第五章作业

1.

- (1)
  - 使用XML语法
  - RDF一句陈述包括：
    - 主词：资源
    - 述词：属性
    - 受词：属性值
- (2)
  - 语义网中会不断出现未知概念，RDFS在RDF基础上增加了一些词汇来拓展RDF的能力，可以自定义词汇用来描述新概念
  - 词汇之间还有联系，例如反义词、同义词等关系，OWL有更强的机械解释能力，可以理解这些语义关系并进一步使计算机理解数据
  - OWL语言的基础是本体概念和计算逻辑，OWL可以通过计算机程序进行推理，从而验证知识一致性或者使隐形知识显性化

2.

- (1)
  1. 启发式规则由结构规则和标签规则两类组成
  2. 结构规则分为三部分：
     1. 将新节点对加入状态s后，检查新的状态s'还是一致状态
     2. 考虑1-lookahead，判断未来一步是否可能构成一致状态s''
     3. 考虑2-lookahead，判断未来两步是否可能构成一致状态s'''
  3. 前驱节点规则：当前状态节点集合的前驱节点和当前状态节点中的关系结构在两个图中要一模一样
  4. 后继结点规则：当前状态节点集合的后继节点和当前状态节点中的关系结构在两个图中要一模一样
  5. 1-lookahead的in节点规则：无论是前驱还是后继节点，其在查询图中的入边的数量不能大于被查询图
  6. 1-lookahead的out节点规则：无论是前驱还是后继节点，其在查询图中的出边的数量不能大于被查询图
  7. 2-lookahead的in和out节点规则：要求与状态中不直接相连的2跳邻居节点，其在查询图中的出边和入边的数量均不能大于被查询图
- (2)

1. 将查询图分解为若干高度为2的树结构的子图stwig，每条边至少在某个子图中出现一次
2. 对每个查询子图stwig，在每一个图引擎节点上，查询满足每个树根节点标签条件的根节点、子节点标签条件的子节点，并将这些根节点和子节点组合成查询子图的候选结果
3. 对于每个查询子图，将每个图引擎节点上的候选查询结果集求并集
4. 对每个图引擎节点，按照查询子图之间的连接条件，将该引擎节点获得的各个查询子图的候选结果进行连接条件的join，形成该图引擎节点获得的全局查询图的候选结果集
5. 将每个图引擎节点的全局查询图的候选结果集求并集，得到最终的全局查询结果集合

3.

- (1)
  - V={N1,N2,N3,N4}
  - L={人，出演，电影}
  - E={E1,E2,E3,E4}
  - A={姓名，性别，角色，来源，名称}
  - N1.L=N3.L=N4.L={人}，N2.L={电影}
  - N1.A=N3.A=N4.A={姓名，性别}，N2.A={名称}，N1[性别]=男，N1[姓名]=张三，N2[名称]=饼侠，N3[姓名]=李四，N3[性别]=男，N4[姓名]=赵某，N4[性别]=女
  - E1=(N1,N2),E2=(N1,N2),E3=(N3,N2),E4=(N4,N2)
  - E1.L=E3.L=E4.L={出演}，E2.L={导演}
  - E1.A=E3.A=E4.A={角色，来源}，E1[角色]=男一号，E1[来源]=百度百科，E3[角色]=男二号，E3[来源]=百度百科，E4[角色]=女一号，E4[来源]=搜狐
- (2)

```
MATCH(S) WHERE id(S)='N1'
MATCH(E) WHERE id(E)='N3'
MATCH P=shortestPath((S)-[*..10]->(E))
RETURN P
```

- (3)

```
MATCH(N1 :人{姓名 :'张三'})-[:出演]->(N2 :电影),
         (N1)-[:导演]->(N2)<-[:出演]-(N3 :人)
WHERE N1<>N3
RETURN N3.姓名,N2.名称
```

4.

- (1)

```
void Compute(MessageIterator* msgs){
for(;!msg->Done();msgs->Next())
        doSomething()
*MultableVertexValue()=...
SendMessageToAllNeighbors(...);
}
```

- (2)

```
Message gather(Vertex u,Edge uv,Vertex v)
{
        Message msg =从v发给u的来自v的i跳邻居ID消息;
        return msg;
}
Message sum(Message msg)
{
        return left+right;//汇总收到的邻居节点ID消息
}
void apply(Vertex u,Message sum)
{
        u.Value=sum;//u节点更新自己即将发出的消息内容为收到的邻居节点ID消息及其生命值的列表
}
void scatter(Vertex u,Edge uv,Vertex v)
{
        uv.value保存不变，消息从u发给v;
        发给邻居点v消息中每个邻居的消息的生命值减一;
}
Job()
{
        初始化所有节点消息生命值为2;
        执行第一轮统计二跳邻居数的gas计算任务;
        执行第二轮统计二跳邻居数的gas计算任务;
        return 每个节点收集到的生命值为0的消息的列表;
}
```