Appendix

```
;;; Food & Wine Pairing Knowledge Base for Joshua
;;; for use with Rule-Based Systems Exercises
;;; 6.871 Spring 2005


;;; Code pertaining to setup of system
;;; Code mostly reused from pset 2

(in-package :ju)

; (ask [wine-to-drink john ?x] #'print-answer-with-certainty)

(defun print-answer-with-certainty (backward-support &optional (stream
*standard-output*))
  (check-type backward-support cons "backward-support from a query")
  (let ((predication (ask-database-predication backward-support)))
    (check-type predication predication "a predication from a query")
    (terpri stream)
    (ji::truth-value-case  (predication-truth-value predication)
      (*true*
       (prin1 predication stream))
      (*false*
       (write-string "[not " stream)
       (ji::print-without-truth-value predication stream)
       (write-string "]" stream)))
    (format stream " ~d" (certainty-factor predication))))

(defgeneric possesive-suffix (predication))
(defgeneric first-prompt (predication))
(defgeneric second-prompt (predication))
(defgeneric third-prompt (predication))
(defgeneric possible-values (predication))
(defgeneric get-an-answer (predication &optional stream))
(defgeneric appropriate-ptype (predication))
(defgeneric accept-prompt (predication))
(defgeneric question-prefix (predication))
(defgeneric remaining-object-string (predication))

;;; The base mixin
(define-predicate-model question-if-unknown-model () () )

(clim:define-gesture-name :my-rule :keyboard (:r :control :shift))
(clim:define-gesture-name :my-help :keyboard (:h :control :shift))
(clim:define-gesture-name :my-why :keyboard (:w :control :shift))

(defparameter *mycin-help-string*
  "
 ctrl-?  - to show the valid answers to this question
 meta-r  - to show the current rule
 meta-y  - to see why this question is asked
 meta-h  - to see this list"
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;
                          ;;;
;;; explaining why we're asking what we're asking
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun print-why (trigger rule &optional (stream *standard-output*))
  (format stream "~%We are trying to determine ")
  (if (predicationp trigger)
    (progn (format stream "~a " (question-prefix trigger)) (say trigger
stream))
    (princ trigger stream))
  (if (null rule)
    (format stream "~%This is a top level query")
    (let* ((debug-info (ji::rule-debug-info rule))
           (sub-goals (let ((ji::*known-lvs* nil))(eval (ji::rule-
debug-info-context debug-info)))))
      (format stream "~%This is being asked for by the rule ~a in order
to determine:~%"
              rule)
      (format stream "~a " (question-prefix ji::*goal*)) (say
ji::*goal* stream)
      (typecase sub-goals
        (ji::and-internal
         (let ((remaining-preds (rest (predication-statement sub-
goals)))
               (good-answers nil)
               (remaining-stuff nil)
               (first-remaining-object-string nil))
           (labels ((do-good-preds ()
                      (when remaining-preds
                        (let ((first (pop remaining-preds)))
                          (cond
                           ((not (predicationp first))
                            (push (copy-object-if-necessary first)
good-answers)
                            (do-good-preds))
                           (t
                            (let ((found-it nil))
                              (ask first
                                   #'(lambda (just)
                                       (push (ask-database-predication
just) good-answers)
                                       (setq found-it t)
                                       (do-good-preds))
                                   :do-backward-rules nil
                                   :do-questions nil)
                              (unless found-it
                                (with-statement-destructured (who
value) first
                                  (declare (ignore who))
                                  (with-unification
                                    (unify trigger first)
                                    (setq first-remaining-object-string
(remaining-object-string first))
```

```
                                    (unify value first-remaining-
object-string)
                                    (setq remaining-stuff
                                          (loop for pred in remaining-
preds
                                                if (predicationp pred)
                                                collect (with-
statement-destructured (who value) pred
                                                                  (declare
(ignore who))
                                                                  (unify value
(if (joshua:unbound-logic-variable-p value)

(remaining-object-string pred)

(joshua:joshua-logic-variable-value value)))
                                                                  (copy-object-
if-necessary pred))
                                                else collect (copy-
object-if-necessary pred))))))))))))
             (do-good-preds))
           (loop for pred in (nreverse good-answers)
                 for first-time = t then nil
                 if first-time
                 do (format stream "~%It has already been determined
whether: ")
                 else do (format stream "~%and whether: ")
                 do (say pred stream))
           (format stream "~%It remains to determine ~a ~a ~a"
                   (question-prefix trigger) first-remaining-object-
string (remaining-stuff-suffix trigger))
           (loop for pred in remaining-stuff
                 do (format stream "~%and ~a ~a ~a" (question-prefix
pred) (remaining-object-string pred) (remaining-stuff-suffix pred)))))
         (otherwise ))
       )))

(defmethod remaining-stuff-suffix ((pred predication)) "is")
(defmethod remaining-stuff-suffix ((expression cons)) "")
(defmethod predication-value-description ((pred predication))
(remaining-object-string pred))


;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;;  PROTOCOL HACKING
;;;
;;;;;;;;;;;;;;;;;;;;;;;;

(defmethod say ((expression cons) &optional (stream *standard-output*))
  (princ expression stream))

(defmethod remaining-object-string ((expression cons)) (format nil "~a"
expression))

(defmethod question-prefix ((expression cons)) "whether")
```

```
(defmethod get-an-answer ((predication question-if-unknown-model)
&optional (stream *standard-output*))
  "Print the prompt for this parameter (or make one up) and read the
reply."
  (fresh-line)
  (flet ((mycin-help (stream action string-so-far)
           (declare (ignore string-so-far))
           (when (member action '(:help :my-help :my-rule :my-why))
             (fresh-line stream)
             (case action
               (:my-why
                (print-why predication ji::*running-rule* stream)
                )
               (:my-rule
                (format stream "You are running the rule ~a"
ji::*running-rule*))
               (:my-help
                (format stream *mycin-help-string*)
                ))
             (fresh-line stream)
             (write-string "You are being asked to enter " stream)
             (clim:describe-presentation-type (appropriate-ptype
predication) stream)
             (write-char #\. stream)
             )))
    (let ((clim:*help-gestures* (list* :my-help :my-why :my-rule
clim:*help-gestures*)))
      (clim:with-accept-help ((:top-level-help #'mycin-help))
        (clim:accept (appropriate-ptype predication)
                     :stream stream
                     :prompt (accept-prompt predication))))))


(defun rules-concluding-predicate (pred)
  (let ((answers nil))
    (map-over-backward-rule-triggers `[,pred ? ?]
                                     #'(lambda (trigger) (pushnew
(ji::backward-trigger-rule trigger) answers)))
    answers))

(defun predicates-rule-relies-on (rule)
  (let ((answers nil))
    (labels ((do-one-level (stuff)
               (let ((connective (when (predication-maker-p stuff)
(predication-maker-predicate stuff))))
                 (case connective
                   ((and or)
                    (with-predication-maker-destructured (&rest more-
stuff) stuff
                      (loop for thing in more-stuff
                            do (do-one-level thing))))
                   ((nil))
                   (otherwise
                    (pushnew connective answers))
                   ))))
      (do-one-level (ji::rule-debug-info-context (ji::rule-debug-info
rule))))
```

```
    answers))


(defun graph-rule-tree (predicates &key (orientation :vertical) (size
:small) (stream *standard-output*))
  (terpri stream)
  (clim:with-text-size (stream size)
    (clim:format-graph-from-roots
     (loop for pred in predicates
           collect (list 'predicate pred))
     #'(lambda (thing stream)
         (destructuring-bind (type name) thing
           (case type
             (predicate
              (clim:surrounding-output-with-border (stream)
                (princ name stream)))
             (rule
              (clim:surrounding-output-with-border (stream :shape
:oval)
                (princ name stream))))))
     #'(lambda (thing)
         (destructuring-bind (type name) thing
           (case type
             (predicate (loop for r in (rules-concluding-predicate
name)
                              collect (list 'rule r)))
             (rule (loop for p in (predicates-rule-relies-on name)
                         collect (list 'predicate p))))))
     :stream stream
     :orientation orientation
     :merge-duplicates t
     :duplicate-test #'equal)))

(clim-env::define-lisp-listener-command (com-graph-rules :name t)
                                        ((predicates `(clim:sequence
(member ,@(loop for pred being the hash-keys of ji::*all-predicates*
collect pred)))
                                                     :prompt "A
sequence of predicates")
                                         &key
                                         (orientation `(clim:member
:vertical :horizontal) :default :vertical)
                                         (size `(clim:member :tiny
:very-small :small :normal :large :very-large :huge)
                                               :default :small)
                                         (to-file 'clim:pathname
:default nil)
                                         (page-orientation
'(clim:member :portrait :landscape)
                                                           :default
:portrait
                                                           :prompt "If
to file, print in portrait or landscape format")
                                         (multi-page 'clim:boolean
:default nil :prompt "If to file, segment into multiple pages")
                                         (scale-to-fit 'clim:boolean
:default nil :prompt "If to file, scale to fit one page"))
```

```
   (if to-file
     (with-open-file (file to-file :direction :output :if-exists
:supersede :if-does-not-exist :create)
       (clim:with-output-to-postscript-stream (stream file
                                                      :multi-page
multi-page
                                                      :scale-to-fit
scale-to-fit
                                                      :orientation
page-orientation)
         (graph-rule-tree predicates :orientation orientation :size
size :stream stream)))
     (graph-rule-tree predicates :orientation orientation :size size)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;
;;;;  Our pseudo mycin contains 3 types of predications
            ;;;
;;;;  boolean valued, numeric valued, and those that take one of
      ;;;
;;;;  a set of values
                        ;;;
;;;;  For each type we provide say methods
                   ;;;
;;;;   and a bunch of subordinate methods to make dialog almost English
      ;;;
;;;;   and to do CLIM accepts correctly
                   ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;

;;;;; boolean values
(define-predicate-model value-is-boolean-mixin () () )

(define-predicate-method (say value-is-boolean-mixin) (&optional
(stream *standard-output*))
  (with-statement-destructured (user yesno) self
    (format stream "~A~A ~A ~A"
            user (possesive-suffix self)
            (if (joshua:joshua-logic-variable-value yesno) (first-
prompt self) (second-prompt self))
            (third-prompt self))))

(defmethod remaining-object-string ((predication value-is-boolean-
mixin))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~A ~A ~a"
            (joshua:joshua-logic-variable-value user)
            (first-prompt predication) (third-prompt predication))))

(defmethod appropriate-ptype ((predication value-is-boolean-mixin))
'(clim:member yes no))

(defmethod accept-prompt ((predication value-is-boolean-mixin))
  (with-statement-destructured (user value) predication
```

```
    (declare (ignore value))
    (format nil "~%Does ~a~a ~a ~a"
            user (possesive-suffix predication)
            (first-prompt predication)
            (third-prompt predication))))

(defmethod question-prefix ((predication value-is-boolean-mixin))
"whether")

(defmethod possible-values ((predication value-is-boolean-mixin))
'("yes" "no"))

(defmethod remaining-stuff-suffix ((pred value-is-boolean-mixin)) "")
(defmethod predication-value-description ((pred value-is-boolean-
mixin)) "foobar")

;;; Other form of boolean questioning

(define-predicate-model value-is-boolean-mixin2 () () )

(define-predicate-method (say value-is-boolean-mixin2) (&optional
(stream *standard-output*))
  (with-statement-destructured (user yesno) self
    (format stream "~A~A ~A ~A"
            user (possesive-suffix self)
            (if (joshua:joshua-logic-variable-value yesno) (first-
prompt self) (second-prompt self))
            (third-prompt self))))

(defmethod remaining-object-string ((predication value-is-boolean-
mixin2))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~A ~A ~a"
            (joshua:joshua-logic-variable-value user)
            (first-prompt predication) (third-prompt predication))))

(defmethod appropriate-ptype ((predication value-is-boolean-mixin2))
'(clim:member yes no))

(defmethod accept-prompt ((predication value-is-boolean-mixin2))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~%Is ~a~a ~a ~a"
            user (possesive-suffix predication)
            (first-prompt predication)
            (third-prompt predication))))

(defmethod question-prefix ((predication value-is-boolean-mixin2))
"whether")

(defmethod possible-values ((predication value-is-boolean-mixin2))
'("yes" "no"))

(defmethod remaining-stuff-suffix ((pred value-is-boolean-mixin2)) "")
(defmethod predication-value-description ((pred value-is-boolean-
mixin2)) "foobar")
```

```
(define-predicate-model value-is-boolean-mixin3 () () )

(define-predicate-method (say value-is-boolean-mixin3) (&optional
(stream *standard-output*))
  (with-statement-destructured (user yesno) self
    (format stream "~A~A ~A ~A"
            user (possesive-suffix self)
            (if (joshua:joshua-logic-variable-value yesno) (first-
prompt self) (second-prompt self))
            (third-prompt self))))

(defmethod remaining-object-string ((predication value-is-boolean-
mixin3))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~A ~A ~a"
            (joshua:joshua-logic-variable-value user)
            (first-prompt predication) (third-prompt predication))))

(defmethod appropriate-ptype ((predication value-is-boolean-mixin3))
'(clim:member yes no))

(defmethod accept-prompt ((predication value-is-boolean-mixin3))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~%Can ~a~a ~a ~a"
            user (possesive-suffix predication)
            (first-prompt predication)
