CLIPS

BASICS

Downloading CLIPS

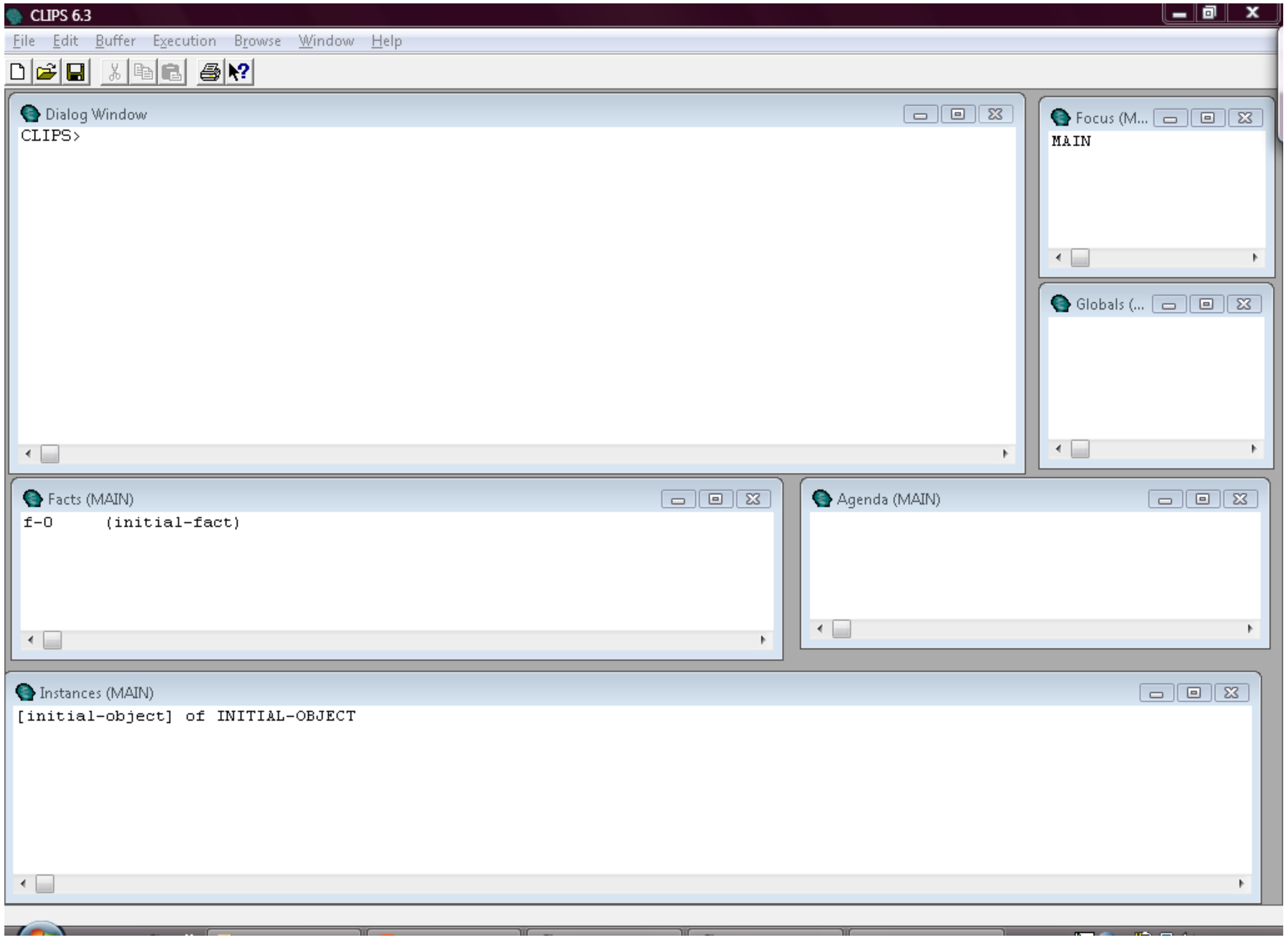
- Go to http://clipsrules.sourceforge.net/ and download

- Documentation available
    - **User's Guide.**
    - Basic Programming Guide.
    - Advanced Programming Guide.

Basic Elements

- CLIPS shell provides:
  - **Fact-list and intance-list:** Global memory for data.
  - **Knowledge-base:** Contains all the rules.
  - **Inference engine:** Controls overall execution of rules.

