

WINEX: The WINe EXpert system

Universidad Veracruzana

May 13, 2019

Outline

- 1 Introducción
- 2 Módulo MAIN
- 3 Módulo QUESTION
- 4 Módulo WINE-QUESTIONS
- 5 Módulo RULES
- 6 Módulo CHOOSE-QUALITIES
- 7 Módulo WINES
- 8 Módulo PRINT-RESULTS
- 9 Ejemplo

- Selecciona el vino apropiado para acompañar una comida.
- Considera grados de certeza.
- Ejemplo codificado para CLIPS 6.0.
- Consta de 7 módulos compuestos por reglas, templates y funciones.
- Para correrlo se necesita ocupar los comandos (reset) y (run).

- Se considera la variedad de vinos con su respectivas características y la descripción de la comida.

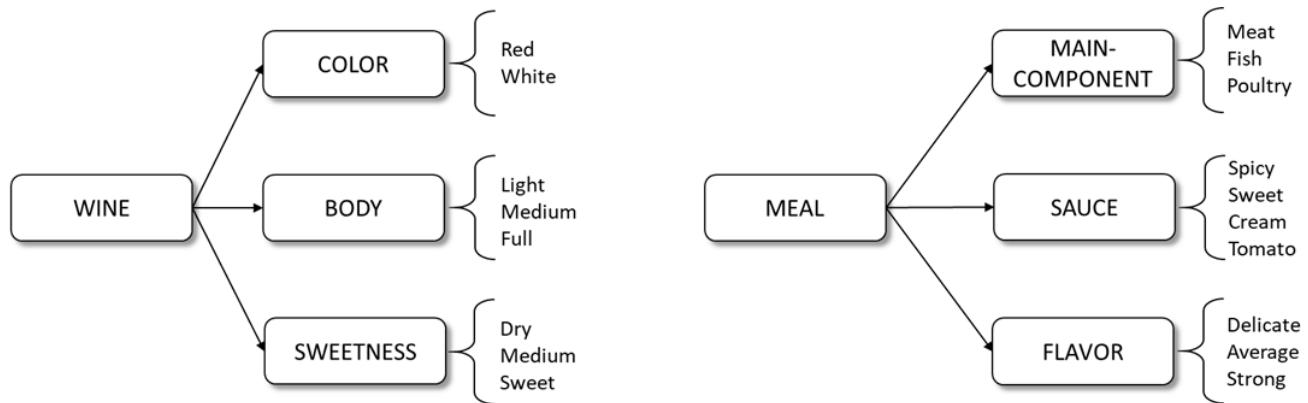


Figure 1: Universo de discurso.

Sintaxis: Módulos

```
1  (defmodule <module-name> [<comment>]
2    <port-specification>*)
3
4  port-specification> ::= (export <port-item>) |
5    (import <module-name> <port-item>)
6
7  <port-item> ::= ?ALL | ?NONE |
8    <port-construct> ?ALL | <port-construct> ?NONE |
9    <port-construct> <construct-name>+
10
11 <port-construct> ::= deftemplate | defclass |
12   defglobal | deffunction | defgeneric
```

Ejemplo: Módulos

```
1  (defmodule MAIN (export ?ALL))
```

Sintaxis: Plantillas

```
1  (deftemplate <defmodule-name>::<deftemplate-name>
2    [<comment>] <slot-definition>*)
3
4  <slot-definition> ::= <single-slot-definition> |
5    <multislot-definition>
6
7  <single-slot-definition> ::= (slot <slot-name>
8    <template-attribute>*)
9
10 <multislot-definition> ::= (multislot <slot-name>
11   <template-attribute>*)
12
13 <template-attribute> ::= <default-attribute>
14
15 <default-attribute> ::= (default ?DERIVE | ?NONE |
16   <expression>*)
```

Ejemplo: Plantillas

```
1 (deftemplate MAIN::attribute
2   (slot name)
3   (slot value)
4   (slot certainty (default 100.0)))
```


Sintaxis: Funciones

```
1  (deffunction <module-name>::<name> [<comment>]
2      (<regular-parameter>* [<wildcard-parameter>])
3      <action>*)
4
5  <regular-parameter> ::= <single-field-variable>
6
7  <wildcard-parameter> ::= <multifield-variable>
```

Ejemplo: Funciones

```
1  (deffunction MAIN::ask-question (?question ?allowed-values)
2    (printout t ?question)
3    (bind ?answer (read))
4    (if (lexemep ?answer)
5        then (bind ?answer (lowercase ?answer)))
6    (while (not (member ?answer ?allowed-values)) do
7        (printout t ?question)
8        (bind ?answer (read))
9        (if (lexemep ?answer)
10            then (bind ?answer (lowercase ?answer))))
11    ?answer)
```

Sintaxis: Reglas

```
1  (defrule <module-name>::<rule-name> [<comment>]
2      <conditional-element>* ; Left-Hand Side (LHS)
3  =>
4      <action>*) ; Right-Hand Side (RHS)
```

Ejemplo: Reglas

```
1 (defrule MAIN::start
2   (declare (salience 10000))
3   =>
4   (set-fact-duplication TRUE)
5   (focus QUESTIONS CHOOSE-QUALITIES WINES PRINT-RESULTS))
```

Sintaxis: Deffacts

```
1 (deffacts <deffacts-name> [<comment>]  
2   <RHS-pattern>*)
```

Ejemplo: Deffacts

```
1 (deffacts any-attributes
2   (attribute (name best-color) (value any))
3   (attribute (name best-body) (value any))
4   (attribute (name best-sweetness) (value any)))
```