            (third-prompt predication))))

(defmethod question-prefix ((predication value-is-boolean-mixin3))
"whether")

(defmethod possible-values ((predication value-is-boolean-mixin3))
'("yes" "no"))

(defmethod remaining-stuff-suffix ((pred value-is-boolean-mixin3)) "")
(defmethod predication-value-description ((pred value-is-boolean-
mixin3)) "foobar")


(define-predicate-model value-is-boolean-mixin4 () () )

(define-predicate-method (say value-is-boolean-mixin4) (&optional
(stream *standard-output*))
  (with-statement-destructured (user yesno) self
    (format stream "~A~A ~A ~A"
            user (possesive-suffix self)
            (if (joshua:joshua-logic-variable-value yesno) (first-
prompt self) (second-prompt self))
            (third-prompt self))))

(defmethod remaining-object-string ((predication value-is-boolean-
mixin4))
  (with-statement-destructured (user value) predication
```

```
    (declare (ignore value))
    (format nil "~A ~A ~a"
            (joshua:joshua-logic-variable-value user)
            (first-prompt predication) (third-prompt predication))))

(defmethod appropriate-ptype ((predication value-is-boolean-mixin4))
'(clim:member yes no))

(defmethod accept-prompt ((predication value-is-boolean-mixin4))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~%Does the ~a~a ~a ~a"
            (possesive-suffix predication)
            user
            (first-prompt predication)
            (third-prompt predication))))

(defmethod question-prefix ((predication value-is-boolean-mixin4))
"whether")

(defmethod possible-values ((predication value-is-boolean-mixin4))
'("yes" "no"))

(defmethod remaining-stuff-suffix ((pred value-is-boolean-mixin4)) "")
(defmethod predication-value-description ((pred value-is-boolean-
mixin4)) "foobar")


;;;; numeric values

(define-predicate-model value-is-numeric-mixin () () )
(define-predicate-method (say value-is-numeric-mixin) (&optional
(stream *standard-output*))
  (with-statement-destructured (user number) self
    (if (joshua:unbound-logic-variable-p number)
      (format stream "is ~a~a ~a"
              user (possesive-suffix self) (first-prompt self))
      (format stream "~A~A ~A is ~A ~A"
              user (possesive-suffix self)
              (first-prompt self)
              (joshua:joshua-logic-variable-value number)
              (second-prompt self)))))

(defmethod remaining-object-string ((predication value-is-numeric-
mixin))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~A~A ~A"
            (joshua:joshua-logic-variable-value user) (possesive-suffix
predication)
            (first-prompt predication))))


(defmethod appropriate-ptype ((predication value-is-numeric-mixin))
'number)

(defmethod accept-prompt ((predication value-is-numeric-mixin))
```

```
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~%What is ~a~a ~a"
            user (possesive-suffix predication) (first-prompt
predication))))

(defmethod question-prefix ((predication value-is-numeric-mixin))
"what")


;;; variety of possible values

(define-predicate-model value-is-option-mixin ()  () )

(define-predicate-method (say value-is-option-mixin) (&optional (stream
*standard-output*))
  (with-statement-destructured (user option) self
    (format stream "~A~A ~A ~A ~A"
            user (possesive-suffix self)
            (first-prompt self)
            (second-prompt self)
            (joshua:joshua-logic-variable-value option))))

(defmethod remaining-object-string ((predication value-is-option-
mixin))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~A~A ~A"
            (joshua:joshua-logic-variable-value user) (possesive-suffix
predication)
            (first-prompt predication))))

(defmethod appropriate-ptype ((predication value-is-option-mixin))
`(member ,@(possible-values predication)))

(defmethod accept-prompt ((predication value-is-option-mixin))
  (with-statement-destructured (user value) predication
    (declare (ignore value))
    (format nil "~%What is ~a~a ~a"
            user (possesive-suffix predication) (first-prompt
predication))))

(defmethod question-prefix ((predication value-is-option-mixin))
"whether")


;;; Predicate defining macro

(defmacro define-predicate-with-ancillary-info ((pred-name mixin)
                                                &key
                                                possesive-suffix
                                                prompt1 prompt2 prompt3
                                                possible-values
                                                missing-value-prompt
                                                )
  `(eval-when (:compile-toplevel :execute :load-toplevel)
```

```
     (define-predicate ,pred-name (user value) (,mixin question-if-
unknown-model cf-mixin ltms:ltms-predicate-model))
     (defmethod possesive-suffix ((predication ,pred-name)) ()
,possesive-suffix)
     (defmethod first-prompt ((predication ,pred-name)) () ',prompt1)
     (defmethod second-prompt ((predication ,pred-name)) () ',prompt2)
     ,(when prompt3 `(defmethod third-prompt ((predication ,pred-name))
() ',prompt3))
     ,(when possible-values `(defmethod possible-values ((predication
,pred-name)) ',possible-values))
     ,(when missing-value-prompt `(defmethod missing-value-prompt
((predication ,pred-name)) ',missing-value-prompt))
  ))

;; using this model, the system will ask the user any time
;; it needs a specific fact to continue backward chaining.

;;; we should only be asking a question under the following
;;; circumstances:
;;;
;;; the predication being asked contains no logic variables
;;; eg. [has-health-insurance matt yes], not
;;; [has-health-insurance matt ?x]
;;;
;;; AND
;;;
;;; that predication is not already in the database
;;;
;;; AND
;;;
;;; any other predication matching the predicate and ?user
;;; eg. [has-health-insurance matt no] is not already in the
;;; database.
;;;
;;; AND
;;;
;;; there is no rule we can use to find out the answer
;;;
;;; this can be told by check [known [has-health-insurance matt ?]]


(define-predicate already-known (predicate object))

;;; if after doing the normal processing nothing is found
;;; then finally ask the guy a question if appropriate
(define-predicate-method (ask question-if-unknown-model) (intended-
truth-value continuation do-backward-rules do-questions)
  (let ((answers nil)
        (predicate (predication-predicate self)))
    (flet ((my-continuation (bs)
             (let* ((answer (ask-query bs))
                    (database-answer (insert (copy-object-if-necessary
answer))))
               (pushnew database-answer answers))))
      (with-statement-destructured (user value) self
        (declare (ignore value))
        (with-unbound-logic-variables (value)
```

```
          (let ((predication `[,predicate ,user ,value]))
            ;; first see if there's an answer already in the database
            ;; may want to change this to asserting already-know
predication, but I'm trying to avoid that
            (ask-data predication intended-truth-value #'my-
continuation)
            (unless answers
              ;; Now go get stuff from rules.
              (when do-backward-rules
                (ask-rules predication intended-truth-value #'my-
continuation do-questions))
              ;; now go hack questions
              (unless answers
                (when do-questions
                  (ask-questions predication intended-truth-value #'my-
continuation))))))
        ;; if he's doing a raws database fetch, don't ask
        (when (and (null answers) (or do-backward-rules do-questions))
          (unless (joshua:unbound-logic-variable-p user)
            (let* ((answer (get-an-answer self))
                 (database-answer (tell `[,predicate ,user ,answer]
                                        :justification '((user-input
1.0)))))
              (pushnew database-answer answers))))))
    (loop for answer in answers
          when (eql (predication-truth-value answer) intended-truth-
value)
          do (with-stack-list (just self intended-truth-value answer)
               (with-unification
                 (unify self answer)
                 (funcall continuation just)))))
  ;; make it clear that there is no interesting return value
  (values))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;  Code pertaining to the food and wine pairing system



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; General predicates that take numeric values
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(define-predicate-with-ancillary-info (minimum-temp value-is-numeric-
mixin)
  :possesive-suffix "'s" :prompt1 "location doesn't go lower than"
:prompt2 "degrees")

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;
                          ;;;
;;; General predicates that take one of a set of values
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (meal value-is-option-mixin)
  :possesive-suffix "'s" :prompt1 "meal type: lunch, dinner or desser"
:prompt2 "is of type"
  :possible-values (lunch dinner dessert))

(define-predicate-with-ancillary-info (food-ethnicity value-is-option-
mixin)
  :possesive-suffix "'s" :prompt1 "meal ethnicity" :prompt2 "is"
  :possible-values (AMERICAN ITALIAN FRENCH CARIBBEAN MEXICAN MIDDLE-
EASTERN INDIAN CHINESE))

(define-predicate-with-ancillary-info (wine-to-drink value-is-option-
mixin)
  :possesive-suffix "'s" :prompt1 "recommended wine" :prompt2 "is"
  :possible-values (PINOT-NOIR CABERNET-SAUVIGNON MERLOT ZINFANDEL
SAUVIGNON-BLANC CHARDONNAY RIESLING))

(define-predicate-with-ancillary-info (meat-round1 value-is-option-
mixin)
  :possesive-suffix "" :prompt1 "might eat" :prompt2 ""
  :possible-values (pork/beef poultry/seafood))

(define-predicate-with-ancillary-info (final-meat value-is-option-
mixin)
  :possesive-suffix "" :prompt1 "is most likely" :prompt2 "eating"
  :possible-values (BBQ beef seafood poultry pork lamb gamebird
shellfish none))

(define-predicate-with-ancillary-info (sauce-type value-is-option-
mixin)
  :possesive-suffix "'s meal" :prompt1 "is most likely based on this
sauce" :prompt2 ""
  :possible-values (BBQ TOMATOES CREAM LIGHT CURRY)
  :prompt3 "sauce")

(define-predicate-with-ancillary-info (month value-is-option-mixin)
  :possesive-suffix "" :prompt1 "month" :prompt2 "is"
  :possible-values (JANUARY FEBRUARY MARCH APRIL MAY JUNE JULY AUGUST
SEPTEMBER OCTOBER NOVEMBER DECEMBER))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Boolean valued predicates relating to herbs/spices
      ;;;
;;;
                          ;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (use-mint value-is-boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "mint in it")
(define-predicate-with-ancillary-info (use-sage value-is-boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "sage in it")
(define-predicate-with-ancillary-info (use-cinnamon value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "cinnamon in it")
(define-predicate-with-ancillary-info (use-black-pepper value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "black pepper in it")
(define-predicate-with-ancillary-info (use-bay-leaf value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "bay leaf in it")
(define-predicate-with-ancillary-info (use-parsley value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "parsley in it")
(define-predicate-with-ancillary-info (use-nutmeg value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "nutmeg in it")
(define-predicate-with-ancillary-info (use-chili value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "chili in it")
(define-predicate-with-ancillary-info (use-curry value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "curry in it")
(define-predicate-with-ancillary-info (use-oregano value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "oregano in it")
(define-predicate-with-ancillary-info (use-basil value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "basil in it")
(define-predicate-with-ancillary-info (use-tarragon value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "tarragon in it")
(define-predicate-with-ancillary-info (use-thyme value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "thyme in it")
(define-predicate-with-ancillary-info (use-clove value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "clove in it")
```

```
(define-predicate-with-ancillary-info (use-garlic value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "garlic in it")
(define-predicate-with-ancillary-info (use-pepper value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "pepper in it")
(define-predicate-with-ancillary-info (use-mustard value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "mustard in it")
(define-predicate-with-ancillary-info (use-ginger value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "ginger in it")
(define-predicate-with-ancillary-info (use-caribbean value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "caribbean spices in it")
(define-predicate-with-ancillary-info (use-dill value-is-boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "dill in it")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Boolean valued predicates relating to sauces
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(define-predicate-with-ancillary-info (use-bbq-sauce value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "BBQ sauce in it")
(define-predicate-with-ancillary-info (has-tomatoes value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "has" :prompt2 "doesn't have"
:prompt3 "tomatoes")
(define-predicate-with-ancillary-info (use-cream-sauce value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "a cream based sauce in it")
(define-predicate-with-ancillary-info (use-light-sauce value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "light sauce in it")
(define-predicate-with-ancillary-info (use-curry-sauce value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "shouldn't have"
:prompt3 "curry sauce in it")

(define-predicate-with-ancillary-info (use-bbq-sauce-r1 value-is-
option-mixin)
```

```
  :possesive-suffix "'s meal" :prompt1 "most likely has" :prompt2
"shouldn't have"
  :possible-values (Yes No maybe))
(define-predicate-with-ancillary-info (has-tomatoes-r1 value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "most likely has" :prompt2
"shouldn't have"
  :possible-values (Yes No maybe))
(define-predicate-with-ancillary-info (use-cream-sauce-r1 value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "most likely has" :prompt2
"shouldn't have"
  :possible-values (Yes No maybe))
(define-predicate-with-ancillary-info (use-curry-sauce-r1 value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "most likely has" :prompt2
"shouldn't have"
  :possible-values (Yes No maybe))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; General boolean valued predicates
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (having-lunch value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "" :prompt2 "not" :prompt3 "having
lunch")
(define-predicate-with-ancillary-info (having-dinner value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "" :prompt2 "not" :prompt3 "having
dinner")
(define-predicate-with-ancillary-info (having-dessert value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "" :prompt2 "not" :prompt3 "having
dessert")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to meats
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(define-predicate-with-ancillary-info (bbq value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a BBQ meal")
(define-predicate-with-ancillary-info (beef value-is-boolean-mixin2)
```

```
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a beef meal")
(define-predicate-with-ancillary-info (seafood value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a seafood meal")
