ΠΑΡΑΡΤΗΜΑ 2

Το Σύστημα Κανόνων CLIPS

Τεχνητή Νοημοσύνη - Β' Έκδοση

Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου

Ιστορικά Στοιχεία...

- **C** **L**anguage **I**ntegrated **P**roduction **S**ystem
- Περιβάλλον προγραμματισμού με κανόνες, αντικείμενα και συναρτήσεις
- Αναπτύχθηκε από τη NASA το 1985
  - Υλοποιήθηκε με τη γλώσσα C
  - Σύνταξη θυμίζει OPS5
  - Λειτουργικότητα όμοια με ART
- Τρέχει σε DOS, Windows, UNIX, VMS
- Υποστηρίζει τον αντικειμενοστραφή προγραμματισμό (COOL)

Δομή του CLIPS

- Λίστα Γεγονότων (facts list)

```
(name george)
```

- Βάση κανόνων (rule/knowledge base)

```
(defrule rain "in case of rain"
    (weather rain) =>
    (assert (action "take umbrella")))
```

- Μηχανισμός Εξαγωγής Συμπερασμάτων (Inference Engine)
  - Στρατηγικές Επίλυσης Ανταγωνισμού (Conflict Resolution Strategies)

Εκτέλεση Προγράμματος

- Πρόγραμμα
  - Ένα σύνολο από κανόνες και γεγονότα
- Εκτέλεση
  - Ακολουθία από πυροδοτήσεις κανόνων των οποίων οι συνθήκες ικανοποιούνται
- Ικανοποίηση συνθηκών
  - Ταυτοποίηση με γεγονότα
- Η εκτέλεση τερματίζεται όταν:
  - Δεν υπάρχουν άλλοι κανόνες προς πυροδότηση
  - Κληθεί συγκεκριμένη εντολή τερματισμού (**`halt`**)

Κύκλος Λειτουργίας CLIPS

1. Εύρεση όλων των κανόνων των οποίων οι συνθήκες ικανοποιούνται και προσθήκη τους στην ατζέντα (*agenda - conflict set*).
2. Αν η ατζέντα είναι κενή τότε η εκτέλεση τερματίζεται.
3. Επιλογή ενός κανόνα με βάση τη στρατηγική επίλυσης ανταγωνισμού (*conflict resolution*) και εκτέλεσή του.
4. Επιστροφή στο βήμα 1, εκτός αν υπάρχει εντολή τερματισμού (*halt*).

Σύνταξη του CLIPS

- Θυμίζει LISP
- Είναι Case-Sensitive
- Δομικά Στοιχεία:
  - Σύμβολα, π.χ. **`you`**, **`why_this`**, **`good-morning`**
  - Αλφαριθμητικά, π.χ. **`"This is a String"`**
  - Αριθμοί, π.χ. **`24`**,**`-45.6`**, **`8e11`**
  - Σχόλια: ότι ακολουθεί τον χαρακτήρα **`;`**
- Μεταβλητές
  - Μονότιμες π.χ. **`?var`**, **`?x`**
  - Πολλαπλών Τιμών π.χ. **`$?fruits`**, **`$?shopping`**

Μεταβλητές

- Εμφανίζονται
  - Στις συνθήκες ενός κανόνα
  - Στις ενέργειες ενός κανόνα
- Παίρνουν τιμές
  - Κυρίως στις συνθήκες των κανόνων μέσω της διαδικασίας ταυτοποίησης
  - Η ανάθεση τιμής σε μεταβλητή στις ενέργειες ενός κανόνα είναι δυνατή με τη χρήση κατάλληλης συνάρτησης, αλλά καλό είναι να αποφεύγεται.
- Η εμβέλεια των μεταβλητών περιορίζεται στον κανόνα που αυτές εμφανίζονται.

