鸡尾酒专家系统——Lab3 文档

陈昕婕 15302010057 软件工程

一. 引言

本次 Lab 我选择 swi-prolog 语言，基于 prolog 的回溯推理匹配算法，实现了一个鸡尾酒推荐的专家系统。这个系统在知识库内在 fact 的基础上，根据用户的不同需求（鸡尾酒物理特性，类别等），为用户挑选一款适合的鸡尾酒或者为用户定制一款全新的鸡尾酒。程序的入口是 intro，所以打开 prolog 命令行，输入 intro.即可开始运行此专家系统。

二. 知识库 Fact 介绍

```
%Fact_for_materials
base_liquor(vodka).
base_liquor(gin).
base_liquor(rum).
base_liquor(tequila).
base_liquor(whiskey).
base_liquor(brandy).

juice(coconat_juice).
juice(orange_juice).
juice(lemonade).
juice(lime_juice).
juice(tomato_juice).

condiment(milk).
condiment(soda_water).
condiment(water).
condiment(cola).
condiment(coffee).
condiment(mint_leaf).
condiment(X) :- juice(X).

liqueur(kahlua).
liqueur(baileys_cream).
liqueur(creme_de_menthe).
liqueur(amarula).
liqueur(vermouth).

syrup(grenadine).
syrup(basic_syrup).
syrup(ginger).
```

```
ingredient(X) :- liqueur(X).
ingredient(X) :- syrup(X).

glass(bullet_cup).
glass(martini_cup).
glass(collins_glass).
glass(old_fashion_glass).
glass(brandy_snifter).

color(white).
color(yellow).
color(black).
color(red).
color(brown).
color(transparent).

category(sours).
category(collins).
category(old_fashion).
category(three_partners).

bitter(vermouth).
bitter(coffee).
bitter(kahlua).
sweet(creme_de_menthe).
sweet(amarula).
sweet(baileys_cream).
sweet(cola).
bitter(X) :- syrup(X).
sweet(X) :- juice(X).

result(tequila_sunrise).
result(b52).
```

base_liquor(X) ——表示 X 是鸡尾酒调酒中用到的基酒。

juice(X) ——表示 X 是鸡尾酒中调酒用到的果汁系列。

condiment(X) ——表示 X 是鸡尾酒中调酒用到的配料，其中包括果汁。

liqueur(X) ——利口酒，用于鸡尾酒调酒，比较好入口。

syrup(X) ——各种味道的糖浆。

ingredient(X) —— 添加料，指鸡尾酒调酒中会加到里面的，包括上面所有。

glass(X) ——鸡尾酒杯，本次 Lab 简单的分类为子弹（shot）杯，马天尼杯，柯林斯杯，古典杯，白兰地杯。

color(X) ——鸡尾酒成份的颜色。

category(X) —— 鸡尾酒的分类。参考鸡尾酒多种分类方式的其中一种标准，分为 4 类，sours, collins, old fashion 和 three partners。

bitter/sweet(X) ——鸡尾酒成份的味道，简单分为苦和甜两种。

result(X) ——存入知识库中的已存在鸡尾酒，有别于系统按照规则为用户生成的新款鸡尾酒。

知识库主要以鸡尾酒调酒用到的不同成份的特性为主，在 prolog 推理的过程中，这些成份根据它们各自的性质（fact）来适配 rules，最终获得符合要求的解法。

三. 基础规则介绍

```
%library
member(X, [X|_]).
member(X, [_|T]) :- member(X,T).
len([], 0).
len([_|T], N) :- len(T,X), N is X+1.
```

为了详细规则书写而写的一些方法，member 用于判断 X 是不是在 list T 里面，采用了递归判断。len 用户获取列表 T 的长度，同样采用了递归，算法会在[]时停止。

```
%basic_rule
contain_alcohol(T) :- member(X,T), base_liquor(X); member(Y, T), liqueur(Y).
contain_condiment(T) :- member(X, T), condiment(X),!.
base_liquor_num([], 0).
base_liquor_num([X|Z], Y) :- base_liquor_num(Z, N), base_liquor(X), Y is N+1.
base_liquor_num([X|Z], Y) :- base_liquor_num(Z, N), \+base_liquor(X), Y is N.
condiment_num([], 0).
condiment_num([X|Z], Y) :- condiment_num(Z, N), condiment(X), Y is N+1.
condiment_num([X|Z], Y) :- condiment_num(Z, N), \+condiment(X), Y is N.
legal([]).
legal([X|T]) :- ingredient(X), legal(T).
```