Conformación del sistema

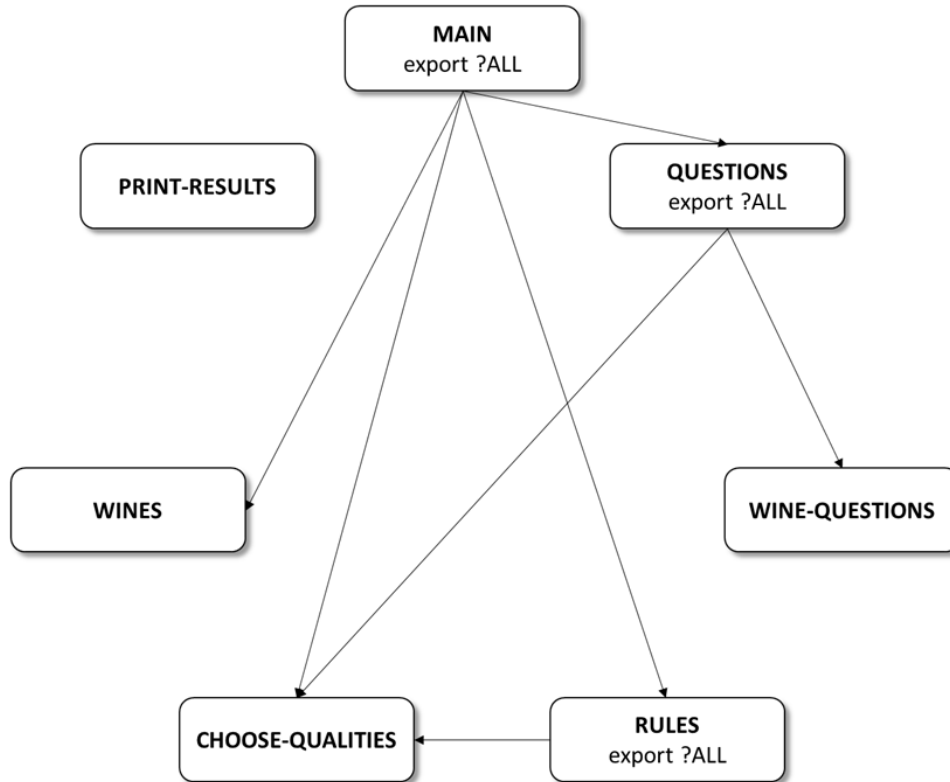


Figure 2: Dependencias de módulos.

```
1 (defmodule MAIN (export ?ALL))
```

- Módulo principal, compuesto por:
 - 1 función: ask-question.
 - 1 plantilla: attribute.
 - 2 reglas: start, combine_certainties.

Función ask-question

```
1  (deffunction MAIN::ask-question (?question ?allowed-values)
2    (printout t ?question)
3    (bind ?answer (read))
4    (if (lexemep ?answer)
5        then (bind ?answer (lowercase ?answer)))
6    (while (not (member ?answer ?allowed-values)) do
7        (printout t ?question)
8        (bind ?answer (read))
9        (if (lexemep ?answer)
10            then (bind ?answer (lowercase ?answer))))
11    ?answer)
```

Función que imprime la pregunta, verifica la entrada, asigna el valor a una variable y regresa el valor.

Plantilla: attribute

```
1 (deftemplate MAIN::attribute
2   (slot name)
3   (slot value)
4   (slot certainty (default 100.0)))
```

Plantilla que define las características de los vinos inferidas con un grado de certeza.

Regla: start

```
1 (defrule MAIN::start
2   (declare (salience 10000))
3   =>
4   (set-fact-duplication TRUE)
5   (focus QUESTIONS CHOOSE-QUALITIES WINES PRINT-RESULTS))
```

Regla que comienza el proceso, tiene una prioridad de 10,000, habilita el duplicado de hechos y define el focus para nuevos módulos.

Regla: combine_certainties

```
1 (defrule MAIN::combine-certainties ""
2   (declare (salience 100)
3     (auto-focus TRUE))
4   ?rem1 <- (attribute (name ?rel) (value ?val)
5     (certainty ?per1))
6   ?rem2 <- (attribute (name ?rel) (value ?val)
7     (certainty ?per2))
8   (test (neq ?rem1 ?rem2))
9   =>
10  (retract ?rem1)
11  (modify ?rem2 (certainty
12    (/ (- (* 100 (+ ?per1 ?per2)) (* ?per1 ?per2)) 100))))
```

Cuando dos atributos coinciden en el nombre y el valor, pero difieren en sus certezas, esta regla elimina el primero y modifica la certeza del segundo.

Módulo: QUESTIONS

```
1 (defmodule QUESTIONS (import MAIN ?ALL) (export ?ALL))
```

- Módulo con reglas para hacer preguntas, compuesto por:
 - 1 plantilla: question.
 - 3 reglas: ask-a-question, precursor-is-satisfied, precursor-is-not-satisfied.

Plantilla: question

```
1 (deftemplate QUESTIONS::question
2   (slot attribute (default ?NONE))
3   (slot the-question (default ?NONE))
4   (multislot valid-answers (default ?NONE))
5   (slot already-asked (default FALSE))
6   (multislot precursors (default ?DERIVE)))
```

Almacena el atributo por el cual pregunta, un string que representa la pregunta, sus respuestas válidas, si ya fue realizada y si tiene una pregunta como antecedente.

Regla: ask-a-question

```
1  (defrule QUESTIONS::ask-a-question
2      ?f <- (question (already-asked FALSE)
3                  (precursors)
4                  (the-question ?the-question)
5                  (attribute ?the-attribute)
6                  (valid-answers $?valid-answers))
7  =>
8      (modify ?f (already-asked TRUE))
9      (assert (attribute (name ?the-attribute)
10                      (value
11                      (ask-question ?the-question ?valid-answers)))))
```

Si existe una plantilla question que no ha sido realizada, modifica su slot already-asked a true y el slot value recibe el valor que regresa la función MAIN::ask-question.

