Part1

2.

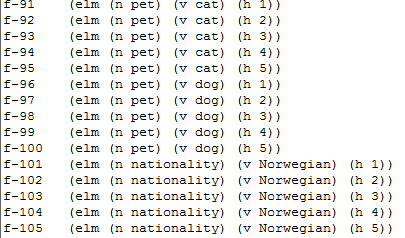
解题思路：

1. 一开始本想从零开始生成一个如下图一样表格

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| nationality | English’ | Swedish | Dane | German | Norwegian |
| drink |  |  |  |  |  |
| cigarette |  |  |  |  |  |
| pet |  |  |  |  |  |
| color |  |  |  |  |  |
| location |  |  |  |  |  |

但是发现，从零开始并不能全部把这个表格填写出来，有很多依赖关系是基于不同对的属性值之间的，比如“英国人住在红色房子”和“喝牛奶的人住在中间”分别使用了两对完全不同的属性，并不能链式地将表格填完整，所以只能更换思路。

1. 第二个思路就是想做减法，将5\*5\*5\*5\*5\*5=15625中情况全部列出来，然后通过题目中的限制条件做减法，比如将”英国人“”和“红色“”不在一起的元组删掉，这就可以使用大部分的条件；然后第5、11、12、15、16这些条件作为提取的正确元组，比如绿色的房子在白色的右侧，但是虽然看样子可行，由于产生了大量的元组，添加事实的时候非常的慢，最后似乎内存占用满了，这个不是一个很好的主意。
2. 还是依照上面的思路，我们换一种表达方式，既然全部排列组合不可行，那么只将属性和对应的房子的位置进行排列组合（如下图），然后删掉不合理的，再提取满足剩余条件的是不是就行了？



然后，发现只是部分的排列组合，并不能做到根据“英国人住在红色房子”条件删掉不满足的，因为红色和英国人对应的数字需要一样，那么他们同时为12345都可以，所以只能进行最后的提取满足条件的工作了。不仅如此，在（一）中分析了，他们之间隐含的关系非常复杂，与其让编程的人来分析如何划分规则，不如整个规则一股脑全部丢出去，让这个推理机自己来做。我想这也是符合这个clips设计的初衷的：推理机就是来帮助平凡的大众进行推理得到不容易推理出来的结果。

所以代码实现中，最主要的一个规则就是将题目中的条件全部写于RHS，最后生成只有5条满足全部条件的新事实。那么问题又来了，该如何写这一个复杂的规则呢？

在写RHS时，我发现了一个非常重要却容易忽略的细节，那就是：同样一种属性的不同值对应的数字（房子位置）应该保证不同，同样一条属性同一个值再次出现对应的数字应该保持不变。比如说，“绿色房子在白色的相邻左侧”和“住在绿色房子里面的人喝咖啡”的规则的书写就需要注意，两个绿色对应的数字变量应该保持相同，那么RHS就能做到操作的是同一条事实，而非两个匹配的不同事实。

**(elm (n color) (v green) (h ?c2&~?c1&~?c5))**

**(elm (n color) (v white) (h ?c3&~?c1&~?c5&~?c2))**

**(test(eq ?c2 (- ?c3 1) ))**

**(elm (n color) (v green) (h ?c2))**

**(elm (n drink) (v coffee) (h ?d2&?c2&~?d1))**

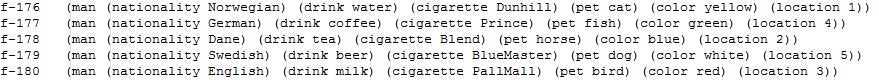
所以，这条规则的右边非常的复杂，要保证不仅每个题中条件都正确书写，还要注意条件之间应保持同一性，即操作的同一条的事实。

最后，生成自定义的模板下的事实

**(assert**

**(man(nationality ?e5)(cigarette ?e4)(pet ?e2)(color ?e3)(drink ?e1)(location ?num))))**

最后是生成了5条符合规则的事实，如下图：



可见，题目的答案是fish是德国人养的！