Templates

(deftemplate dish

(slot name (type STRING))

(slot cuisine (type SYMBOL))

(slot vegetarian (type SYMBOL) (default FALSE))

(slot sweet (type SYMBOL) (default FALSE))

(slot sour (type SYMBOL) (default FALSE))

(slot spiciness (type NUMBER))

(slot lowcal (type SYMBOL) (default FALSE))

(slot lowna (type SYMBOL) (default FALSE))

(slot lowfat (type SYMBOL) (default FALSE))

(slot highfiber (type SYMBOL) (default FALSE)))

**Explanation :**

Dish is a structure like in C, which contains attributes name, cuisine, vegetarian etc.

(deftemplate preference

(slot property (type SYMBOL))

(slot symbolValue (type SYMBOL))

(slot numberValue (type NUMBER)))

**Explanation :**

Preference is also a structure which contains attributes like property, symbolValue and numberValue. This is used to store customer preferences.

(deftemplate drink

(slot name (type STRING))

(slot category (type SYMBOL))

)

Facts

(deffacts drinks

(drink (name "Thumbsup") (category cool) )

(drink (name "Sprite") (category cool) )

(drink (name "GreenTea") (category hot) )

(drink (name "Maaza") (category cool) )

(drink (name "Coffee") (category hot) )

(drink (name "Tea") (category hot))

)

(deffacts dishes

(dish (name "Bak kut teh") (cuisine Chinese)(taste spicy) (level 2) )

(dish (name "Mee pok") (cuisine Chinese) (taste sour) (level 0)(lowfat TRUE)(lowcal TRUE))

(dish (name "Char Kway Teow") (cuisine Chinese)(vegetarian TRUE) (taste spicy) (level 2)(lowfat TRUE))

(dish (name "Chicken rice") (cuisine Chinese) (taste spicy)(level 3)(lowcal TRUE))

(dish (name "Ban mian") (cuisine Chinese) (vegetarian TRUE) (taste spicy) (level 1) (lowfat TRUE) )

(dish (name "You char kuey") (cuisine Chinese) (taste spicy) (level 2)(lowcal TRUE) )

(dish (name "Kaya toast ") (cuisine Chinese)(taste sweet) (level 2) )

(dish (name "Seafood hor fun") (cuisine Chinese) (taste spicy) (level 3) )

(dish (name "Chai tow Kway") (cuisine Chinese) (taste spicy)(level 1)(lowcal TRUE))

(dish (name "Vegetarian Manchuria") (cuisine Chinese) (vegetarian TRUE) (taste sweet)(level 0) (lowfat TRUE))

(dish (name "Noodles") (cuisine Chinese) (taste sour)(level 1) )

(dish (name "Tandoori chicken") (cuisine Indian) (taste spicy) (level 4)(lowcal TRUE))

(dish (name "Roti parata") (cuisine Indian) (vegetarian TRUE) (taste sour) (level 3) )

(dish (name "Chicken Butter Masala") (cuisine Indian) (taste spicy) (level 2)(lowcal TRUE))

(dish (name "Idlee Saambaar") (cuisine Indian) (vegetarian TRUE) (taste sweet) (level 2) (lowfat TRUE) )

(dish (name "Paneer Butter Masala ") (cuisine Indian) (vegetarian TRUE) (taste sweet)(level 1) (highfiber TRUE))

(dish (name "Garbanzo Tomato Pasta ") (cuisine Indian) (vegetarian TRUE) (taste sweet)(level 1) (highfiber TRUE))

(dish (name "Masala dosa") (cuisine Indian) (vegetarian TRUE) (taste spicy) (level 2) )

(dish (name "Chicken Biryani") (cuisine Indian) (taste spicy) (level 4)(lowcal TRUE))

(dish (name "Gulab Jamun") (cuisine Indian) (vegetarian TRUE) (taste sweet) (level 3) (lowfat TRUE))

(dish (name "Rasgulla") (cuisine Indian) (vegetarian TRUE) (taste sweet) (level 4)(lowcal TRUE))

(dish (name "Lemon Rice") (cuisine Indian) (vegetarian TRUE) (taste sour) (level 3) (lowfat TRUE))

(dish (name "Black pepper crab") (cuisine Seafood) (taste spicy) (level 2) (lowfat TRUE) )

(dish (name "Fried lobster omelette") (cuisine Seafood) (taste sweet) (level 0)(lowcal TRUE))

(dish (name "Crispy Manchuria") (cuisine Seafood) (vegetarian TRUE) (taste sour) (level 2) )

(dish (name "Fish head curry") (cuisine CrossCultural) (taste spicy) (level 4) (lowfat TRUE))

(dish (name "Laksa") (cuisine CrossCultural) (taste spicy) (level 3)(lowcal TRUE))

(dish (name "Ayam buah keluak") (cuisine CrossCultural) (taste sweet)(level 2))

(dish (name "Kueh pie tee") (cuisine CrossCultural) (taste sweet) (level 1) (lowfat TRUE) (lowcal TRUE) )