Regla: precursor-is-satisfied

```
1 (defrule QUESTIONS::precursor-is-satisfied
2   ?f <- (question (already-asked FALSE)
3               (precursors ?name is ?value $?rest))
4   (attribute (name ?name) (value ?value))
5   =>
6   (if (eq (nth 1 ?rest) and)
7       then (modify ?f (precursors (rest$ ?rest)))
8       else (modify ?f (precursors ?rest))))
```

Pre-condiciones: existe una plantilla question con el slot already-asked FALSE y existe un atributo registrado como su antecedente que contenga el valor especificado.

Acción: si el primer campo del elemento es and, precursor toma el valor (rest\$?rest) y si no toma el valor de ?rest.

Regla: precursor-is-not-satisfied

```
1 (defrule QUESTIONS::precursor-is-not-satisfied
2   ?f <- (question (already-asked FALSE)
3     (precursors ?name is-not ?value $?rest))
4   (attribute (name ?name) (value ~?value))
5   =>
6   (if (eq (nth 1 ?rest) and)
7     then (modify ?f (precursors (rest$ ?rest)))
8     else (modify ?f (precursors ?rest))))
```

Pre-condiciones: existe una plantilla question con el slot already-asked FALSE y existe un atributo registrado como su antecedente que no contenga el valor especificado.

Acción: si el primer campo del elemento es and, precursor toma el valor (rest\$?rest) y si no toma el valor de ?rest.

Módulo: WINE-QUESTIONS

```
1 (defmodule WINE-QUESTIONS (import QUESTIONS ?ALL))
```

- Módulo con preguntas acerca de los vinos, compuesto por:
 - 1 deffact: question-attributes.

Defacts: question-attributes

```
1 (defacts WINE-QUESTIONS::question-attributes
2   (question (attribute main-component)
3             (the-question ''Is the main component of
4             the meal meat, fish, or poultry?''))
5             (valid-answers meat fish poultry unknown))
6   (question (attribute has-turkey)
7             .
8             .
9             .
10  (question (attribute preferred-sweetness)
11            (the-question ''Do you generally prefer dry,
12            medium, or sweet wines?''))
13            (valid-answers dry medium sweet unknown)))
```

Define plantillas question con las preguntas que son necesarias para generar inferencias.

```
1 (defmodule RULES (import MAIN ?ALL) (export ?ALL))
```

- Módulo con reglas para inferir información de los template rule, compuesto por:
 - 1 plantilla: rule.
 - 6 reglas: throw-away-and-in-consequent,
throw-away-and-in-antecedent,
remove-is-not-condition-when-satisfied,
remove-is-condition-when-satisfied,
perform-rule-consequent-with-certainty,
perform-rule-consequent-without-certainty.

Plantilla: rule

```
1 (deftemplate RULES::rule
2   (slot certainty (default 100.0))
3   (multislot if)
4   (multislot then))
```

Representa una regla, el slot if es el antecedente, el slot then es el consecuente y el slot certain es la certeza.

Rule: throw-away-and-in-antecedent

```
1 (defrule RULES::throw-away-and-in-antecedent
2   ?f <- (rule (if and $?rest))
3   =>
4   (modify ?f (if ?rest)))
```

Si una platilla rule tiene el símbolo and en el primer campo del slot if, se elimina.

Rule: throw-away-and-in-consequent

```
1 (defrule RULES::throw-away-and-in-consequent
2   ?f <- (rule (then and $?rest))
3   =>
4   (modify ?f (then ?rest)))
```

Si una platilla rule tiene el símbolo and en el primer campo del slot then, se elimina.

Rule: remove-is-condition-when-satisfied

```
1 (defrule RULES::remove-is-condition-when-satisfied
2   ?f <- (rule (certainty ?c1)
3             (if ?attribute is ?value $?rest))
4   (attribute (name ?attribute)
5             (value ?value)
6             (certainty ?c2))
7   =>
8   (modify ?f (certainty (min ?c1 ?c2)) (if ?rest)))
```

Si una plantilla rule tiene el mismo atributo y valor que una plantilla attribute, el valor de certeza de rule es modificado con el valor más pequeño.

Rule: remove-is-not-condition-when-satisfied

```
1 (defrule RULES::remove-is-not-condition-when-satisfied
2   ?f <- (rule (certainty ?c1)
3             (if ?attribute is-not ?value $?rest))
4   (attribute (name ?attribute) (value ~?value)
5   (certainty ?c2))
6   =>
7   (modify ?f (certainty (min ?c1 ?c2)) (if ?rest)))
```

Si una plantilla attribute tiene el mismo atributo y un valor distinto a la condición is-not de una plantilla rule, el valor de certeza de rule es modificado con el valor más pequeño.

Rule: perform-rule-consequent-with-certainty

```
1 (defrule RULES::perform-rule-consequent-with-certainty
2   ?f <- (rule (certainty ?c1)
3     (if)
4     (then ?attribute is ?value with certainty ?c2 $?rest))
5   =>
6   (modify ?f (then ?rest))
7   (assert (attribute (name ?attribute)
8     (value ?value)
9     (certainty (/ (* ?c1 ?c2) 100)))))
```

Si una platilla rule no tiene información en el slot if, se agrega un hecho con el primer atributo que incluye certeza en el slot then y la certeza del nuevo hecho se calcula como la multiplicación de ambas certezas divididas entre 100.

Rule: perform-rule-consequent-without-certainty

```
1 (defrule RULES::perform-rule-consequent-without-certainty
2   ?f <- (rule (certainty ?c1)
3     (if)
4     (then ?attribute is ?value $?rest))
5   (test (or (eq (length$ ?rest) 0)
6     (neq (nth 1 ?rest) with)))
7   =>
8   (modify ?f (then ?rest))
9   (assert (attribute (name ?attribute) (value ?value)
10     (certainty ?c1))))
```

Si una platilla rule no tiene información en el slot if, se agrega un hecho con el primer atributo que no incluye certeza en el slot then y la certeza del nuevo hecho se calcula como la multiplicación de ambas certezas divididas entre 100.

