

- 可满足性 (satisfiability)
  - 本体可满足性
    - 检查一个本体是否可满足，即检查该本体是否有模型。如果本体不可满足，说明存在不一致
  - 概念可满足性
    - 检查某一概念的可满足性，即检查是否具有模型，使得针对该概念的解释不是空集

一个不可满足的本体

$\text{Man} \sqcap \text{Women} \sqsubseteq \bot$

Man(Allen)

Women(Allen)

一个不可满足的概念

$\text{Eternity} \sqsubseteq \bot$

针对可满足性推理采用Tableaux算法

OWL本体推理

- 分类 (classification)
  - 针对Tbox的推理，计算新的概念包含关系

$$\text{Mother} \sqsubseteq \text{Women},\ \text{Women} \sqsubseteq \text{Person} \Rightarrow \text{Mother} \sqsubseteq \text{Person}$$

注意：这里的分类和机器学习中分类的区别！

OWL本体推理

分类的例子

Apple ⊑ ∃beInvestedBy.(Fidelity ⊓BlackStone)

苹果由富达和黑石投资。

∃beFundedBy.Fidelity ⊑ InnovativeCompanies

借助富达融资的公司都是创新企业。

∃beFundedBy.BlackStone ⊑ InnovativeCompanies

借助黑石融资的公司都是创新企业。

beInvestedBy ⊑ beFundedBy

投资即是帮助融资。

富达基金和黑石基金都喜欢投资高新科技公司。

OWL本体推理

$Apple \sqsubseteq \exists beInvestedBy.(Fidelity \sqcap BlackStone)$

$\exists beFundedBy.Fidelity \sqsubseteq InnovativeCompanies$

$\exists beFundedBy.BlackStone \sqsubseteq InnovativeCompanies$

$beInvestedBy \sqsubseteq beFundedBy$

$Apple \sqsubseteq \exists beInvestedBy.Fidelity$

$Apple \sqsubseteq \exists beFundedBy.Fidelity$

$Apple \sqsubseteq InnovativeCompanies$

OWL本体推理

- 实例化 (materialization)
  - 实例化即计算属于某个概念或关系的所有实例的集合

计算新的类实例信息

Mother(Alice)
Mother ⊑ Women

Women(Alice)

计算新的二元关系

has_son(Alice, Bob)
has_son ⊑ has_child

has_child(Alice, Bob)

OWL本体推理

实例化(materialization)的一个例子:

一个兼并重组(可以是业务兼并，不是收购)套利策略:
策略思想:与大盘股公司兼并重组的上市企业有很高的预期收益。

$$\exists merge.BigCapital \sqsubseteq ValueSecurity$$

定义什么是大盘股(按照策略自己调整):
上证50和沪深300指数中的标的属于大盘股。

$$SZ50 \sqsubseteq BigCapital,\ HS300 \sqsubseteq BigCapital,\ SZ180 \sqsubseteq HS300$$

选股目标:找出兼并重组策略下所有高预期公司:

使用OWL的实例化推理

OWL本体推理

实例化(materialization)的一个例子:

一个兼并重组套利策略:

$\exists$ merge.BigCapital $\sqsubseteq$ ValueSecurity
SZ50 $\sqsubseteq$ BigCapital,
HS300 $\sqsubseteq$ BigCapital,
SZ180 $\sqsubseteq$ HS300

选股目标：兼并重组策略下所有收益预期高的公司：

merge(SZ300377, SH600570) 赢时胜和恒生电子在区块链方面有业务兼并
SZ180(SH600570) 恒生电子是上证180的成分股

推理：
HS300(SH600570)
BigCapital(SH600570)
ValueSecurity(SZ300377)

结论: SZ300377赢时胜在短期内是一家高收益公司。
这本质上用基于消息面的套利，推理机可以完成复杂股票筛选的过程。

典型推理算法——Tableaux算法

Tableaux运算

- 适用场合
  - 检查某一本体概念的**可满足性**，以及**实例检测**
- 基本思想
  - 通过一系列规则**构建Abox**，以检测可满足性，或者检测某一实例是否存在于某概念
  - 基本思想类似于一阶逻辑的**归结反驳**

基于Tableaux运算的方法

- Tableaux运算规则（以主要DL算子举例）

初始情况下，$\emptyset$ 是原始的Abox，迭代运用如下规则：

$\sqcap^{+}$-规则：若$C \sqcap D(x) \in \emptyset$，且$C(x), D(x) \notin \emptyset$，则$\emptyset := \emptyset \cup \{C(x), D(x)\}$；

$\sqcap^{-}$-规则：若$C(x), D(x) \in \emptyset$，且$C \sqcap D(x) \notin \emptyset$，则$\emptyset := \emptyset \cup \{C \sqcap D(x)\}$；

$\exists$-规则：若$\exists R.C(x) \in \emptyset$，且$R(x,y), C(y) \notin \emptyset$，则$\emptyset := \emptyset \cup \{R(x,y), C(y)\}$，