(define-predicate-with-ancillary-info (poultry value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a chicken meal")
(define-predicate-with-ancillary-info (pork value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a pork meal")
(define-predicate-with-ancillary-info (lamb value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a lamb meal")
(define-predicate-with-ancillary-info (gamebird value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a gamebird meal")
(define-predicate-with-ancillary-info (shellfish value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "having" :prompt2 "is not having"
:prompt3 "a shellfish meal")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to location
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (is-near-coast value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "located" :prompt2 "is not located"
:prompt3 "near the coast")
(define-predicate-with-ancillary-info (is-south value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "located" :prompt2 "is not located"
:prompt3 "in the south")
(define-predicate-with-ancillary-info (is-midwest value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "located" :prompt2 "is not located"
:prompt3 "in the midwest")
(define-predicate-with-ancillary-info (is-north value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "located" :prompt2 "is not located"
:prompt3 "in the north")
(define-predicate-with-ancillary-info (is-east value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "located" :prompt2 "is not located"
:prompt3 "in the east")
(define-predicate-with-ancillary-info (has-boat value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "has" :prompt2 "doesn't have" :prompt3
"a boat")
```

```
(define-predicate-with-ancillary-info (goes-to-the-beach value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "go often" :prompt2 "is not" :prompt3
"to the beach")
(define-predicate-with-ancillary-info (likes-country value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "like" :prompt2 "doesn't like" :prompt3
"country music")
(define-predicate-with-ancillary-info (says-pop value-is-boolean-mixin)
  :possesive-suffix "" :prompt1 "say" :prompt2 "doesn't say" :prompt3
"pop when talking about coke/pepsi")


(define-predicate-with-ancillary-info (is-near-coast-r1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "is located" :prompt2 "is not located"
  :possible-values (yes no idk))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to seasons
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (is-spring value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "the current season" :prompt2 "is not
currently in" :prompt3 "spring")
(define-predicate-with-ancillary-info (is-summer value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "the current season" :prompt2 "is not
currently in" :prompt3 "summer")
(define-predicate-with-ancillary-info (is-fall value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "the current season" :prompt2 "is not
currently in" :prompt3 "fall")
(define-predicate-with-ancillary-info (is-winter value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "the current season" :prompt2 "is not
currently in" :prompt3 "winter")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to wine characteristics
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(define-predicate-with-ancillary-info (prefers-sweet-wines value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "prefer" :prompt2 "does not like"
:prompt3 "sweet wines")
(define-predicate-with-ancillary-info (prefers-dry-wines value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "prefer" :prompt2 "does not like"
:prompt3 "dry wines")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Random Meal Stuff
            ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (has-cheese value-is-boolean-
mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "doesn't have"
:prompt3 "cheese in it")
(define-predicate-with-ancillary-info (is-pasta value-is-boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "have" :prompt2 "doesn't have"
:prompt3 "pasta in it")
(define-predicate-with-ancillary-info (is-vegetarian value-is-boolean-
mixin2)
  :possesive-suffix "" :prompt1 "" :prompt2 "is not" :prompt3
"vegetarian")
(define-predicate-with-ancillary-info (is-spicy value-is-boolean-
mixin2)
  :possesive-suffix "'s meal" :prompt1 "" :prompt2 "is not" :prompt3
"spicy")
(define-predicate-with-ancillary-info (like-indian value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "also enjoy" :prompt2 "doesn't really
like" :prompt3 "indian food")
(define-predicate-with-ancillary-info (likes-savory value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "enjoy" :prompt2 "doesn't really like"
:prompt3 "savory foods")

(define-predicate-with-ancillary-info (on-date value-is-boolean-mixin2)
  :possesive-suffix "" :prompt1 "" :prompt2 "is not" :prompt3 "on a
date")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to health consciousness
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(define-predicate-with-ancillary-info (health-conscious-r1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "is health conscious?  Yes No idk"
:prompt2 "is not"
  :possible-values (yes no idk))


(define-predicate-with-ancillary-info (health-conscious value-is-
boolean-mixin2)
  :possesive-suffix "" :prompt1 "" :prompt2 "is not" :prompt3 "health
conscious")
(define-predicate-with-ancillary-info (has-dieted-in-past-year value-
is-boolean-mixin)
  :possesive-suffix "" :prompt1 "" :prompt2 "doesn't" :prompt3 "diet or
has followed some sort of diet during the past year")
(define-predicate-with-ancillary-info (exercises value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "" :prompt2 "doesn't" :prompt3
"exercise more than twice a week")
(define-predicate-with-ancillary-info (follows-the-pyramid value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "pay" :prompt2 "doesn't pay" :prompt3
"attention to what he/she eats")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to savory foods
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (like-savory value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "like" :prompt2 "doesn't like" :prompt3
"savory foods")
(define-predicate-with-ancillary-info (has-herb-garden value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "have" :prompt2 "doesn't have" :prompt3
"a herb garden")
(define-predicate-with-ancillary-info (has-several-spices value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "have" :prompt2 "doesn't have" :prompt3
"more than 5 herbs and spices in his/her spice cabinet")
(define-predicate-with-ancillary-info (enjoy-steak-sauce value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "like" :prompt2 "doesn't like" :prompt3
"steak sauce")

(define-predicate-with-ancillary-info (like-savory-r1 value-is-option-
mixin)
```

```
  :possesive-suffix "" :prompt1 "likes savory foods?  Yes No idk"
:prompt2 "doesn't"
  :possible-values (yes no idk))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to drink preference
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



(define-predicate-with-ancillary-info (prefers-fruit-punch value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "prefer" :prompt2 "does not prefer"
:prompt3 "fruit punch over iced tea")
(define-predicate-with-ancillary-info (prefers-iced-tea value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "prefer" :prompt2 "does not prefer"
:prompt3 "iced tea over fruit punch")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to lactose/allergies
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (eat-cheese value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "" :prompt2 "cannot" :prompt3 "eat
cheese")
(define-predicate-with-ancillary-info (eat-ice-cream value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "" :prompt2 "cannot" :prompt3 "eat ice
cream")
(define-predicate-with-ancillary-info (drink-milk value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "" :prompt2 "cannot" :prompt3 "drink-
milk")
(define-predicate-with-ancillary-info (lactose-tolerant value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "" :prompt2 "cannot" :prompt3 "tolerate
lactose")

(define-predicate-with-ancillary-info (lactose-tolerant-r1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "can tolerate lactose?  Yes No idk"
:prompt2 "can't tolerate"
```

```
  :possible-values (yes no idk))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to exotic taste
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (tries-new-dishes value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "like " :prompt2 "doesn't like"
:prompt3 "to try new dishes")
(define-predicate-with-ancillary-info (tries-dishes-w-weird-names
value-is-boolean-mixin)
  :possesive-suffix "" :prompt1 "like " :prompt2 "doesn't like"
:prompt3 "to try dishes with weird names")
(define-predicate-with-ancillary-info (likes-to-travel value-is-
boolean-mixin)
  :possesive-suffix "" :prompt1 "like " :prompt2 "doesn't like"
:prompt3 "to travel")
(define-predicate-with-ancillary-info (hunts value-is-boolean-mixin)
  :possesive-suffix "" :prompt1 "like " :prompt2 "doesn't like"
:prompt3 "hunt")
(define-predicate-with-ancillary-info (exotic-taste value-is-boolean-
mixin)
  :possesive-suffix "" :prompt1 "have" :prompt2 "doesn't have" :prompt3
"exotic tastes")


(define-predicate-with-ancillary-info (exotic-taste-r1 value-is-option-
mixin)
  :possesive-suffix "" :prompt1 "has exotic taste?  Yes No idk"
:prompt2 "doesn't"
  :possible-values (yes no idk))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to restaurant expense
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(define-predicate-with-ancillary-info (has-valet value-is-boolean-
mixin)
  :possesive-suffix "restaurant" :prompt1 "is in has " :prompt2
"doesn't have" :prompt3 "valet parking")
(define-predicate-with-ancillary-info (has-value-menu value-is-boolean-
mixin)
```

```
  :possesive-suffix "restaurant" :prompt1 "is in has " :prompt2
"doesn't have" :prompt3 "a value meal menu")
(define-predicate-with-ancillary-info (has-dress-code value-is-boolean-
mixin)
  :possesive-suffix "restaurant" :prompt1 "is in requires " :prompt2
"doesn't require" :prompt3 "a dress code")
(define-predicate-with-ancillary-info (has-host value-is-boolean-mixin)
  :possesive-suffix "restaurant" :prompt1 "is in have " :prompt2
"doesn't have" :prompt3 "a host or hostess")
(define-predicate-with-ancillary-info (is-expensive value-is-boolean-
mixin)
  :possesive-suffix "restaurant" :prompt1 "is going" :prompt2 "is not"
:prompt3 "expensive")

(define-predicate-with-ancillary-info (is-expensive-r1 value-is-option-
mixin)
  :possesive-suffix "" :prompt1 "is at an expensive restaurant?  Yes No
idk" :prompt2 "is not "
  :possible-values (yes no idk))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to American ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;; Predicates relating to meal type
(define-predicate-with-ancillary-info (full-meal-american value-is-
boolean-mixin2)
  :possesive-suffix "" :prompt1 "eating a f" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-american value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")

(define-predicate-with-ancillary-info (meat-american-round1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef poultry/seafood))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to Caribbean ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Predicates relating to meal type
```

```
(define-predicate-with-ancillary-info (full-meal-caribbean value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-caribbean value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")

(define-predicate-with-ancillary-info (meat-caribbean-round1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef poultry/seafood))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to Chinese ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Predicates relating to meal type
(define-predicate-with-ancillary-info (full-meal-chinese value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-chinese value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")

(define-predicate-with-ancillary-info (meat-chinese-round1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef poultry/seafood))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to Italian ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Predicates relating to meal type
(define-predicate-with-ancillary-info (full-meal-italian value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-italian value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")
```

```
(define-predicate-with-ancillary-info (cheese-meal-italian value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"based only on cheese")
(define-predicate-with-ancillary-info (meat-italian-round1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef poultry/seafood))

(define-predicate-with-ancillary-info (seafood/poultry-italian value-
is-boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"seafood/poultry")

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to Mexican ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Predicates relating to meal type
(define-predicate-with-ancillary-info (full-meal-mexican value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-mexican value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")

(define-predicate-with-ancillary-info (meat-mexican-round1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to French ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Predicates relating to meal type
(define-predicate-with-ancillary-info (full-meal-french value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-french value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")
```

```
(define-predicate-with-ancillary-info (meat-french-round1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef poultry/seafood))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to Indian ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Predicates relating to meal type
(define-predicate-with-ancillary-info (full-meal-indian value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-indian value-is-
boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")

(define-predicate-with-ancillary-info (meat-indian-round1 value-is-
option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef poultry/seafood))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
                          ;;;
;;; Predicates relating to Middle Eastern ethnicity
      ;;;
;;;
                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Predicates relating to meal type
(define-predicate-with-ancillary-info (full-meal-middle-eastern value-
is-boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"full")
(define-predicate-with-ancillary-info (dessert-meal-middle-eastern
value-is-boolean-mixin)
  :possesive-suffix "'s meal" :prompt1 "is" :prompt2 "is not" :prompt3
"dessert")

(define-predicate-with-ancillary-info (meat-middle-eastern-round1
value-is-option-mixin)
  :possesive-suffix "" :prompt1 "might" :prompt2 "eat"
  :possible-values (pork/beef poultry/seafood))

;;; The following sections are for questions for the specific
ethnicities

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  American Food   Specific
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;; Meal Type

(defrule full-meal-american (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user american]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-american ?user yes])

(defrule dessert-meal-american (:backward :certainty 1.0 :importance
91)
  if [and   [food-ethnicity ?user american]
                  [meal ?user dessert]]
  then      [dessert-meal-american ?user yes])

(defrule full-meal-american-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-american ?user yes]
