基于分解法模板的逻辑问题求解

第一章 基于图的建模

1.1 分解法模型表示

将分解法模型定义为一个带属性的有向无环图$\mathrm{G}=(V,E,A)$。

结点：$\mathrm{V}=\{v_{\mathrm{answer}},v_{\mathrm{obj}},v_{\mathrm{rule}}\}\cup \mathrm{Options}\cup \mathrm{Propositions}\cup Rules$。其中$v_{answer}$表示问题的根结点，$v_{obj}$表示求解目标的根结点，$v_{rule}$表示约束规则的根节点，$\mathrm{Options}=\{op_1,op_2,\cdots,op_m\}$表示求解目标结点，$\mathrm{Propositions}=\{p_1,p_2,\cdots,p_n\}$为命题结点,$Rules=\{r_1,r_2,\cdots,r_n\}$表示具体的规则结点。

边：$\mathrm{E}=\{(v_i,v_j)\}$，其中$v_i,v_j\in V$为关系双方，v_i为父结点，v_j为子结点。

属性：

$\mathrm{A}=\{least_\mathrm{accepted_num},most_accepted_num,precondition,constrain\}$为每个结点所包含的属性。$least_accepted_num$表示该结点的子结点被接受的最小个数，；$most_accepted_num$表示该结点的子结点被接受的最大个数。precondition表示该结点所遵守的约束；$constrain$表示该结点的子结点所遵守的共同约束。

在分解法模型中:

v_{answer}为模型的根结点，含义为模型的最终求解目标为使得问题有解，这一目标可以进一步分解为使问题选项的求解目标和问题的规则同时成立。其子结点为v_{obj}和v_{rule}，表示为边(v_{answer},v_{obj})和(v_{answer},v_{rule})。属性$least_\mathrm{accepted_num}=most_accepted_num=2$，表示其子结点为$v_{\mathrm{obj}}$和$v_{\mathrm{rule}}$均被接受。

v_{obj}为求解目标的根结点，含义为题目的求解目标必须被满足，可以进一步分解为题目中的各选项或各具体求解目标同时满足的最小和最大个数。其子结点为Options，表示为边(v_{obj},op_i)，其中$op_i\in Options$。其属性$least_\mathrm{accepted_num}$、$most_accepted_num$的值分别表示逻辑题目中选项成立的最小和最大个数。

v_{rule}为题目中规则的根结点，含义为题目中的限定规则必须被遵守，可以进一步分解为题目中的各命题同时成立的最小和最大个数。其子结点为$Propositions$，表示为边(v_{rule},p_i)，其中$p_i\in Propositions$。其属性$least_\mathrm{accepted_num}$、$most_accepted_num$的值分别表示逻辑题目中命题成立的最小和最大个数。

$\mathrm{Options}=\{op_1,op_2,\cdots,op_m\}$为题目中具体求解目标的结点，含义为题目中可能成立的具体求解目标，可以分解为该具体求解目标的具体规则。其子结点为$\{r_j|r_j\in Rules,\exists op_i\in Options\ 使得(op_i,r_j)\in E\}$。其属性$least_\mathrm{accepted_num}$、$most_accepted_num$的值分别表示该具体求解目标下各规则成立的最小和最大个数，若该具体求解目标只有一条规则，则此属性可缺省；属性constrain的内容表示该具体求解目标下的具体规则所遵守的共同约束，若无需传递共同约束，则此属性可缺省。

$\mathrm{Propositions}=\{p_1,p_2,\cdots,p_n\}$为题目中的命题结点，含义为题目中可能成立的命题，可以分解为该命题的具体规则。其子结点为$\{r_j|r_j\in Rules,\exists p_i\ 使得(p_i,r_j)\in E\}$。其属性

$least$_accepted_num、$most_accepted_num$的值分别表示该命题下各规则成立的最小和最大个数，若该命题只有一条规则，则此属性可缺省；属性constrain的内容表示该命题下的具体规则所遵守的共同约束，若无需传递共同约束，则此属性可缺省。

$Rules = \{r_1, r_2, \cdots, r_n\}$为具体的规则结点，含义为具体的逻辑规则，可以进一步分解为其他的逻辑规则。其可以作为p_i、$\mathrm{op_i}$、r_i的子结点，表示为边(p_i, r_j)、(op_i, r_j)、(r_i, r_j)，其中$p_i \in Prepositions$，$op_i \in Options$，$r_i, \mathrm{r_j} \in Rules$。规则$r_i$的子结点为$\{r_j | \exists r_i, r_j \in Rules$使得$(r_i, r_j) \in E\}$。其属性$least$_accepted_num、$most_accepted_num$的值分别表示该规则下各子规则成立的最小和最大个数，若该规则只可以分解为一条子规则，则此属性可缺省；属性$precondition$的内容表示该所需遵守的约束，若无需遵守约束，则此属性可缺省；属性constrain的内容表示该规则下的子规则所遵守的共同约束，若无需传递共同约束，则此属性可缺省。