其中，y是新加进来的个体；

$\forall$-规则：若$\forall R.C(x), R(x,y) \in \emptyset$，且$C(y) \notin \emptyset$，则$\emptyset := \emptyset \cup \{C(y)\}$；

$\sqsubseteq$-规则：若$C(x) \in \emptyset, C \sqsubseteq D$，且$D(x) \notin \emptyset$，则$\emptyset := \emptyset \cup \{D(x)\}$；

$\bot$-规则：若$\bot (x) \in \emptyset$，则拒绝$\emptyset$；

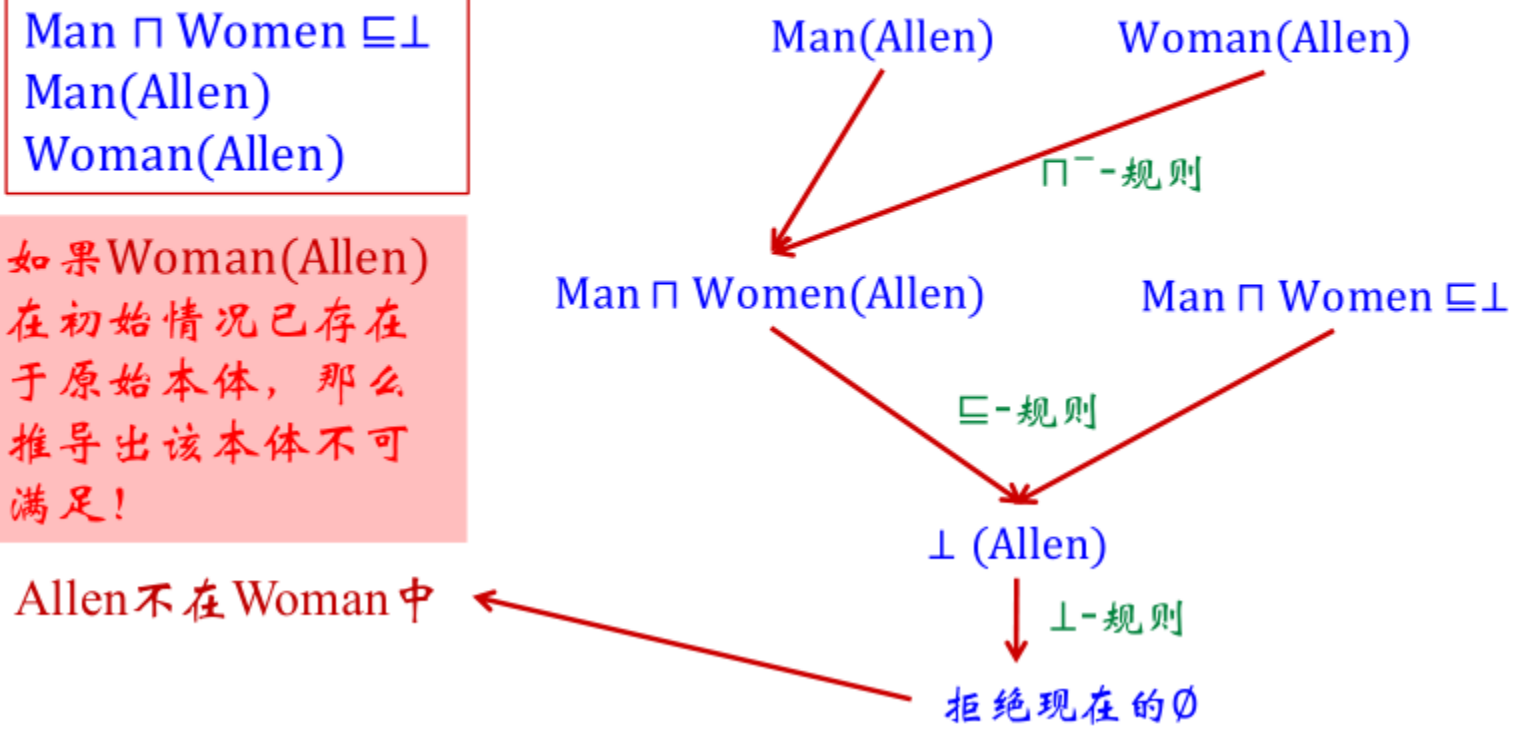
基于Tableaux运算的方法

- Tableaux运算规则举例

给定如下本体，检测是否可满足？

Man ⊓ Women ⊑ ⊥
Man(Allen)
Woman(Allen)

如果Woman(Allen)在初始情况已存在于原始本体，那么推导出该本体不可满足！



相关工具简介

- FaCT++
  - 曼彻斯特大学开发的描述逻辑推理机
  - 使用C++实现，且能与Protégé集成
  - Java版本名为Jfact，基于OWL API
- 使用举例

构建推理机

```
OWLReasonerFactory reasonerFactory = new JFactFactory();
OWLReasoner reasoner = this.reasonerFactory.createReasoner(ontology);
```

