CLIPS

● Breve historia y aplicaciones del lenguaje asignado

CLIPS: es una herramienta de desarrollo para la producción y ejecución de sistemas expertos. Fue creado en el año 1984 en el space center de la NASA Lyndon B. Johnson. Su nombre es un acrónimo de C **L**anguage **I**ntegrated **P**roduction **S**ystem (Sistema de Producción Integrado en Lenguaje C).

Durante el año 1984, en el Lyndon B. Johnson Space Center, la sección de inteligencia artificial había desarrollado alrededor de una docena de prototipos de sistemas expertos usando hardware y software de aquella época. A pesar del demostrado potencial de los sistemas expertos, la mayoría de aquellos prototipos no estaban siendo usados regularmente, según la NASA esto se debió a que el lenguaje de programación usado para el desarrollo de estas aplicaciones era LISP. Se encontraron varias debilidades de

LISP, de las cuales se destacan tres:
- No estaba disponible para una amplia variedad de equipos de cómputo.
- No era fácilmente integrable con otras aplicaciones.
- Su costo era muy elevado.

Sus aplicaciones pueden comprender diferentes tipos de algoritmos:
- Búsqueda (ciega, con heurística, con coste y heurística)
- Sistemas de control
- Árboles de decisión

● Palabras reservadas seleccionadas

!=
*
**
+
-
/
<
<=
<>
=
>
>=
abs
acos
acosh
acot

acoth
acsc
acsch
activeduplicateinstance
active-initialize-instance
active-make-instance
activemessageduplicateinstance
active-message-modify-instance
active-modify-instance
agenda
and
any-instancep
apropos
asec
asech
asin
asinh
assert
assert-string
atan
atanh
batch
bind
bload
bload-instances
break
browse-classes
bsave
bsave-instances
build
call-next-handler
call-next-method
call-specific-method
class
class-abstractp
class-existp
class-reactivep
class-slots
class-subclasses
class-superclasses
clear
clear-focus-stack
close
conserve-mem
constructs-to-c
cos
cosh
cot
coth
create$
csc
csch
defclass-module
deffacts-module

deffunction-module
defgeneric-module
defglobal-module
definstances-module
defrule-module
deftemplate-module
deg-grad
deg-rad
delayed-do-for-all-instances
delete$
delete-instance
dependencies
dependents
describe-class
direct-mv-delete
direct-mv-insert
direct-mv-replace
div
do-for-all-instances
do-for-instance
dribble-off
dribble-on
duplicate
duplicate-instance
duplicate-instance
dynamic-get
dynamic-put
edit
eq
eval
evenp
exit
exp
expand$
explode$
fact-index
facts
fetch
find-all-instances
find-instance
first$
float
floatp
focus
format
gensym
gensym*
get
get-auto-float-dividend
get-current-module
get-defclass-list
get-deffacts-list
get-deffunction-list
get-defgeneric-list

get-defglobal-list
get-definstances-list
get-defmessage-handler-list
get-defmethod-list
get-defmodule-list
get-defrule-list
get-deftemplate-list
get-dynamic-constraint-checking
get-fact-duplication
get-focus
get-focus-stack
get-function-restrictions
get-incremental-reset
get-method-restrictions
get-reset-globals
get-salience-evaluation
get-sequence-operator-recognition
get-static-constraint-checking
get-strategy
grad-deg
halt
help
help-path
if
implode$
init-slots
initialize-instance
initialize-instance
insert$
instance-address
instance-addressp
instance-existp
instance-name
instance-name-to-symbol
instance-namep
instancep
instances
integer
integerp
length
length$
lexemep
list-defclasses
list-deffacts
list-deffunctions
list-defgenerics
list-defglobals
list-definstances
list-defmessage-handlers
list-defmethods
list-defmodules
list-defrules
list-deftemplates
list-focus-stack

list-watch-items
load
load-facts
load-instances
log
log10
loop-for-count
lowcase
make-instance
make-instance
matches
max
mem-requests
mem-used
member
member$
message-duplicate-instance
message-duplicate-instance
message-handler-existp
message-modify-instance
message-modify-instance
min
mod
modify
modify-instance
modify-instance
multifieldp
mv-append
mv-delete
mv-replace
mv-slot-delete
mv-slot-insert
mv-slot-replace
mv-subseq
neq
next-handlerp
next-methodp
not
nth
nth$
numberp
object-pattern-match-delay
oddp
open
options
or
override-next-handler
override-next-method
pi
pointerp
pop-focus
ppdefclass
ppdeffacts
ppdeffunction