成份判断时用到的一些规则。contain_alcohol 判断配方 T 中含有酒精（基酒或者利口酒），
base_liquor_num 递归计算出成份中含有基酒的种类数。condiment_num 递归计算出成份中含有配料的种类数。legal 用于判断配方中的成份都属于 ingredient，即符合知识库可以食用的材料。

以上函数单个都没有什么具体用处，但在其他具体规则的判断中经常会用到，所以单列出来。

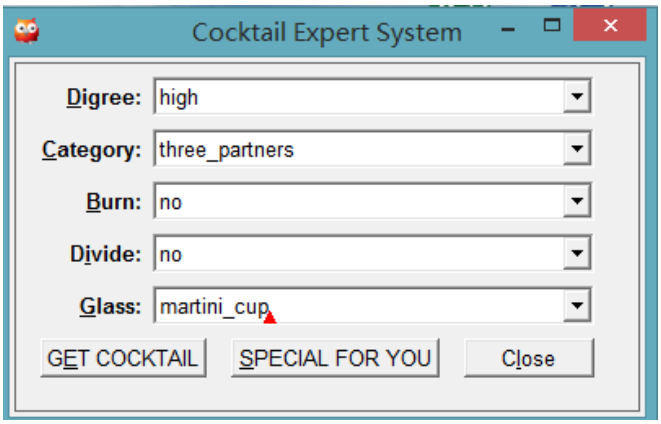
```
%formula
cocktail(tequila_sunrise, X) :- X = [grenadine, tequila, orange_juice].
cocktail(b52, X) :- X = [kahlua, vodka, baileys_cream].
cocktail(white_russian, X) :- X = [vodka, kahlua, milk].
cocktail(black_russian, X) :- X = [vodka, kahlua].
cocktail(long_island_iced_tea, X) :- X = [gin, vodka, tequila, rum, lemonade, cola
, basic_syrup, creme_de_menthe].
cocktail(mojito, X) :- X = [rum, soda_water, basic_syrup, mint_leaf, lime_juice].
cocktail(singapore_sling, X) :- X = [gin, brandy, lemonade, soda_water, base_liquo
r].
cocktail(bloody_mary, X) :- X = [vodka, tomato_juice].
cocktail(whiskey_rickey, X) :- X = [whiskey, soda_water, lime_juice].
cocktail(whiskey_sling, X) :- X = [lemonade, whiskey, water].
cocktail(martini, X) :- X = [gin].
```

Line: 124

储存系统已知的鸡尾酒。配方相同的即可认定为同一款酒，系统已存的酒拥有固定的学名以及图片，在获取结果的时候展示给用户。

以上就是基础规则，在详细判定的时候会用到。其余规则在第四点中功能介绍里详述。

四. 功能介绍



系统主要有以下功能：

1. 按照酒精度数来调制鸡尾酒。
2. 按照鸡尾酒的成份来调制鸡尾酒。
3. 按照鸡尾酒是否可燃（视觉效果）来调制鸡尾酒。
4. 按照鸡尾酒是否分层（视觉效果）来调制鸡尾酒。
5. 按照引用鸡尾酒采用的鸡尾酒杯种类来调制鸡尾酒。
6. 从系统知识库中匹配已存在的鸡尾酒。
7. 按照用户选择的前五点条件从配料生成系统不存在的鸡尾酒。
8. 用户标定对鸡尾酒喜不喜欢。

1. **酒精度数**

```
%degree
high_degree(X) :- shot(X).
high_degree(X) :- base_liquor_num(X, Y), Y > 2.
high_degree(X) :- len(X, 2), member(Z, X), base_liquor(Z).
low_degree(X) :- \+high_degree(X), condiment_num(X, Y), Y > 2.
normal_degree(X) :- \+high_degree(X), \+low_degree(X).
get_degree(X, high) :- high_degree(X).
get_degree(X, low) :- low_degree(X).
get_degree(X, medium) :- normal_degree(X).

%drink
shot(X) :- cocktail(b52, X).
shot(T) :- len(T, 1), base_liquor_num(T, 1).
long_drink(X) :- low_degree(X); normal_degree(X).
short_drink(X) :- high_degree(X), \+shot(X).
```

