Input = {Customer(x).Has\_Height, Customer(x).Has\_Weight, Customer(x).Has\_Age, Customer(x).Gender, Illness[n], Target, HasActiveLevel(Customer(x), PhysicActiveLevel(act\_level\_input)}

Known = {Height, Weight, Gender, Age, PhysicalActiveLevel, ...}

(Fact, Rule)

Fact:

{

}

Function:

1. FindAvoidFood(): return

Example\_rule:

Note: Thỏa 1 luật thì quay lại vòng lặp luật

{ sk\_giả thiết | sk\_suy diễn } ;

Thỏa sk giả thiết mới xét sk\_suy diễn

[a, b] giải thích:

A =1 và B có thuộc Known => True or else: false

A = 2 -> B = [b1, b2, b3] : b1 thuộc Known, if: b2.value == b3 => True

[a,b] , [a,b] -> [] and []

A : Fact và A phải chưa thuộc known , B : Funct(biến) -> A.value => Append A vào trong Know

Nhãn có nhiều biến: CaloForEachMeal(3, biến);

Note: listavoidfood = (extract(avoidgroupfood))

Rule:

{

{[1, Height], [1,Weight] | [BMIvalue, funct:BMIvalueCalc(height,weight)]},

{[1, BMIvalue] | [BMIrange, funct:BMIrangeCalc(bmi\_value)]},

{[1, BMIrange] | [BMIlevel, funct:BMIlevelCalc(bmi\_range)]},

{[2, Gender, gender, “Nam”)], [1, Height], [1,Weight], [1, Age] | [BMRvalue, funct:BMRvalueOfMaleCalc(height, weight, age)]},

{[2, Gender, gender, “Nữ”], [1, Height], [1,Weight], [1, Age] | [BMRvalue, funct:BMRvalueOfFemaleCalc(height, weight, age)]}

{[1, PhysicalActiveLevel] | [PhysicalActiveValue, funct:PAvalueCalc(pa\_level)]},

{[1, BMRvalue], [1,PhysicalActiveValue], [1, Target] | [CaloPerDay, funct:CaloCalc(bmr\_value, pa\_value,target)]},

{[2, Target, target,“Giữ cân”],[1,CaloPerDay] | [CaloForEachMeal,funct:CaloToKeepWeightCalc(calo\_per\_day)]}

{[2, Target, target, “Tăng cân”],[1,CaloPerDay] | [CaloForEachMeal,funct:CaloToGainWeightCalc(calo\_per\_day)]}

{[2, Target, target, “Giảm cân”],[1,CaloPerDay] | [CaloForEachMeal,funct:CaloToLoseWeightCalc(calo\_per\_day)]}

{[1, ListIllness] | [ListAvoidFood, funct:FindAvoidFood(list\_illness)]},//Tách nhóm groupavoid

{[1, ListIllness] | [ListNeedFood, funct:FindNeedFood(list\_illness)]}//Tách nhóm groupneed

{[1, ListIllness] | [ListLimitFood, funct:FindLimitFood(list\_illness)]}//Tách nhóm grouplimit

{[1, ListAvoidFood], [1, ListAllFood] | [ListFood, funct:FilterFood(list\_illness, list\_food)]},

}

Lập thực đơn:

- Tính BMI

- Tính BMR

- Tính Chỉ số hoạt động

- Tính lượng Calo cần trong 1 ngày

- Tính lượng calo tùy đối tượng

- Tìm danh sách các thực phẩm, nhóm thực phẩm phụ thuộc từng tình trạng, nhu cầu, dị ứng (nếu có)

- Lọc ra các thực phẩm không dùng chung

- Đánh giá mức độ quan trọng của từng thực phẩm đối với nhu cầu cũng như tình trạng

- Tùy vào độ ưu tiên, chọn ra các danh sách thực đơn

- Thuwcj

- Tính định lượng từng thực phẩm trong mỗi thực đơn dựa vào tổng calo cần thiết.

- Xếp hạng cho các thực đơn

List of food (food\_name: score (1,0,-1)

Limit\_food\_and\_their\_calo(rau = 20-30 calo)

Món ăn vào menu:

Có: Calo - phương sai

List of food - nb

Món\_fix - calo đi cùng (example: rau - 50calo)

Cần: Buổi sáng: rule = random(list\_of\_rule(name\_food, portion\_food))

Food\_portion = calo/(

Buổi trưa:

Rule = {random(rau, 2), random(meat, seafood, egg\_products; 1-2), randome(fruits, yes\_no)}

Buổi tối:

Rule = {}