进行推理（分类）

```
reasoner.precomputeInferences(InferenceType.CLASS_HIERARCHY);
```

http://owl.man.ac.uk/factplusplus/

相关工具简介

- Racer
  - 美国Franz Inc.公司开发的以描述逻辑为基础的本体推理，也可以用作语义知识库
  - 支持OWL DL，支持部分OWL 2 DL
  - 支持单机和客户端/服务器两种模式
  - 用Allegro Common Lisp实现
- 使用举例

```
(classify-tbox &optional (tbox (current-tbox)))
```

进行TBox推理

```
(realize-abox &optional (abox (current-abox)))
```

进行ABox推理

https://www.ifis.uni-luebeck.de/index.php?id=385

相关工具简介

- Pellet

  https://github.com/stardog-union/pellet

  - 马里兰大学开发的本体推理机
  - 支持OWL DL的所有特性，包括枚举类和XML数据类型的推理
  - 支持OWL API以及Jena的接口

- 使用举例

```
PelletReasoner reasoner =
PelletReasonerFactory.getInstance().createReasoner( ontology);
```

构建一个推理机

```
NodeSet<OWLNamedIndividual> individuals =
reasoner.getInstances(Person, true);
```

通过查询接口进行推理

相关工具简介

- HermiT
  - 牛津大学开发的本体推理机
  - 基于hypertableau运算，更加高效
  - 支持OWL 2规则
- 使用举例

```
Reasoner hermit = new Reasoner(ontology);
```

构建一个推理机

```
System.out.println(hermit.isConsistent());
```

一致性检测

http://www.hermit-reasoner.com/

基于逻辑编程改写的方法

- 规则推理
  - 本体推理的局限:

    (1) 仅支持预定义的本体公理上的推理

    (无法针对自定义的词汇支持灵活推理)

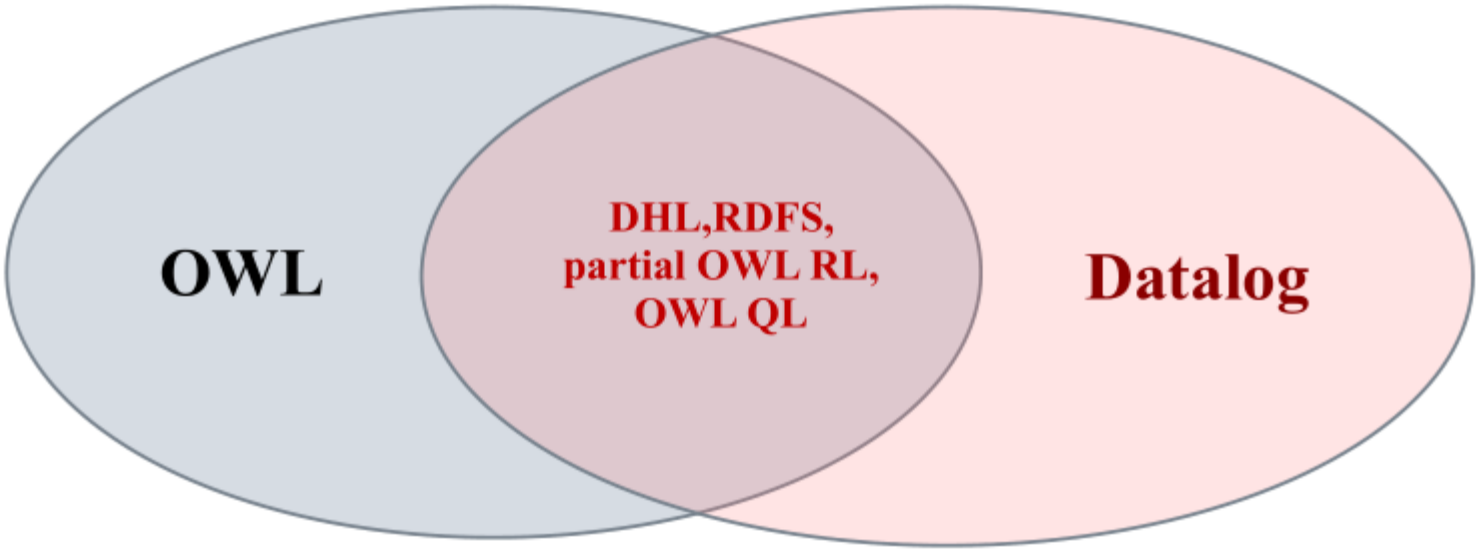
    (2) 用户无法定义自己的推理过程
  - 引入规则推理

    (1) 可以根据特定的场景定制规则，以实现用户自定义的推理过程

    (2) Datalog语言可以结合本体推理和规则推理

基于逻辑编程改写的方法

- Datalog语言
  - 面向知识库和数据库设计的逻辑语言，表达能力与OWL相当，支持递归
  - 便于撰写规则，实现推理



基于逻辑编程改写的方法

- Datalog语法
  - 原子 (Atom)
    - $p(t_1, t_2, \ldots, t_n)$
    - 其中p是谓词，n是目数，t_i是项(变量或常量)
    - 例如：has_child(X, Y)
  - 规则 (Rule)
    - $H: -B_1, B_2, \ldots, B_m.$
    - 由原子构建，其中H是头部原子，$B_1, B_2, \ldots, B_m$是体部原子
    - 例如：has_child(X, Y): −has_son(X, Y).