(dish (name "Tauhu goreng") (cuisine CrossCultural) (vegetarian TRUE) (taste sour) (level 2)(lowcal TRUE))

(dish (name "Kueh tutu") (cuisine CrossCultural) (vegetarian TRUE) (taste sweet) (level 0) (lowfat TRUE))

(dish (name "Chicken chop") (cuisine Western) (taste spicy) (level 1) (lowcal TRUE))

(dish (name "Pasta") (cuisine Western) (vegetarian TRUE) (taste sour) (level 0) (lowfat TRUE))

(dish (name "Sausage roll") (cuisine Western) (vegetarian TRUE) (taste sour) (level 2))

(dish (name "Japanese Ramen") (cuisine Japanese) (taste sour) (level 2)(lowfat TRUE))

(dish (name "Takoyaki") (cuisine Japanese) (taste spicy) (level 3)(lowcal TRUE))

(dish (name "Seaweed Roll") (cuisine Japanese) (vegetarian TRUE) (taste sour) (level 0)) )

**Explanation :**

These are facts that are stored in the knowledge base. These facts are the dishes available in the restaurant.

(deffacts start

(state start))

This is the fact to start the expert system.

Rules

(defrule start

?state <- (state start)

=>

(printout t crlf)

(printout t "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" crlf)

(printout t "Welcome to the world's most advanced food recommendation system [citation needed]" crlf)

(printout t "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" crlf)

(printout t crlf)

(retract ?state)

(assert (state question))))

**Explanation :**

This rule enables the expert system to ask questions to customers.

(defrule ask-cuisine

(declare (salience 85))

?state <- (state question)

(not (asked cuisine))

=>

(printout t "What cuisine would you prefer ?" crlf)

(printout t "(Chinese/Indian/Seafood/Western/Japanese/CrossCultural)" crlf)

(bind ?response (read))

(switch ?response

(case Chinese then (assert (preference (property cuisine) (symbolValue Chinese))))

(case Indian then (assert (preference (property cuisine) (symbolValue Indian))))

(case Seafood then (assert (preference (property cuisine) (symbolValue Seafood))))

(case Western then (assert (preference (property cuisine) (symbolValue Western))))

(case Japanese then (assert (preference (property cuisine) (symbolValue Japanese))))

(case CrossCultural then (assert (preference (property cuisine) (symbolValue CrossCultural))))

(default (printout t "Great, lemme search across all cuisines!" crlf)))

(assert (asked cuisine)))

Explanation :

This rule is used to select a particular cuisine and retracts other facts which do not belong to that cuisine.

(defrule ask-vegetarian

(declare (salience 75))

?state <- (state question)

(not (asked vegetarian))

=>

(printout t "Do you prefer vegetarian food? (yes/no/Either)" crlf)

(bind ?response (read))

(switch ?response

(case yes then (assert (preference (property vegetarian) (symbolValue TRUE))))

(case no then (assert (preference (property vegetarian) (symbolValue FALSE))))

(default (printout t "Great, so I'll search across vegetarian AND non-vegetarian dishes!" crlf)))

(assert (asked vegetarian)))

Explanation :

This rule is used to separate the vegetarians and non-vegetarians for further queries.

(defrule ask-taste

(declare (salience 65))

?state <- (state question)

(not (asked taste))

=>

(printout t "choose your taste preference -- sweet / spicy / sour " crlf)

(bind ?response (read))

(switch ?response

(case spicy then

(assert (preference (property taste) (symbolValue spicy) )))

(case sour then

(assert (preference (property taste) (symbolValue sour) )))

(case sweet then

(assert (preference (property taste) (symbolValue sweet) )))

(default

(printout t "Didn't catch that. I'm going to assume you're a 2." crlf)

(assert (preference (property spiciness) (numberValue 2)))))

(assert (asked taste))

(assert (explore level))

)

Explanation :

This rule is used to select the taste and a fact which determines the taste preference of a customer.

Remaining Facts:

(defrule ask-value

(declare (salience 60))

?state <- (state question)

(not (asked value))

(explore level)

=>

(printout t "On a scale of 0-4, what's your tolerance (or preference) of your chosen taste?" crlf)

(bind ?response (read))

(switch ?response

(case 0 then

(assert (preference (property tasted) (numberValue 0))))

(case 1 then

(assert (preference (property tasted)(numberValue 1))))

(case 2 then

(assert (preference (property tasted)(numberValue 2))))

(case 3 then

(assert (preference (property tasted) (numberValue 3))))

(case 4 then

(assert (preference (property tasted) (numberValue 4))))

(default

(assert (preference (property tasted) (numberValue 2)))))

(assert (asked value)))

(defrule process-value

(declare (salience 58))

(preference (property tasted)(numberValue ?preferred))

?dish <- (dish (level ?found))

=>

(if (> ?found ?preferred)

then

(retract ?dish)

)

)

(defrule check-after-value

(declare (salience 56))