Módulo: CHOOSE-QUALITIES

```
1 (defmodule CHOOSE-QUALITIES (import RULES ?ALL)
2                               (import QUESTIONS ?ALL)
3                               (import MAIN ?ALL))
```

- Módulo que define reglas para la inferencias de vinos, compuesto por:
 - 1 deffacts: wine-rules.
 - 7 reglas para seleccionar el mejor cuerpo.
 - 9 reglas para seleccionar el mejor color.
 - 11 reglas para seleccionar el mejor sabor.

Reglas para seleccionar el mejor cuerpo

```
1      (rule (if has-sauce is yes and
2              sauce is spicy)
3              (then best-body is full))
4
5      (rule (if tastiness is delicate)
6              (then best-body is light))
```

Si la comida tiene salsa y la salsa es picante, el mejor cuerpo es completo.

Si el sabor de la comida es delicado, el mejor cuerpo es ligero.

Reglas para seleccionar el mejor color

```
1      (rule (if main-component is meat)
2            (then best-color is red with certainty 90))
3
4      (rule (if main-component is poultry and
5            has-turkey is no)
6            (then best-color is white with certainty 90 and
7                  best-color is red with certainty 30))
```

Si el componente principal de la comida es carne, entonces el mejor color es rojo con certeza del 90.

Si el principal componente de la comida es un ave de corral que no sea pavo, el mejor color es rojo con certeza del 30.

Reglas para seleccionar el mejor sabor

```
1      (rule (if preferred-sweetness is dry)
2            (then best-sweetness is dry with certainty 40))
3
4      (rule (if preferred-sweetness is medium)
5            (then best-sweetness is medium with certainty 40))
```

Si la dulzura preferida es del tipo seco, el mejor tipo de dulzura es seco con certeza del 40.

Si la dulzura preferida es media, el mejor tipo de dulzura es media con certeza del 40.

```
1 (defmodule WINES (import MAIN ?ALL))
```

- Módulo que define vinos y sus características preferidas, compuesto por:
 - 2 deffacts: any-attributes, the-wine-list.
 - 1 template: wine.
 - 1 regla: generate-wines

Plantilla: wine

```
1 (deftemplate WINES::wine
2   (slot name (default ?NONE))
3   (multislot color (default any))
4   (multislot body (default any))
5   (multislot sweetness (default any)))
```

Representa un vino junto con sus características color, cuerpo y dulzura.

Deffacts: any-attributes

```
1 (deffacts any-attributes
2   (attribute (name best-color) (value any))
3   (attribute (name best-body) (value any))
4   (attribute (name best-sweetness) (value any)))
```

Define plantillas attribute con valores any.

Deffacts: the-wine-list

```
1 (deffacts WINES::the-wine-list
2   (wine (name Geverztraminer) (color white) (body full))
3   (wine (name Valpolicella) (color red) (body light))
4   .
5   .
6   .
7   (wine (name Burgundy) (color red) (body full))
8   (wine (name Zinfandel) (color red)
9     (sweetness dry medium)))
```

Define plantillas wine con distintos tipos de vino.

Rule: generate-wines

```
1      (wine (name ?name)
2            (color $? ?c $?)
3            (body $? ?b $?)
4            (sweetness $? ?s $?))
5      (attribute (name best-color) (value ?c)
6                (certainty ?certainty-1))
7      (attribute (name best-body) (value ?b)
8                (certainty ?certainty-2))
9      (attribute (name best-sweetness)
10               (value ?s) (certainty ?certainty-3))
11     =>
12     (assert (attribute (name wine) (value ?name)
13                       (certainty
14                         (min ?certainty-1 ?certainty-2 ?certainty-3))))
```

Si una platilla vino concuerda en valores con atributos best-color, best-body, best-sweetness se agrega un hecho con el vino y la certeza más pequeña.

Módulo: PRINT-RESULTS

```
1 (defmodule PRINT-RESULTS (import MAIN ?ALL))
```

- Módulo que imprime las inferencias, compuesto por:
 - 4 reglas: header, print-wines, remove-poor-wine-choices, end-spaces.

Rule: header

```
1 (defrule PRINT-RESULTS::header ""
2   (declare (salience 10))
3   =>
4   (printout t t)
5   (printout t ''           SELECTED WINES'' t t)
6   (printout t '' WINE           CERTAINTY'' t)
7   (printout t '' -----'' t)
8   (assert (phase print-wines)))
```

Imprime un encabezado antes de presentar la información inferida.

Rule: print-wines

```
1 (defrule PRINT-RESULTS::print-wine ""
2   ?rem <- (attribute (name wine) (value ?name)
3     (certainty ?per))
4   (not (attribute (name wine)
5     (certainty ?per1&:(> ?per1 ?per))))
6   =>
7   (retract ?rem)
8   (format t ' ' %-24s %2d%%n' ' ?name ?per))
```

Imprime los vinos de forma descendente de acuerdo a su certeza, y elimina las plantillas attribute de los mismos.

Rule: remove-poor-wine-choices

```
1 (defrule PRINT-RESULTS::remove-poor-wine-choices ""
2   ?rem <- (attribute (name wine)
3     (certainty ?per&:(< ?per 20)))
4   =>
5   (retract ?rem))
```

Elimina todas las plantillas que contengan certezas menores a 20.