  then [and [dessert-meal-american ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-american-exclusion (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user american]
                  [meal ?user dessert]]
  then [full-meal-american ?user no])

      ;;; Figure out meat

            ;;; General selection

(defrule meat-type-healthy-american (:backward :certainty 1.0
:importance 96)
  if [and   [food-ethnicity ?user american]
                  [full-meal-american ?user yes]
                  [health-conscious ?user yes]]
  then [meat-american-round1 ?user poultry/seafood])

(defrule meat-type-not-healthy-american (:backward :certainty 1.0
:importance 95)
  if [and   [food-ethnicity ?user american]
                  [full-meal-american ?user yes]
                  [health-conscious ?user no]]
  then [meat-american-round1 ?user pork/beef])

            ;;; final selection

 (defrule bbq-american (:backward :certainty 1.0 :importance 94)
   if [and [food-ethnicity ?user american]
               [full-meal-american ?user yes]
               [meat-american-round1 ?user pork/beef]
```

```
               [is-south ?user yes]
               [is-summer ?user yes]
               [use-bbq-sauce ?user yes]]
   then     [final-meat ?user BBQ])

 (defrule beef-american (:backward :certainty 1.0 :importance 93)
   if [and  [food-ethnicity ?user american]
                 [full-meal-american ?user yes]
                 [meat-american-round1 ?user pork/beef]
                 [is-midwest ?user yes]]
   then [beef ?user yes])

  (defrule beef-american-no (:backward :certainty 1.0 :importance 93)
   if [and  [food-ethnicity ?user american]
                 [full-meal-american ?user yes]
                 [meat-american-round1 ?user pork/beef]
                 [is-midwest ?user no]]
   then [beef ?user no])

 (defrule seafood-american (:backward :certainty 1.0 :importance 92)
   if [and  [food-ethnicity ?user american]
                 [full-meal-american ?user yes]
                 [meat-american-round1 ?user poultry/seafood]
                 [is-near-coast ?user yes]]
   then [seafood ?user yes])

  (defrule seafood-american-no (:backward :certainty 1.0 :importance
92)
   if [and  [food-ethnicity ?user american]
                 [full-meal-american ?user yes]
                 [meat-american-round1 ?user poultry/seafood]
                 [is-near-coast ?user no]]
   then [seafood ?user no])

 (defrule poultry-american (:backward :certainty 1.0 :importance 91)
   if [and  [food-ethnicity ?user american]
                 [full-meal-american ?user yes]
                 [meat-american-round1 ?user poultry/seafood]
                 [is-near-coast ?user no]]
   then [poultry ?user yes])

  (defrule poultry-american-no (:backward :certainty 1.0 :importance
91)
   if [and  [food-ethnicity ?user american]
                 [full-meal-american ?user yes]
                 [meat-american-round1 ?user poultry/seafood]
                 [is-near-coast ?user yes]]
   then [poultry ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Caribbean Food Specific
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
      ;;; Meal Type

(defrule full-meal-caribbean (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user caribbean]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-caribbean ?user yes])

(defrule dessert-meal-caribbean (:backward :certainty 1.0 :importance
91)
  if [and   [food-ethnicity ?user caribbean]
                  [meal ?user dessert]]
  then      [dessert-meal-caribbean ?user yes])

(defrule full-meal-caribbean-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-caribbean ?user yes]
  then [and [dessert-meal-caribbean ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-caribbean-exclusion (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user caribbean]
                  [meal ?user dessert]]
  then [full-meal-caribbean ?user no])

      ;;; Figure out meat

            ;;; General selection

(defrule meat-type-healthy-caribbean (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user caribbean]
                  [full-meal-caribbean ?user yes]
                  [health-conscious ?user yes]]
  then [meat-caribbean-round1 ?user poultry/seafood])

(defrule meat-type-not-healthy-caribbean (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user caribbean]
                  [full-meal-caribbean ?user yes]
                  [health-conscious ?user no]]
  then [meat-caribbean-round1 ?user pork/beef])

            ;;; final selection

 (defrule pork-caribbean (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user pork/beef]
                  [is-expensive ?user yes]]
   then [pork ?user yes])

  (defrule pork-caribbean-no (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user pork/beef]
                  [is-expensive ?user no]]
   then [pork ?user no])
```

```
 (defrule pork-caribbean-no2 (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user poultry/seafood]]
   then [pork ?user yes])

 (defrule beef-caribbean (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user pork/beef]
              [is-expensive ?user no]]
   then [beef ?user yes])

 (defrule beef-caribbean-no (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user pork/beef]
              [is-expensive ?user yes]]
   then [beef ?user no])

 (defrule beef-caribbean-no2 (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user poultry/seafood]]
   then [beef ?user no])

 (defrule seafood-caribbean (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user poultry/seafood]
              [is-near-coast ?user yes]]
   then [seafood ?user yes])

  (defrule seafood-caribbean-no (:backward :certainty 1.0 :importance
10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user poultry/seafood]
              [is-near-coast ?user no]]
   then [seafood ?user no])

  (defrule seafood-caribbean-no2 (:backward :certainty 1.0 :importance
10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user pork/beef]]
   then [seafood ?user no])

 (defrule poultry-caribbean (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user poultry/seafood]
              [is-near-coast ?user no]]
   then [poultry ?user yes])

 (defrule poultry-caribbean-no (:backward :certainty 1.0 :importance
10)
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user poultry/seafood]
              [is-near-coast ?user yes]]
   then [poultry ?user no])

 (defrule poultry-caribbean-no2 (:backward :certainty 1.0 :importance
10)
```

```
   if [and  [food-ethnicity ?user caribbean]
                  [meat-caribbean-round1 ?user pork/beef]]
   then [poultry ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Italian Food Specific
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;; Meal Type

(defrule full-meal-italian (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user italian]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-italian ?user yes])

(defrule dessert-meal-italian (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user italian]
                  [meal ?user dessert]]
  then      [dessert-meal-italian ?user yes])

(defrule full-meal-italian-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-italian ?user yes]
  then [and [dessert-meal-italian ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-italian-exclusion (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user italian]
                  [meal ?user dessert]]
  then [full-meal-italian ?user no])

(defrule vegetarian (:backward :certainty 1.0 :importance 10)
      if [is-vegetarian ?user yes]
      then [cheese-meal-italian ?user yes])

      ;;; Figure out meat

            ;;; General selection

(defrule meat-type-healthy-italian (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user italian]
                  [full-meal-italian ?user yes]
                  [health-conscious ?user yes]]
  then [seafood/poultry-italian ?user yes])

(defrule meat-type-healthy-italian-no1 (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user italian]
                  [full-meal-italian ?user yes]
                  [health-conscious ?user no]]
```

```
  then [seafood/poultry-italian ?user no])

           ;;; final selection

(defrule meat-type-not-healthy-italian (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user italian]
                [full-meal-italian ?user yes]
                [health-conscious ?user no]]
  then [beef ?user yes])

(defrule meat-type-not-healthy-italian-no (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user italian]
                [full-meal-italian ?user yes]
                [health-conscious ?user yes]]
  then [beef ?user no])

(defrule seafood-italian (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user italian]
                [seafood/poultry-italian ?user yes]
              [is-near-coast ?user yes]]
   then [seafood ?user yes])

(defrule seafood-italian-no (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user italian]
                [seafood/poultry-italian ?user yes]
              [is-near-coast ?user no]]
   then [seafood ?user no])

(defrule poultry-italian (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user italian]
                [seafood/poultry-italian ?user yes]
              [is-near-coast ?user no]]
   then [poultry ?user yes])

 (defrule poultry-italian-no (:backward :certainty 1.0 :importance 10)
   if [and  [food-ethnicity ?user italian]
                [seafood/poultry-italian ?user yes]
              [is-near-coast ?user yes]]
   then [poultry ?user no])

                ;;; ethnicity specific

(defrule cheese-italian (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user italian]
                [full-meal-italian ?user yes]
                [lactose-tolerant ?user yes]]
  then [has-cheese ?user yes])

(defrule cheese-italian-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user italian]
                [full-meal-italian ?user yes]
                [lactose-tolerant ?user no]]
  then [has-cheese ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Mexican Food Specific
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;; Meal Type

(defrule full-meal-mexican (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user mexican]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-mexican ?user yes])

(defrule dessert-meal-mexican (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user mexican]
                  [meal ?user dessert]]
  then      [dessert-meal-mexican ?user yes])

(defrule full-meal-mexican-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-mexican ?user yes]
  then [and [dessert-meal-mexican ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-mexican-exclusion (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user mexican]
                  [meal ?user dessert]]
  then [full-meal-mexican ?user no])

      ;;; Figure out meat

            ;;; General selection

(defrule meat-type-healthy-mexican (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [health-conscious ?user yes]]
  then [poultry ?user yes])

(defrule meat-type-healthy-mexican-no (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [health-conscious ?user no]]
  then [poultry ?user no])

(defrule meat-not-type-healthy-mexican (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [health-conscious ?user no]]
  then [meat-mexican-round1 ?user pork/beef])
```

```
            ;;; final selection

(defrule meat-not-type-healthy-mexican-beef (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [meat-mexican-round1 ?user pork/beef]
                  [is-expensive ?user yes]]
  then [beef ?user yes])

(defrule meat-not-type-healthy-mexican-beef-no (:backward :certainty
1.0 :importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [meat-mexican-round1 ?user pork/beef]
                  [is-expensive ?user no]]
  then [beef ?user no])

(defrule meat-not-type-healthy-mexican-pork (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [meat-mexican-round1 ?user pork/beef]
                  [is-expensive ?user no]]
  then [pork ?user yes])

(defrule meat-not-type-healthy-mexican-pork-no (:backward :certainty
1.0 :importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [meat-mexican-round1 ?user pork/beef]
                  [is-expensive ?user no]]
  then [pork ?user no])

            ;;; ethnicity specific

(defrule spicy-mexican (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [is-spicy ?user yes]]
  then [use-chili ?user yes])

(defrule spicy-mexican-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [is-spicy ?user no]]
  then [use-chili ?user no])

(defrule cheese-mexican (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
                  [lactose-tolerant ?user yes]]
  then [has-cheese ?user yes])

(defrule cheese-mexican-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user mexican]
                  [full-meal-mexican ?user yes]
```

```
                  [lactose-tolerant ?user no]]
  then [has-cheese ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                      ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Chinese Food Specific
      ;;;
;;;                                                                      ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;; Meal Type

(defrule full-meal-chinese (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user chinese]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-chinese ?user yes])

(defrule dessert-meal-chinese (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user chinese]
                  [meal ?user dessert]]
  then      [dessert-meal-chinese ?user yes])

(defrule full-meal-chinese-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-chinese ?user yes]
  then [and [dessert-meal-chinese ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-chinese-exclusion (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user chinese]
                  [meal ?user dessert]]
  then [full-meal-chinese ?user no])

      ;;; Figure out meat

            ;;; General selection

(defrule meat-type-healthy-chinese (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [health-conscious ?user yes]]
  then [meat-chinese-round1 ?user poultry/seafood])

(defrule meat-type-unhealthy-chinese (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [health-conscious ?user no]]
  then [meat-chinese-round1 ?user pork/beef])

            ;;; final selection

(defrule meat-seafood-chinese (:backward :certainty 1.0 :importance 10)
```

```
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user poultry/seafood]
                  [is-near-coast ?user yes]]
  then [seafood ?user yes])

(defrule meat-seafood-chinese-no (:backward :certainty 1.0 :importance
11)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user pork/beef]]
  then [seafood ?user no])

 (defrule meat-seafood-chinese-no2 (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user poultry/seafood]
                  [is-near-coast ?user no]]
  then [seafood ?user no])

(defrule meat-poultry-chinese (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user poultry/seafood]
                  [is-near-coast ?user no]]
  then [poultry ?user yes])

(defrule meat-poultry-chinese-no (:backward :certainty 1.0 :importance
11)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user pork/beef]]
  then [poultry ?user no])

(defrule meat-poultry-chinese-no2 (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user poultry/seafood]
                  [is-near-coast ?user yes]]
  then [poultry ?user no])

(defrule meat-beef-chinese (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user pork/beef]
                  [is-expensive ?user yes]]
  then [beef ?user yes])

(defrule meat-beef-chinese-no (:backward :certainty 1.0 :importance 11)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user poultry/seafood]]
  then [beef ?user no])
```

```
 (defrule meat-beef-chinese-no2 (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user pork/beef]