CLIPS Interface Example Windows



```
CLIPS>
```

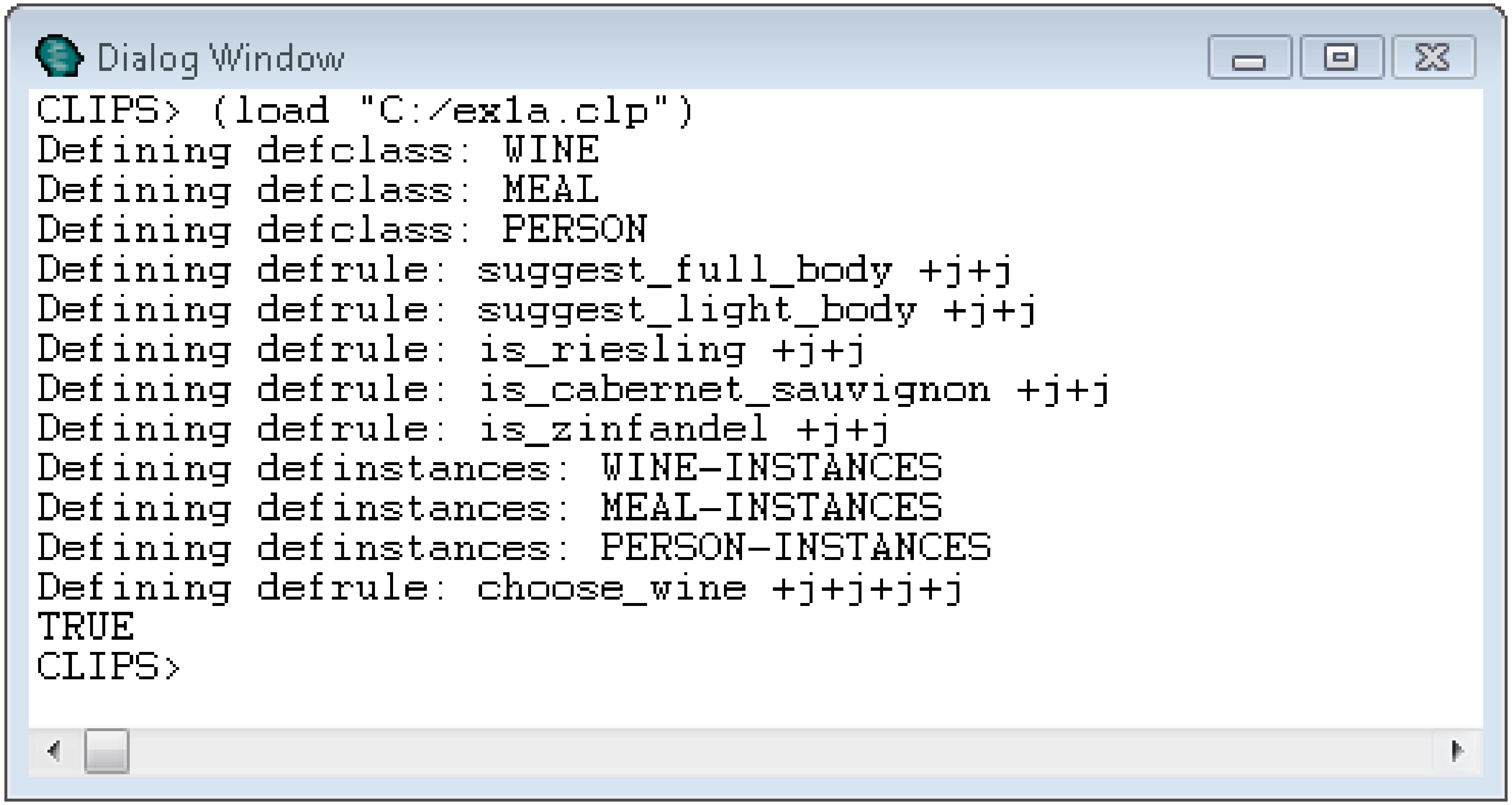
```
MAIN
```

```
f-0     (initial-fact)
```

```
[initial-object] of INITIAL-OBJECT
```

CLIPS Interface Example

Click File->Load and select the file “C:\ex1a.CLP”

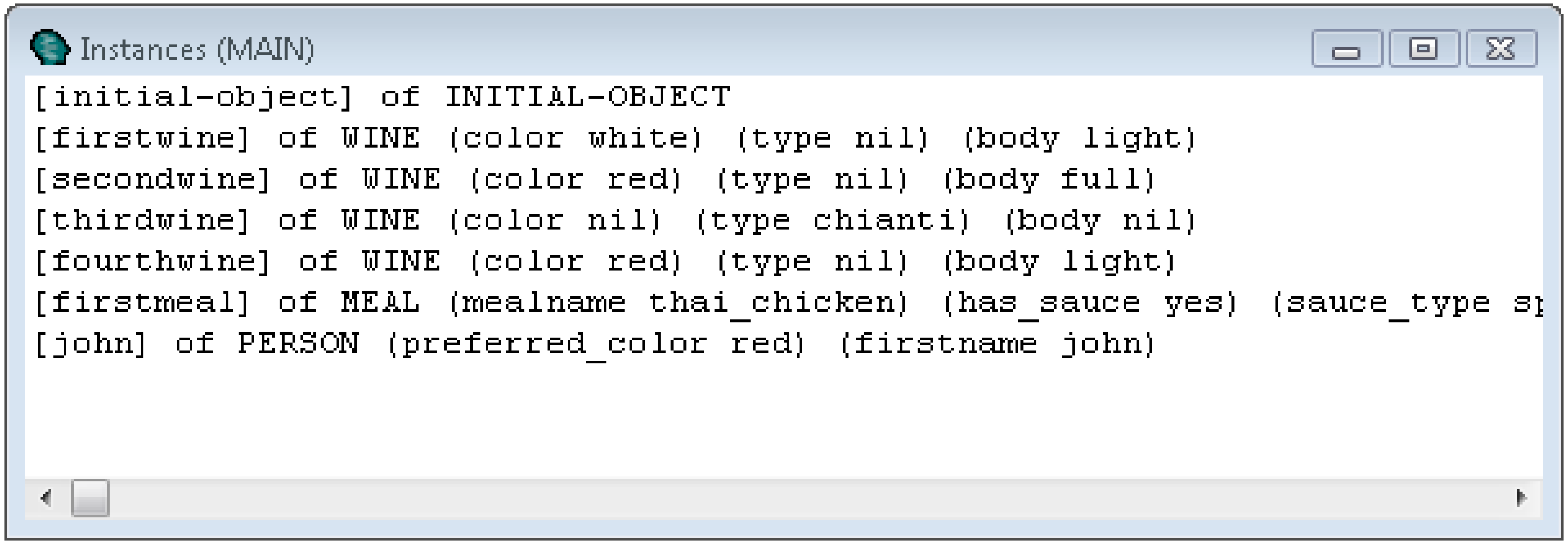


```
CLIPS> (load "C:/ex1a.clp")
Defining defclass: WINE
Defining defclass: MEAL
Defining defclass: PERSON
Defining defrule: suggest_full_body +j+j
Defining defrule: suggest_light_body +j+j
Defining defrule: is_riesling +j+j
Defining defrule: is_cabernet_sauvignon +j+j
Defining defrule: is_zinfandel +j+j
Defining definstances: WINE-INSTANCES
Defining definstances: MEAL-INSTANCES
Defining definstances: PERSON-INSTANCES
Defining defrule: choose_wine +j+j+j+j
TRUE
CLIPS>
```