由于鸡尾酒成分复杂，想要完全的分出高低度数也是不可能的，并且度数的高低定义仅供参考因人而异。

度数高：是 shot，或者含有的基酒数量大于 2 种。（由于六种基酒都是 35-55 度的烈酒）。

度数低：度数不高，并且加入的非酒精配料大于 2 种。（非酒精材料比如果汁能够很好的稀释烈酒，所以配料多度数相对也低）。

度数中等：不满足度数高的规则也不满足度数低的规则。

其中在判定度数的时候也用到了鸡尾酒的属性（长短饮），这也有一套判定规则。
shot：纯基酒或者 B52。用子弹杯装，一般一口喝完，度数高。
长饮：度数相对没那么高，可以饮用 20 到 30 分钟，适合长时间引用。
短饮：度数相对高，短时间饮用完毕，20 分钟内喝完为佳。

2. 鸡尾酒种类

参考国外鸡尾酒网站的其中一种分类方式，分为 4 类：

```
%categories
divide_category(sours, X) :- member(Z, X), syrup(Z), member(Y, X), base_liquor(Y),
 member(A, X), juice(A).
divide_category(collins, X) :- member(Z, X), syrup(Z), member(lime_juice, X);membe
r(Z, X), syrup(Z), member(lemonade, X).
divide_category(old_fashion, X) :- member(Z, X), bitter(Z), member(Y, X), sweet(Y)
.
divide_category(three_partners, X) :- len(X, 3), member(Z, X), base_liquor(Z), mem
ber(A, X), liqueur(A).
```

Sours：Fresh squeezed juices are critical here, often paired with simple syrup (sugar dissolved in an equal volume of water) to sweeten.简单来说就是成份中还有糖浆以及一些果汁。一般使用摇晃来调酒，不过这不在本次专家系统设计的考虑范围内。

Collins(又名 Fizz)：Throw in a few lemons and limes, and you can whip up a Moscow mule, a rye and ginger, a Campari gin and tonic, a mojito, a Tom Collins or a paloma anytime.简单判定就是含有糖浆（ginger 之类的），还加入了柠檬汁或者青柠檬汁。

Old fashion(又名 lowball)：These old school short cocktails are simply sweetened liquor zipped up with something bitter. 比较古典的调酒方式，成份中含有苦的和甜的即可。

Three partners：These drinks are the most spirit-forward of the collection, a three-part combo of booze plus what’s known as a “modifier” (a lower-alcohol ingredient like vermouth) plus a bitter or a syrup.成份有三种，一种是烈的基酒，搭配利口酒（vermouth 比较多），再搭配甜的或者苦的材料。

根据定义和成份类别，成份口味即可得知属于哪一种种类。但是上面的定义都是比较宽泛的定义，可以区分绝大部分酒，但有可能出现个别酒同时满足两个类别的情况。

3. 鸡尾酒是否可分层

分层的判定主要用到了物理特性，不同酒成份的密度不同，一般来说含糖量大的密度大，容易沉在地下，基酒密度最小，浮在上方。但是不同密度不相容不代表就分层，还要判断不相容的两种液体的颜色不同，视觉效果才会分层。

用到的鸡尾酒密度规则：

```
%density
density_stronger_basic(kahlua, baileys_cream).
density_stronger_basic(X, Y) :- syrup(X), liqueur(Y).
density_stronger_basic(X, Y) :- liqueur(X), condiment(Y).
density_stronger_basic(X, Y) :- liqueur(X), base_liquor(Y).
density_stronger(X, Z) :- density_stronger_basic(X,Z); ingredient(Y), density_stro
nger_basic(X, Y), density_stronger(Y, Z).
divide(X) :- ingredient(Y), member(Y, X), ingredient(Z), member(Z, X), density_str
onger(Y, Z), not(color_equal(Y, Z)),!.
divide_output(X, can_divide) :- divide(X).
divide_output(X, cannot_divide) :- \+divide(X).
```