分解法模型的架构如图 1 所示。

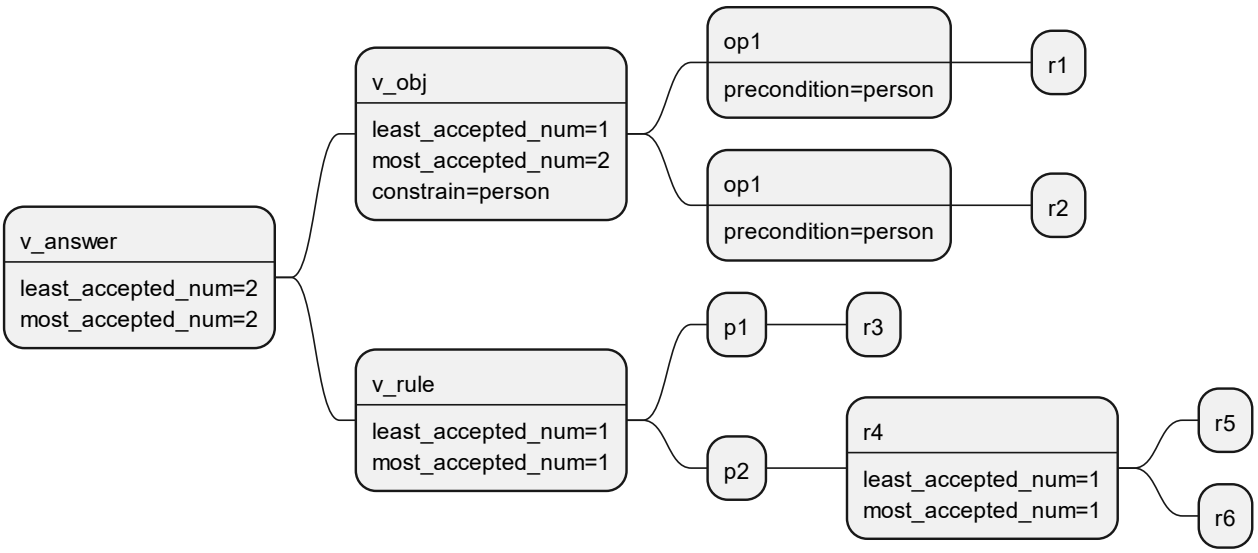

图 1 分解法模型架构

第二章 基于 ASP 的求解

2.1 分解法模型到 ASP 实现的转换

基于分解法模型可以将逻辑问题转化为 ASP（answer set programming）代码进行求解，代码结构分为 5 部分，分别为：“事实”、“常识”、“框架代码”、“求解目标”、“命题”。

1. 事实代码

“事实”代码由题目中确定存在的信息得到，在 ASP 中可以用事实型陈述来表示，例如：

person(Alice).

2. 常识代码

“常识”代码可以由相应场景的框架提供，也可以手动进行定义，通常为题目中未明确说明的一般性知识，例如：

1{eat(P, F, D):food(F)}1 :- person(P), day(D).

3. 框架代码

“框架代码”为分解法固定包含的代码，在分解法模型中对应为结点v_{answer}、v_{obj}和v_{rule}和关系(v_{answer}, v_{obj})和(v_{answer}, v_{rule})。其中v_{answer}的属性$least_\text{accepted_num} = most_accepted_num = 2$。$v_{answer}$、$v_{\text{obj}}$和$v_{\text{rule}}$分别可以用谓词“answer”“objective”“rule”表示。

关系$\{(v_i, v_j)|v_i, v_j \in V, (v_i, v_j) \in E\}$可以将父结点的对应代码置于语句体部，子节点的对应代码置于代码头部表示。代码可以表示为：

answer.

2{objective；rule}2 :- answer.