CLIPS Interface Example

Click Execution->Reset or type (reset)

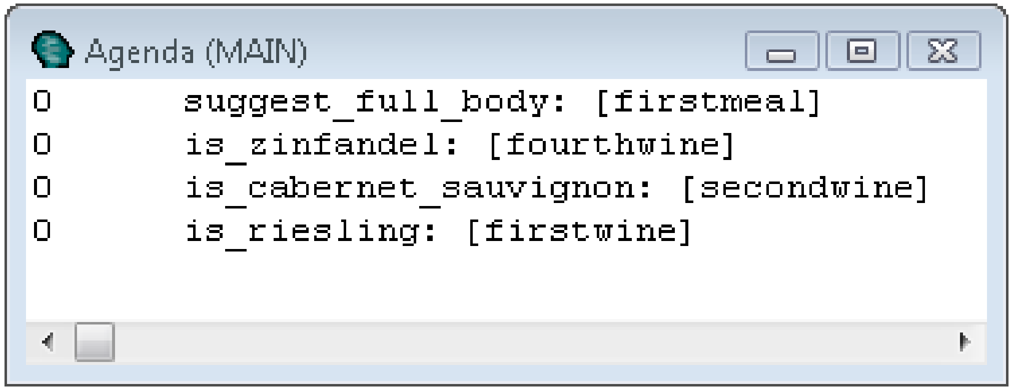
Type “(instances) [Enter]” to see the defined instances or click Window->Instances Window to open a window displaying the instances



```
[initial-object] of INITIAL-OBJECT
[firstwine] of WINE (color white) (type nil) (body light)
[secondwine] of WINE (color red) (type nil) (body full)
[thirdwine] of WINE (color nil) (type chianti) (body nil)
[fourthwine] of WINE (color red) (type nil) (body light)
[firstmeal] of MEAL (mealname thai_chicken) (has_sauce yes) (sauce_type s
[john] of PERSON (preferred_color red) (firstname john)
```

CLIPS Interface Example

Type “(agenda) [Enter]” to see the current rules agenda, or click Window->Agenda Window to open a window displaying the current agenda



```
0       suggest_full_body: [firstmeal]
0       is_zinfandel: [fourthwine]
0       is_cabernet_sauvignon: [secondwine]
0       is_riesling: [firstwine]
```

Things You Should Know.

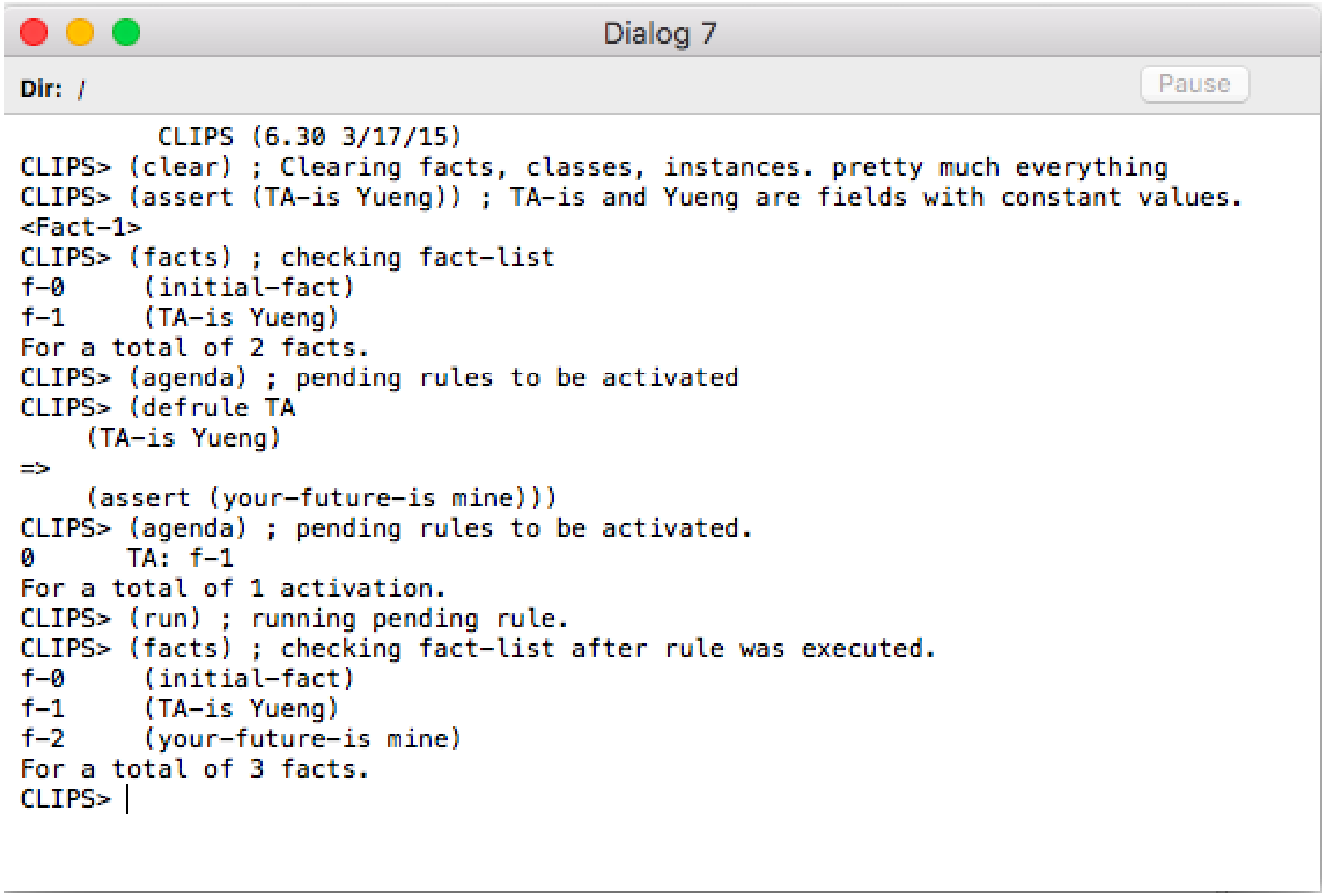
- **All statements have to be inside parentheses.**
  - (assert), (exit), (clear), (reset), etc.
- **A field is a placeholder that may have a value associated with it**.
  - A fact consists of one or more fields.
  - (field), (field1 field2), (field1 field2 … fieldn)
  - (Yueng), (Yueng Delahoz), etc. Order matters.
  - Types of fields:
    - **Float, integer, symbol, string, external-address, fact-address, instance-name and instance-address.**
- **The comment begins with a semicolon.**
  - *; hey! Whatup. Je suis un commentaire.*

How to: Create rules.