                  [is-expensive ?user no]]
  then [beef ?user no])

(defrule meat-pork-chinese (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user pork/beef]
                  [is-expensive ?user no]]
  then [pork ?user yes])

(defrule meat-pork-chinese-no (:backward :certainty 1.0 :importance 11)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user poultry/seafood]]
  then [pork ?user no])

 (defrule meat-pork-chinese-no2 (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [meat-chinese-round1 ?user pork/beef]
                  [is-expensive ?user yes]]
  then [pork ?user no])

           ;;; ethnicity specific

(defrule spicy-chinese1 (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [is-spicy ?user yes]]
  then [use-ginger ?user yes])

(defrule spicy-chinese1-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [is-spicy ?user no]]
  then [use-ginger ?user no])

(defrule spicy-chinese2 (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [is-spicy ?user yes]]
  then [use-chili ?user yes])

(defrule spicy-chinese2-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user chinese]
                  [full-meal-chinese ?user yes]
                  [is-spicy ?user no]]
  then [use-chili ?user no])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Middle Eastern Food Specific
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;; Meal Type

(defrule full-meal-middle-eastern (:backward :certainty 1.0 :importance
91)
  if [and   [food-ethnicity ?user middle-eastern]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-middle-eastern ?user yes])

(defrule dessert-meal-middle-eastern (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user middle-eastern]
                  [meal ?user dessert]]
  then      [dessert-meal-middle-eastern ?user yes])

(defrule full-meal-middle-eastern-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-middle-eastern ?user yes]
  then [and [dessert-meal-middle-eastern ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-middle-eastern-exclusion (:backward :certainty
1.0 :importance 91)
  if [and   [food-ethnicity ?user middle-eastern]
                  [meal ?user dessert]]
  then [full-meal-middle-eastern ?user no])

      ;;; Figure out meat

            ;;; General selection

(defrule meat-type-healthy-middle-eastern (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [health-conscious ?user yes]]
  then [meat-middle-eastern-round1 ?user poultry/seafood])

(defrule meat-type-not-healthy-middle-eastern (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [health-conscious ?user no]]
  then [meat-middle-eastern-round1 ?user beef/lamb])

            ;;; final selection

(defrule meat-type-poultry-middle-eastern (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
```

```
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user poultry/seafood]
                  [is-near-coast ?user no]]
  then [poultry ?user yes])

(defrule meat-type-poultry-middle-eastern-no (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user poultry/seafood]
                  [is-near-coast ?user yes]]
  then [poultry ?user no])

(defrule meat-type-poultry-middle-eastern-no2 (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user beef/lamb]]
  then [poultry ?user no])

(defrule meat-type-seafood-middle-eastern (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user poultry/seafood]
                  [is-near-coast ?user yes]]
  then [seafood ?user yes])

(defrule meat-type-seafood-middle-eastern-no (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user poultry/seafood]
                  [is-near-coast ?user no]]
  then [seafood ?user no])

(defrule meat-type-seafood-middle-eastern-no2 (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user beef/lamb]]
  then [seafood ?user no])

(defrule meat-type-lamb-middle-eastern (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user beef/lamb]
                  [like-indian ?user yes]]
  then [lamb ?user yes])

(defrule meat-type-lamb-middle-eastern-no (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                  [full-meal-middle-eastern ?user yes]
                  [meat-middle-eastern-round1 ?user beef/lamb]
                  [like-indian ?user no]]
```

```
  then [lamb ?user no])

(defrule meat-type-lamb-middle-eastern-no2 (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                [full-meal-middle-eastern ?user yes]
                [meat-middle-eastern-round1 ?user poultry/seafood]]
  then [lamb ?user no])

(defrule meat-type-beef-middle-eastern (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                [full-meal-middle-eastern ?user yes]
                [meat-middle-eastern-round1 ?user beef/lamb]
                [like-indian ?user no]]
  then [beef ?user yes])

(defrule meat-type-beef-middle-eastern-no (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                [full-meal-middle-eastern ?user yes]
                [meat-middle-eastern-round1 ?user beef/lamb]
                [like-indian ?user yes]]
  then [beef ?user no])

(defrule meat-type-beef-middle-eastern-no2 (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                [full-meal-middle-eastern ?user yes]
                [meat-middle-eastern-round1 ?user poultry/seafood]]
  then [beef ?user no])

           ;;; ethnicity specific

(defrule tarragon-middle-eastern (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user middle-eastern]
                [full-meal-middle-eastern ?user yes]
                [likes-savory ?user yes]]
  then [use-tarragon ?user yes])

(defrule tarragon-middle-eastern-no (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                [full-meal-middle-eastern ?user yes]
                [likes-savory ?user no]]
  then [use-tarragon ?user no])

(defrule dill-middle-eastern (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user middle-eastern]
                [full-meal-middle-eastern ?user yes]
                [likes-savory ?user yes]]
  then [use-dill ?user yes])

(defrule dill-middle-eastern-no (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user middle-eastern]
```

```
                  [full-meal-middle-eastern ?user yes]
                  [likes-savory ?user no]]
  then [use-dill ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Indian Food Specific
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;; Meal Type

(defrule full-meal-indian (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user indian]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-indian ?user yes])

(defrule dessert-meal-indian (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user indian]
                  [meal ?user dessert]]
  then      [dessert-meal-indian ?user yes])

(defrule full-meal-indian-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-indian ?user yes]
  then [and [dessert-meal-indian ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-indian-exclusion (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user indian]
                  [meal ?user dessert]]
  then [full-meal-indian ?user no])

      ;;; Figure out meat


(defrule meat-type-poultry-indian(:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user indian]
                  [full-meal-indian ?user yes]
                  [health-conscious ?user yes]]
  then [poultry ?user yes])

(defrule meat-type-poultry-indian-no(:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user indian]
                  [full-meal-indian ?user yes]
                  [health-conscious ?user no]]
  then [poultry ?user no])

(defrule meat-type-lamb-indian(:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user indian]
                  [full-meal-indian ?user yes]
```

```
                  [health-conscious ?user no]]
  then [lamb ?user yes])

(defrule meat-type-lamb-indian-no(:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user indian]
                  [full-meal-indian ?user yes]
                  [health-conscious ?user yes]]
  then [lamb ?user no])

            ;;; ethnicity specific

(defrule spicy-indian (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user indian]
            [full-meal-indian ?user yes]
            [is-spicy ?user yes]]
  then [use-curry ?user yes])

(defrule spicy-indian-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user indian]
            [full-meal-indian ?user yes]
            [is-spicy ?user no]]
  then [use-curry ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;  Inference Rules (For importance, higher values go first.)         ;;;
;;;  Domain:  French Food Specific
      ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;; Meal Type

(defrule full-meal-french (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user french]
                  [or [meal ?user lunch]
                        [meal ?user dinner]]]
  then [full-meal-french ?user yes])

(defrule dessert-meal-french (:backward :certainty 1.0 :importance 91)
  if [and   [food-ethnicity ?user french]
                  [meal ?user dessert]]
  then      [dessert-meal-french ?user yes])

(defrule full-meal-french-exclusion (:forward :certainty 1.0
:importance 90)
      if [full-meal-french ?user yes]
  then [and [dessert-meal-french ?user no]
                  [having-dessert ?user no]])

(defrule dessert-meal-french-exclusion (:backward :certainty 1.0
:importance 91)
  if [and   [food-ethnicity ?user french]
                  [meal ?user dessert]]
  then [full-meal-french ?user no])
```

```
      ;;; Figure out meat

            ;;; General selection

(defrule meat-type-healthy-french (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user french]
                  [full-meal-french ?user yes]
                  [health-conscious ?user yes]]
  then [meat-french-round1 ?user poultry/seafood])

(defrule meat-type-unhealthy-french (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes]
                  [health-conscious ?user no]]
  then [meat-french-round1 ?user pork/gamebird])

            ;;; final selection

(defrule meat-seafood-french (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes]
                  [meat-french-round1 ?user poultry/seafood]
                  [is-near-coast ?user yes]]
  then [seafood ?user yes])

 (defrule meat-seafood-french-no (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes]
                  [meat-french-round1 ?user poultry/seafood]
                  [is-near-coast ?user no]]
  then [seafood ?user no])

  (defrule meat-seafood-french-no2 (:backward :certainty 1.0
:importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes]
                  [meat-french-round1 ?user pork/gamebird]]
  then [seafood ?user no])

(defrule meat-poultry-french (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes]
                  [meat-french-round1 ?user poultry/seafood]
                  [is-near-coast ?user no]]
  then [poultry ?user yes])

(defrule meat-poultry-french-no (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes]
                  [meat-french-round1 ?user poultry/seafood]
                  [is-near-coast ?user yes]]
  then [poultry ?user no])
```

```
(defrule meat-poultry-french-no2 (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes]
                  [meat-french-round1 ?user pork/gamebird]]
  then [poultry ?user no])

(defrule meat-gamebird-french (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes ]
                  [meat-french-round1 ?user pork/gamebird]
                  [exotic-taste ?user yes]]
  then [gamebird ?user yes])

(defrule meat-gamebird-french-no (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes ]
                  [meat-french-round1 ?user pork/gamebird]
                  [exotic-taste ?user no]]
  then [gamebird ?user no])

(defrule meat-gamebird-french-no2 (:backward :certainty 1.0 :importance
10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes ]
                  [meat-french-round1 ?user poultry/seafood]]
  then [gamebird ?user no])

(defrule meat-pork-french (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes ]
                  [meat-french-round1 ?user pork/gamebird]
                  [exotic-taste ?user no]]
  then [pork ?user yes])

(defrule meat-pork-french-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes ]
                  [meat-french-round1 ?user pork/gamebird]
                  [exotic-taste ?user yes]]
  then [pork ?user no])

(defrule meat-pork-french-no2 (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes ]
                  [meat-french-round1 ?user poultry/seafood]]
  then [pork ?user no])

            ;;; ethnicity specific

(defrule garlic-french (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                  [full-meal-french ?user yes ]
                  [on-date ?user no]]
  then [use-garlic ?user yes])
```

```
(defrule garlic-french-no (:backward :certainty 1.0 :importance 10)
  if [and   [food-ethnicity ?user french ]
                [full-meal-french ?user yes ]
                [on-date ?user yes]]
  then [use-garlic ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                          ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  General Rules Based on Ethnicities
      ;;;
;;;                                                                          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule no-meat (:forward :certainty 1.0 :importance 90)
      if [meal ?user dessert]
      then [final-meat ?user none])

(defrule no-meat2 (:forward :certainty 1.0 :importance 90)
      if [final-meat ?user none]
      then [and   [beef ?user no]
                        [seafood ?user no]
                        [poultry ?user no]
                        [pork ?user no]
                        [lamb ?user no]
                        [gamebird ?user no]
                        [shellfish ?user no]])

(defrule BBQ (:forward :certainty 1.0 :importance 90)
      if [final-meat ?user bbq]
      then [and   [beef ?user yes]
                        [seafood ?user no]
                        [poultry ?user yes]
                        [pork ?user yes]
                        [lamb ?user no]
                        [gamebird ?user no]
                        [shellfish ?user no]])

(defrule beef (:forward :certainty 1.0 :importance 91)
      if [beef ?user yes]
      then [and   [seafood ?user no]
                        [poultry ?user no]
                        [pork ?user no]
                        [lamb ?user no]
                        [gamebird ?user no]
                        [shellfish ?user no]])

(defrule seafood (:forward :certainty 1.0 :importance 90)
      if [seafood ?user yes]
      then [and   [beef ?user no]
                        [poultry ?user no]
                        [pork ?user no]
                        [lamb ?user no]
                        [gamebird ?user no]
                        [shellfish ?user yes]])

(defrule poultry (:forward :certainty 1.0 :importance 90)
```

```
      if [poultry ?user yes]
      then [and   [beef ?user no]
                  [seafood ?user no]
                  [pork ?user no]
                  [lamb ?user no]
                  [gamebird ?user no]
                  [shellfish ?user no]])

(defrule pork (:forward :certainty 1.0 :importance 90)
      if [pork ?user yes]
      then [and   [beef ?user no]
                  [seafood ?user no]
                  [poultry ?user no]
                  [lamb ?user no]
                  [gamebird ?user no]
                  [shellfish ?user no]])

(defrule lamb (:forward :certainty 1.0 :importance 90)
      if [lamb ?user yes]
      then [and   [beef ?user no]
                  [seafood ?user no]
                  [poultry ?user no]
                  [pork ?user no]
                  [gamebird ?user no]
                  [shellfish ?user no]])

