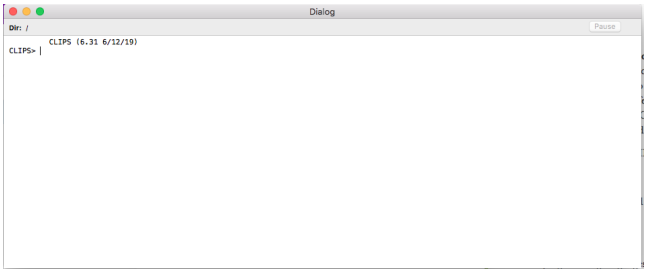
CLIPS 1 - Intro

Παρασκευή, 27 Μαρτίου 2020 9:08 πμ

Κατεβάζουμε CLIPS από http://www.clipsrules.net/
ή για linux "**sudo apt-get install clips**"
Εγκαθιστούμε...
Τρέχουμε το command line περιβάλλον.



ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ (NOC122)
https://e-class.ionio.gr/courses/NOC122/index.php

CLIPS (6.31 6/12/19)

```
CLIPS> (assert (colour green)) ; προσθέτουμε ένα γεγονός
<Fact-1>                       ; η απόκριση της clips
CLIPS> (facts)                 ; ποια facts υπάρχουν
f-0     (initial-fact)
f-1     (colour green)
For a total of 2 facts.
CLIPS> assert (colour red))
assert
CLIPS> (assert (colour red))
<Fact-2>
CLIPS> (facts)
f-0     (initial-fact)
f-1     (colour green)
f-2     (colour red)
For a total of 3 facts.
CLIPS> (retract 1)             ; αφαίρεση ενός fact με το id
του
CLIPS> (facts)
f-0     (initial-fact)
f-2     (colour red)
For a total of 2 facts.
CLIPS> (watch facts)           ; "παρακολουθούμε" τα γεγονότα
που γίνονται assert ή retract
CLIPS> (assert (colour green))
==> f-3     (colour green)     ; προσθέτουμε ένα γεγονός,
ίδιο με το προηγούμενο color green, αλλά διαφορετικό!
Παίρνει άλλο id
<Fact-3>
CLIPS> (facts)
f-0     (initial-fact)
```

```
f-2     (colour red)
f-3     (colour green)
For a total of 3 facts.
CLIPS> (assert (colour green))  ; προσθέτουμε ένα ακόμη ίδιο
γεγονός; εφόσον υπάρχει αυτή τη στιγμή στη βάση γνώσης, δεν
μπορούμε
FALSE
CLIPS> (facts)
f-0     (initial-fact)
f-2     (colour red)
f-3     (colour green)
For a total of 3 facts.
CLIPS> (defrule duck           ; προσθέτουμε ένα κανόνα
(animal-is duck)               ; έχει όνομα, συνθήκη(/-ές)
=>                             ; και παράγει κάτι στο δεξί
μέλος
(assert (sound-is quack))
)
CLIPS> (rules)                 ; ποιοι κανόνες υπάρχουν;
duck
For a total of 1 defrule.
CLIPS> (ppdefrule duck)        ; βλέπουμε πώς έχει οριστεί ο κανόνας
duck
(defrule MAIN::duck
   (animal-is duck)
   =>
   (assert (sound-is quack)))
CLIPS> (facts)
f-0     (initial-fact)
f-2     (colour red)
f-3     (colour green)
For a total of 3 facts.
CLIPS> (rules)
duck
For a total of 1 defrule.
CLIPS> (assert (animal-is duck))
==> f-4     (animal-is duck)
<Fact-4>
CLIPS> (facts)
f-0     (initial-fact)
f-2     (colour red)
f-3     (colour green)
f-4     (animal-is duck)
For a total of 4 facts.
CLIPS> (agenda)                ; όταν ικανοποιούνται οι
προϋποθέσεις τους, οι κανόνες περιμένουν ενεργοποίηση στην agenda
0      duck: f-4
For a total of 1 activation.
CLIPS> (run)                   ; η run προκαλεί την εκτέλεση όλων
των κανόνων
==> f-5     (sound-is quack)
CLIPS> (facts)
```

```
f-0     (initial-fact)
f-2     (colour red)
f-3     (colour green)
f-4     (animal-is duck)
f-5     (sound-is quack)
For a total of 5 facts.
CLIPS> (defrule is-it-a-duck (animal-has webbed-feet)
(animal-has feathers) => (assert (animal-is duck)) )
CLIPS> (rules)
```
; προσθέτουμε ένα νέο κανόνα
```
duck
is-it-a-duck
For a total of 2 defrules.
CLIPS> (facts)
f-0     (initial-fact)
f-2     (colour red)
f-3     (colour green)
f-4     (animal-is duck)
f-5     (sound-is quack)
For a total of 5 facts.
CLIPS> (reset)
```
; η reset αφαιρεί όλα τα γεγονότα (πλην του initial fact) αλλά αφήνει τους κανόνες στη βάση γνώσης
```
<== f-0     (initial-fact)
<== f-2     (colour red)
<== f-3     (colour green)
<== f-4     (animal-is duck)
<== f-5     (sound-is quack)
==> f-0     (initial-fact)
CLIPS> (rules)
duck
is-it-a-duck
For a total of 2 defrules.
CLIPS> (assert (animal-has webbed-feet))
```
; προσθέτουμε το πρώτο γεγονός που περιμένει ο κανόνας is-it-a-duck
```
==> f-1     (animal-has webbed-feet)
<Fact-1>
CLIPS> (assert (animal-has feathers))
```
; προσθέτουμε το δεύτερο γεγονός που περιμένει ο κανόνας is-it-a-duck
```
==> f-2     (animal-has feathers)
<Fact-2>
CLIPS> (facts)
f-0     (initial-fact)
f-1     (animal-has webbed-feet)
f-2     (animal-has feathers)
For a total of 3 facts.
CLIPS> (agenda)
```
; ο κανόνας is-it-a-duck περιμένει εκτέλεση
```
0      is-it-a-duck: f-1,f-2
For a total of 1 activation.
CLIPS> (run)
```
; εκτελούμε...
```
==> f-3     (animal-is duck)
```
; πρώτο νέο γεγονός, από ενεργοποίηση του κανόνα is-it-a-duck

```
==> f-4     (sound-is quack)
```
; δεύτερο νέο γεγονός, από ενεργοποίηση του κανόνα duck

```
CLIPS> (facts)
f-0     (initial-fact)
f-1     (animal-has webbed-feet)
f-2     (animal-has feathers)
f-3     (animal-is duck)
f-4     (sound-is quack)
For a total of 5 facts.
CLIPS> (defrule duck (animal-is duck) => (assert (sound-is
quack)) (printout t "it’s a duck" crlf))
```
; τροποποίηση κανονα για να κάνει δύο πράγματα στο THEN μέρος του, ένα assert και ένα printout

```
CLIPS> (rules)
is-it-a-duck
duck
For a total of 2 defrules.
CLIPS> (ppdefrule duck)
```
; ο κανονας όντως άλλαξε

```
(defrule MAIN::duck
   (animal-is duck)
   =>
   (assert (sound-is quack))
   (printout t "it’s a duck" crlf))
CLIPS> (reset)
```
; αφαιρούμε όλα τα facts από τη βάση γνώσης