Keep IF THEN analogy in mind.

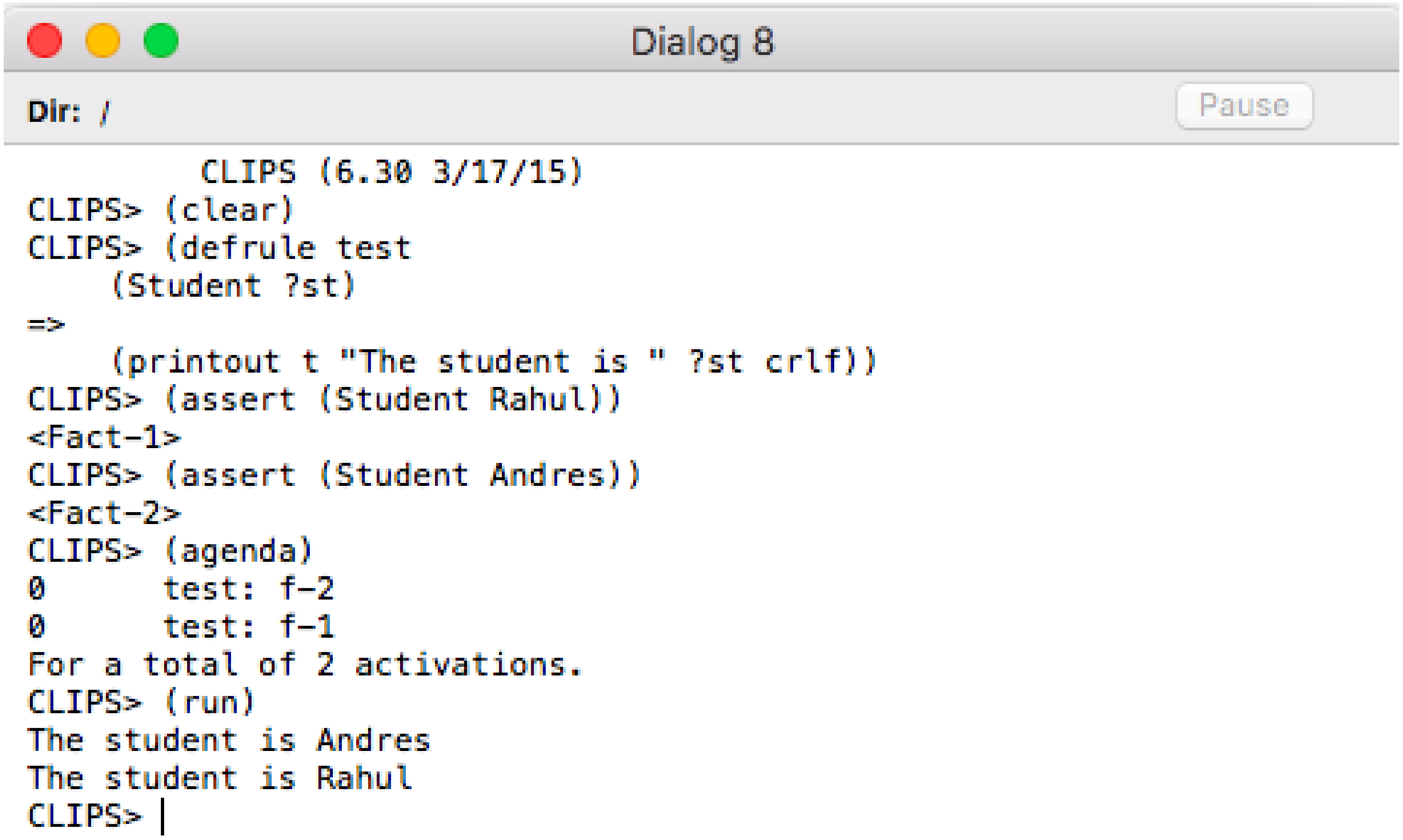
```
(defrule rule_name “optional comment”
        (pattern_1)     ; comment 1
        (pattern_2)     ; comment 2
        …
        (pattern_n)
 =>
        (action_1)      ; comment 3
        (action_2)
        …
        (action_n)
```

How to: Create rules. Example.