4. 求解目标代码

“求解目标”代码为题目中各个选项的求解目标以及选项的成立情况对 ASP 代码的映射。在分解法模型中为结点v_{obj}、Options、$\{r_j|r_j \in Rules, \exists op_i \in Options\ 使得 (op_i, r_j) \in E\}$和关系$\{(v_{obj}, op_i)|op_i \in Options\}$、$\{(op_i, r_j)|op_i \in Options, r_j \in Rules, (op_i, r_j) \in E\}$、$\{(r_i, r_j)|r_i, r_j \in Rules, (r_i, r_j) \in E\}$到 ASP 代码的映射。

$\{op_i|op_i \in Options\}$可以用谓词“option(X)”表示，其中 X==i。

v_{obj}的属性$least_\text{accepted_num}$、$most_accepted_num$的值分别表示逻辑题目中选项成立的最小和最大个数，在代码中表示为 MINO 和 MAXO。

因此关系$\{(v_{obj}, op_i)|op_i \in Options\}$的代码可以表示为：

MINO{option(1..ON)}MAXO :- objective.

$\{r_i|r_i \in Rules\}$可以用基于知识的具体规则表示。

$\{op_i|op_i \in Options\}$的属性$least$_accepted_num、$most_accepted_num$的值该具体求解目标下各规则成立的最小和最大个数，若该具体求解目标只有一条规则，则此属性可缺省，可以通过分别向头部的选择约束添加下区间和上区间实现；属性constrain的内容表示该具体求解目标下的具体规则所遵守的共同约束，若无需传递共同约束，则此属性可缺省，可以通过为体部谓词增加变量实现。

关系$\{(op_i,\ r_j)|op_i \in Options, r_j \in Rules, (op_i r_j) \in E\}$的代码可以将父结点的对应代码置于语句体部，子节点的对应代码置于代码头部表示。代码可以表示为：

2{eat(P, ham, yes); eat(P, pork, tod)}2 :- answer(P).

$\{r_i|r_i \in Rules\}$的属性$least$_accepted_num、$most_accepted_num$的值该规则下子规则成立的最小和最大个数，若该规则只有一条规则，则此属性可缺省，可以通过分别向头部的选择约束添加下区间和上区间实现；属性constrain的内容表示该规则下子规则所遵守的共同约束，若无需传递共同约束，则此属性可缺省，可以通过为体部谓词增加变量实现。

关系$\{(r_i, r_j)|r_i, r_j \in Rules, (r_i, r_j) \in E\}$的代码可以将父结点的对应代码置于语句体部，子节点的对应代码置于代码头部表示。代码可以表示为：

1{p(21;22)}1:- p(2).

2{join(w;l)}2 :- p(21).

5. 命题代码

“命题”代码为题目中各个命题的成立情况对 ASP 代码的映射。在分解法模型中为结点v_{rule}、$Propositons$、$\{r_j|r_j \in Rules, \exists p_i \in Propositions\ 使得 (p_i, r_j) \in E\}$和关系$\{(v_{rule}, p_i)|p_i \in Propositions\}$、$\{(p_i, r_j)|p_i \in Propositions, r_j \in Rules, (p_i, r_j) \in E\}$、$\{(r_i, r_j)|r_i, r_j \in Rules, (r_i, r_j) \in E\}$到 ASP 代码的映射。

$\{p_i|p_i \in Propositions\}$可以用谓词“p(X)”表示，其中 X==i。

v_{rule}的属性$least$_accepted_num、$most_accepted_num$的值分别表示逻辑题目中命题成立的最小和最大个数，在代码中表示为 MINP 和 MAXP。

因此关系$\{(v_{rule}, p_i)|p_i \in Propositions\}$的代码可以表示为：

MINP{propsition(1..PN)}MAXP :- rule.

$\{r_i|r_i \in Rules\}$可以用基于知识的具体规则表示。

$\{p_i|p_i \in Propositions\}$的属性$least$_accepted_num、$most_accepted_num$的值该命题下各规则成立的最小和最大个数，若该命题只有一条规则，则此属性可缺省，可以通过分别向头部的选择约束添加下区间和上区间实现；属性constrain的内容表示该命题下的具体规则所遵守的共同约束，若无需传递共同约束，则此属性可缺省，可以通过为体部谓词增加变量实现。

关系$\{(p_i, r_j)|p_i \in Propositions, r_j \in Rules, (p_i, r_j) \in E\}$的代码可以将父结点的对应代码置于语句体部，子节点的对应代码置于代码头部表示。代码可以表示为：

1{p(11,D);p(12,D)}1 :-p(1,D).