Γεγονότα

- Λίστες από σύμβολα που περικλείονται σε παρενθέσεις π.χ.

```
(name John Papas)
(shopping_list cheese wine bread book)
(days Monday Friday Sunday)
```

- Κάθε γεγονός έχει μοναδικό αριθμό-ταυτότητα (fact index) που καθορίζεται αυτόματα
- Εμφάνιση γεγονότων

```
CLIPS> (facts)
f-0     (name John Papas)
for a total of 1 fact.
```

Εισαγωγή Γεγονότων

- Με τη χρήση της εντολής **assert**

(assert <fact>)

```
CLIPS> (assert (gift book))
<Fact-0>
CLIPS> (facts)
f-0      (gift book)
for a total of 1 fact.
CLIPS> (assert (day Monday) (phone 891363))
CLIPS> (facts)
f-0      (gift book)
f-1      (day Monday)
f-2      (phone 891363)
for a total of 3 facts.
```


Εισαγωγή Γεγονότων

- Με τη χρήση της εντολής **deffacts** (μαζί με **reset**)

```
(deffacts <name> "comments"
     (<fact1>)
     (<fact2>)
     …
     (<fact n>)
)
```

Εντολή **deffacts**

```
CLIPS> (deffacts colours  "this is to insert
  some colours" (colour red)(colour blue)(colour
  green))
CLIPS> (reset)
CLIPS> (facts)
f-0      (initial-fact)
f-1      (colour red)
f-2      (colour blue)
f-3      (colour green)
for a total of 4 facts.
CLIPS> (facts 2)
f-2      (colour blue)
f-3      (colour green)
for a total of 2 facts.
```


Επιτρέπονται διπλά γεγονότα?

```
CLIPS> (assert (name nick))
<Fact-0>
CLIPS> (facts)
f-0 (name nick)
For a total of 1 facts.
CLIPS> (assert (name nick))
FALSE
CLIPS> (facts)
f-0 (name nick)
For a total of 1 facts.
```

Διαγραφή Γεγονότων

- Χρήση της εντολής **(retract)**

  (retract <fact-index>)

```
CLIPS> (retract 1 3)
CLIPS> (facts)
f-0      (initial-fact)
f-2      (colour blue)
for a total of 2 facts.
CLIPS> (retract 1)
[PRNUTIL1] Unable to find fact-1
```

- Χρήση της εντολής **(clear)**

Παρακολούθηση Προσθήκης και Διαγραφής Γεγονότων

- Χρήση της εντολής **(watch facts)**

```
CLIPS> (watch facts)
CLIPS> (assert (age 23))
==> f-0   (age 23)
<Fact-0>
CLIPS>  (reset)
<== f-0     (age 23)
==> f-0     (initial-fact)
CLIPS> (assert (mobile 0977233445))
==> f-1     (mobile 0977233445)
<Fact-1>
CLIPS> (retract 1)
<== f-1           (mobile 0977233445)
CLIPS> (facts)
f-0   (initial-fact)
for a total of 1 fact.
```

Κανόνες

- Μορφή:
  - **`if`** (Συνθήκες) **`then`** (Ενέργειες)
- Συνθήκες:
  - Γεγονότα & Μεταβλητές
- Ενέργειες:
  - Πράξεις (εντολές) μετά την ενεργοποίηση του κανόνα
- Σημασία:
  - **Εάν** ικανοποιούνται οι συνθήκες (δηλαδή ταυτοποιούνται με τη λίστα γεγονότων),
  - **Τότε** εκτέλεσε τις ενέργειες

Κανόνες

- Σύνταξη:

```
(defrule <όνομα κανόνα>
  "<σχόλια>"
  (<συνθήκη 1>)
  ...
  (<συνθήκη n>)
 =>
  (<εντολή 1>)
  ...
  (<εντολή m>)
)
```

Κανόνες

- Παράδειγμα

```
(defrule soccer-time
  "Warns for your soccer time"
  (day sunday)
  (time afternoon)
=>
  (assert (go for soccer)))
```


Ταυτοποίηση

Συνθήκη	Γεγονός	Αναθέσεις τιμών
`(day ?d ?t)`	`(day fri 12)`	`?d=fri` `?t=12`
`(list $?lst)`	`(list a b c d)`	`$?lst=(a b c d)`

`(car ?c model $?m license ?l)`	`(car 1 model BMW FIAT license wqw45)`	`?c=1`
		`$?m=(BMW FIAT)`
		`?l=wqw45`

Ταυτοποίηση

Συνθήκη	Γεγονός
`(day ?d ?t)`	`(days fri 12)`
`(list a b $?lst)`	`(list 1 2 c d e f)`
`(car ?c type $?m license ?l)`	`(car 1 2 type BMW FIAT license wqw45)`