(defrule gamebird (:forward :certainty 1.0 :importance 90)
      if [gamebird ?user yes]
      then [and   [beef ?user no]
                  [seafood ?user no]
                  [poultry ?user no]
                  [pork ?user no]
                  [lamb ?user no]
                  [shellfish ?user no]])

(defrule shellfish (:forward :certainty 1.0 :importance 90)
      if [shellfish ?user yes]
      then [and   [beef ?user no]
                  [seafood ?user no]
                  [poultry ?user no]
                  [pork ?user no]
                  [lamb ?user no]
                  [gamebird ?user no]])

(defrule no-pasta (:forward :certainty 1.0 :importance 99)
      if    [or         [food-ethnicity ?user american]
                        [food-ethnicity ?user french]
                        [food-ethnicity ?user caribbean]
                        [food-ethnicity ?user mexican]
                        [food-ethnicity ?user indian]
                        [food-ethnicity ?user middle-eastern]
                        [food-ethnicity ?user chinese]]
      then [is-pasta ?user no])

(defrule no-cheese (:forward :certainty 1.0 :importance 99)
      if    [or [lactose-tolerant ?user no]
                [or         [food-ethnicity ?user american]
```

```
                              [food-ethnicity ?user french]
                              [food-ethnicity ?user caribbean]
                              [food-ethnicity ?user indian]
                              [food-ethnicity ?user middle-eastern]
                              [food-ethnicity ?user chinese]]]
      then [has-cheese ?user no])

;;; the sauces yes or no rules

(defrule BBQ-sauce-yes (:backward :certainty 1.0 :importance 99)
      if    [or   [use-bbq-sauce-r1 ?user yes]
                  [and  [use-bbq-sauce-r1 ?user maybe]
                              [and  [is-south ?user yes]
                                          [is-summer ?user yes]]]]
      then  [use-bbq-sauce ?user yes])

(defrule BBQ-sauce-no (:backward :certainty 1.0 :importance 99)
      if    [or   [use-bbq-sauce-r1 ?user no]
                  [and  [use-bbq-sauce-r1 ?user maybe]
                              [or               [is-south ?user no]
                                                [is-summer ?user no]]]]
      then  [use-bbq-sauce ?user no])

(defrule tomato-sauce-yes (:backward :certainty 1.0 :importance 99)
      if    [or   [has-tomatoes-r1 ?user yes]
                  [and  [has-tomatoes-r1 ?user maybe]
                              [or   [and  [lactose-tolerant ?user no]
                                                [is-summer ?user yes]]
                                    [and  [lactose-tolerant ?user yes]
                                                [is-summer ?user
yes]]]]]
      then  [has-tomatoes ?user yes])

(defrule tomato-sauce-no (:backward :certainty 1.0 :importance 99)
      if    [or   [has-tomatoes-r1 ?user no]
                  [and  [has-tomatoes-r1 ?user maybe]
                              [or   [and  [lactose-tolerant ?user no]
                                                [is-summer ?user no]]
                                    [and  [lactose-tolerant ?user yes]
                                                [is-summer ?user
no]]]]]
      then  [has-tomatoes ?user no])

(defrule cream-sauce-yes (:backward :certainty 1.0 :importance 99)
      if    [or   [use-cream-sauce-r1 ?user yes]
                  [and  [use-cream-sauce-r1 ?user maybe]
                              [lactose-tolerant ?user yes]]]
      then  [use-cream-sauce ?user yes])

(defrule cream-sauce-no (:backward :certainty 1.0 :importance 99)
      if    [or   [use-cream-sauce-r1 ?user no]
                  [and  [use-cream-sauce-r1 ?user maybe]
                              [lactose-tolerant ?user no]]]
      then  [use-cream-sauce ?user no])

(defrule curry-sauce-yes (:backward :certainty 1.0 :importance 99)
      if    [or   [use-curry-sauce-r1 ?user yes]
```

```
                  [and  [use-curry-sauce-r1 ?user maybe]
                              [use-curry ?user yes]]]
      then  [use-curry-sauce ?user yes])

(defrule curry-sauce-no (:backward :certainty 1.0 :importance 99)
      if    [or   [use-curry-sauce-r1 ?user no]
                  [and  [use-curry-sauce-r1 ?user maybe]
                              [use-curry ?user no]]]
      then  [use-curry-sauce ?user no])

;;; the herbs/sauces
;;; This is a real cheap way to do this, there has to be a simpler way

(defrule american-herbs-sauces (:forward :certainty 1.0 :importance 99)
      if    [food-ethnicity ?user american]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user no]
                        [use-oregano ?user yes]
                        [use-basil ?user yes]
                        [use-tarragon ?user yes]
                        [use-thyme ?user no]
                        [use-clove ?user no]
                        [use-garlic ?user yes]
                        [use-pepper ?user no]
                        [use-mustard ?user yes]
                        [use-ginger ?user yes]
                        [use-dill ?user no]
                        [use-caribbean ?user no]
                        [use-bbq-sauce-r1 ?user maybe]
                        [use-light-sauce ?user no]
                        [has-tomatoes-r1 ?user no]
                        [use-cream-sauce-r1 ?user no]
                        [use-curry-sauce-r1 ?user no]
                        [is-pasta ?user no]
                        [lamb ?user no]
                        [gamebird ?user no]])

(defrule caribbean-herbs-sauces (:forward :certainty 1.0 :importance
99)
      if    [food-ethnicity ?user caribbean]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user no]
                        [use-curry ?user no]
                        [use-oregano ?user yes]
```

```
                        [use-basil ?user yes]
                        [use-tarragon ?user yes]
                        [use-thyme ?user no]
                        [use-clove ?user no]
                        [use-garlic ?user yes]
                        [use-pepper ?user no]
                        [use-mustard ?user yes]
                        [use-ginger ?user yes]
                        [use-dill ?user no]
                        [use-caribbean ?user yes]
                        [use-bbq-sauce-r1 ?user no]
                        [use-light-sauce ?user yes]
                        [has-tomatoes-r1 ?user maybe]
                        [use-cream-sauce-r1 ?user maybe]
                        [use-curry-sauce-r1 ?user no]
                        [is-pasta ?who no]
                        [lamb ?user no]
                        [gamebird ?user no]
                        ])

(defrule italian-herbs-sauces (:forward :certainty 1.0 :importance 99)
      if    [food-ethnicity ?user italian]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user no]
                        [use-oregano ?user yes]
                        [use-basil ?user yes]
                        [use-tarragon ?user yes]
                        [use-thyme ?user no]
                        [use-clove ?user no]
                        [use-garlic ?user yes]
                        [use-pepper ?user no]
                        [use-mustard ?user yes]
                        [use-ginger ?user yes]
                        [use-dill ?user no]
                        [use-caribbean ?user yes]
                        [use-bbq-sauce-r1 ?user no]
                        [use-light-sauce ?user no]
                        [has-tomatoes-r1 ?user maybe]
                        [use-cream-sauce-r1 ?user maybe]
                        [use-curry-sauce-r1 ?user no]
                        [is-pasta ?who yes]
                        [lamb ?user no]
                        [gamebird ?user no]])

(defrule mexican-herbs-sauces (:forward :certainty 1.0 :importance 99)
      if    [food-ethnicity ?user mexican]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
```

```
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]

                        [use-curry ?user no]
                        [use-oregano ?user yes]
                        [use-basil ?user yes]
                        [use-tarragon ?user yes]
                        [use-thyme ?user no]
                        [use-clove ?user no]
                        [use-garlic ?user yes]
                        [use-pepper ?user no]
                        [use-mustard ?user yes]
                        [use-ginger ?user yes]
                        [use-dill ?user no]
                        [use-caribbean ?user yes]
                        [use-bbq-sauce-r1 ?user no]
                        [use-light-sauce ?user yes]
                        [has-tomatoes-r1 ?user maybe]
                        [use-cream-sauce-r1 ?user maybe]
                        [use-curry-sauce-r1 ?user no]
                        [lamb ?user no]
                        [gamebird ?user no]
                        [seafood ?user no]
                        [shellfish ?user no]])

(defrule chinese-herbs-sauces (:forward :certainty 1.0 :importance 99)
      if    [food-ethnicity ?user chinese]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]

                        [use-curry ?user no]
                        [use-oregano ?user yes]
                        [use-basil ?user yes]
                        [use-tarragon ?user yes]
                        [use-thyme ?user no]
                        [use-clove ?user no]
                        [use-garlic ?user yes]
                        [use-pepper ?user no]
                        [use-mustard ?user yes]

                        [use-dill ?user no]
                        [use-caribbean ?user yes]
                        [use-bbq-sauce-r1 ?user no]
                        [use-light-sauce ?user yes]
                        [has-tomatoes-r1 ?user no]
                        [use-cream-sauce-r1 ?user maybe]
                        [use-curry-sauce-r1 ?user no]
                        [is-pasta ?who no]
                        [lamb ?user no]
                        [gamebird ?user no]])
```

```
(defrule middle-eastern-herbs-sauces (:forward :certainty 1.0
:importance 99)
      if    [food-ethnicity ?user middle-eastern]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user no]
                        [use-oregano ?user yes]
                        [use-basil ?user yes]

                        [use-thyme ?user no]
                        [use-clove ?user no]
                        [use-garlic ?user yes]
                        [use-pepper ?user no]
                        [use-mustard ?user yes]
                        [use-ginger ?user yes]

                        [use-caribbean ?user yes]
                        [use-bbq-sauce-r1 ?user no]
                        [use-light-sauce ?user yes]
                        [has-tomatoes-r1 ?user no]
                        [use-cream-sauce-r1 ?user maybe]
                        [use-curry-sauce-r1 ?user no]
                        [is-pasta ?who no]
                        [pork ?user no]
                        [gamebird ?user no]
                        ])

(defrule indian-herbs-sauces (:forward :certainty 1.0 :importance 99)
      if    [food-ethnicity ?user indian]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]

                        [use-oregano ?user yes]
                        [use-basil ?user yes]
                        [use-tarragon ?user yes]
                        [use-thyme ?user no]
                        [use-clove ?user no]
                        [use-garlic ?user yes]
                        [use-pepper ?user no]
                        [use-mustard ?user yes]
                        [use-ginger ?user yes]
                        [use-dill ?user no]
                        [use-caribbean ?user yes]
                        [use-bbq-sauce-r1 ?user no]
                        [use-light-sauce ?user no]
```

```
                        [has-tomatoes-r1 ?user no]
                        [use-cream-sauce-r1 ?user maybe]

                        [use-curry-sauce-r1 ?user maybe]
                        [is-pasta ?who no]
                        [beef ?user no]
                        [pork ?user no]
                        [gamebird ?user no]
                        [seafood ?user no]
                        [shellfish ?user no]
                        ])

(defrule french-herbs-sauces (:forward :certainty 1.0 :importance 99)
      if    [food-ethnicity ?user french]
      then [and   [use-mint ?user no]
                        [use-sage ?user yes]
                        [use-cinnamon ?user yes]
                        [use-black-pepper ?user yes]
                        [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user no]
                        [use-oregano ?user yes]
                        [use-basil ?user yes]
                        [use-tarragon ?user yes]
                        [use-thyme ?user no]
                        [use-clove ?user no]

                        [use-pepper ?user no]
                        [use-mustard ?user yes]
                        [use-ginger ?user yes]
                        [use-dill ?user no]
                        [use-caribbean ?user yes]
                        [use-bbq-sauce-r1 ?user no]
                        [use-light-sauce ?user yes]
                        [has-tomatoes-r1 ?user maybe]
                        [use-cream-sauce-r1 ?user maybe]
                        [use-curry-sauce-r1 ?user no]
                        [is-pasta ?who no]
                        [lamb ?user no]
                        [beef ?user no]])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Drink Type Preference
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule sweet-drinks (:backward :certainty 1.0 :importance 60)
      if [prefers-fruit-punch ?user yes]
      then  [prefers-sweet-wines ?user yes])

(defrule sweet-drinks2 (:backward :certainty 1.0 :importance 60)
      if [prefers-fruit-punch ?user no]
```

```
      then  [prefers-sweet-wines ?user no])

(defrule dry-drinks (:backward :certainty 1.0 :importance 59)
      if [prefers-iced-tea ?user yes]
      then [prefers-dry-wines ?user yes])

(defrule dry-drinks2 (:backward :certainty 1.0 :importance 59)
      if [prefers-iced-tea ?user no]
      then [prefers-dry-wines ?user no])

(defrule wine-exclusion1 (:forward :certainty 1.0 :importance 69)
      if [prefers-sweet-wines ?user yes]
      then [prefers-dry-wines ?user no])

(defrule wine-exclusion2 (:forward :certainty 1.0 :importance 69)
      if [prefers-dry-wines ?user yes]
      then [prefers-sweet-wines ?user no])

;;; The following rules are used as a second layer of knowledge.  It
;;; helps to determine some facts that the user does not know
;;; To implement it as a "backup" or "secondary" layer, i need to
;;; change the values for these domain from simple booleans to a list
of values
;;; and then have the check for a "I don't know" response in the
antecedent


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Location
           ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



(defrule near-coastl (:backward :certainty 1.0 :importance 10)
  if [or    [is-near-coast-r1 ?user yes]
                [and  [is-near-coast-r1 ?user idk]
                            [or   [goes-to-the-beach ?user yes]
                                  [has-boat ?user yes]]]]
  then [is-near-coast ?user yes])

(defrule near-coast-no (:backward :certainty 1.0 :importance 10)
  if [or    [is-near-coast-r1 ?user no]
                [and  [is-near-coast-r1 ?user idk]
                            [or   [goes-to-the-beach ?user no]
                                  [has-boat ?user no]]]]
  then [is-near-coast ?user no])

(defrule warm-state (:backward :certainty 1.0 :importance 58)
  if [and   [minimum-temp ?user ?x]