关系$\{(r_i, r_j)|r_i, r_j \in Rules, (r_i, r_j) \in E\}$的代码与求解目标代码部分中的规则关系映射方法一致。

2.2 基于分解法的 ASP 模板

根据 2.1 中的映射规则，转换的 ASP 程序的内容如下所示：

```
fact.
common_sense.

answer.

2{objective；rule}2 :- answer.

MINO{option(1..ON)}MAXO :- objective.
2{eat(a, X); eat(b, X)}2:- option(1, X).

MINP{p(1..PN)}MAXP :- rule.
2{eat(a, X); eat(b, X)}2:- p(1, X).

#show option/1.
```

第三章 案例应用

3.1 应用结果

在附录中，列举了应用分解法解决实际逻辑题目的案例。其中，问题：火腿还是猪排、问题：谁参加预选赛和问题：谁要参加活动这三个题目能够按照模式直接映射到可运行正确结果的 ASP 代码。但是问题：身份匹配和问题：谁射中了鹿这两个问题，模型则无法直接适用。

1. 谁射中了鹿

在谁射中了鹿的问题中，题目的规则约束是五个命题中的两个命题成立，若以此分解，则会生成如““-p(X)”定义缺失时”部分所示的结果。即按原有分解方法只能保证有两个命题成立，而无法保证仅有两个命题成立。

因此代码中加入了类似定义法的实现，即将规则约束分解为两个命题成立，三个命题不成立。如求解结果所示。

2. 身份匹配

身份匹配题目的求解目标是比较不同职业的人之间的籍贯、年龄关系，而这些职业对应的具体的人是未知的。

若按照分解法模型，求解目标代码应如下所示：

```
0{option(1);option(2);option(3);option(4)}4 :- options.
1{percondition(X,P1,P2):person(P1),person(P2)}1 :- option(X).

3{elder(P1, P2); identity_match(P1, writer); identity_match(P2, teacher)}3 :- percondition(1,P1,P2).
```

但解得的结果有类似问题，只能确保命题为真时，命题规则成立，而不能确保命题不为真时，命题规则不成立。如分解法表示求解目标时的错误答案中所示。

若使用类似于定义法的形式进行修改，如下所示：

```
-option(1) :- not option(1).
1{-percondition(1,P1,P2):person(P1),person(P2)}1 :- -option(1).

3{elder(P1, P2); identity_match(P1, writer); identity_match(P2, teacher)}3 :- percondition(1,P1,P2).
0{elder(P1, P2); identity_match(P1, writer); identity_match(P2, teacher)}2 :- -percondition(1,P1,P2).
```

则会输出过多不必要的答案集。因为对于每一个-percondition 都会有 $\sum_{n=0}^{2} C_3^n$种成立的组合。四个选项共会生成$(\sum_{n=0}^{2} C_3^n)^4$种答案集组合。

定义选项不成立条件：

3{elder(P1, P2); identity_match(P1, writer); identity_match(P2, teacher)}3 :- percondition(1,P1,P2).

3{not elder(P1, P2); identity_match(P1, writer); identity_match(P2, teacher)}3 :- -percondition(1,P1,P2).

3{same_hometown(P1, P2); identity_match(P1, doctor); identity_match(P2, lawyer)}3 :- percondition(2,P1,P2).

3{not same_hometown(P1, P2); identity_match(P1, doctor); identity_match(P2, lawyer)}3 :- -percondition(2,P1,P2).

3{elder(P1, P2); identity_match(P1, doctor); identity_match(P2, writer)}3 :- percondition(3,P1,P2).

3{not elder(P1, P2); identity_match(P1, doctor); identity_match(P2, writer)}3 :- -percondition(3,P1,P2).

3{different_hometown(P1, P2); identity_match(P1, teacher); identity_match(P2, lawyer)}3 :- percondition(4,P1,P2).

3{not different_hometown(P1, P2); identity_match(P1, teacher); identity_match(P2, lawyer)}3 :- -percondition(4,P1,P2).

此时，答案集：

Answer: 1

option(1) option(3) percondition(1,a,d) percondition(3,b,a) -option(2) -option(4) different_hometown(b,c) different_hometown(c,b) unknown_hometown(b,a) unknown_hometown(d,a) unknown_hometown(a,b) unknown_hometown(d,c) unknown_hometown(a,d) unknown_hometown(c,d)

Answer: 2

option(1) option(3) option(4) percondition(1,a,d) percondition(3,b,a) percondition(4,d,c) -option(2) different_hometown(c,d) different_hometown(d,c) different_hometown(b,c) different_hometown(c,b) unknown_hometown(b,a) unknown_hometown(d,a) unknown_hometown(a,b) unknown_hometown(a,d)

Answer: 3

option(1) option(3) percondition(1,a,c) percondition(3,b,a) -option(2) -option(4) different_hometown(b,d) different_hometown(d,b) unknown_hometown(d,a) unknown_hometown(d,c) unknown_hometown(a,d) unknown_hometown(c,d)

Answer: 4

option(1) option(3) option(4) percondition(1,a,c) percondition(3,b,a) percondition(4,c,d) -option(2) different_hometown(c,d) different_hometown(d,c) different_hometown(b,d) different_hometown(d,b) unknown_hometown(d,a) unknown_hometown(a,d)

SATISFIABLE

cautious 答案集：

Solving...