ppdefgeneric
ppdefglobal
ppdefinstances
ppdefmessage-handler
ppdefmethod
ppdefmodule
ppdefrule
ppdeftemplate
ppinstance
preview-generic
preview-send
primitives-info
print-region
printout
progn
progn$
put
rad-deg
random
read
readline
refresh
refresh-agenda
release-mem
remove
remove-break
rename
replace$
reset
rest$
restore-instances
retract
return
round
rule-complexity
rules
run
save
save-facts
save-instances
sec
sech
seed
send
sequencep
set-auto-float-dividend
set-break
set-current-module
set-dynamic-constraint-checking
set-fact-duplication
set-incremental-reset
set-reset-globals
set-salience-evaluation
set-sequence-operator-recognition

set-static-constraint-checking
set-strategy
setgen
show-breaks
show-defglobals
show-fht
show-fpn
show-joins
show-opn
sin
sinh
slotallowedvalues
slot-cardinality
slot-delete$
slot-direct-accessp
slot-direct-delete$
slot-direct-insert$
slot-direct-replace$
slot-existp
slot-facets
slot-initablep
slot-insert$
slot-publicp
slot-range
slot-replace$
slot-sources
slot-types
slot-writablep
sqrt
str-assert
str-cat
str-compare
str-explode
str-implode
str-index
str-length
stringp
sub-string
subclassp
subseq$
subset
subsetp
superclassp
switch
sym-cat
symbol-to-instance-name
symbolp
system
tan
tanh
time
toss
type
type

undefclass
undeffacts
undeffunction
undefgeneric
undefglobal
undefinstances
undefmessage-handler
undefmethod
undefrule
undeftemplate
unmake-instance
unwatch
upcase
watch
while
wordp

● Expresión regular para los identificadores

Una expresión regular denota un conjunto de secuencias de símbolos válidos que se construyen en base al alfabeto existente de un lenguaje. Una secuencia de símbolos 'S' es una expresión regular que denota a un conjunto que contiene a 'S'.

● Lista de operadores y caracteres especiales seleccionados

Operadores:
- Restricción tilde: La restricción conectiva tilde actua negando el valor sobre el que actúa. Para aclarar este concepto, supongamos que queremos escribir una regla que niegue el paso a toda persona de la base de conocimiento que no tenga de nombre Juan. La regla sería algo parecido a:

```
1  (defrule prohibirpaso
2      (persona (nombre ~"Juan"))
3      =>
4      (printout t "prohibdo el paso" crlf)
5      (assert (pasar no))
6  )
```

Como podemos comprobar en el ejemplo, la regla se activará para todos los hechos del tipo persona cuyo nombre no sea Juan, y actuará dando un mensaje prohibiendo el paso y produciendo el hecho (pasar no), que se podría utilizar en una regla que sirviera, por ejemplo, para controlar la apertura de la puerta.

- Restricción barra: La restricción conectiva barra se utiliza para combinar varios hechos. Supongamos que queremos modificar la regla anterior para que se permita el paso a Juan y Pedro. La regla que realizaría esta acción sería:

```
(defrule prohibirpaso
    (persona (nombre ~"Juan"))
    =>
    (printout t "prohibdo el paso" crlf)
    (assert (pasar no))
)
```

- Restricción &: Esta última restricción se utiliza para conectar varias restricciones en unión.

```
(defrule permitirpaso2
     (persona (nombre ?name&"Juan"|"Pedro"))
=>
    (printout t "Puedes pasar" ?name crlf)
    (assert (pasar si))
)
```

- Forma de construcción de comentarios en el lenguaje

La construcción de comentarios en CLIPS comienza con un punto y coma. Todo lo que le sigue al punto y coma será ignorado por el compilador.

```
(defrule duck                          ; Rule header
    "Here comes the quack"             ; Comment
    (animal-is duck)                   ; Pattern
    =>                                 ; THEN arrow
    (assert (sound-is quack))          ; Action
)
```

- Un ejemplo de programa para escribir “Hola mundo” en el lenguaje

```
(defrule hw
    (f ?x)
    =>
    (printout t ?x crlf)
)

(assert (f "Hola mundo"))
(run)
```