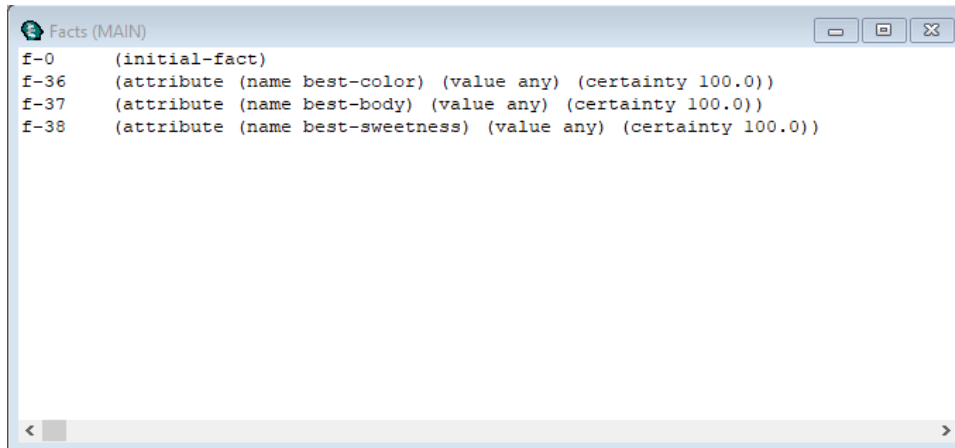
Rule: end-spaces

```
1 (defrule PRINT-RESULTS::end-spaces ""
2   (not (attribute (name wine)))
3   =>
4   (printout t t))
```

Imprime un espacio al final cuando ya no existen atributos tipo vino.

Ejemplo

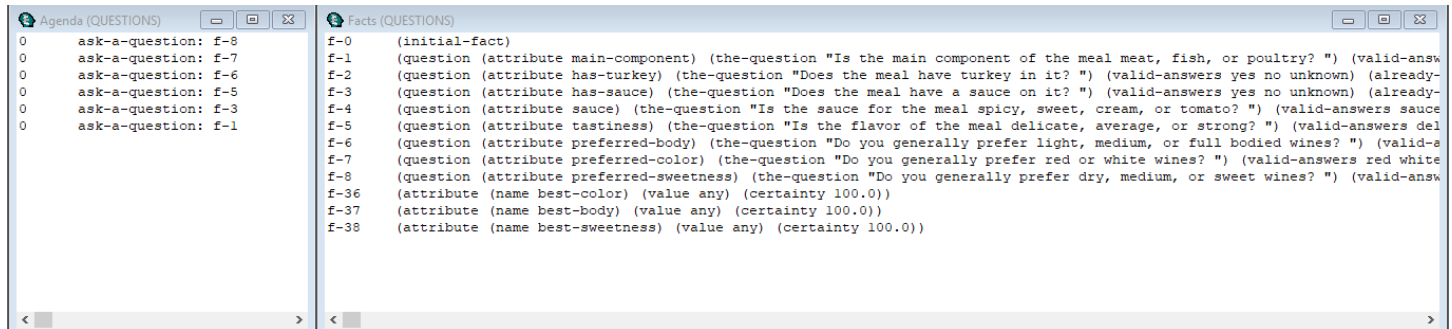
- Para correr el sistema experto sólo es necesario usar los comandos (reset) y (run).



```
Facts (MAIN)
f-0      (initial-fact)
f-36     (attribute (name best-color) (value any) (certainty 100.0))
f-37     (attribute (name best-body) (value any) (certainty 100.0))
f-38     (attribute (name best-sweetness) (value any) (certainty 100.0))
```

Ejemplo

- Focus-stack: QUESTIONS, CHOOSE-QUALITIES, WINES, PRINT-RESULTS y MAIN.
- La agenda se satura con reglas del módulo QUESTIONS.



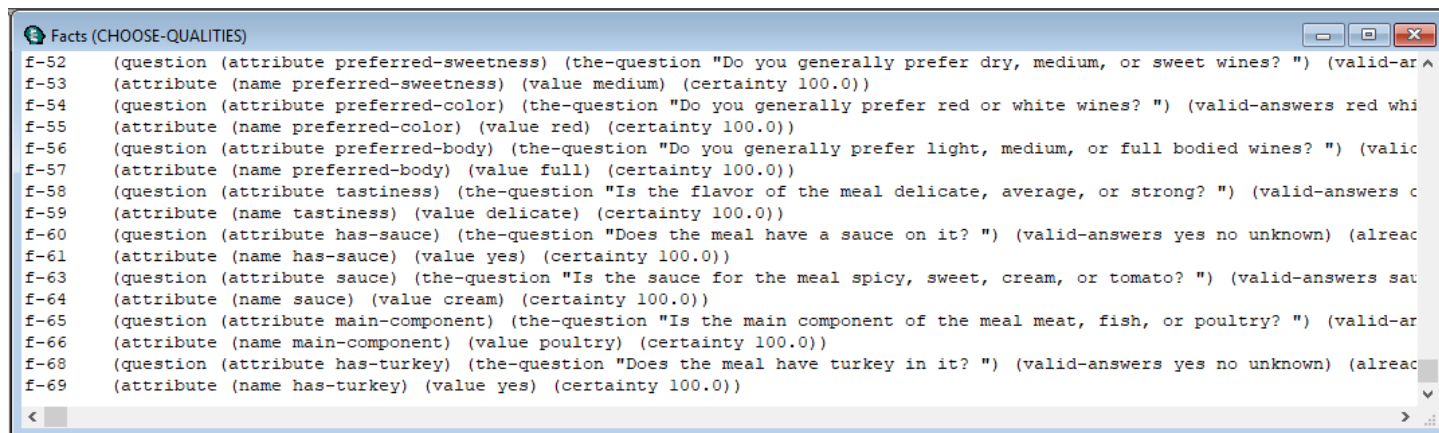
The screenshot shows two Prolog windows. The left window, titled 'Agenda (QUESTIONS)', displays a list of five 'ask-a-question' goals, each associated with a fact identifier (f-8, f-7, f-6, f-5, f-3, f-1). The right window, titled 'Facts (QUESTIONS)', displays a list of 38 facts, including initial facts, questions about meal components and attributes, and attribute values for best-color and best-sweetness.

```
Agenda (QUESTIONS)
0 ask-a-question: f-8
0 ask-a-question: f-7
0 ask-a-question: f-6
0 ask-a-question: f-5
0 ask-a-question: f-3
0 ask-a-question: f-1

Facts (QUESTIONS)
f-0 (initial-fact)
f-1 (question (attribute main-component) (the-question "Is the main component of the meal meat, fish, or poultry? ") (valid-answ
f-2 (question (attribute has-turkey) (the-question "Does the meal have turkey in it? ") (valid-answers yes no unknown) (already-
f-3 (question (attribute has-sauce) (the-question "Does the meal have a sauce on it? ") (valid-answers yes no unknown) (already-
f-4 (question (attribute sauce) (the-question "Is the sauce for the meal spicy, sweet, cream, or tomato? ") (valid-answers sauce
f-5 (question (attribute tastiness) (the-question "Is the flavor of the meal delicate, average, or strong? ") (valid-answers del
f-6 (question (attribute preferred-body) (the-question "Do you generally prefer light, medium, or full bodied wines? ") (valid-a
f-7 (question (attribute preferred-color) (the-question "Do you generally prefer red or white wines? ") (valid-answers red white
f-8 (question (attribute preferred-sweetness) (the-question "Do you generally prefer dry, medium, or sweet wines? ") (valid-answ
f-36 (attribute (name best-color) (value any) (certainty 100.0))
f-37 (attribute (name best-body) (value any) (certainty 100.0))
f-38 (attribute (name best-sweetness) (value any) (certainty 100.0))
```

