

```
1 ;Samedi 6 Avril 2002 11:18:0 file : evalsym.vlisp ;
2
3 ;----- evaluateur partiel [hw];
4 ; adapte du pretty-print de [pg];
5 ; 24.3.02 (hw);
6
7 (setq --x-- careful careful nil)
8
9 (de symeval (exp) symeval
10 (setq activity 'symb)
11 (let ((x exp) (y (interpret exp))) (if (equal x y) x (self y (interpret y))))))
12
13 (de interpret (l ;; al xx) interpret
14 (initializeInterprete)
15 (cond
16 ((null l) (casNil))
17 ((atom l) (casAtom))
18 ((and (eq (car l) quote) (null (cddr l))) (casQuote))
19 ((and (listp (car l)) (eq (caar l) lambda)) (casLambda))
20 (t (initGeneralCase)
21 (let
22 ((x
23 (let ((f (and (litatom (car l)) (get (car l) activity))))
24 (if f (apply f nil) (defaultCase)))))) (exitGeneralCase x))))))
25
26 (de casNil nil nil)
27
28 (defmacro casAtom () casAtom
29 '(let (val (assoc l al)) (if val (cadr val) l)))
30
31 (defmacro casQuote () casQuote
32 '(cadr l))
33
34 (defmacro casLambda () casLambda
35 '(progn
36 (setq
37 l
38 ['let
39 (let ((v (cadar l)) (a (cdr l)))
40 (if (null v) nil [[(nextl v) (nextl a)] . (self v a)])
41 . (cddar l))
42 (interpret l)))
43
44 (de initGeneralCase nil nil)
45
46 (de initializeInterprete nil nil)
47
48 (de exitGeneralCase (x) exitGeneralCase
49 x)
50
51 (defmacro defaultCase () defaultCase
52 '(cond
53 ((or (null (get (car l) 'body)) (listp (car l)))
```

```

54 [(interpret (nextl l) al) . (mapcar l '(lambda (x) (interpret x al)))]
55 (t
56 (let (largs (mapcar (cdr l) '(lambda (x) (interpret x al))))
57 (let (all (redpair (car l) (get (car l) 'vars) largs))
58 (if (eq all 'no)
59 [(car l) . largs]
60 (interpret (get (car l) 'body) (nconc all al))))))
61
62 (put 'cdr
63 'symb
64 '(lambda ()
65 (let ((x (interpret (cadr l) al)))
66 (cond ((null x) nil) ((equal (car x) 'cons) (caddr x)) (t ['cdr x])))))
67
68 (put '1+
69 'symb
70 '(lambda ()
71 (let ((x (interpret (cadr l) al)))
72 (cond ((numbp x) (1+ x)) ((atom x) ['1+ x]) ((eq (car x) '1-) (cadr x))
73 (t ['1+ x]))))
74
75 (put '1-
76 'symb
77 '(lambda ()
78 (let ((x (interpret (cadr l) al)))
79 (cond ((numbp x) (1- x)) ((atom x) ['1- x]) ((eq (car x) '1+) (cadr x))
80 (t ['1+ x]))))
81
82 (put 'zerop
83 'symb
84 '(lambda ()
85 (let ((x (interpret (cadr l) al)))
86 (cond ((numbp x) (if (zerop x) t nil)) ((atom x) [zerop x]) ((eq (car x) '1+) nil)
87 (t ['zerop x]))))
88
89 (put 'car
90 'symb
91 '(lambda ()
92 (let ((x (interpret (cadr l) al)))
93 (cond ((null x) nil) ((equal (car x) 'cons) (cadr x)) (t ['car x]))))
94
95 (put 'null
96 'symb
97 '(lambda ()
98 (let ((x (interpret (cadr l) al)))
99 (cond ((null x) t) ((and (listp x) (eq (car x) 'cons)) nil) (t ['null x]))))
100
101 (put 'if
102 'symb
103 '(lambda ()
104 (let ((x (interpret (cadr l) al)))
105 (cond ((or (eq x t) (and (listp x) (eq (car x) 'cons))) (interpret (caddr l) al)) ((null x) (interpret (cadr (caddr l)) al))
106 (t ['null . [x . (caddr l)]])))
107
108 (put 'de 'symb '(lambda () (eval l) (analyse (cadr l)) (cadr l)))
109