Answer: 1

option(3) percondition(3,b,a) option(1) percondition(1,a,d) -option(2) -option(4) different_hometown(b,c) different_hometown(c,b) unknown_hometown(b,a) unknown_hometown(d,a) unknown_hometown(a,b) unknown_hometown(d,c) unknown_hometown(a,d) unknown_hometown(c,d)

Consequences: [2;14]

Answer: 2

option(3) percondition(3,b,a) option(1) percondition(1,a,d) -option(2) different_hometown(b,c) different_hometown(c,b) unknown_hometown(b,a) unknown_hometown(d,a) unknown_hometown(a,b) unknown_hometown(a,d)

Consequences: [2;11]

Answer: 3

option(1) option(3) percondition(3,b,a) -option(2) unknown_hometown(d,a) unknown_hometown(a,d)

Consequences: [6;6]

SATISFIABLE

此外，还可使用使用类似于约束满足方法来实现，此时能够输出正确结果，如分解法解答所示。

第四章 总结与展望

4.1 总结

基于图定义了一个用于解决逻辑题目的分解法模型，并基于 ASP 进行了实现。

4.2 不足

初始的分解法模型中，选项与求解目标之间为充分非必要条件，使得选项不成立时无法使求解目标的情况不成立；同样命题不成立时无法使其规则不成立。对于模型的改进可以有以下方法：

1. **参照定义法的形式，分别对命题成立和命题不成立的情况进行分解。**

例如在谁射中了鹿的问题中，按原有分解方法只能保证有两个命题成立，而无法保证仅有两个命题成立。因此可以按照定义法的思想分解为两个命题成立，三个命题不成立。

2. **参照约束满足法的形式，将具体规则和命题的分解关系进行调换，规则和选项的分解关系进行调换，可以满足求解目标是选项的必要条件。**

例如在身份匹配问题中，求解目标为多选问题，意为选项成立时其规则被满足，选项不成立时规则也不成立。但原始分解法模型无法做到后者。

规则与命题/求解目标进行对调更符合其本质含义，即由规则来决定选项是否成立，而非选项决定规则是否成立。

3. **参照约束满足法的形式，无需将具体规则和命题的分解关系进行调换，在转化为 ASP 代码时将其头部、体部关系进行转换，可以满足求解目标是选项的必要条件。但如此可能会与“分解法”本身的含义不相符。**

4. **定义新的分解法模型，改变分解依据。**

此外，根据常理，模型输出的题解答案应唯一，但是当能够同时满足规则和约束条件的命题组合多于一种时，本模型的答案输出将等同于满足上述条件的命题组合的个数。

对于常识的定义也未包含在分解法模型内，而是使用类似于定义法的思想进行描述。

4.3 展望

可以通过阅读相关论文以更多了解分解法相关的理论研究。对于逻辑类型、问题类型不同的其他逻辑思维题目，本模型存在的潜在缺陷也待进一步验证。

附录

A. 火腿还是猪排

1. 问题

阿德里安、布福德和卡特三人去餐馆吃饭，他们每人要的不是火腿就是猪排。

（1） 如果阿德里安要的是火腿，那么布福德要的就是猪排。

（2） 阿德里安或卡特要的是火腿，但是不会两人都要火腿。

（3） 布福德和卡特不会两人都要猪排。

谁昨天要的是火腿，今天要的是猪排？

2. 分解法解答

```
person(a).
person(b).
person(c).
food(ham).
food(pork).
day(yes).
day(tod).
1{eat(P, F, D):food(F)}1 :- person(P), day(D).

answer.
2{goal; rule}2 :- answer.

1{answer(P) : person(P)}3 :- goal.
2{eat(P, ham, yes); eat(P, pork, tod)}2 :- answer(P).

6{p(1, yes); p(2, yes);p(3, yes); p(1, tod);p(2, tod);p(3, tod)}6 :- rule.

1{p(11,D);p(12,D)}1                    :-p(1,D).
2{eat(a,ham,D);eat(b,pork,D)}2    :-p(11,D).
not eat(a,ham,D)                       :-p(12,D).
1{eat(a,ham,D);eat(c,ham,D)}1      :-p(2,D).
0{eat(b,pork,D);eat(c,pork,D)}1 :-p(3,D).

#show answer/1.
```

