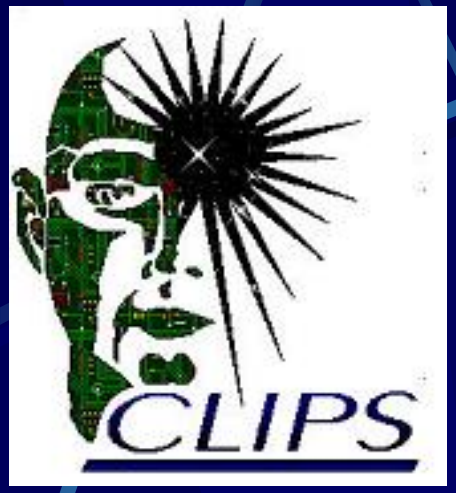
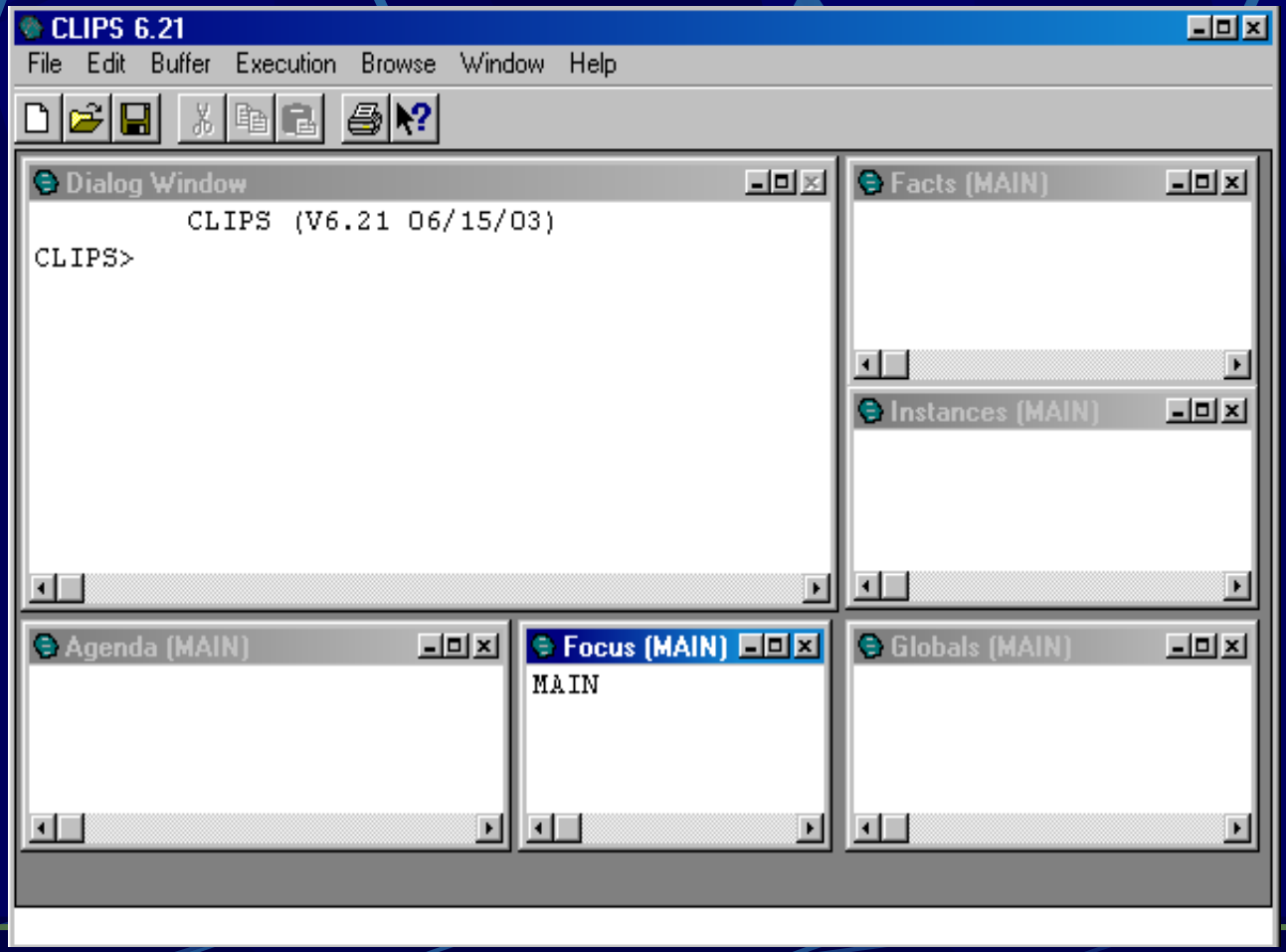


