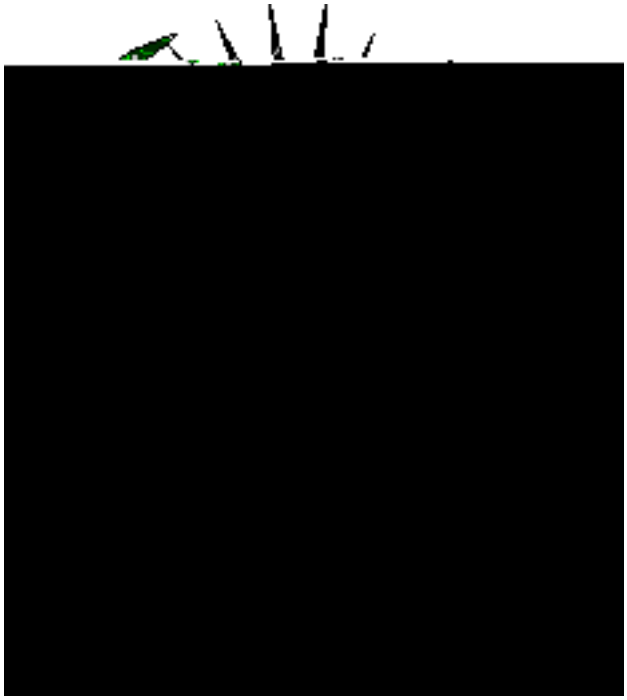


User!s Guide

Quicksilver Beta

December 31st 2007

byQ q 9 0 5 2 1n

CLIPS User!s Guide

Table of Contents

Your Prim 0 Tm ter r1ir6 (-1f40 594 743 re W n/Cs1 cs 0 0 0 sc q 0.2000000 0 0 0.2000000 90 744.4cm BT 60 1

- *Object-oriented programming*

language such as Jav

```
f-0     (initial-fact)
f-1     (duck)
For a total of 2 facts.
CLIPS>
```

The terms *f-0*

f-1 (a)
f-2 (b)
f-3 (c)
For a total of 4 facts.

example, think of a field as analogous to a mailbox. There's a big difference between an empty mailbox, and no mailbox at all. Without the *nil*, the fact becomes a single-field fact

- double quotes, **"**

```
"duck soup is good!!!"
```

The third and fourth types of field are **numeric fields**. A field which represents a number which can be either an **integer** or **floating-point** type field. A floating-

f-2 (x 1.5)
f-

```
For a total of 2 facts.
CLIPS>
```

As you can see, CLIPS replaced the carriage returns and tabs with single spaces.

```
CLIPS> (clear)
CLIPS> (assert (animal-is duck))
<Fact-
```

to the right means an entering fact or activation while an arrow to the left would mean an exiting fact or activation.

```
CLIPS> (clear)
CLIPS> (defrule duck
   (animal-is duck)
=>
```

```
   (animal-is duck)
   =>
   (assert (sound-is quack)))
CLIPS>
```

CLIPS puts different parts of the rule on different lines for the sake of readability. The patterns before the arrow are still considered the LHS and the actions after the arrow are still considered the RHS of the rule. The term *MAIN* refers to the MAIN module that this rule is in by default. You can define modules to put rules in analogous to the state

The output is the text within the double quotes. Be sure to type the letter "**t**" following the printout command. This tells CLIPS to send the output to the **standard output device** of your computer. Generally, the standard output device is your terminal (hence the letter "t" after printout.) However, this may be redefined so that the standard output

Chapter 3 Adding Details

Notice that the (printout) prints the fact-index of ?duck, <Fact-

The move-to-

Attributes	***Value***
name	

As you can see, CLIPS has

```
    (default rich))    ;default value of field assets
  (slot age            ;name of field
    (default 80)))     ;default value of field age
CLIPS>
(defrule matrimonial_candidate
  (prospect (name ?name) (assets ?net_worth) (age ?months))
=>
  (printout t "Prospect: " ?name crlf
                           ?net_worth crlf
                           ?months " months old" crlf))
CLIPS> (assert (prospect (name "Dopey Wonderful") (age 99)))
<Fact-1>
CLIPS> (run)
Prospect: Dopey Wonderful
rich
99 months old
CLIPS>
```