3. 求解结果

Answer: 1

answer(b) eat(b,pork,tod) eat(c,ham,yes) eat(c,ham,tod) eat(a,pork,tod) eat(b,ham,yes) eat(a,pork,yes)

SATISFIABLE

B. 谁要参加活动

1. 问题

某医院刘佳、郑毅、郭斌、丁晓、吴芳、施文 6 位医生拟报名参加“一心向党，健康为民”进社区义诊活动，已知下列情况为真：

（1）要么刘佳参加，要么郑毅参加；

（2）只有吴芳参加，刘佳才参加；

（3）如果郭斌和吴芳都参加，那么施文也会参加；

（4）或者丁晓不参加，或者郭斌参加；

（5）施文、丁晓至少有 1 人参加。

现施文确定无法参加，那么 6 位医生中最后参加义诊活动的是：

A.刘佳、郭斌、丁晓

B.郑毅、郭斌、丁晓

C.郑毅、丁晓、吴芳

D.刘佳、丁晓、吴芳

2. 分解法解答

```
person(l;z;g;d;w;s).
-join(X) :- not join(X), person(X).

answer.

2{options; rule}2 :- answer.

2{pre; ops}2 :- options.

0{join(s)}0 :- pre.

1{option(a;b;c;d)}1 :- ops.

3{join(l;g;d)}3 :- option(a).
3{join(z;g;d)}3 :- option(b).
3{join(z;d;w)}3 :- option(c).
3{join(l;d;w)}3 :- option(d).
```

```
5{p(1;2;3;4;5)}5 :- rule.

1{join(l;z)}1                    :- p(1).
1{p(21;22)}1                     :- p(2).
2{join(w;l)}2                    :- p(21).
not join(l)                      :- p(22).
1{p(31;32)}1                     :- p(3).
3{join(g;w;s)}3                  :- p(31).
0{join(g;w)}1                    :- p(32).
1{not join(d); join(g)}1 :- p(4).
1{join(s;d)}1                    :- p(5).

% #show join/1.
% #show -join/1.
#show option/1.
```

3. 求解结果

```
Answer: 1
option(b) -join(l) join(z) join(g) join(d) -join(w) -join(s)
SATISFIABLE
```

C. 身份匹配

1. 问题

甲、乙、丙、丁 4 人，一人是教师，一人是医生，一人是作家，一人是律师。现已知:
①甲的年龄比教师大；
②乙和律师的籍贯不同；
③丙与作家的籍贯相同；
④作家的年龄比乙小；
⑤甲与律师来自相同的城市；
⑥教师的籍贯与乙相同。
x1,x2,x3,x4 -年龄
y1,y2,y3,y4 -籍贯
根据以上的信息，以下说法不正确的是:
A.作家的年龄比教师大
B.医生与律师的籍贯相同
C.医生的年龄比作家大
D.律师与教师的籍贯不同