Εισαγωγή στο κέλυφος ανάπτυξης έμπειρων συστημάτων του CLIPS

Το περιβάλλον του CLIPS



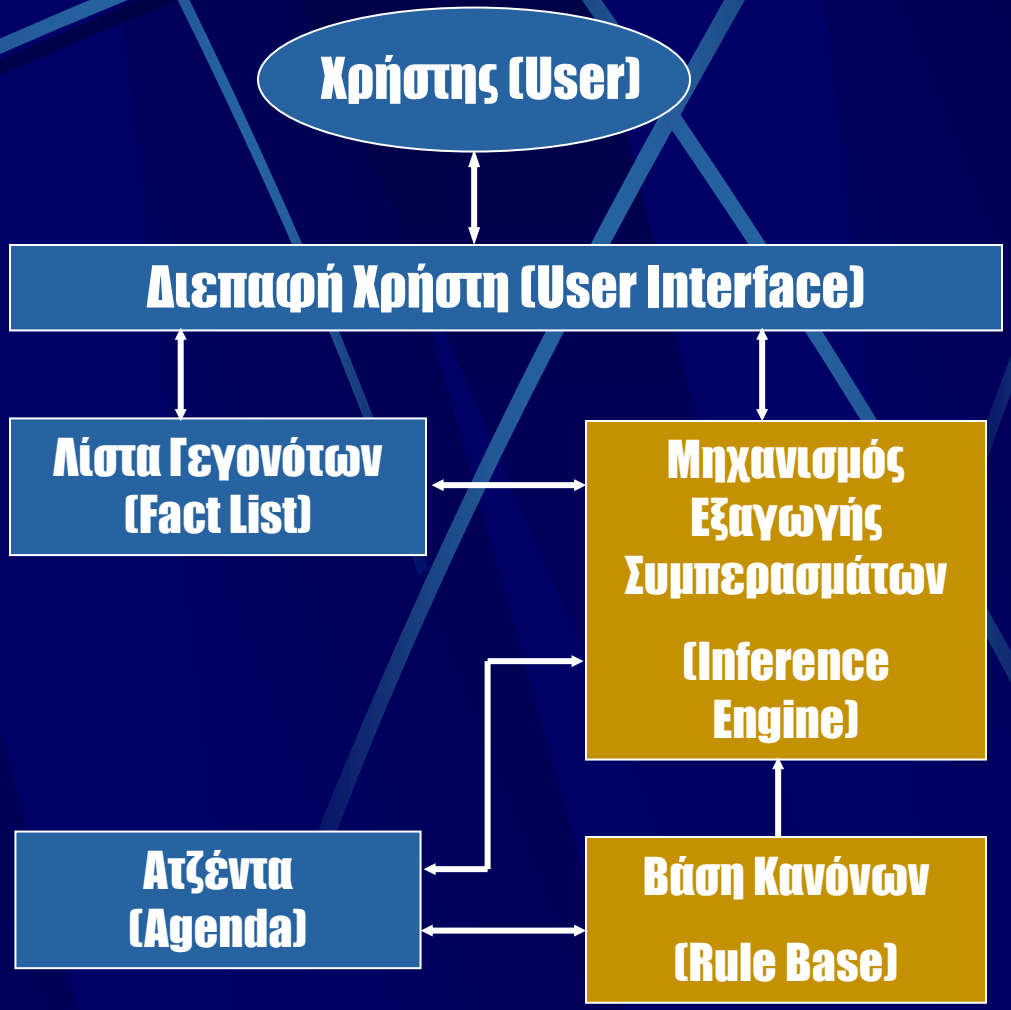
Τι είναι το CLIPS

Το CLIPS μπορεί να θεωρηθεί σαν ένα γενικό εργαλείο ανάπτυξης συστημάτων λογισμικού.

Το CLIPS εμπεριέχει ένα κέλυφος έμπειρου συστήματος, διότι διαθέτει τα βασικά μέρη ενός έμπειρου συστήματος

- *Λίστα γεγονότων (fact-list)*
- *Βάση κανόνων (rule-base)*
- *Μηχανισμός εξαγωγής συμπερασμάτων (inference engine)*

Δομή Κελύφους ΕΣ του CLIPS



Κύκλος Λειτουργίας Εξαγωγής Συμπερασμάτων

Ο κύκλος λειτουργίας είναι ο τυπικός κύκλος λειτουργίας ενός συστήματος παραγωγής:

1. **Εύρεση όλων των κανόνων των οποίων οι συνθήκες ικανοποιούνται και προσθήκη τους στην ατζέντα (agenda - conflict set).**
2. **Αν η ατζέντα είναι κενή ή ο μέγιστος αριθμός πυροδοτήσεων έχει συμπληρωθεί τότε η εκτέλεση τερματίζεται.**
3. **Διάταξη των κανόνων στην ατζέντα με βάση τη στρατηγική επίλυσης συγκρούσεων (conflict resolution).**
4. **Πυροδότηση του πρώτου κανόνα στην ατζέντα.**
5. **Επιστροφή στο βήμα 1, εκτός αν υπάρχει εντολή τερματισμού (halt).**

(Το CLIPS ακολουθεί την ορθή αλυσίδωση σαν μέθοδο εξαγωγής συμπερασμάτων.)