Ejemplo

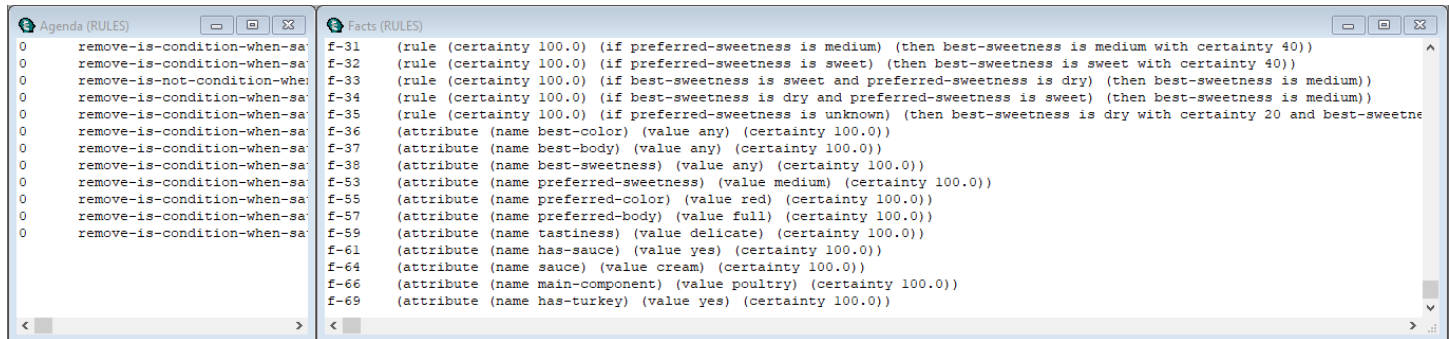
- Focus-stack: CHOOSE-QUALITIES, WINES, PRINT-RESULTS y MAIN.
- Se agregan plantillas **attributes** cuando una pregunta se responde y se modifican plantillas **question**.



```
f-52 (question (attribute preferred-sweetness) (the-question "Do you generally prefer dry, medium, or sweet wines? ") (valid-answers dry medium sweet))
f-53 (attribute (name preferred-sweetness) (value medium) (certainty 100.0))
f-54 (question (attribute preferred-color) (the-question "Do you generally prefer red or white wines? ") (valid-answers red white))
f-55 (attribute (name preferred-color) (value red) (certainty 100.0))
f-56 (question (attribute preferred-body) (the-question "Do you generally prefer light, medium, or full bodied wines? ") (valid-answers light medium full))
f-57 (attribute (name preferred-body) (value full) (certainty 100.0))
f-58 (question (attribute tastiness) (the-question "Is the flavor of the meal delicate, average, or strong? ") (valid-answers delicate average strong))
f-59 (attribute (name tastiness) (value delicate) (certainty 100.0))
f-60 (question (attribute has-sauce) (the-question "Does the meal have a sauce on it? ") (valid-answers yes no unknown) (already-asked no))
f-61 (attribute (name has-sauce) (value yes) (certainty 100.0))
f-63 (question (attribute sauce) (the-question "Is the sauce for the meal spicy, sweet, cream, or tomato? ") (valid-answers spicy sweet cream tomato))
f-64 (attribute (name sauce) (value cream) (certainty 100.0))
f-65 (question (attribute main-component) (the-question "Is the main component of the meal meat, fish, or poultry? ") (valid-answers meat fish poultry))
f-66 (attribute (name main-component) (value poultry) (certainty 100.0))
f-68 (question (attribute has-turkey) (the-question "Does the meal have turkey in it? ") (valid-answers yes no unknown) (already-asked no))
f-69 (attribute (name has-turkey) (value yes) (certainty 100.0))
```

Ejemplo

- Focus-stack: RULES, CHOOSE-QUALITIES, WINES, PRINT-RESULTS y MAIN.
- Se satura la agenda con reglas del módulo RULES.