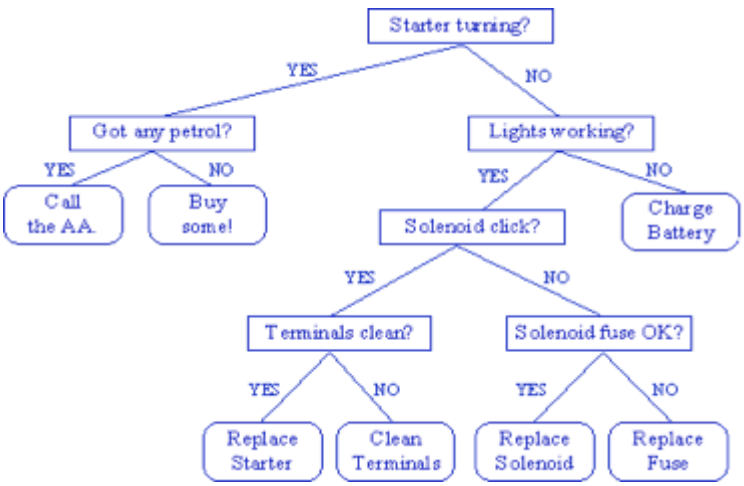
```
<== f-0     (initial-fact)
<== f-1     (animal-has webbed-feet)
<== f-2     (animal-has feathers)
<== f-3     (animal-is duck)
<== f-4     (sound-is quack)
==> f-0     (initial-fact)
CLIPS> (facts)
f-0     (initial-fact)
For a total of 1 fact.
CLIPS> (rules)
is-it-a-duck
duck
For a total of 2 defrules.
CLIPS> (assert (animal-has webbed-feet))
```
; προσθέτουμε ένα fact

```
==> f-1     (animal-has webbed-feet)
<Fact-1>
CLIPS> (assert (animal-has feathers) )
```
; προσθέτουμε ένα ακόμη fact

```
==> f-2     (animal-has feathers)
<Fact-2>
CLIPS> (facts)
f-0     (initial-fact)
f-1     (animal-has webbed-feet)
f-2     (animal-has feathers)
For a total of 3 facts.
CLIPS> (agenda)
```
; τα δύο προηγούμενα facts αρκούν για να γίνει ενεργοποίηση του is-it-a-duck, έχει λοιπον προστεθεί στην agenda

```
0      is-it-a-duck: f-1,f-2
For a total of 1 activation.
CLIPS> (run)                                  ; εκτέλεση
FIRE    1 is-it-a-duck: f-3,f-2               ; ενεργοποίηση is-it-a-duck
==> f-4     (animal-is duck)                  ; Ο is-it-a-duck κάνει assert
```
ένα νέο fact, το οποίο ενεργοποιεί τον κανόνα duck
```
FIRE    2 duck: f-4                           ; ενεργοποίηση duck
==> f-5     (sound-is quack)                  ; Ο duck κάνει assert ένα
```
ακόμα fact και κάνει και ένα printout
```
it's a duck                                   ; Το printout
C
CLIPS> (reset)                                ; Η reset αφαιρεί facts,
```
διατηρεί rules
```
<== f-0     (initial-fact)
<== f-1     (animal-has webbed-feet)
<== f-2     (animal-has feathers)
<== f-3     (animal-is duck)
<== f-4     (sound-is quack)
==> f-0     (initial-fact)
CLIPS> (rules)
is-it-a-duck
duck
For a total of 2 defrules.
CLIPS> (clear)                                ; Η clearαφαιρεί facts
```
ΚΑΙ rules
```
==> f-0     (initial-fact)
CLIPS> (rules)
CLIPS>
```