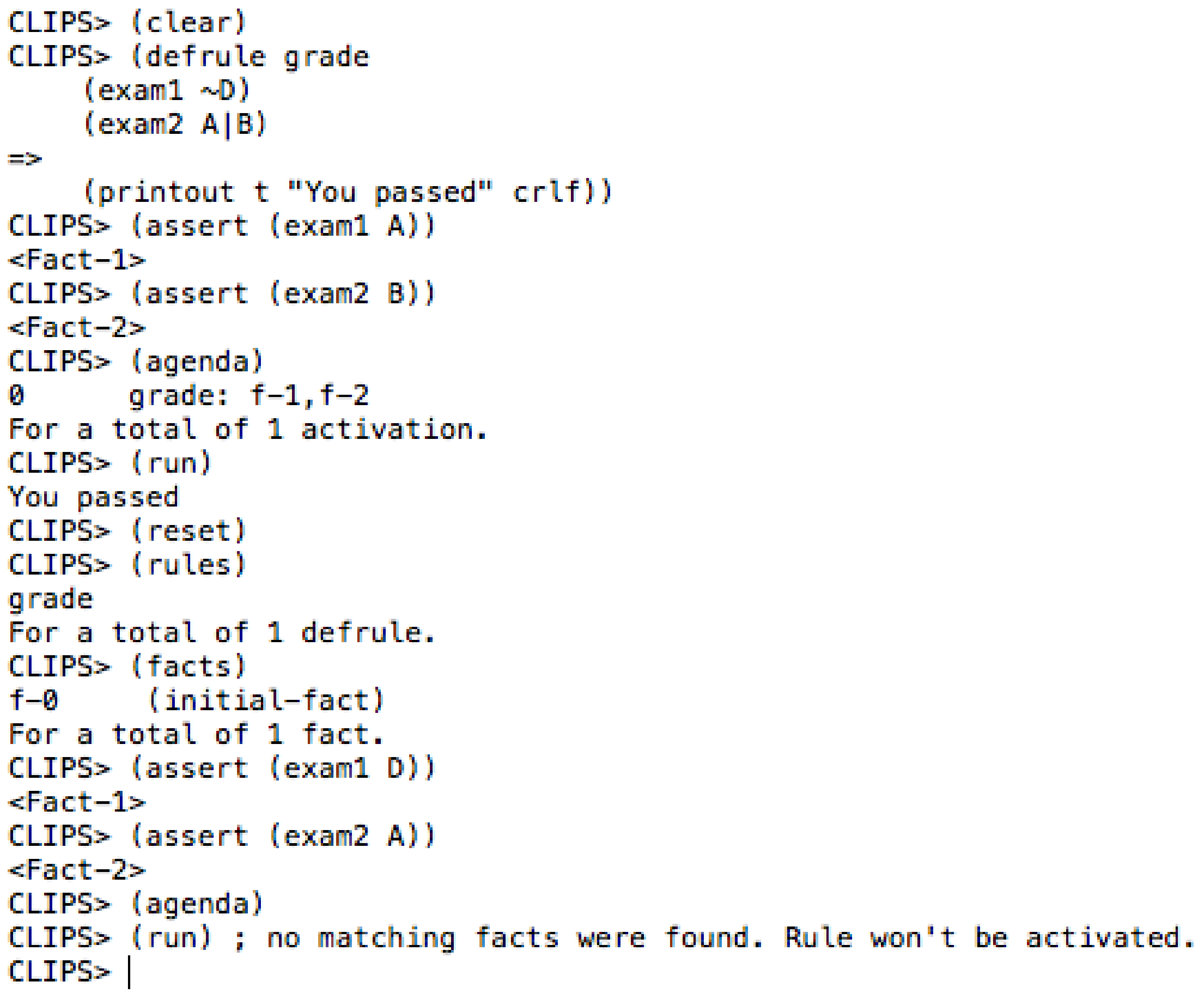
```
         CLIPS (6.30 3/17/15)
CLIPS> (clear) ; Clearing facts, classes, instances. pretty much everything
CLIPS> (assert (TA-is Yueng)) ; TA-is and Yueng are fields with constant values.
<Fact-1>
CLIPS> (facts) ; checking fact-list
f-0     (initial-fact)
f-1     (TA-is Yueng)
For a total of 2 facts.
CLIPS> (agenda) ; pending rules to be activated
CLIPS> (defrule TA
    (TA-is Yueng)
=>
    (assert (your-future-is mine)))
CLIPS> (agenda) ; pending rules to be activated.
0      TA: f-1
For a total of 1 activation.
CLIPS> (run) ; running pending rule.
CLIPS> (facts) ; checking fact-list after rule was executed.
f-0     (initial-fact)
f-1     (TA-is Yueng)
f-2     (your-future-is mine)
For a total of 3 facts.
CLIPS> |
```

How to: Create rules with a variable.



```
           CLIPS (6.30 3/17/15)
CLIPS> (clear)
CLIPS> (defrule test
    (Student ?st)
=>
    (printout t "The student is " ?st crlf))
CLIPS> (assert (Student Rahul))
<Fact-1>
CLIPS> (assert (Student Andres))
<Fact-2>
CLIPS> (agenda)
0      test: f-2
0      test: f-1
For a total of 2 activations.
CLIPS> (run)
The student is Andres
The student is Rahul
CLIPS> 
```