                (> ?x 0)]
  then [is-south ?user yes])

(defrule warm-state-no (:backward :certainty 1.0 :importance 58)
```

```
  if [and   [minimum-temp ?user ?x]
                (< ?x 0)]
  then [is-south ?user no])

(defrule cold-n-country (:backward :certainty 1.0 :importance 58)
  if [or    [and [minimum-temp ?user ?x]
                           (< ?x 0)
                       [likes-country ?user yes]]
                [says-pop ?user yes]]
  then [is-midwest ?user yes])

(defrule midwest-no (:backward :certainty 1.0 :importance 58)
  if  [and  [minimum-temp ?user ?x]
                                        (< ?x 0)
                      [likes-country ?user no]
                      [says-pop ?user no]]
  then [is-midwest ?user no])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Health Conscious
           ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule health-conscious-yes (:backward :certainty 1.0 :importance 90)
  if [or    [health-conscious-r1 ?user yes]
                [and  [health-conscious-r1 ?user idk]
                           [or         [has-dieted-in-past-year
?user yes]
                                       [exercises ?user yes]
                                       [follows-the-pyramid ?user
yes]
                                       [is-vegetarian ?user yes]]]]
  then [health-conscious ?user yes])

(defrule health-conscious-no (:backward :certainty 1.0 :importance 90)
  if [or    [health-conscious-r1 ?user no]
                [and  [health-conscious-r1 ?user idk]
                           [has-dieted-in-past-year ?user no]
                           [exercises ?user no]
                           [follows-the-pyramid ?user no]
                           [is-vegetarian ?user no]]]
  then [health-conscious ?user no])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Savory Food
           ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule likes-savory (:backward :certainty 1.0 :importance 58)
```

```
  if [or    [like-savory-r1 ?user yes]
                [and  [like-savory-r1 ?user idk]
                            [or         [has-herb-garden ?user yes]
                                        [has-several-spices ?user
yes]
                                        [enjoy-steak-sauce ?user
yes]]]]
  then [likes-savory ?user yes])

(defrule likes-savory-no (:backward :certainty 1.0 :importance 58)
  if [or    [like-savory-r1 ?user no]
                [and  [like-savory-r1 ?user idk]
                            [has-herb-garden ?user no]
                            [has-several-spices ?user no]
                            [enjoy-steak-sauce ?user no]]]
  then [likes-savory ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Restaurant Expense
      ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule expensive-restaurant (:backward :certainty 1.0 :importance 58)
  if [or    [is-expensive-r1 ?user yes]
                [and  [is-expensive-r1 ?user idk]
                            [or         [has-valet ?user yes]
                                        [has-dress-code ?user yes]
                                        [has-host ?user yes]]]]
  then [is-expensive ?user yes])

(defrule cheap-restaurant-1 (:backward :certainty 1.0 :importance 58)
  if [or    [is-expensive-r1 ?user no]
                [or   [and  [is-expensive-r1 ?user idk]
                                  [has-valet ?user no]
                                  [has-dress-code ?user no]
                                  [has-host ?user no]]
                      [and  [is-expensive-r1 ?user idk]
                                  [has-value-menu ?user yes]]]]
  then [is-expensive ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Exotic taste
           ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule exotic-taste (:backward :certainty 1.0 :importance 82)
      if    [or   [exotic-taste-r1 ?user yes]
                            [exotic-taste-r1 ?user idk]
                            [or         [tries-new-dishes ?user yes]
                                        [tries-dishes-w-weird-names
?user yes]
```

```
                                          [likes-to-travel ?user yes]
                                          [hunts ?user yes]]]
      then [exotic-taste ?user yes])

(defrule no-exotic-taste (:backward :certainty 1.0 :importance 82)
      if [or      [exotic-taste-r1 ?user no]
                  [and  [exotic-taste-r1 ?user idk]
                              [tries-new-dishes ?user no]
                              [tries-dishes-w-weird-names ?user no]
                              [likes-to-travel ?user no]
                              [hunts ?user no]]]
      then [exotic-taste ?user no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                  ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Season Questions
            ;;;
;;;                                                                  ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; Spring is March 20 through June 19
;;; Summer is June 20 through September 21
;;; Fall is September 22 through December 20
;;; Winter is December 21 through March 19

(defrule spring-time (:backward :certainty 1.0 :importance 82)
      if [or      [month ?user march]
                  [month ?user april]
                  [month ?user may]]
      then  [is-spring ?user yes])

(defrule spring-time-2 (:forward :certainty 1.0 :importance 82)
      if          [is-spring ?user yes]
      then [and   [is-summer ?user no]
                        [is-fall ?user no]
                        [is-winter ?user no]])

(defrule summer-time (:backward :certainty 1.0 :importance 82)
      if [or      [month ?user june]
                  [month ?user july]
                  [month ?user august]]
      then  [is-summer ?user yes])

(defrule summer-time-2 (:forward :certainty 1.0 :importance 82)
      if          [is-summer ?user yes]
      then [and   [is-spring ?user no]
                        [is-fall ?user no]
                        [is-winter ?user no]])

(defrule fall-time (:backward :certainty 1.0 :importance 82)
      if [or      [month ?user september]
                  [month ?user october]
                  [month ?user november]]
      then  [is-fall ?user yes])

(defrule fall-time-2 (:forward :certainty 1.0 :importance 82)
```

```
      if          [is-fall ?user yes]
      then [and   [is-summer ?user no]
                        [is-spring ?user no]
                        [is-winter ?user no]])

(defrule winter-time (:backward :certainty 1.0 :importance 82)
      if [or      [month ?user december]
                  [month ?user january]
                  [month ?user february]]
      then  [is-winter ?user yes])

(defrule winter-time-2 (:forward :certainty 1.0 :importance 82)
      if          [is-winter ?user yes]
      then [and   [is-summer ?user no]
                        [is-fall ?user no]
                        [is-spring ?user no]])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Lactose/allergy Questions
      ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule lactose-tolerant (:backward :certainty 1.0 :importance 82)
      if  [or [lactose-tolerant-r1 ?user yes]
                  [and  [lactose-tolerant-r1 ?user idk]
                              [or [eat-cheese ?user yes]
                                    [drink-milk ?user yes]
                                    [eat-ice-cream ?user yes]]]]
      then [lactose-tolerant ?user yes])

(defrule lactose-intolerant (:backward :certainty 1.0 :importance 81)
      if  [or     [lactose-tolerant-r1 ?user no]
                  [and  [lactose-tolerant-r1 ?user idk]
                              [eat-cheese ?user no]
                              [drink-milk ?user no]
                              [eat-ice-cream ?user no]]]
      then [lactose-tolerant ?user no])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;  Inference Rules (For importance, higher values go first.)  ;;;
;;;  Domain:  Wine Choices
            ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;;
                        ;;;
      ;;; Pinot Noir
                  ;;;
      ;;;
                        ;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule pinot-noir-1 (:backward :certainty 1.0 :importance 98)
  if [and   [or [poultry ?user yes]
                [pork ?user yes]
                [lamb ?user yes]]
          [or [use-mint ?user yes]
                [use-sage ?user yes]
                [use-cinnamon ?user yes]]
          [has-cheese ?user yes]
          [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user pinot-noir])

(defrule pinot-noir-2 (:backward :certainty 1.0 :importance 87)
  if [and   [or [poultry ?user yes]
                [pork ?user yes]
                [lamb ?user yes]]
          [has-cheese ?user yes]
          [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user pinot-noir])

(defrule pinot-noir-3 (:backward :certainty 1.0 :importance 86)
  if [and   [or [poultry ?user yes]
                [pork ?user yes]
                [lamb ?user yes]]
          [or [use-mint ?user yes]
                [use-sage ?user yes]
                [use-cinnamon ?user yes]]
          [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user pinot-noir])

(defrule pinot-noir-4 (:backward :certainty 1.0 :importance 85)
  if [and   [or [poultry ?user yes]
                [pork ?user yes]
                [lamb ?user yes]]
          [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user pinot-noir])

(defrule pinot-noir-5 (:backward :certainty 1.0 :importance 84)
  if [and   [or [use-mint ?user yes]
                [use-sage ?user yes]
                [use-cinnamon ?user yes]]
          [has-cheese ?user yes]
          [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user pinot-noir])

(defrule pinot-noir-6 (:backward :certainty 1.0 :importance 83)
  if [and   [has-cheese ?user yes]
          [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user pinot-noir])


      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;;
                        ;;;
      ;;; Cabernet Sauvignon
            ;;;
```

```
    ;;;
                      ;;;
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defrule cabernet-sauvignon-1 (:backward :certainty 1.0 :importance 97)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [or   [has-cheese ?user yes]
                        [is-pasta ?user yes]]
                  [has-tomatoes ?user yes]
                  [or [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-2 (:backward :certainty 1.0 :importance 83)
  if  [and  [is-pasta ?user yes]
                  [has-tomatoes ?user yes]
                  [or [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-3 (:backward :certainty 1.0 :importance 83)
  if [and   [has-cheese ?user yes]
                  [has-tomatoes ?user yes]
                  [or [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-4 (:backward :certainty 1.0 :importance 83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [or   [has-cheese ?user yes]
                        [is-pasta ?user yes]]
                  [or [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-5 (:backward :certainty 1.0 :importance 83)
  if [and   [is-pasta ?user yes]
                  [or [use-bay-leaf ?user yes]
```

```
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-6 (:backward :certainty 1.0 :importance 83)
  if [and   [has-cheese ?user yes]
                  [or [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-7 (:backward :certainty 1.0 :importance 83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [or   [has-cheese ?user yes]
                        [is-pasta ?user yes]]
                  [has-tomatoes ?user yes]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-8 (:backward :certainty 1.0 :importance 83)
  if [and   [is-pasta ?user yes]
                  [has-tomatoes ?user yes]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-9 (:backward :certainty 1.0 :importance 83)
  if [and   [has-cheese ?user yes]
                  [has-tomatoes ?user yes]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-10 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [or   [has-cheese ?user yes]
                        [is-pasta ?user yes]]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-11 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [is-pasta ?user yes]
                  [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-12 (:backward :certainty 1.0 :importance
83)
```

```
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                [has-cheese ?user yes]
                [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-13 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                [prefers-sweet-wines ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-14 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                [or   [has-cheese ?user yes]
                      [is-pasta ?user yes]]
                [has-tomatoes ?user yes]
                [or [use-bay-leaf ?user yes]
                      [use-parsley ?user yes]
                      [use-nutmeg ?user yes]
                      [use-chili ?user yes]
                      [use-curry ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-15 (:backward :certainty 1.0 :importance
83)
  if  [and  [is-pasta ?user yes]
                [has-tomatoes ?user yes]
                [or [use-bay-leaf ?user yes]
                      [use-parsley ?user yes]
                      [use-nutmeg ?user yes]
                      [use-chili ?user yes]
                      [use-curry ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-16 (:backward :certainty 1.0 :importance
83)
  if [and   [has-cheese ?user yes]
                [has-tomatoes ?user yes]
                [or [use-bay-leaf ?user yes]
                      [use-parsley ?user yes]
                      [use-nutmeg ?user yes]
                      [use-chili ?user yes]
                      [use-curry ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-17 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                [or   [has-cheese ?user yes]
                      [is-pasta ?user yes]]
                [or [use-bay-leaf ?user yes]
                      [use-parsley ?user yes]
```

```
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-18 (:backward :certainty 1.0 :importance
83)
  if [and   [is-pasta ?user yes]
                  [or [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-19 (:backward :certainty 1.0 :importance
83)
  if [and   [has-cheese ?user yes]
                  [or [use-bay-leaf ?user yes]
                        [use-parsley ?user yes]
                        [use-nutmeg ?user yes]
                        [use-chili ?user yes]
                        [use-curry ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-20 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [or   [has-cheese ?user yes]
                        [is-pasta ?user yes]]
                  [has-tomatoes ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-21 (:backward :certainty 1.0 :importance
83)
  if [and   [is-pasta ?user yes]
                  [has-tomatoes ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-22 (:backward :certainty 1.0 :importance
83)
  if [and   [has-cheese ?user yes]
                  [has-tomatoes ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-23 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [or   [has-cheese ?user yes]
                        [is-pasta ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-24 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
```

```
                        [lamb ?user yes]]
                  [is-pasta ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-25 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]
                  [has-cheese ?user yes]]