2. 分解法解答

```
person(a;b;c;d).
identity(teacher;doctor;writer;lawyer).


:- identity_match(a,teacher).
:- identity_match(b,lawyer).
:- identity_match(c,writer).
:- identity_match(b,writer).
:- identity_match(a,lawyer).
:- identity_match(b,teacher).


same_hometown(P1, P2)          :- person(P1), person(P2), -different_hometown(P1, P2), P1 != P2.
different_hometown(P1, P2)    :- person(P1), person(P2), -same_hometown(P1, P2), P1 != P2.
different_hometown(P1, P2)    :- person(P1), person(P2), different_hometown(P2, P1), P1 != P2.
-same_hometown(P1, P2)         :- person(P1), person(P2), different_hometown(P1, P2), P1 != P2.
-different_hometown(P1, P2) :- person(P1), person(P2), same_hometown(P1, P2), P1 != P2.
```

```
same_hometown(P1, P2) :- person(P1), person(P2), person(P3), same_hometown(P1, P3), same_hometown(P2, P3), P1 != P2.
same_hometown(P1, P2) :- person(P1), person(P2), same_hometown(P2, P1), P1 != P2.
unknown_hometown(P1, P2) :- person(P1), person(P2), not same_hometown(P1, P2), not different_hometown(P1, P2), P1 != P2.

elder(P1, P2) :- person(P1), person(P2), -smaller(P1, P2), P1 != P2.
smaller(P1, P2) :- person(P1), person(P2), -elder(P1, P2), P1 != P2.
-elder(P1, P2) :- person(P1), person(P2), smaller(P1, P2), P1 != P2.
-smaller(P1, P2) :- person(P1), person(P2), elder(P1, P2), P1 != P2.
elder(P1, P2) :- person(P1), person(P2), smaller(P2, P1), P1 != P2.
smaller(P1, P2) :- person(P1), person(P2), elder(P2, P1), P1 != P2.
elder(P1, P2) :- person(P1), person(P2), person(P3), elder(P1, P3), elder(P2, P3), P1 != P2.
smaller(P1, P2) :- person(P1), person(P2), person(P3), smaller(P1, P3), smaller(P2, P3), P1 != P2.
unknown_age(P1, P2) :- person(P1), person(P2), not elder(P1, P2), not smaller(P1, P2), P1 != P2.

1{identity_match(P, I) : identity(I)}1 :- person(P).
:- identity_match(P1, I), identity_match(P2, I), P1 != P2.

answer.

2{options; rule}2 :- answer.

% 0{option(1);option(2);option(3);option(4)}4 :- options.

answer(a) :- elder(P1, P2), identity_match(P1, writer), identity_match(P2, teacher).
answer(b) :- same_hometown(P1, P2), identity_match(P1, doctor), identity_match(P2, lawyer).
answer(c) :- elder(P1, P2), identity_match(P1, doctor), identity_match(P2, writer).
answer(d) :- different_hometown(P1, P2), identity_match(P1, lawyer), identity_match(P2, teacher).

5{situation(1..5)}5 :- rule.
```

```
1{elder(a, P) : identity_match(P, teacher)}1                    :- situation(1).
1{different_hometown(b, P) : identity_match(P, lawyer)}1 :- situation(2).
1{same_hometown(c, P) : identity_match(P, writer)}1          :- situation(3).
1{smaller(P, b) : identity_match(P, writer)}1                  :- situation(4).
1{same_hometown(b, P) : identity_match(P, teacher)}1        :- situation(5).

#show option/1.
```

3. 求解结果

```
Answer: 1
answer(c) answer(a)
Answer: 2
answer(c) answer(a)
SATISFIABLE
```

4. 约束满足法解答

```
% rule
elder(a, P)                    :- person(P), identity_match(P, teacher).
different_hometown(b, P)       :- person(P), identity_match(P, lawyer).
same_hometown(c, P)            :- person(P), identity_match(P, writer).
smaller(P, b)                  :- person(P), identity_match(P, writer).
same_hometown(b, P)            :- person(P), identity_match(P, teacher).

% options
answer(a) :- elder(P1, P2), identity_match(P1, writer), identity_match(P2, teacher).
answer(b) :- same_hometown(P1, P2), identity_match(P1, doctor), identity_match(P2, lawyer).
answer(c) :- elder(P1, P2), identity_match(P1, doctor), identity_match(P2, writer).
answer(d) :- different_hometown(P1, P2), identity_match(P1, lawyer), identity_match(P2, teacher).
```

5. 分解法表示求解目标时的错误答案

```
Answer: 1
percondition(1,a,d)
Answer: 2
```

Answer: 3
percondition(1,a,d) percondition(3,b,a)
Answer: 4
percondition(3,b,a)
Answer: 5
percondition(1,a,d) percondition(4,c,d)
Answer: 6
percondition(4,c,d)
Answer: 7
percondition(1,a,d) percondition(4,c,d) percondition(3,b,a)
Answer: 8
percondition(4,c,d) percondition(3,b,a)
Answer: 9
percondition(1,a,c)
Answer: 10
percondition(1,a,c) percondition(3,b,a)
Answer: 11
percondition(1,a,c) percondition(4,d,c)
Answer: 12
percondition(1,a,c) percondition(4,d,c) percondition(3,b,a)
Answer: 13

Answer: 14
percondition(3,b,a)
Answer: 15
percondition(4,d,c)
Answer: 16
percondition(4,d,c) percondition(3,b,a)

D. 谁射中了鹿

1. 问题

古代一位国王和他的张、王、李、赵、钱五位将军，一同出外打猎，各人的箭上都刻有自己的姓氏。打猎中，一只鹿中箭倒下，但不知是何人所射。

张说：“或者是我射中的，或者是李将军射中的。”