How to: Create rules. AND, OR, NOT



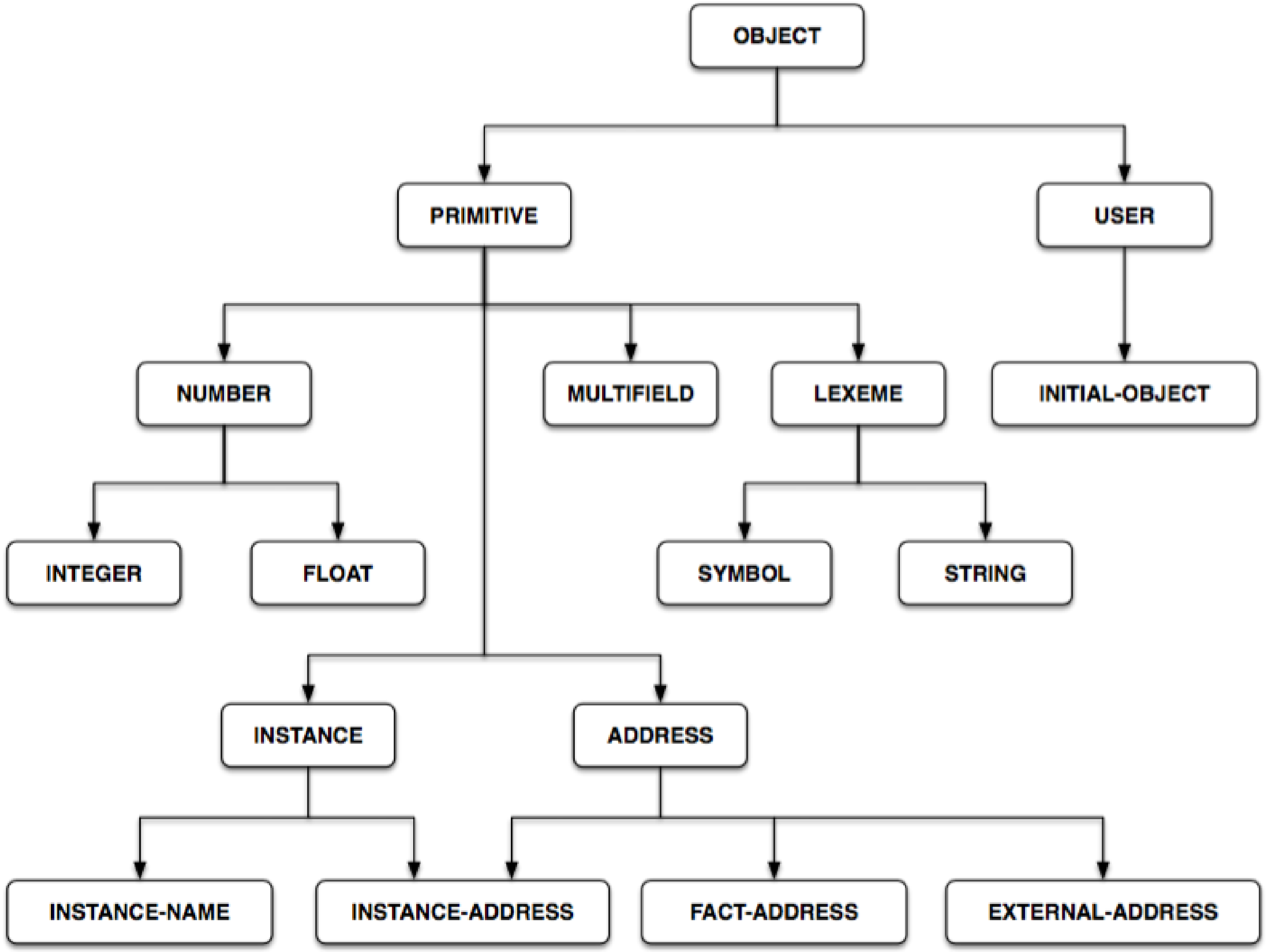
```
CLIPS> (clear)
CLIPS> (defrule grade
    (exam1 ~D)
    (exam2 A|B)
=>
    (printout t "You passed" crlf))
CLIPS> (assert (exam1 A))
<Fact-1>
CLIPS> (assert (exam2 B))
<Fact-2>
CLIPS> (agenda)
0      grade: f-1,f-2
For a total of 1 activation.
CLIPS> (run)
You passed
CLIPS> (reset)
CLIPS> (rules)
grade
For a total of 1 defrule.
CLIPS> (facts)
f-0     (initial-fact)
For a total of 1 fact.
CLIPS> (assert (exam1 D))
<Fact-1>
CLIPS> (assert (exam2 A))
<Fact-2>
CLIPS> (agenda)
CLIPS> (run) ; no matching facts were found. Rule won't be activated.
CLIPS> 
```

How to: Read from keyboard

```
CLIPS> (clear)
CLIPS>
(defrule read-input
=>
   (printout t "Name a primary color" crlf)
   (assert (color (read))))
CLIPS>
(defrule check-input
   ?color <-
      (color ?color-read&red|yellow|blue)
=>
   (retract ?color)
   (printout t "Correct" crlf))
CLIPS> (reset)
CLIPS> (agenda)
0       read-input: *
For a total of 1 activation.
```

```
CLIPS> (run)
Name a primary color
red
Correct
CLIPS> (reset)
CLIPS> (run)
Name a primary color
green
CLIPS>     ; No "correct"
```

How to: OOP


OBJECT
PRIMITIVE
USER
NUMBER
MULTIFIELD
LEXEME
INITIAL-OBJECT
INTEGER
FLOAT
SYMBOL
STRING
INSTANCE
ADDRESS
INSTANCE-NAME
INSTANCE-ADDRESS
FACT-ADDRESS
EXTERNAL-ADDRESS

How to: OOP. Defining a Class.

```
(defclass <class>
   (is-a <direct-superclasses>))
```

```
CLIPS> (defclass STUDENT (is-a USER)
           (slot Name)
           (slot Age))
```

How to: OOP. Object’s Behavior. Message-handlers.

```
(defmessage-handler <class-name>
   <message-name> [handler-type]
   [comment]
   (<parameters>* [wildcard-parameter])
   <action>*)
```

```
CLIPS>
; ?arg is argument of handler
(defmessage-handler NUMBER + (?arg)
   ; Function addition of handler
   (+ ?self ?arg))
CLIPS> (send 1 + 2)
3
```

CLIPS File Example

ex1a.CLP

```
(defclass WINE
                (is-a USER)
                (role concrete)
                (slot color)
                (slot type)
                (slot body))

(defclass MEAL
                (is-a USER)
                (role concrete)
                (slot mealname)
                (slot has_sauce)
                (slot sauce_type)
                (slot main_component)
                (slot suggested_wine_body))

(defclass PERSON
                (is-a USER)
                (role concrete)
                (slot preferred_color)
                (slot firstname)
)
```

CLIPS File Example

ex1a.CLP - continued

```
(defrule suggest_full_body
?ins <- (object (is-a MEAL) (has_sauce yes) (sauce_type spicy))
=> (send ?ins put-suggested_wine_body full) )

(defrule suggest_light_body
?ins <- (object (is-a MEAL) (has_sauce no) (main_component fish))
=> (send ?ins put-suggested_wine_body light) )

(defrule is_riesling
?ins <- (object (is-a WINE) (color white) (body light))
=> (send ?ins put-type riesling) )

(defrule is_cabernet_sauvignon
?ins <- (object (is-a WINE) (color red) (body full))
=> (send ?ins put-type cabernet_sauvignon) )

(defrule is_zinfandel
?ins <- (object (is-a WINE) (color red) (body light))
=> (send ?ins put-type zinfandel) )
```