基于逻辑编程改写的方法

- Datalog语法
  - 事实 (Fact)
    - $F(c_1, c_2, \ldots, c_n):-$
    - 没有体部且没有变量的规则
    - 例如：has_child(Alice, Bob):−
  - Datalog程序是规则的集合
    - 例如：

Datalog程序P:
has_child(X, Y): −has_son(X, Y).
has_child(Alice, Bob): −

基于逻辑编程改写的方法

- Datalog推理举例

规则集

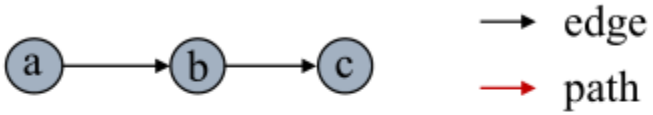
$path(x, y) :- edge(x, y).$

$path(x, y) :- path(x, z), path(z, y).$

事实集

edge(a, b).

edge(b, c).



基于逻辑编程改写的方法

- Datalog推理举例

规则集

$path(x, y) :- edge(x, y).$

$path(x, y) :- path(x, z), path(z, y).$

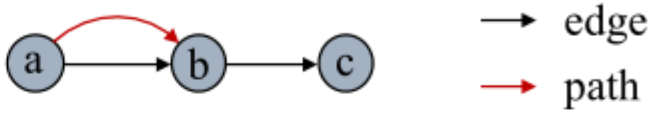
事实集

edge(a, b).

edge(b, c).

结果集

path(a, b).



基于逻辑编程改写的方法

- Datalog推理举例

规则集

$path(x, y) :- edge(x, y).$

$path(x, y) :- path(x, z), path(z, y).$

事实集

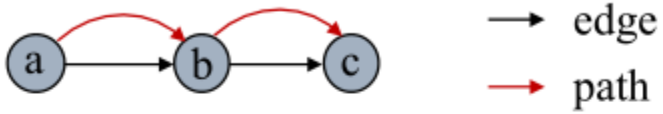
edge(a, b).

edge(b, c).

结果集

path(a, b).

path(b, c).



基于逻辑编程改写的方法

- Datalog推理举例

规则集

$path(x, y) :- edge(x, y).$

$path(x, y) :- path(x, z), path(z, y).$

事实集

$edge(a, b).$

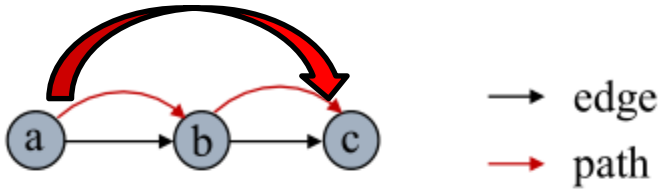
$edge(b, c).$

结果集

$path(a, b).$

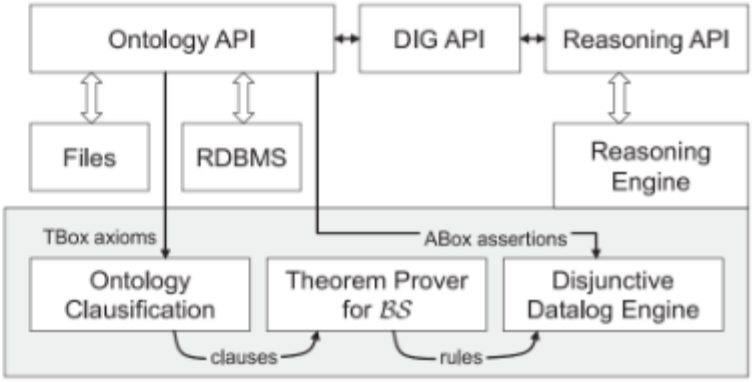
$path(b, c).$

$path(a,c).$



相关工具简介

- KAON2
  - OWL推理机、本体管理API
  - 基于一阶消解原理
  - 针对大规模ABox进行优化



- 使用举例

http://kaon2.semanticweb.org/

创建本体

```
OntologyManager ontologyManager=KAON2Manager.newOntologyManager();
Ontology ontology=ontologyManager.createOntology("http://kaon2.semanticweb.org/example04",new
HashMap<String,Object>());
```

创建推理机并用于查询

```
Reasoner reasoner=ontology.createReasoner();
Query whatDoPeopleKnowAbout=reasoner.createQuery(new Literal[] {
        KAON2Manager.factory().literal(true,person,new Term[] { X }),
        KAON2Manager.factory().literal(true,personKnowsAboutTopic,new Term[] { X,Y }),
    },new Variable[] { X,Y});
```