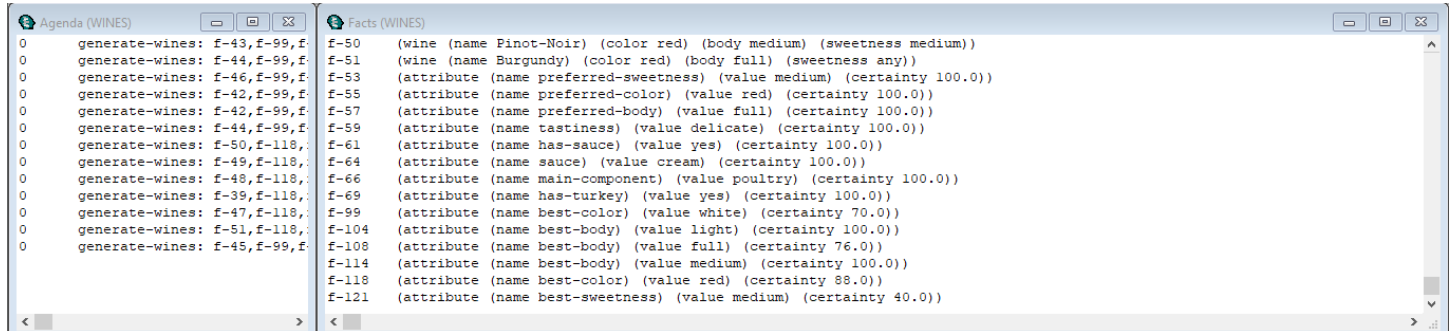
The screenshot shows two windows from a Prolog environment. The left window, titled 'Agenda (RULES)', displays a list of 10 identical rules: 'remove-is-condition-when-sa'. The right window, titled 'Facts (RULES)', displays a list of 15 facts (f-31 to f-69) defining rules and attributes for wine quality. The facts include rules for sweetness levels based on preferred sweetness and attributes for wine characteristics like body, color, and taste.

```
Agenda (RULES)
0  remove-is-condition-when-sa
0  remove-is-condition-when-sa
0  remove-is-not-condition-when-sa
0  remove-is-condition-when-sa
0  remove-is-condition-when-sa
0  remove-is-condition-when-sa
0  remove-is-condition-when-sa
0  remove-is-condition-when-sa
0  remove-is-condition-when-sa
0  remove-is-condition-when-sa

Facts (RULES)
f-31  (rule (certainty 100.0) (if preferred-sweetness is medium) (then best-sweetness is medium with certainty 40))
f-32  (rule (certainty 100.0) (if preferred-sweetness is sweet) (then best-sweetness is sweet with certainty 40))
f-33  (rule (certainty 100.0) (if best-sweetness is sweet and preferred-sweetness is dry) (then best-sweetness is medium))
f-34  (rule (certainty 100.0) (if best-sweetness is dry and preferred-sweetness is sweet) (then best-sweetness is medium))
f-35  (rule (certainty 100.0) (if preferred-sweetness is unknown) (then best-sweetness is dry with certainty 20 and best-sweetne
f-36  (attribute (name best-color) (value any) (certainty 100.0))
f-37  (attribute (name best-body) (value any) (certainty 100.0))
f-38  (attribute (name best-sweetness) (value any) (certainty 100.0))
f-53  (attribute (name preferred-sweetness) (value medium) (certainty 100.0))
f-55  (attribute (name preferred-color) (value red) (certainty 100.0))
f-57  (attribute (name preferred-body) (value full) (certainty 100.0))
f-59  (attribute (name tastiness) (value delicate) (certainty 100.0))
f-61  (attribute (name has-sauce) (value yes) (certainty 100.0))
f-64  (attribute (name sauce) (value cream) (certainty 100.0))
f-66  (attribute (name main-component) (value poultry) (certainty 100.0))
f-69  (attribute (name has-turkey) (value yes) (certainty 100.0))
```

Ejemplo

- Focus-stack: WINES, PRINT-RESULTS y MAIN.
- Cuando no hay más reglas de producción del módulo RULES que aplicar, ya se han generado una serie de inferencias sobre qué características de los vinos son mejores con sus certezas.

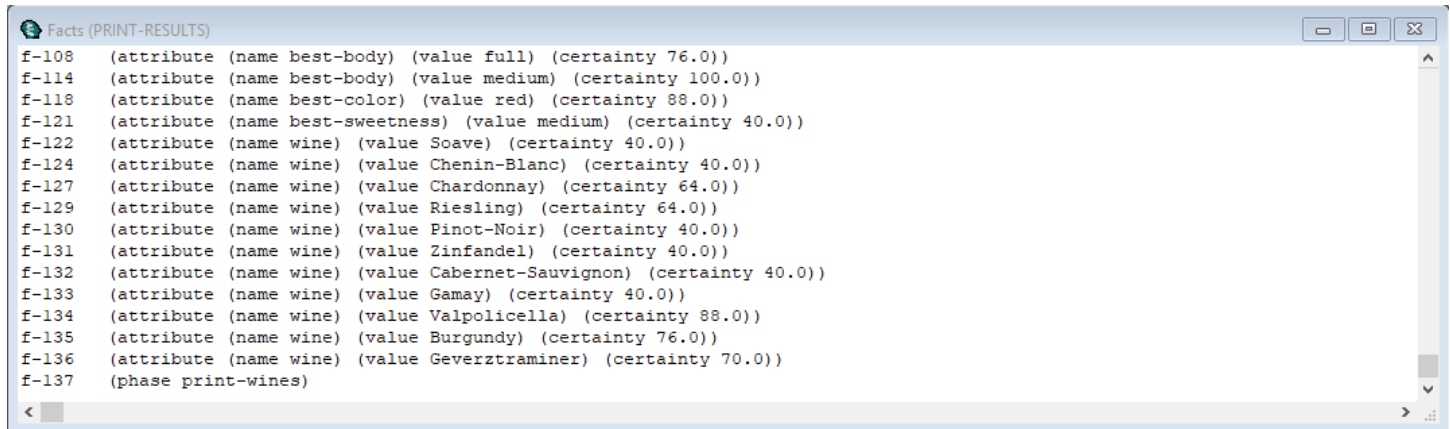


The screenshot displays two windows from a Prolog interpreter:

- Agenda (WINES):** Shows a list of goals to be executed, each preceded by a line number (0). The goals are: `generate-wines: f-43,f-99,f`, `generate-wines: f-44,f-99,f`, `generate-wines: f-46,f-99,f`, `generate-wines: f-42,f-99,f`, `generate-wines: f-42,f-99,f`, `generate-wines: f-44,f-99,f`, `generate-wines: f-50,f-118,`, `generate-wines: f-49,f-118,`, `generate-wines: f-48,f-118,`, `generate-wines: f-39,f-118,`, `generate-wines: f-47,f-118,`, and `generate-wines: f-51,f-118,`. The last line is `generate-wines: f-45,f-99,f`.
- Facts (WINES):** Shows a list of facts, each preceded by a line number (f-50 to f-121). The facts are: `(wine (name Pinot-Noir) (color red) (body medium) (sweetness medium))`, `(wine (name Burgundy) (color red) (body full) (sweetness any))`, `(attribute (name preferred-sweetness) (value medium) (certainty 100.0))`, `(attribute (name preferred-color) (value red) (certainty 100.0))`, `(attribute (name preferred-body) (value full) (certainty 100.0))`, `(attribute (name tastiness) (value delicate) (certainty 100.0))`, `(attribute (name has-sauce) (value yes) (certainty 100.0))`, `(attribute (name sauce) (value cream) (certainty 100.0))`, `(attribute (name main-component) (value poultry) (certainty 100.0))`, `(attribute (name has-turkey) (value yes) (certainty 100.0))`, `(attribute (name best-color) (value white) (certainty 70.0))`, `(attribute (name best-body) (value light) (certainty 100.0))`, `(attribute (name best-body) (value full) (certainty 76.0))`, `(attribute (name best-body) (value medium) (certainty 100.0))`, `(attribute (name best-color) (value red) (certainty 88.0))`, and `(attribute (name best-sweetness) (value medium) (certainty 40.0))`.

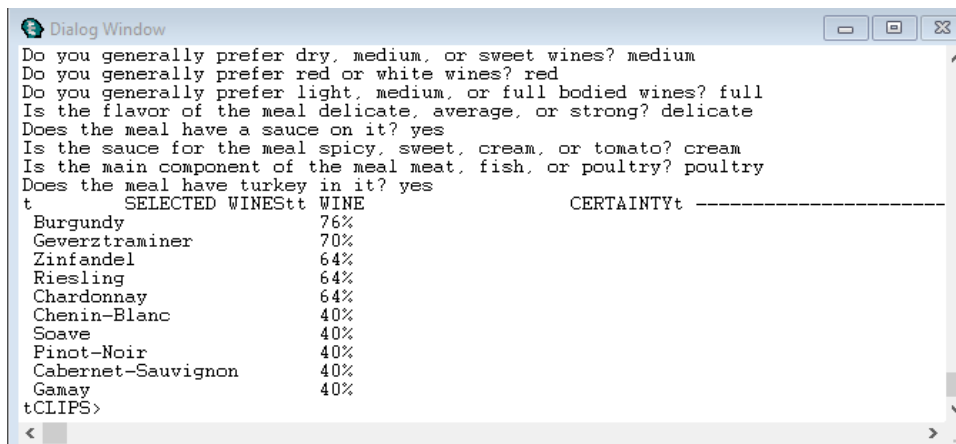
Ejemplo

- Focus-stack: PRINT-RESULTS y MAIN.
- Una vez generados los vinos, el focus se le otorga a PRINT-RESULTS para desplegar la salida.



```
Facts (PRINT-RESULTS)
f-108 (attribute (name best-body) (value full) (certainty 76.0))
f-114 (attribute (name best-body) (value medium) (certainty 100.0))
f-118 (attribute (name best-color) (value red) (certainty 88.0))
f-121 (attribute (name best-sweetness) (value medium) (certainty 40.0))
f-122 (attribute (name wine) (value Soave) (certainty 40.0))
f-124 (attribute (name wine) (value Chenin-Blanc) (certainty 40.0))
f-127 (attribute (name wine) (value Chardonnay) (certainty 64.0))
f-129 (attribute (name wine) (value Riesling) (certainty 64.0))
f-130 (attribute (name wine) (value Pinot-Noir) (certainty 40.0))
f-131 (attribute (name wine) (value Zinfandel) (certainty 40.0))
f-132 (attribute (name wine) (value Cabernet-Sauvignon) (certainty 40.0))
f-133 (attribute (name wine) (value Gamay) (certainty 40.0))
f-134 (attribute (name wine) (value Valpolicella) (certainty 88.0))
f-135 (attribute (name wine) (value Burgundy) (certainty 76.0))
f-136 (attribute (name wine) (value Geverztraminer) (certainty 70.0))
f-137 (phase print-wines)
```

- Resultados.



```
Dialog Window
Do you generally prefer dry, medium, or sweet wines? medium
Do you generally prefer red or white wines? red
Do you generally prefer light, medium, or full bodied wines? full
Is the flavor of the meal delicate, average, or strong? delicate
Does the meal have a sauce on it? yes
Is the sauce for the meal spicy, sweet, cream, or tomato? cream
Is the main component of the meal meat, fish, or poultry? poultry
Does the meal have turkey in it? yes
t      SELECTED WINEStt WINE      CERTAINTYt -----
t      Burgundy      76%
t      Gevertztraminer 70%
t      Zinfandel      64%
t      Riesling       64%
t      Chardonnay     64%
t      Chenin-Blanc   40%
t      Soave          40%
t      Pinot-Noir     40%
t      Cabernet-Sauvignon 40%
t      Gamay          40%
tCLIPS>
```




Culbert, C., Riley, G., & Donnell, B. (2015). CLIPS–Reference Manual Volume I Basic Programming Guide (Version 6.30).