CLIPS File Example

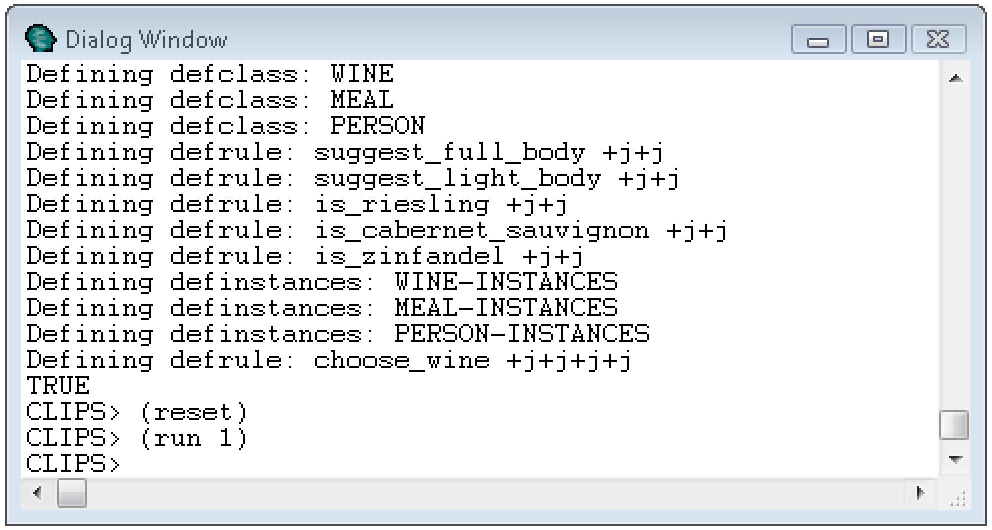
ex1a.CLP - continued

```
(definstances WINE-INSTANCES
        (firstwine of WINE (color white) (body light))
        (secondwine of WINE (color red) (body full))
        (thirdwine of WINE (type chianti))
        (fourthwine of WINE (color red) (body light))
)

(definstances MEAL-INSTANCES
        (firstmeal of MEAL (has_sauce yes) (sauce_type spicy) (mealname thai_chicken))
)

(definstances PERSON-INSTANCES
        (john of PERSON (preferred_color red) (firstname john))
)

(defrule choose_wine (declare (salience -10))
(object (is-a MEAL) (suggested_wine_body ?swb) (mealname ?mn))
(object (is-a PERSON) (preferred_color ?pc) (firstname ?fn))
(object (is-a WINE) (color ?pc) (body ?swb) (type ?t))
=>
(printout t "If " ?fn " chooses " ?mn " then he should take " ?t " for his wine selection." crlf)
)
```

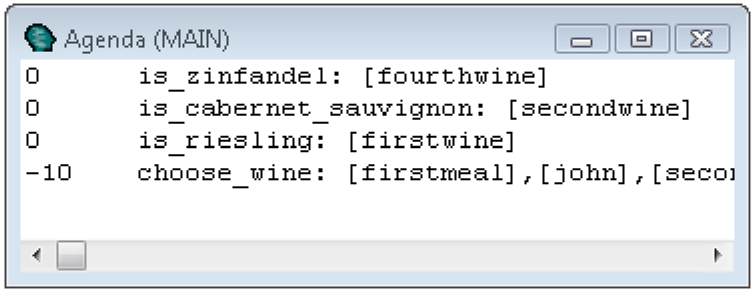
CLIPS Interface Example

Type “(run 1) [Enter]” or click Execution->Step to run the system through one step