相关工具简介

- RDFox

  https://www.cs.ox.ac.uk/isg/tools/RDFox/

  - 由牛津大学开发的可扩展、跨平台、基于内存的RDF三元组存储系统
  - 支持并行Datalog推理、SPARQL查询

- 使用举例

创建本体与存储

```
OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
OWLOntology ontology =
manager.loadOntologyFromOntologyDocument(IRI.create("test.owl"));
DataStore store = new DataStore(DataStore.StoreType.ParallelSimpleNN, true);
```

导入本体进行推理

```
store.importOntology(ontology);
store.applyReasoning();
```

基于产生式规则的方法

- 产生式系统
  - 一种前向推理系统，可以按照一定机制执行规则从而达到某些目标，与一阶逻辑类似，也有区别
  - 应用
    - 自动规划
    - 专家系统
- 产生式系统的组成
  - 事实集合 (Working Memory)
  - 产生式/规则集合
  - 推理引擎

Feigenbaum研制的化学分子结构专家系统DENDRAL
Shortliffe研制的的诊断感染性疾病的专家系统MYCIN
…

基于产生式规则的方法

- 事实集/运行内存 (Working Memory, WM)
  - ✓ 事实 (WME)的集合
  - ✓ 用于存储当前系统中所有事实
- 事实 (Working Memory Element, WME)
  - ✓ 描述对象
    - 形如(type $attr_1$: val_1 $attr_2$: val_2 ... $attr_n$: val_n)，其中type, $attr_i$, val_i均为**原子（常量）**
    - 例如: (student name: Alice age: 24)

    类比类和对象
  - ✓ 描述关系 (Refication)
    - 例如: (basicFact relation: olderThan firstArg: John secondArg: Alice)
      简记为(olderThan John Alice)

基于产生式规则的方法

- 产生式集合 (Production Memory, PM)
  - 产生式的集合
- 产生式
  - **IF** *conditions* **THEN** *actions*
  - *conditions*是由条件组成的集合，又称为LHS（Left Hand Side）
  - *actions*是由动作组成的序列，又称为RHS（Right Hand Side)

基于产生式规则的方法

- LHS
  - 条件 (condition)的集合，各条件之间是且的关系
  - 当LHS中所有条件均被满足，则该规则触发
  - 每个条件形如(type $attr_1$: $spec_1$ $attr_2$: $spec_2$ ... $attr_n$: $spec_n$)
    - 其中$spec_i$ 表示对$attr_i$ 的约束，形式可取下列中的一种
      - 原子，如：Alice　　(person name:Alice)
      - 变量，如：x (斜体)　　(person name:x)
      - 表达式，如：$[n+4]$　　(person age:$[n+4]$)
      - 布尔测试，如：$\{>10\}$　　(person age:$\{>10\}$)
      - 约束的与、或、非操作

基于产生式规则的方法

- RHS
  - 动作 (action) 的序列，执行时依次执行
  - 动作的种类如下：
    - **ADD *pattern***
      - 向WM中加入形如*pattern*的WME
    - **REMOVE *i***
      - 从WM中移除当前规则第*i*个条件匹配的WME
    - **MODIFY *i* *(attr spec)***
      - 对于当前规则第*i*个条件匹配的WME，将其对应于*attr*属性的值改为*spec*

基于产生式规则的方法

- 产生式 LHS RHS
  - **IF** *conditions* **THEN** *actions*
  - 例如：

IF (Student name: x)
Then ADD (Person name: x)

亦可写作（具体语法因不同系统而异）
(Student name: x) ⇒ ADD (Person name: x)