Να υλοποιήσετε αυτό το expert system:



Μερικοί πιθανοί κανόνες που να λύνουν (εν μέρη) το ζητούμενο:

```
(defrule charge-battery
      (lights-working no)(starter-turning no) =>
      (printout t "Charge battery" crlf))
(defrule replace-solenoid
      (solenoid-fuse-ok yes)(solenoid-click no)(lights-working yes)
      (starter-turning no) =>
```

```
        (printout t "Replace Solenoid" crlf))
(defrule replace-fuse
        (solenoid-fuse-ok no)(solenoid-click no)(lights-working yes)
        (starter-turning no) =>
        (printout t "Replace fuse" crlf))
```

```
CLIPS> (watch rules)
CLIPS> (watch facts)
CLIPS> (watch activations)
CLIPS> (facts)
f-0     (initial-fact)
For a total of 1 fact.
CLIPS> (rules)
replace-fuse
replace-solenoid
charge-battery
For a total of 3 defrules.
CLIPS> (agenda)
CLIPS> (assert (solenoid-click no))
==> f-1     (solenoid-click no)
<Fact-1>
CLIPS> (agenda)
CLIPS> (assert (lights-working no))
==> f-2     (lights-working no)
<Fact-2>
CLIPS> (assert (starter-turning no))
==> f-3     (starter-turning no)
==> Activation 0      charge-battery: f-2,f-3
<Fact-3>
CLIPS> (run)
FIRE    1 charge-battery: f-2,f-3
Charge battery
CLIPS> (facts)
f-0     (initial-fact)
f-1     (solenoid-click no)
f-2     (lights-working no)
f-3     (starter-turning no)
For a total of 4 facts.
CLIPS> (assert (light-working yes))
==> f-4     (light-working yes)
<Fact-4>
CLIPS> (facts)
```

f-0 (initial-fact)

f-1 (solenoid-click no)

f-2 (lights-working no)

f-3 (starter-turning no)

f-4 (light-working yes)

For a total of 5 facts.

CLIPS> (retract 2)

<== f-2 (lights-working no)

CLIPS> (facts)

f-0 (initial-fact)

f-1 (solenoid-click no)

f-3 (starter-turning no)

f-4 (light-working yes)

For a total of 4 facts.

CLIPS> (assert solenoid-fuse-ok no))

[PRNTUTIL2] Syntax Error: Check appropriate syntax for RHS patterns.

CLIPS> (assert (solenoid-fuse-ok no))

==> f-5 (solenoid-fuse-ok no)

<Fact-5>

CLIPS> (facts)

f-0 (initial-fact)

f-1 (solenoid-click no)

f-3 (starter-turning no)

f-4 (light-working yes)

f-5 (solenoid-fuse-ok no)

For a total of 5 facts.

CLIPS> (agenda)

CLIPS> (matches replace-fuse)

Matches for Pattern 1

f-5

Matches for Pattern 2

f-1

Matches for Pattern 3

None

Matches for Pattern 4

f-3

Partial matches for CEs 1 - 2

f-5,f-1

Partial matches for CEs 1 - 3

None

Partial matches for CEs 1 - 4

```
None
Activations
None
(3 1 0)
CLIPS> (ppdefrule replace-fuse)
(defrule MAIN::replace-fuse
  (solenoid-fuse-ok no)
  (solenoid-click no)
  (lights-working yes)
  (starter-turning no)
  =>
  (printout t "Replace fuse" crlf))
CLIPS> (facts)
f-0     (initial-fact)
f-1     (solenoid-click no)
f-3     (starter-turning no)
f-4     (light-working yes)
f-5     (solenoid-fuse-ok no)
For a total of 5 facts.
CLIPS> (retract 4)
<== f-4     (light-working yes)
CLIPS> (assert (lights-working yes))
==> f-6     (lights-working yes)
==> Activation 0      replace-fuse: f-5,f-1,f-6,f-3
<Fact-6>
CLIPS> (run)
FIRE    1 replace-fuse: f-5,f-1,f-6,f-3
Replace fuse
```