(assets rich)

```
=>
   (printout t "Don't walk" crlf))
```

The previous rules are simplified and don't cover all cases such as the breakdown of the traffic-light. For example, what does the robot do if the light is red or yellow and the walk-sign says walk?

```
CLIPS> (clear)
CLIPS> (defrule cautious
   (light yellow|blinking-yellow)
```

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

```
(defrule addition
   (numbers ?x ?y ?z)
=>
   (assert (answer-plus (+ ?x ?y ?z))))      ; ?x + ?y + ?z
```

conflict with variable names in a rule if they are the same. The term dummy argument is sometimes called a **parameter**

Function

Chapter 7 How to Be in Control

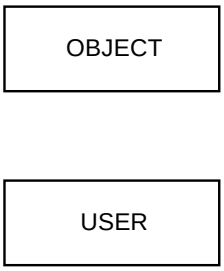
```
CLIPS> (run)
Name a primary color
red
Correct
CLIPS> (reset)
CLIPS> (run)
Name a primary color
green
CLIPS>              ; No "correct"
```

The rule is designed to use keyboard input on the RHS, so it's convenient to trigger

information written to it will be saved.

The **logical name**

wheel and the accelerator. That is, these people are not concerned with the hundreds of

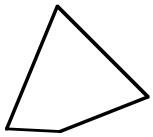

OBJECT
USER

```
(defclass UPPIE (is-a USER))
```

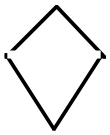
Notice the simi

validation nee ll t (h) Q q 9(d) -sq 9 40 5 ( ) 11 (a) dnt (11 (a)g5 (e) -11 (e)s 5 ( ) f5 ( ) 11 (a)c 4 (i) 5

QUADRILATERAL



KITE



The OOP paradigm is quite different from the subroutine library approach in which

by brackets if it is a user-

The second rule encourages the idea that classes are intended as a template to produce multiple objects of the same kind. Of course you can start out with zero or one instance. However, if you'll never need more than one instance in a class, you should consider modifying its superclass to accomodate the instance rather than defining a new subclass. If all

Chapter 10 Fascinating Facets

If you want to have class, then act, dress, and talk like your friends.

every new type.

```
CLIPS> (describe-class INTEGER)
==========================================================
**********************************************************
Abstract: direct instances of this class cannot be created.====
```

```
CLIPS> (send 2.5 + 3)
5.5
CLIPS> (send 2.5 + 2.6)
5.1
CLIPS> (describe-class NUMBER)
==============================================================
**************************************************************
Abstract: direct instances of th
```

assume that *[Dorky_]*

```
(slot sound (default quack))
(slot age))
```

effect on an instance and so are considered basic actions like get.

Daemons are easily implemented using before and after handlers since these will be executed before and after their primary handler. Implementing daemons like this is called **declarative implementation** because no explicit actions on the part of the handler is necessary for it to be executed. That is, CLIPS will always execute a before handler before its primary and will always execute an after handler after its primary. In a declarative daemon implementation, the normal operation of CLIPS will cause the

```
CLIPS> (make-instance [Dorky_Duck] of DUCK (age 3))
[Dorky_Duck]
CLIPS> (send [Dorky_Duck] lie-about-age 1)
*** Starting to print ***
I am only 2
*** Finished printing ***
```

Now that you unde

```
(multislot position  (propagation no-inherit)))
```

```
(1 0)
CLIPS> (send (send (send [Triangle1] get-line3) get-point2) put-position -1
0)
(-1 0)
CLIPS>
```

The stored values are as follows.

a slot in the specified cla

deffunctions are explicitly called and then executed. Just because a rule is activated does not mean it will be executed. Deffunctions are completely procedural in nature because once called by name, their code is executed in a procedural manner, statement by statement. Also, no pattern matching involving constraints is used in a deffunction to decide if its actions should be executed. Instead, any arguments that match the number expected by the deffunction argument list will satisfy the deffunction and cause its actions to be executed.

```
CLIPS> (> "duck1" "duck2")
FALSE
CLIPS>
```

The (defgeneric) acts as a **header declaration**

Support Information

exponentiation

warm-inis1 Tf h5(al1 (n) 10 (n) z[(i) -at10 (i) -o1 (m)] TJ ET Q Q q 9 40 594 743 re W n /Cs1cs 0 0 0 sc q 0.2000000 0 0 0.20000