如果有一个学生名为?x，那么向事实集中加入一个事实，表示有一个名为?x的人

基于产生式规则的方法

- 推理引擎
  - 控制系统的执行

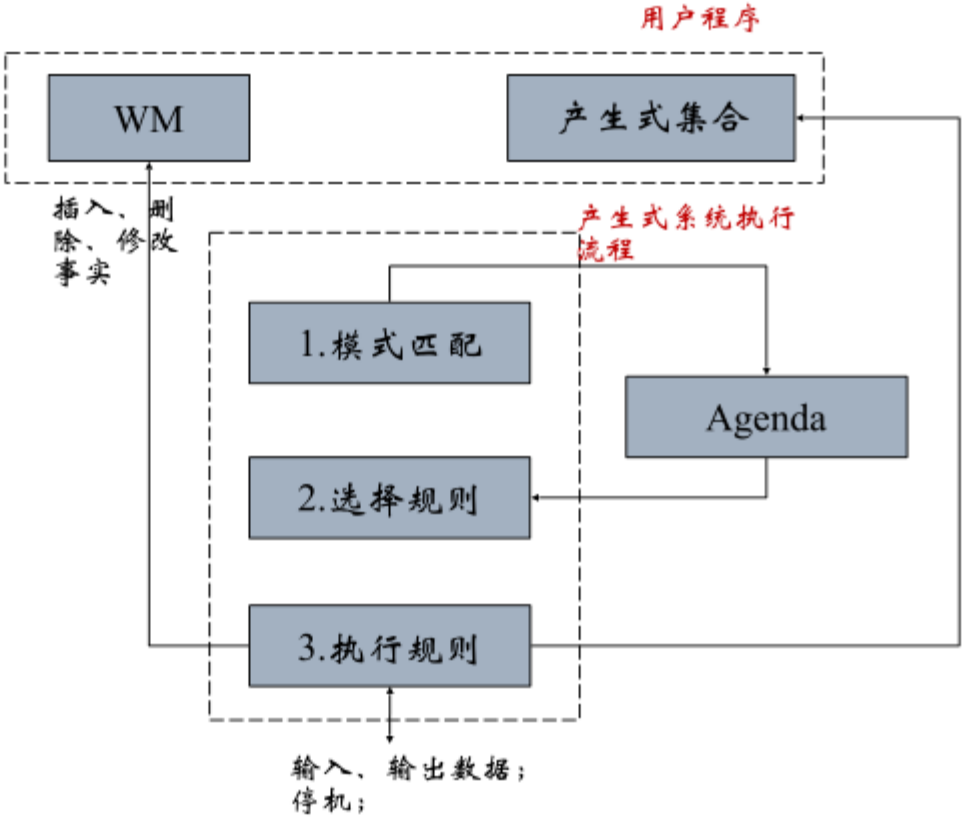
该步骤是产生式系统的核心
具体算法在后面介绍

    - 模式匹配
      - 用规则的条件部分匹配事实集中的事实，整个LHS都被满足的规则被触发，并被加入议程(agenda)
    - 解决冲突
      - 按一定的策略从被触发的多条规则中选择一条
    - 执行动作
      - 执行被选择出来的规则的RHS，从而对WM进行一定的操作