(preference (numberValue ?preferred))

(not (checked-after value))

=>

(assert (check-remaining))

(assert (checked-after value)))

(defrule ask-nutrition

(declare (salience 35))

?state <- (state question)

(not (asked nutrition))

=>

(printout t "Do you want to specify any nutrition preferences? (yes/no)" crlf)

(bind ?response (read))

(if (eq ?response yes)

then

(printout t "Got it. Let me ask you a few more questions about your nutrition preferences." crlf)

(assert (explore nutrition))

else

(retract ?state)

(assert (state suggest))

(assert (asked all)))

(assert (asked nutrition)))

(defrule ask-lowcal

(declare (salience 30))

?state <- (state question)

(not (asked lowcal))

(explore nutrition)

=>

(printout t "Do you prefer food that has low calories? (yes/no/Either)" crlf)

(bind ?response (read))

(switch ?response

(case yes then (assert (preference (property lowcal) (symbolValue TRUE))))

(case no then (assert (preference (property lowcal) (symbolValue FALSE)))))

(assert (asked lowcal)))

(defrule process-lowcal

(declare (salience 28))

(preference (property lowcal) (symbolValue ?preferred))

?dish <- (dish (lowcal ?found))

=>

(if (not (eq ?found ?preferred))

then

(retract ?dish)))

(defrule check-after-lowcal

(declare (salience 26))

(preference (property lowcal))

(not (checked-after lowcal))

=>

(assert (check-remaining))

(assert (checked-after lowcal)))

(defrule ask-lowfat

(declare (salience 18))

?state <- (state question)

(not (asked lowfat))

(explore nutrition)

=>

(printout t "Do you prefer food that has low fat? (yes/no/Either)" crlf)

(bind ?response (read))

(switch ?response

(case yes then (assert (preference (property lowfat) (symbolValue TRUE))))

(case no then (assert (preference (property lowfat) (symbolValue FALSE)))))

(assert (asked lowfat)))

(defrule process-lowfat

(declare (salience 16))

(preference (property lowfat) (symbolValue ?preferred))

?dish <- (dish (lowfat ?found))

=>

(if (not (eq ?found ?preferred))

then

(retract ?dish)))

(defrule check-after-lowfat

(declare (salience 14))

(preference (property lowfat))

(not (checked-after lowfat))

=>

(assert (check-remaining))

(assert (checked-after lowfat))

)

(defrule ask-highfiber

(declare (salience 12))

?state <- (state question)

(not (asked highfiber))

(explore nutrition)

=>

(printout t "Do you prefer food that has high fiber? (yes/no/Either)" crlf)

(bind ?response (read))

(switch ?response

(case yes then (assert (preference (property highfiber) (symbolValue TRUE))))

(case no then (assert (preference (property highfiber) (symbolValue FALSE)))))

(assert (asked highfiber)))

(defrule process-highfiber

(declare (salience 10))

(preference (property highfiber) (symbolValue ?preferred))

?dish <- (dish (highfiber ?found))

=>

(if (not (eq ?found ?preferred))

then

(retract ?dish)))

(defrule check-after-highfiber

(declare (salience 8))

(preference (property highfiber))

(not (checked-after highfiber))

?state <- (state question)

=>

(assert (check-remaining))

(assert (checked-after highfiber))

(retract ?state)

(assert (state suggest))

(assert (asked all)))

(defrule start-drink-module

(declare (salience 7))

(state question2)

(state drink)

(not (asked cool/hot-drink))

=>

(printout t "What do you prefer? (cool/hot/Either)" crlf)

(bind ?response (read))

(switch ?response

(case cool then (assert (preference (property drink) (symbolValue cool))))

(case hot then (assert (preference (property drink) (symbolValue hot))))

(default (printout t "Great, so I'll search across cool AND hot drinks!" crlf)))

(assert (asked cool/hot-drink))

)

(defrule process-drink

(declare (salience 6))

(preference (property drink) (symbolValue ?preferred))

?dis <- (drink (category ?found))

=>

(if (not (eq ?preferred ?found))

then

(retract ?dis)))

(defrule check-after-drink

(declare (salience 5))

(state drink)

(not (asked all2))

(not (checked-after drink))

=>

(assert (check-remaining))

(assert (checked-after drink))

(assert (asked all2)))

(defrule out-of-questions2

(declare (salience 3))

?state <- (state question2)

(asked all2)

=>

(retract ?state)

(assert (state suggest2)))

(defrule out-of-questions

(declare (salience 4))

?state <- (state question)

(asked all)

=>

(retract ?state)

(assert (state suggest)))

**Explanation:**

Rules like

defrule ask-nutrition -> this rule asks about nutrition preference in the selected category by above rules. If the customer prefers nutrition rules like high-fibre, low-fat, low-cal are fired.

After execution of above said rules a dish preferred by customer is suggested to him.

Now the expert system asks for the drink preference.

After that rules like process-drink and check-after drink is executed.