甘露咖啡比百利甜重（由于它们属于同一层，只能单列出来）。糖浆比利口酒密度大，利口酒比基酒和非酒精饮料密度大。非酒精饮料和基酒虽然密度不同，但是很接近，时间长或者用搅拌勺搅拌一下有可能混在一起，所以不判定他们的密度关系。其中判定密度的时候用到了根据规则的适配，自动匹配隔了一层密度关系的两种成分的关系（这就体现了 prolog 的优秀了）。

用到的鸡尾酒成份颜色判定：

```
%color
match_color(X, white) :- X = milk; X = coconat_juice; X = baileys_cream.
match_color(X, yellow) :- X = lemonade; X = orange_juice; X = ginger.
match_color(X, black) :- X = cola; X = coffee; X = kahlua.
match_color(X, red) :- X = tomato_juice; X = grenadine.
match_color(X, brown) :- X = amarula.
match_color(X, transparent) :- \+match_color(X, white), \+match_color(X, yellow),
                \+match_color(X, black), \+match_color(X, red), \+match_color(X
, brown).
color_equal(X, Y) :- color(Z), match_color(X, Z), match_color(Y, Z).
```

同样用到了变量匹配，如果一种颜色同时匹配了两种成份，则说明他们的颜色不一样。

4. 是否可燃（视觉效果）

```
%burn
burn(X) :- len(X, 1), base_liquor(Z), member(Z, X).
burn(X) :- divide(X), base_liquor(Y), member(Y, X), not(contain_condiment(X)), bas
e_liquor_num(X, 1).
burn_output(X, can_burn) :- burn(X).
burn_output(X, cannot_burn) :- \+burn(X).
```

烈酒可燃，所以判断是否可燃就两个条件，一是纯烈酒，成份只有 1 种并且是基酒，二是它分层，并且烈酒在最上面一层，这样烈酒也是纯的可燃。由于烈酒的密度最小，所以烈酒肯定在最上面一层，只要成份中不含有其他非酒精饮料（因为它们容易和烈酒混起来导致烈酒度数降低），即可初步判定可燃。

5. 酒杯种类

```
%glass
cup(bullet_cup, X) :- \+cocktail(martini, X), shot(X); \+cocktail(martini, X), len
(X, 1), X = [N|_], base_liquor(N).
cup(martini_cup, X) :- len(X, Y), Y < 5, cocktail(martini, X); len(X, Y), Y < 5, s
hort_drink(X).
cup(brandy_snifter, X) :- base_liquor_num(X, 1), member(brandy, X).
cup(collins_glass, X) :- long_drink(X), \+cup(brandy_snifter, X), \+cocktail(marti
ni, X).
cup(old_fashion_glass, X) :- short_drink(X),\+shot(X).
```

简单分成五类酒杯：

子弹杯（bullet cup）：用于饮用高浓度烈酒 shot 或者 B52 这种酒。

马天尼杯：由于马天尼杯容量较小，一般包含的液体成份低于 5 种。容量小适合喝短饮类的鸡尾酒或者马天尼类酒。

白兰地杯：由于白兰地的特性，一般要用手温了来品酒，所以杯子比较独特。所以适用于白兰地为基酒的酒。

柯林斯杯：杯子容量比较大，适合喝长饮，度数比较低的酒类。

古典杯：杯子容量比柯林斯杯小，适合喝短饮，度数偏高。

6. 标定 dislike

用户选择完条件后，系统会为用户推荐适合的酒，如果用户不喜欢推荐的酒，可以在 cocktail card 鸡尾酒名片页面点击下方的 dislike 按钮，系统则会把用户的选择加入知识库，在下次推荐时会避开用户不喜欢的酒。