产生式系统=事实集+产生式集合+推理引擎

基于产生式规则的方法

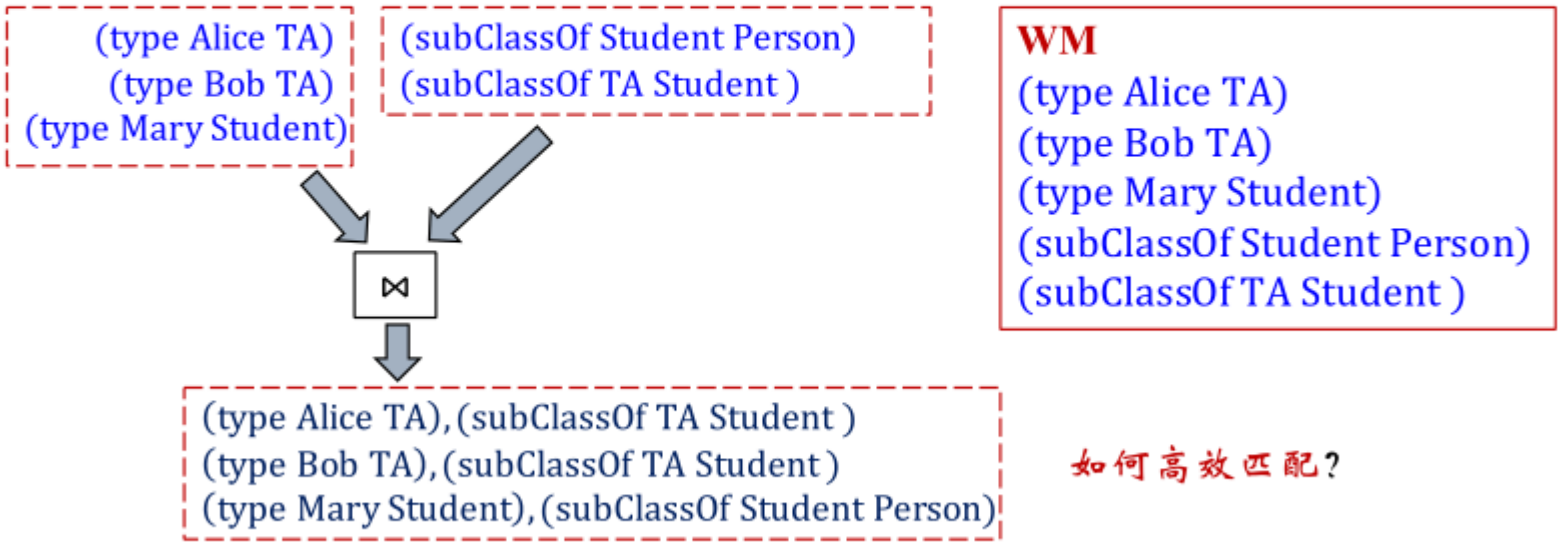
- 产生式系统执行流程



基于产生式规则的方法

- 模式匹配
  - 用每条规则的条件部分匹配当前WM

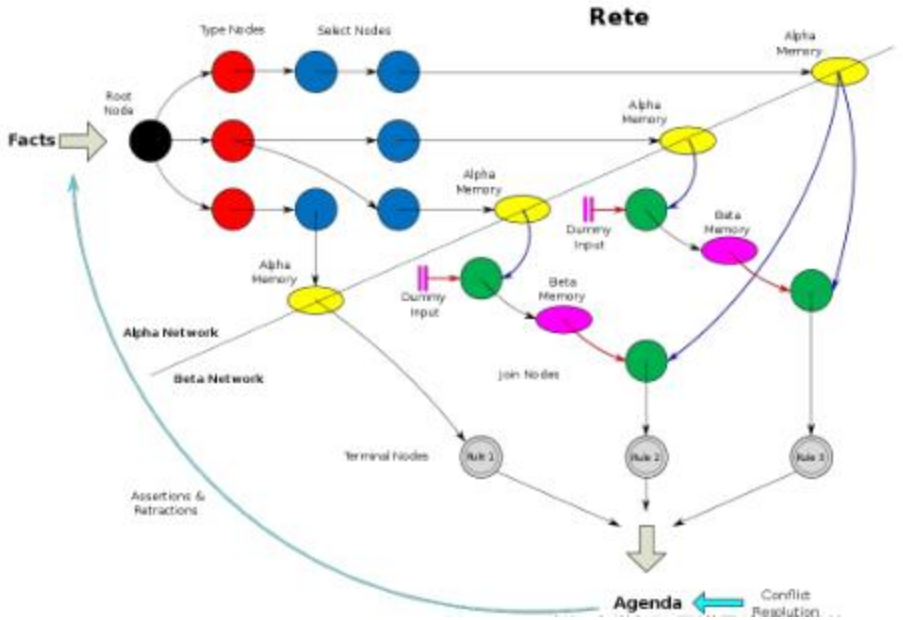
$$(\text{type } x\ y), (\text{subClassOf } y\ z) \Rightarrow \text{ADD } (\text{type } x\ z)$$



如何高效匹配?

基于产生式规则的方法

- 模式匹配
  - RETE算法
    - 1979年由Charles Forgy (CMU)提出
    - 将产生式的LHS组织成判别网络形式
    - 用空间换时间

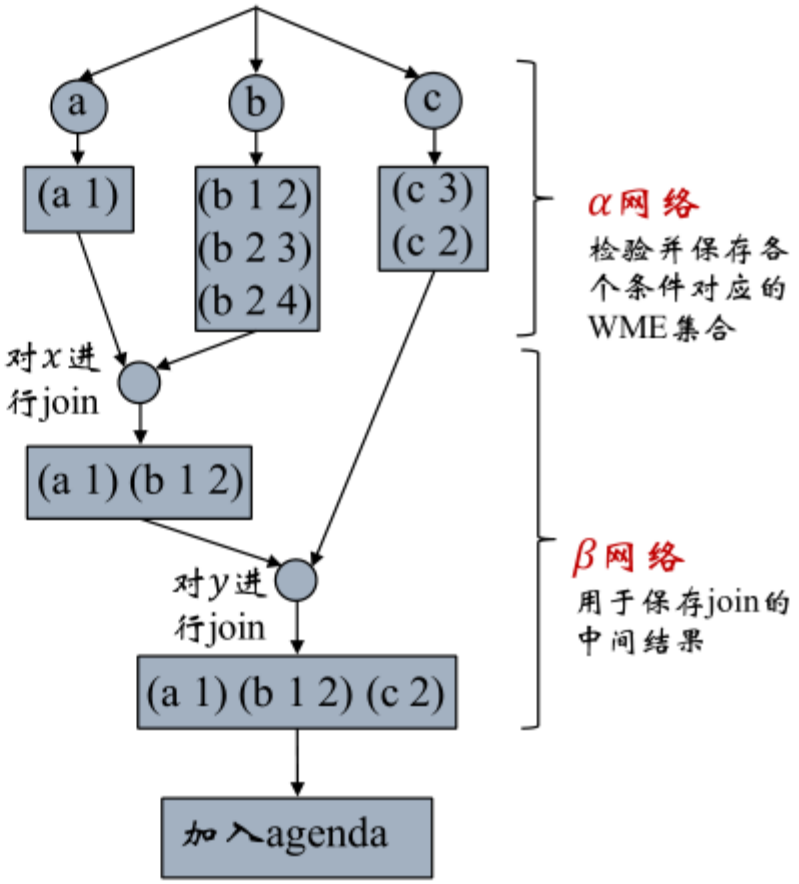


基于产生式规则的方法

- 模式匹配
  - RETE算法

Production Memory
(a x), (b x y), (c y) ⇒ …

Working Memory
(a 1)
(b 1 2)
(b 2 3)
(b 2 4)
(c 3)
(c 2)



基于产生式规则的方法

- 冲突解决
  - 从被触发的多条规则中选择一条
  - 常见策略
    - 随机选择 在推理的场景下，被触发的多条规则可全被执行
      - 从被触发的规则中随机选择一条执行
    - 具体性 (specificity)
      - 选择最具体的规则

      (Student name: x) ⇒ …

      (Student name: x age: 20) ⇒ …
    - 新近程度 (recency)
      - 选择最近没有被触发的规则执行动作

相关工具介绍

规则格式

```
rule "name"
    attributes
    when
        LHS
    then
        RHS
end
```