  then [wine-to-drink ?user cabernet-sauvignon])

(defrule cabernet-sauvignon-26 (:backward :certainty 1.0 :importance
83)
  if [and   [or [beef ?user yes]
                        [lamb ?user yes]]]
  then [wine-to-drink ?user cabernet-sauvignon])

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;;
                        ;;;
      ;;; Merlot
                  ;;;
      ;;;
                        ;;;
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule merlot-1 (:backward :certainty 1.0 :importance 96)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [or   [use-oregano ?user yes]
                              [use-basil ?user yes]
                              [use-nutmeg ?user yes]
                              [use-chili ?user yes]
                              [use-curry ?user yes]
                              [use-garlic ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user merlot])

(defrule merlot-2 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [or   [use-oregano ?user yes]
                              [use-basil ?user yes]
                              [use-nutmeg ?user yes]
                              [use-chili ?user yes]
                              [use-curry ?user yes]
                              [use-garlic ?user yes]]
```

```
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user merlot])

(defrule merlot-3 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [use-oregano ?user yes]
                              [use-basil ?user yes]
                              [use-nutmeg ?user yes]
                              [use-chili ?user yes]
                              [use-curry ?user yes]
                              [use-garlic ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user merlot])

(defrule merlot-4 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user merlot])

(defrule merlot-5 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user merlot])

(defrule merlot-6 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user merlot])

(defrule merlot-7 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [prefers-sweet-wines ?user yes]]
```

```
      then [wine-to-drink ?user merlot])

(defrule merlot-8 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [or   [use-oregano ?user yes]
                              [use-basil ?user yes]
                              [use-nutmeg ?user yes]
                              [use-chili ?user yes]
                              [use-curry ?user yes]
                              [use-garlic ?user yes]]]
      then [wine-to-drink ?user merlot])

(defrule merlot-9 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [or   [use-oregano ?user yes]
                              [use-basil ?user yes]
                              [use-nutmeg ?user yes]
                              [use-chili ?user yes]
                              [use-curry ?user yes]
                              [use-garlic ?user yes]]]
      then [wine-to-drink ?user merlot])

(defrule merlot-10 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [use-oregano ?user yes]
                              [use-basil ?user yes]
                              [use-nutmeg ?user yes]
                              [use-chili ?user yes]
                              [use-curry ?user yes]
                              [use-garlic ?user yes]]]
      then [wine-to-drink ?user merlot])

(defrule merlot-11 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
```

```
                              [use-bbq-sauce ?user yes]]]
      then [wine-to-drink ?user merlot])

(defrule merlot-12 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]]
      then [wine-to-drink ?user merlot])

(defrule merlot-13 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [beef ?user yes]
                              [gamebird ?user yess]
                              [lamb ?user yes]
                              [seafood ?user yes]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]]
      then [wine-to-drink ?user merlot])

(defrule merlot-14 (:backward :certainty 1.0 :importance 83)
      if    [or [beef ?user yes]
                  [gamebird ?user yess]
                  [lamb ?user yes]
                  [seafood ?user yes]]
      then [wine-to-drink ?user merlot])

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;;
                        ;;;
      ;;; Zinfandel
                  ;;;
      ;;;
                        ;;;
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defrule zinfandel-1 (:backward :certainty 1.0 :importance 95)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-2 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-tomatoes ?user yes]
```

```
                              [use-bbq-sauce ?user yes]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-3 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-4 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-5 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-6 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-7 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [prefers-sweet-wines ?user yes]]
```

```
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-8 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-9 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-10 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-11 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [has-cheese ?user yes]
                              [is-pasta ?user yes]]
                        [or   [has-tomatoes ?user yes]
                              [use-bbq-sauce ?user yes]]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-12 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                              [gamebird ?user yess]]
                        [or   [use-tarragon ?user yes]
                              [use-thyme ?user yes]
                              [use-clove ?user yes]
                              [use-garlic ?user yes]
                              [use-pepper ?user yes]]]
```

```
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-13 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                            [gamebird ?user yess]]
                      [or   [has-tomatoes ?user yes]
                            [use-bbq-sauce ?user yes]]]
      then [wine-to-drink ?user zinfandel])

(defrule zinfandel-14 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [pork ?user yes]
                            [gamebird ?user yess]]
                      [or   [has-cheese ?user yes]
                            [is-pasta ?user yes]]]
      then [wine-to-drink ?user zinfandel])

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;;
                        ;;;
      ;;; Sauvignon Blanc
                  ;;;
      ;;;
                        ;;;
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule sauvignon-blanc-1 (:backward :certainty 1.0 :importance 94)
      if    [and  [or [poultry ?user yes]
                            [seafood ?user yes]]
                      [is-pasta ?user yes]
                      [or   [use-garlic ?user yes]
                            [use-oregano ?user yes]
                            [use-black-pepper ?user yes]]
                      [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user sauvignon-blanc])

(defrule sauvignon-blanc-2 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                            [seafood ?user yes]]
                      [or   [use-garlic ?user yes]
                            [use-oregano ?user yes]
                            [use-black-pepper ?user yes]]
                      [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user sauvignon-blanc])

(defrule sauvignon-blanc-3 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                            [seafood ?user yes]]
                      [is-pasta ?user yes]
                      [or   [use-garlic ?user yes]
                            [use-oregano ?user yes]
                            [use-black-pepper ?user yes]]
                      [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user sauvignon-blanc])

(defrule sauvignon-blanc-4 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                            [seafood ?user yes]]
```

```
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user sauvignon-blanc])

(defrule sauvignon-blanc-5 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [seafood ?user yes]]
                        [is-pasta ?user yes]
                        [or   [use-garlic ?user yes]
                              [use-oregano ?user yes]
                              [use-black-pepper ?user yes]]]
      then [wine-to-drink ?user sauvignon-blanc])

(defrule sauvignon-blanc-6 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [seafood ?user yes]]
                        [or   [use-garlic ?user yes]
                              [use-oregano ?user yes]
                              [use-black-pepper ?user yes]]]
      then [wine-to-drink ?user sauvignon-blanc])

(defrule sauvignon-blanc-7 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [seafood ?user yes]]
                        [is-pasta ?user yes]
                        [or   [use-garlic ?user yes]
                              [use-oregano ?user yes]
                              [use-black-pepper ?user yes]]]
      then [wine-to-drink ?user sauvignon-blanc])

(defrule sauvignon-blanc-8 (:backward :certainty 1.0 :importance 83)
      if          [or [poultry ?user yes]
                        [seafood ?user yes]]
      then [wine-to-drink ?user sauvignon-blanc])


      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;;
                        ;;;
      ;;; Chardonnay
                  ;;;
      ;;;
                        ;;;
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defrule chardonnay-1 (:backward :certainty 1.0 :importance 93)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
                              [use-ginger ?user yes]
```

```
                              [use-caribbean ?user yes]]
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-2 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
                              [use-ginger ?user yes]
                              [use-caribbean ?user yes]]
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-3 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
                              [use-ginger ?user yes]
                              [use-caribbean ?user yes]]
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-4 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-5 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
                              [use-ginger ?user yes]
                              [use-caribbean ?user yes]]
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user chardonnay])
```

```
(defrule chardonnay-6 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-7 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]
                        [prefers-dry-wines ?user yes]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-8 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
                              [use-ginger ?user yes]
                              [use-caribbean ?user yes]]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-9 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
                              [use-ginger ?user yes]
                              [use-caribbean ?user yes]]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-10 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
```

```
                              [use-ginger ?user yes]
                              [use-caribbean ?user yes]]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-11 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-12 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [or   [use-mustard ?user yes]
                              [use-sage ?user yes]
                              [use-clove ?user yes]
                              [use-ginger ?user yes]
                              [use-caribbean ?user yes]]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-13 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [or   [use-light-sauce ?user yes]
                              [use-cream-sauce ?user yes]]]
      then [wine-to-drink ?user chardonnay])

(defrule chardonnay-14 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [pork ?user yes]
                              [seafood ?user yes]]
                        [has-tomatoes ?user no]
                        [has-cheese ?user yes]]
      then [wine-to-drink ?user chardonnay])


      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;;;
                        ;;;
      ;;; Riesling
                  ;;;
      ;;;
                        ;;;
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defrule riesling-desert (:backward :certainty 0.9 :importance 90)
      if    [meal ?user dessert]
      then [wine-to-drink ?user riesling])
```

```
(defrule riesling-desert-2 (:backward :certainty 1.0 :importance 91)
      if    [and  [meal ?user dessert]
                  [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])

(defrule riesling-1 (:backward :certainty 1.0 :importance 92)
      if    [and  [or [poultry ?user yes]
                        [shellfish ?user yes]]
                  [has-cheese ?user yes]
                  [use-light-sauce ?user yes]
                  [or   [use-dill ?user yes]
                        [use-sage ?user yes]
                        [use-clove ?user yes]
                        [use-ginger ?user yes]]
                  [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])

(defrule riesling-2 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                        [shellfish ?user yes]]
                  [use-light-sauce ?user yes]
                  [or   [use-dill ?user yes]
                        [use-sage ?user yes]
                        [use-clove ?user yes]
                        [use-ginger ?user yes]]
                  [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])

(defrule riesling-3 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                        [shellfish ?user yes]]
                  [has-cheese ?user yes]
                  [or   [use-dill ?user yes]
                        [use-sage ?user yes]
                        [use-clove ?user yes]
                        [use-ginger ?user yes]]
                  [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])

(defrule riesling-4 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                        [shellfish ?user yes]]
                  [has-cheese ?user yes]
                  [use-light-sauce ?user yes]
                  [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])

(defrule riesling-5 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                        [shellfish ?user yes]]
                  [or   [use-dill ?user yes]
                        [use-sage ?user yes]
                        [use-clove ?user yes]
                        [use-ginger ?user yes]]
                  [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])
```

```
(defrule riesling-6 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [shellfish ?user yes]]
                      [use-light-sauce ?user yes]
                      [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])

(defrule riesling-7 (:backward :certainty 1.0 :importance 83)
      if    [and  [or [poultry ?user yes]
                              [shellfish ?user yes]]
                      [has-cheese ?user yes]
                      [prefers-sweet-wines ?user yes]]
      then [wine-to-drink ?user riesling])


;;;;;;;;;;;;;;;;;;;;   End of File
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```