Build-in Συναρτήσεις

Ορισμένες από το σύστημα

- Βασικές Αριθμητικές Συναρτήσεις

```
(+ <ορίσματα>)        (- <ορίσματα>)
(* <ορίσματα>)        (/ <ορίσματα>)
```

- Συναρτήσεις σύγκρισης αριθμών

```
(=  <ορίσματα>)       (<  <ορίσματα>)
(>=  <ορίσματα>)      (>  <ορίσματα>)
(<=  <ορίσματα>)      (<>  <ορίσματα>)
```

- Οι ακόλουθες συναρτήσεις επιστρέφουν **TRUE**

```
(>= 5 5 4 2 2 1)
(<=  2 3 3 4 6)
(<> 3 5)
```

Λογικές Συναρτήσεις

- Δυνατότητα να εκφραστούν πολύπλοκες συνθήκες κανόνων

```
(and  <ορίσματα>)
(or  <ορίσματα>)
(not <όρισμα>)
(eq <ορίσματα>)
(neq  <ορίσματα>)
```

- Οι ακόλουθες συναρτήσεις επιστρέφουν **TRUE**

```
(and (>= 5 5) (> 3 2 1) (= 10 10))
(and (not (= 10 7)) (= 9 9))
```


Λογικές Συναρτήσεις

```
(defrule days
   (month Jan mon)
   (or (hour 12) (hour 13))
  =>
   (assert (day is Monday noon time))
)

(defrule days
   (month Jan mon)
   (or (and (hour 12) (lunch break))
       (and (hour 17) (departure time))
   )
  =>
   (assert (office closed at this hour))
)
```

Συναρτήσεις εισόδου - εξόδου

(printout <device> <expression>)

- Αποστέλλει την έκφραση **<expression>** στη συσκευή **<device>**
  - Η συσκευή μπορεί να είναι ένα αρχείο ή η οθόνη
  - Για οθόνη, χρησιμοποιούμε **t** (terminal)

```
(printout t "The day was " ?type crlf)
The day was sunny
```

- Το σύμβολο **crlf** δηλώνει αλλαγή γραμμής

Συναρτήσεις εισόδου - εξόδου

(read)

- Εισάγει σύμβολο από το πληκτρολόγιο
- Συνήθως χρησιμοποιείται σε συνδυασμό με την εντολή **bind,** για ανάθεση τιμής σε μεταβλητή στις ενέργειες ενός κανόνα

```
(defrule get-user-answer
  (initial-fact)
  =>
  (printout t "What's your name: ")
  (bind ?name (read))
  (assert (user-name ?name))
)
```

Ανάθεση τιμής σε μεταβλητή

(bind <variable> <value>)

- Ανατίθεται η τιμή **<value>** σε μια μεταβλητή **<variable>** στις ενέργειες των κανόνων

```
(defrule rule1 "example rule"
  (oldcost ?oldcost)
  (newcost ?newcost)
  =>
  (bind ?total_cost (+ ?newcost ?oldcost))
  (assert (cost ?total_cost))
  (printout t "The total cost is "
  ?total_cost crlf)
)
```


Έλεγχος ροής προγράμματος

Συνάρτηση while

- Έστω ότι υπάρχει ένα γεγονός **(num 1)**
- Τύπωσε όλους τους αριθμούς από το 1 μέχρι το 9, με τη φράση "**the num is:**" σαν πρόθεμα

```
(defrule test
  (num ?n)
  =>
  (while (< ?n 10)
     do
   (printout t "the num is: " ?n crlf)
   (bind ?n (+ 1 ?n))
  )
)
```

Έλεγχος ροής προγράμματος

Συνάρτηση if-then-else

- Τύπωσε **positive, negative** ή **zero**, αν ο αριθμός **?n** είναι μεγαλύτερος, μικρότερος ή ίσος με μηδέν

```
(defrule sign
     (num ?n)
  =>
   (if (> ?n 0)
    then (printout t "positive" crlf))
    else    (if (< ?n 0)
             then (printout t "negative" crlf)
             else (printout t "zero" crlf))
  )
)
```


Η χρήση του **not**

- Αν εμφανίζονται μεταβλητές μέσα σε κάποιο **(not ...)** τότε δε γίνεται ανάθεση τιμών σε αυτές

```
(defrule wrong-rule
  (not (element ?b))
  =>
  (printout t "not element" ?b
  crlf) )
```