- Drools

  - 商用规则管理系统，其中提供了一个规则推理引擎
  - 核心算法基于RETE算法改进
  - 提供规则定义语言，支持嵌入Java代码

https://www.drools.org/

- 使用举例

创建容器与会话

```
KieServices ks = KieServices.Factory.get();
KieContainer kContainer = ks.getKieClasspathContainer();
KieSession kSession = kContainer.newKieSession("ksession-rules");
```

触发规则

```
kSession.fireAllRules();
```

相关工具介绍

- Jena

[ruleHasStudent: (?s :hasClass ?c) (?p :teaches ?c) -> (?p :hasStudent ?s)]

规则格式

  - 用于构建语义网应用的Java框架
  - 提供了处理RDF、RDFS、OWL数据的接口，还提供了一个规则引擎
  - 提供三元组的内存存储与查询

http://jena.apache.org/

- 使用举例

```
Model m = ModelFactory.createDefaultModel();
```

创建模型

```
Reasoner reasoner = new
GenericRuleReasoner(Rule.rulesFromURL("file:rule.txt"));
InfModel inf = ModelFactory.createInfModel(reasoner, m);
```

创建规则推理机

相关工具介绍

```
CONSTRUCT { ?
p :relatesTo :Cryptography } WHERE
{ { :Bob ?p :Alice } UNION { :Alice ?
p :Bob } }
```

规则格式(SPARQL)

- RDF4J (原Sesame)
  - 一个处理RDF数据的开源框架
  - 支持语义数据的解析、存储、推理和查询
  - 能够关联几乎所有RDF存储系统
  - 能够用于访问远程RDF存储

http://rdf4j.org/

- 使用举例

创建规则

```
String pre = "PREFIX : <http://foo.org/bar#>\n";
String rule = pre + "CONSTRUCT { ?p :relatesTo :Cryptography } WHERE
" + "{ { :Bob ?p :Alice } UNION { :Alice ?p :Bob } }";
```

创建推理知识库

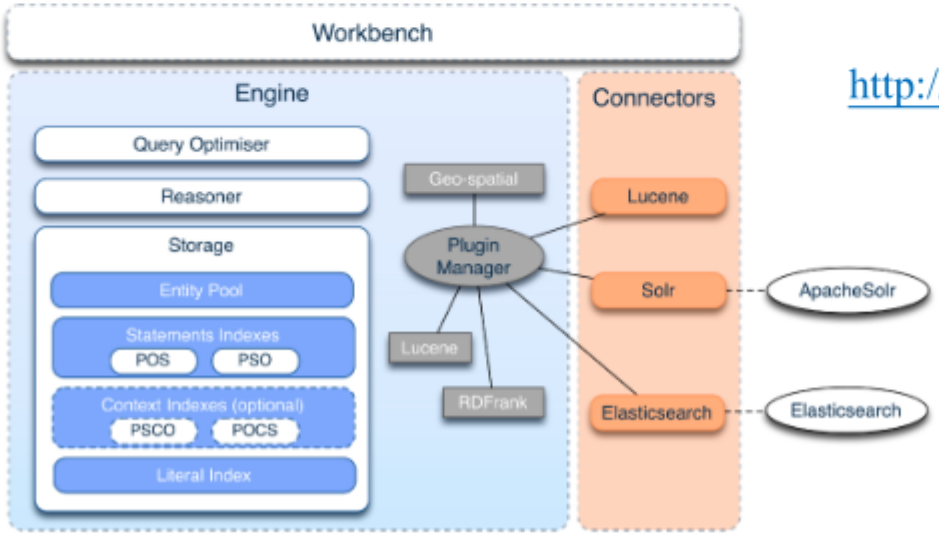
```
Repository repo = new SailRepository(new
CustomGraphQueryInferencer(new MemoryStore(),
QueryLanguage.SPARQL, rule, “”));
```

相关工具介绍

```
Id: rdfs2
  x a y [Constraint a != <rdf:type>]
  a <rdfs:domain> z [Constraint z != <rdfs:Resource>]
  ------------------------------
  x  <rdf:type>  z
```

规则格式

- GraphDB (原OWLIM)
  - 一个可扩展的语义数据存储系统（基于RDF4J）
  - 包含三元组存储、推理引擎、查询引擎
  - 支持RDFS，OWL DLP，OWL Horst，OWL 2 RL推理



http://graphdb.ontotext.com/

知识图谱推理总结

- **逻辑推理按照推理方式的不同包含两大类：演绎推理和归纳推理。其中，归纳推理又包含了溯因推理和类比推理等。**
- **知识图谱推理的主要方法**
  - ✓ 基于演绎的知识图谱推理，如基于描述逻辑 、Datalog、产生式规则等；
  - ✓ 基于归纳的知识图谱推理，如路径推理 、表示学习 、规则学习 等。
- **知识图谱推理的新进展**
  - ✓ 时序预测推理
  - ✓ 基于强化学习的推理
  - ✓ 基于元学习的少样本推理