Look at the Agenda to see what has changed



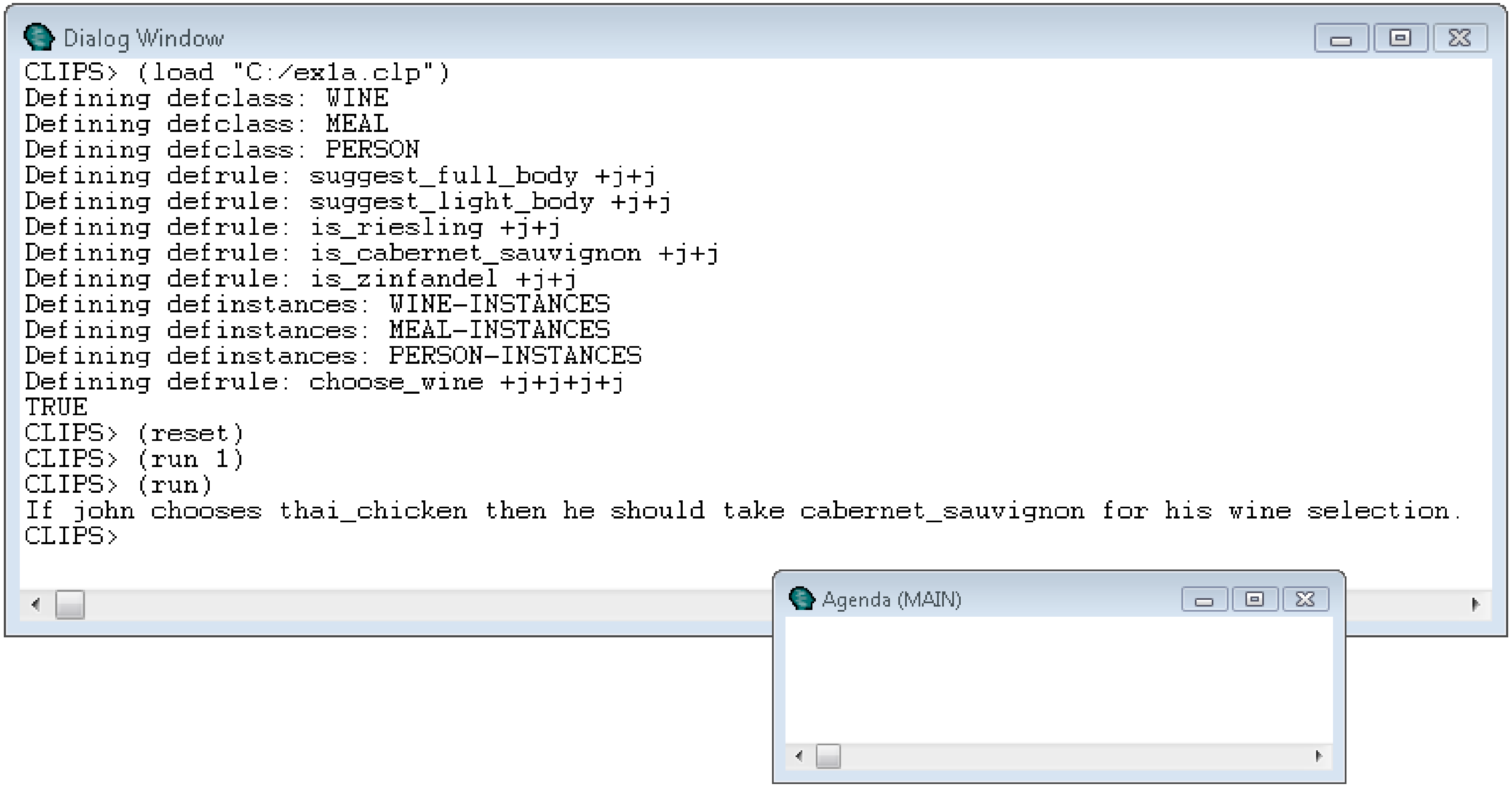
```
Defining defclass: WINE
Defining defclass: MEAL
Defining defclass: PERSON
Defining defrule: suggest_full_body +j+j
Defining defrule: suggest_light_body +j+j
Defining defrule: is_riesling +j+j
Defining defrule: is_cabernet_sauvignon +j+j
Defining defrule: is_zinfandel +j+j
Defining definstances: WINE-INSTANCES
Defining definstances: MEAL-INSTANCES
Defining definstances: PERSON-INSTANCES
Defining defrule: choose_wine +j+j+j+j
TRUE
CLIPS> (reset)
CLIPS> (run 1)
CLIPS>
```



```
0      is_zinfandel: [fourthwine]
0      is_cabernet_sauvignon: [secondwine]
0      is_riesling: [firstwine]
-10    choose_wine: [firstmeal],[john],[seco
```

CLIPS Interface Example

8. Type “(run) [Enter]” or click Execution->Run to finish running the system



```
CLIPS> (load "C:/ex1a.clp")
Defining defclass: WINE
Defining defclass: MEAL
Defining defclass: PERSON
Defining defrule: suggest_full_body +j+j
Defining defrule: suggest_light_body +j+j
Defining defrule: is_riesling +j+j
Defining defrule: is_cabernet_sauvignon +j+j
Defining defrule: is_zinfandel +j+j
Defining definstances: WINE-INSTANCES
Defining definstances: MEAL-INSTANCES
Defining definstances: PERSON-INSTANCES
Defining defrule: choose_wine +j+j+j+j
TRUE
CLIPS> (reset)
CLIPS> (run 1)
CLIPS> (run)
If john chooses thai_chicken then he should take cabernet_sauvignon for his wine selection.
CLIPS>
```

CLIPS File Example #2

ahex.CLP

```
(defclass ANIMAL
		(is-a USER)
		(slot myname)
		(slot skin_covering)
		(slot step_length)
		(slot step_frequency)
)

(defclass BIRD
		(is-a ANIMAL)
		(slot flies)
		(slot neck_length)
		(slot leg_length)
		(slot color))

(defclass MAMMAL
		(is-a ANIMAL)
		(slot food_type))
```

CLIPS File Example #2

ahex.CLP - continued

```
(defclass UNGULATE
        (is-a MAMMAL)
        (slot neck_length)
        (slot leg_length)
        (slot spots_stripes))

(defclass CARNIVORE
        (is-a MAMMAL)
        (slot color)
        (slot spots_stripes))


(defmessage-handler ANIMAL speed ()
        (* ?self:step_length ?self:step_frequency))

(defrule veryfast
        ?ins <- (object (is-a ANIMAL) (myname ?mn))
        (test (> (send ?ins speed) 100))
        =>
        (printout t ?mn " is a very fast animal!" crlf))
```

CLIPS File Example #2

ahex.CLP - continued

```
(defrule fast
        ?ins <- (object (is-a ANIMAL) (myname ?mn))
        (test (and (<= (send ?ins speed) 100) (> (send ?ins speed) 50)))
        =>
        (printout t ?mn " is a fast animal." crlf))

(defrule normal
        ?ins <- (object (is-a ANIMAL) (myname ?mn))
        (test (and (<= (send ?ins speed) 50) (> (send ?ins speed) 25)))
        =>
        (printout t ?mn " is a normal animal." crlf))

(defrule slow
        ?ins <- (object (is-a ANIMAL) (myname ?mn))
        (test (and (<= (send ?ins speed) 25) (> (send ?ins speed) 10)))
        =>
        (printout t ?mn " is a slow animal." crlf))
```

CLIPS File Example #2

ahex.CLP - continued

```
(defrule veryslow
        ?ins <- (object (is-a ANIMAL) (myname ?mn))
        (test (<= (send ?ins speed) 10))
        =>
        (printout t ?mn " is a very slow animal!" crlf))

(definstances myinstances
        (a of ANIMAL (myname tiger) (step_length 8) (step_frequency 10))
        (b of ANIMAL (myname sheep) (step_length 5) (step_frequency 5))
        (c of ANIMAL (myname sloth) (step_length 1) (step_frequency .1))
        (d of ANIMAL (myname falcon) (step_length 1) (step_frequency 101)))
```