```

```

110 (de analyse (f ;; x vars body decvars)                                analyse
111   (setq x (fval f) vars (car x) body (cadr x))
112   (let (e body)
113     (cond
114       ((atom e))
115       ((member (car e) '(car cdr 1-))
116         (if (atom (cadr e))
117             (and
118              (member (cadr e) vars)
119              (or (member (cadr e) decvars) (newl decvars (cadr e))))
120             (self (cadr e))))
121       (t (self (nextl e)) (self e))))
122   (put f 'vars vars)
123   (put f 'body body)
124   (put f 'decvars decvars))
125
126 ; ----- controleur de deploiement -----;
127
128 (de redpair (fun vars largs ;; decvars res v l)                        redpair
129   (setq decvars (get fun 'decvars))
130   (while vars
131     (setq v (nextl vars) l (nextl largs))
132     (if (member v decvars) (and l (neq l 0) (neq (car l) 'cons) (neq (car l) '1+)) (exit 'no)))
133     (newl res [v l]))
134   res)
135
136 (de app (x y)                                                         app
137   (if (null x) y [(car x) . (app (cdr x) y)]))
138
139 (analyse 'app)
140
141 (de rev (x)                                                           rev
142   (if (null x) nil (app (rev (cdr x)) [(car x) . nil])))
143
144 (analyse 'rev)
145
146 (de len (l)                                                         len
147   (if (null l) 0 (1+ (len (cdr l)))))
148
149 (analyse 'len)
150
151 (de p (x y)                                                         p
152   (if (zerop x) y (1+ (p (1- x) y))))
153
154 (analyse 'p)
155
156 (de m (x y)                                                         m
157   (if (zerop x) 0 (p y (1- x) y)))
158
159 (analyse 'm)
160
161 (de loop ()                                                         loop
162   (print "eval sym :")
163   (let (x (read)) (if (null x) 'ok (print (symeval x)) (print "eval sym :") (self (read)))))
164
165 (setq careful --x--)
```

166
167

Cross Reference

analyse de 110
app de 136
casAtom defmacro 28
casLambda defmacro 34
casNil de 26
casQuote defmacro 31
defaultCase defmacro 51
exitGeneralCase de 48
initGeneralCase de 44
initializeInterprete de 46
interprete de 13
len de 146
loop de 161
m de 156
p de 151
redpair de 128
rev de 141
symeval de 9

1	--x--	7 165
2	a	39 40 40
3	activity	10 23
4	al	13 29 54 54 56 60 65 71 78 85 92 98 104 105
5	all	57 58 60
6	analyse	108 110 139 144 149 154 159
7	app	136 137 139 142
8	body	53 60 110 111 112 123 123
9	casAtom	17 28
10	casLambda	19 34
11	casNil	16 26
12	casQuote	18 31
13	decvars	110 119 119 124 124 128 129 129 132
14	defaultCase	24 51

15	defmacro	28 31 34 51
16	e	112 114 115 116 118 119 119 120 121 121
17	exitGeneralCase	24 48
18	f	23 24 24 110 111 122 123 124
19	fun	128 129
20	initGeneralCase	20 44
21	initializeInterprete	14 46
22	interprete	11 11 13 42 54 54 56 60 65 71 78 85 92 98 104 105
23	l	13 16 17 18 18 19 19 23 23 29 29 32 37 39 39 41 42 53 53 54 54 56 57 57 59 60 65 71 78 85 92 98 104 105 106 108 108 108 128 131 132 132 132 132 133 146 147 147
24	larges	56 57 59 128 131
25	len	146 147 149
26	loop	161
27	m	156 159
28	no	58 132
29	ok	163
30	p	151 152 154 157
31	redpair	57 128
32	res	128 133 134
33	rev	141 142 144
34	symb	10 63 69 76 83 90 96 102 108
35	symeval	9 163
36	v	39 40 40 40 128 131 132 133
37	val	29 29 29
38	vars	57 110 111 118 122 122 128 130 131
39	x	11 11 11 22 24 48 49 54 54 56 56 65 66 66 66 66 71 72 72 72 72 73 78 79 79 79 79 80 85 86 86 86 86 86 87 92 93 93 93 93 98 99 99 99 99 104 105 105 105 106 110 111 111 111 111 136 137 137 137 141 142 142 142 151 152 152 156 157 157 163 163 163

40 **xx** 13

41 **y** 11 11 11 11 136 137 137 151 152 152 156 157 157

;Samedi 6 Avril 2002 11:18:1 end of file : evalsym.vlisp ;