Μεγάλα/Σύνθετα Γεγονότα

- Σε μεγάλα προγράμματα χρειάζεται να αναπαρασταθεί η πληροφορία με μεγάλα ή σύνθετα γεγονότα
- Π.χ. βάση δεδομένων μαθητών:

```
(student name <name> surname <surname>
         sex <sex> age <age>
         classes <classes>)

(student name john surname ref sex male
         age 28
         classes math physics chem)
```


Agenda και Εκτέλεση Κανόνων

- Όλοι οι κανόνες των οποίων οι συνθήκες ικανοποιούνται εισάγονται στην agenda
  - Σύνολο συγκρούσεων (conflict set)
- Από την agenda επιλέγεται κάθε φορά 1 μόνο κανόνας, ο οποίος και πυροδοτείται με βάση 2 κριτήρια:
  - την προτεραιότητα των κανόνων, και
  - τη στρατηγική επίλυσης συγκρούσεων.

Η Agenda ως Στοίβα

- Η ατζέντα συμπεριφέρεται σαν στοίβα (stack) όπου όσο μεγαλύτερη προτεραιότητα έχει ένας κανόνας τόσο πιο ψηλά βρίσκεται σε αυτή.
- Κάθε φορά εκτελείται ο κανόνας που βρίσκεται στην κορυφή της στοίβας.
- Ένας νέος κανόνας τοποθετείται στην ατζέντα σύμφωνα με τα ακόλουθα κριτήρια:
  - Προτεραιότητα (salience)
  - Στρατηγική Επίλυσης Συγκρούσεων
  - Αυθαίρετη σειρά

Τοποθέτηση Κανόνα στην Agenda (1)

- Οι νέοι κανόνες μπαίνουν "πάνω" από όλους τους κανόνες με μικρότερη ή ίση προτεραιότητα (salience) και "κάτω" από όλους τους κανόνες με μεγαλύτερη προτεραιότητα.
- Στους κανόνες με ίδια προτεραιότητα χρησιμοποιείται η τρέχουσα στρατηγική επίλυσης συγκρούσεων για να καθοριστεί η σειρά τους.

Τοποθέτηση Κανόνα στην Agenda (2)

- Εάν κάποιοι κανόνες ενεργοποιήθηκαν από το ίδιο σύνολο γεγονότων και τα προηγούμενα βήματα δεν μπόρεσαν να ορίσουν μία σειρά, τότε δίνεται σε αυτούς μια αυθαίρετη σειρά (όχι τυχαία), η οποία εξαρτάται από την υλοποίηση του συστήματος.

Προτεραιότητα Κανόνων

- Σύνταξη (μέσα στον ορισμό του κανόνα)

```
(declare (salience <number>))
```

- Παράδειγμα:

```
(defrule cartesian
  (declare (salience 30))
  (element ?a)
  (element ?b)
  =>
  (printout t "Elements: " ?a " " ?b
  crlf))
```

Ιδιότητες Προτεραιότητας Κανόνα

- Είναι ακέραια αριθμητική τιμή.
- Όσο μεγαλύτερη είναι, τόσο μεγαλύτερη είναι και η προτεραιότητα του κανόνα.
- Οι επιτρεπτές τιμές είναι από -10000 έως 10000.
- Εάν δεν υπάρχει δήλωση, ο κανόνας θεωρείται ότι έχει την προκαθορισμένη τιμή μηδέν.

Χρήση Προτεραιότητας

```
(defrule MAIN::start
   (declare (salience 10000))
   =>
   (set-fact-duplication TRUE)
   (focus QUESTIONS CHOOSE-QUALITIES WINES PRINT-RESULTS))

(defrule MAIN::combine-certainties ""
   (declare (salience 100))
   ?rem1 <- (attribute (name ?rel) (value ?val) (certainty ?per1))
   ?rem2 <- (attribute (name ?rel) (value ?val) (certainty ?per2))
   (test (neq ?rem1 ?rem2))
   =>
   (retract ?rem1)
   (modify ?rem2 (certainty (/ (- (* 100 (+ ?per1 ?per2)) (* ?per1 ?per2)) 100))))
```