王说：“不是钱将军射中的。”

李说：“如果不是赵将军射中的，那么一定是王将军射中的。”

赵说：“既不是我射中的，也不是王将军射中的。”

钱说：“既不是李将军射中的，也不是张将军射中的。”

国王让人把射中鹿的箭拿来，看了看，说：“你们五位将军的猜测，只有两个人的话是真的。”

请根据国王的话，判定以下哪项是真的？

（A） 张将军射中此鹿。
（B） 王将军射中此鹿。
（C） 李将军射中此鹿。
（D） 赵将军射中此鹿。
（E） 钱将军射中此鹿。

2. 分解法解答

```
person(z).
person(w).
person(l).
person(zhao).
person(q).

answer.
```

```
2{options; rule}2 :- answer.

1{shoot(X):person(X)}1 :- options.

-p(X) :- not p(X), person(X).
2{p(z;w;l;zhao;q)}2 :- rule.

1{shoot(z); shoot(l)}1          :- p(z).
not shoot(q)                    :- p(w).
1{shoot(zhao); shoot(w)}1       :- p(l).
0{shoot(zhao); shoot(w)}0       :- p(zhao).
0{shoot(l); shoot(z)}0          :- p(q).

0{shoot(z); shoot(l)}0          :- -p(z).
shoot(q)                        :- -p(w).
0{shoot(zhao); shoot(w)}0       :- -p(l).
1{shoot(zhao); shoot(w)}1       :- -p(zhao).
1{shoot(l); shoot(z)}1          :- -p(q).

#show p/1.
#show shoot/1.
```

3. 求解结果

```
Answer: 1
p(zhao) p(q) shoot(q)
SATISFIABLE
```

4. 约束满足法解答

```
p(z) :- shoot(z), not shoot(l).
p(z) :- not shoot(z), shoot(l).
p(w) :- not shoot(q).
p(l) :- shoot(zhao).
p(l) :- not shoot(zhao), shoot(w).
p(zhao) :- not shoot(zhao), not shoot(w).
p(q) :- not shoot(l), not shoot(z).
```

5. “-p(X)”定义缺失时

Answer: 1

p(z) p(w) shoot(z)

Answer: 2

p(z) p(zhao) shoot(z)

Answer: 3

p(z) p(w) shoot(l)

Answer: 4

p(z) p(zhao) shoot(l)

Answer: 5

p(w) p(q) shoot(w)

Answer: 6

p(w) p(l) shoot(w)

Answer: 7

p(w) p(q) shoot(zhao)

Answer: 8

p(w) p(l) shoot(zhao)

Answer: 9

p(l) p(q) shoot(w)

Answer: 10

p(l) p(q) shoot(zhao)

Answer: 11

p(zhao) p(q) shoot(q)

Answer: 12

p(w) p(zhao) shoot(z)

Answer: 13

p(w) p(zhao) shoot(l)

E. 谁参加预选赛

1. 问题

甲、乙、丙、丁 4 人参加预选赛。对于预选赛结果,几位教练预测如下:
(1)如果甲、乙均未通过，则丙通过
(2)如果乙、丙至少有 1 人通过，则丁也通过
(3)如果甲、乙至少有 1 人通过，则丙也通过，但是丁不通过。
根据几位教练的预测，可以推出:
A．丙和丁通过
B．甲和丁通过
C． 甲和乙通过
D．乙和丙通过

2. 分解法解答

```
person(a;b;c;d).

answer.

2{options; rule}2 :- answer.

1{option(a;b;c;d)}1 :- options.

2{join(c;d)}2 :- option(a).
2{join(d;a)}2 :- option(b).
2{join(a;b)}2 :- option(c).
2{join(c;b)}2 :- option(d).

3{p(1;2;3)}3 :- rule.

1{p(11;12)}1 :- p(1).
3{not join(a);not join(b);join(c)}3 :- p(11).
0{not join(a);not join(b)}1 :- p(12).

1{p(21;22)}1 :- p(2).
2{borc; join(d)}2:- p(21).
1{join(b;c)}2 :- borc.
```

```
0{join(b;c)}0 :- p(22).

1{p(31;32)}1 :- p(3).
3{aorb; join(c); not join(d)}3:- p(31).
1{join(a;b)}2 :- aorb.
0{join(a;b)}0 :- p(32).

#show join/1.
#show option/1.
```

3. 求解结果

```
Answer: 1
option(a) join(c) join(d)
SATISFIABLE
```