由于选择是存在知识库里，所以用户只能 dislike 知识库已存有的 12 种鸡尾酒。

五. 用户交互 UI

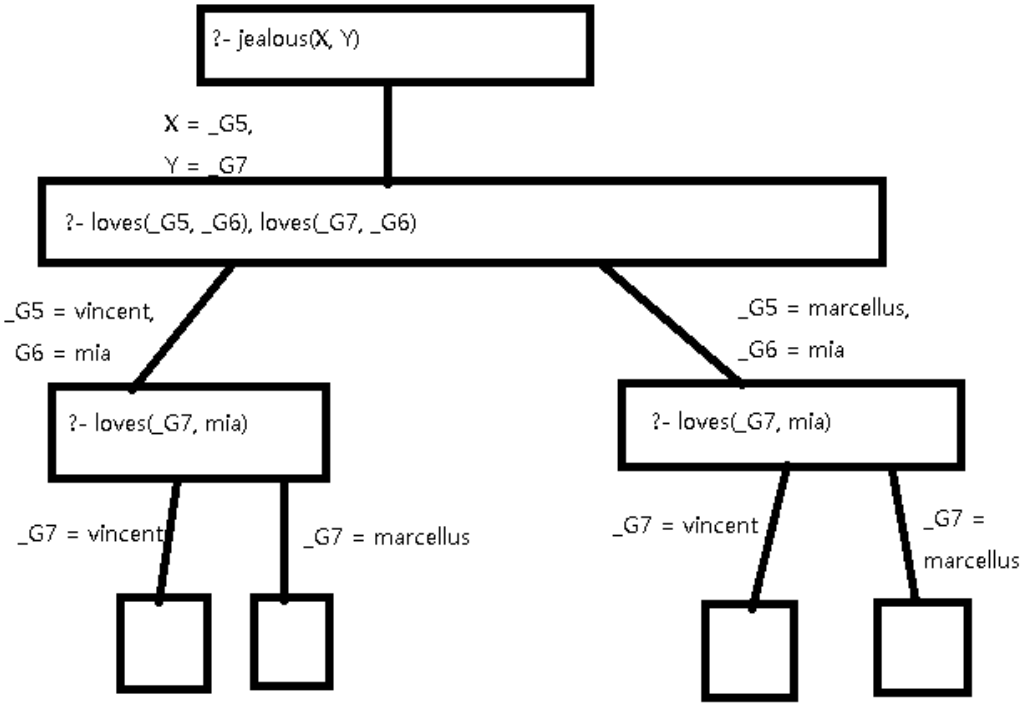
我采用了 Prolog 的自带 UI 插件 XPCE，加入下拉菜单等供用户选择。用户选择完后点击 GET COCKTAIL 则获取知识库已存在的鸡尾酒，点击 SPECIAL FOR YOU 会根据选择生成知识库没保存的全新鸡尾酒。点击 close 关掉窗口。

对于知识库已存在的鸡尾酒，如果用户不想下次再见到，可以点击 dislike 来选择结果。

六. 对 Prolog 的理解

Prolog 采用了逻辑推理自动回溯匹配的算法。和一般我们习惯用的编程语言不一样，他不是按顺序执行代码，而是对于大写字母表示的变量，系统自动去匹配。系统根据知识库中已经存有的信息，对于满足条件需要达成的规则 rules，用知识库的存储去匹配其中的变量，最终找出一种或者多种符合所有 rules 的变量的值，即为算法得出的最终结果。

Prolog 最强大的地方在于他的推理机制。



从图里大概可以看出，Prolog 每次遇到新的变量，都会发展分支去匹配它。所以在 Prolog 编程中，最独特的地方在于我们不用考虑它具体是怎么推出结果的，只需要去书写满足结果的规则。

Prolog 感觉在递归的时候最好用，只要写结束时的规则和递归时的规则，Prolog 的变量匹配会慢慢去回溯到最开始的地方，一步一步的找出结果。

我在 PROLOG 编程中遇到了两种最大的 bug:

1. 语句顺序

由于 prolog 是按照知识库的顺序来回溯的，所以语句的顺序非常重要。一不小心

就有可能不小心运行出死循环。尤其是在递归这种情况，终结语句要写并且要写在一般语句的前面。Prolog 遇到终结条件时就会适配终结语句停止算法。

2. 变量设置

为了比较 A 和 B，有时候可能会引入满足两方的第三个条件，有时候会陷入第三个条件的死循环，导致最终内存爆炸。书写的时候也要注意。

Prolog 比较符合我们思考方式的还有一个地方，就是它对于知识库中不存在的东西，他不会去猜测，会判定为 false。而且他比起 clips，算术语句等和人类正常的思维习惯一样，用的中缀遍历来书写。

七. 总结

通过这次 LAB，我掌握了 prolog 这门语言的语法，跳出以前编程语言的惯用思维方式，体会到了 prolog 回溯算法的神奇之处。