Βασικά Στοιχεία (1)

1. Τύποι δεδομένων

Σύμβολα (symbols): *Οποιαδήποτε ακολουθία εκτυπώσιμων χαρακτήρων που δεν ξεκινά από* <, I , &, (,), $, ?, +, - *και δεν περιέχει κανένα από τους* <, I , &, (,), ;

Αλφαριθμητικά (strings): *Οποιαδήποτε ακολουθία εκτυπώσιμων χαρακτήρων ανάμεσα σε διπλά εισαγωγικά. Π.χ.* "This is a program" , “2A”

Αριθμοί (numbers): 23, -23, +23 (*integers*)

23.34, +23.0, 23e4, -23.2e-5 (*floats*)

Σχόλια: *από ";" μέχρι το τέλος της γραμμής*

Βασικά Στοιχεία (2)

2. Μεταβλητές

Μονότιμες (singlevalue) : *ξεκινούν με ? (Π.χ.* ?x, ?day*).* Τιμές: 32, flight34, mary, “a12”

Πολλαπλών τιμών (multivalue) : *ξεκινούν με* $?. *Π.χ.*

$?days. Τιμές : (28 29 30 31), (mon tues wedn)

Καθολικές μεταβλητές (global variables)

Προσπελάσιμες από παντού:

(defglobal ?*<σύμβολο>* = <έκφραση>)

Π.χ. (defglobal ?*x* = 0)

Τοπικές μεταβλητές (local variables)

Προσπελάσιμες μόνο μέσα από τη δομή (π.χ. κανόνας, συνάρτηση) που χρησιμοποιούνται.

Βασικά Στοιχεία (3)

3. Γεγονότα (facts)

Διατεταγμένα: Λίστες από τιμές, που παριστάνουν σχετικά απλά γεγονότα, όπου η σειρά των τιμών παίζει ρόλο: (flight 734 DELTA), (person George), (class “A1” 1995)

Μη διατεταγμένα ή προτύπου: Ονοματοποιημένες λίστες που περιέχουν λίστες δύο στοιχείων, οι οποίες σχετίζονται με αντίστοιχο πρότυπο γεγονότων (fact template), και παριστάνουν πιο σύνθετα γεγονότα: (student (age 19) (year a) (sex male))

Βασικά Στοιχεία (4)

4. Κανόνες (rules)

   Δομές της μορφής:

   if *(συνθήκες)* then *(ενέργειες)*

   που αναπαριστούν ευρετική γνώση (heuristic knowledge) στη βάση γνώσης.

5. Συναρτήσεις (functions)

   Τμήματα προγράμματος για την αναπαράσταση διαδικαστικής γνώσης (procedural knowledge). Συνήθως επιστρέφουν κάποια τιμή, ως πλευρικό αποτέλεσμα.

Γεγονότα (1)

Εισαγωγή γεγονότων

assert: Εισαγωγή ενός γεγονότος

Παράδειγμα:

(assert (water-tank empty))

(assert <fact>)

deffacts: Εισαγωγή πολλών γεγονότων

Παράδειγμα:

(deffacts pref-cars
  (car AlfaRomeo156 1600 16 nai idrayliki 22910)
  (car AudiA4 1600 8 nai idrayliki 25700)
  (car FordMondeo 1800 16 nai idrayliki 19289)
  (car NissanPrimera 1600 16 oxi idrayliki 17990)
  (car VWPassat 1600 8 oxi idrayliki 18910))

(deffacts <όνομα>
["<σχόλιο>"]
<fact1>
<fact2>
…
<factn>)

Γεγονότα (2)

Διαγραφή γεγονότων

retract: Διαγραφή ενός ή όλων των γεγονότων

(retract <fact-index>)

(retract *)

Παράδειγμα:

(retract 2)

Συνήθως όμως, σ' ένα κανόνα δεν ξέρουμε τον αριθμό του γεγονότος. Τότε χρησιμοποιούμε τον ειδικό τελεστή '<-'.

Παράδειγμα:

?x <- (car AudiA4 1600 8 nai idravliki 25700)

(retract ?x)

Πρότυπα γεγονότων (1)

Ορισμός

```
(deftemplate <όνομα προτύπου>
   (slot/multislot <slot-name1> <constraint-form>*)
   (slot/multislot <slot-name2> <constraint-form>*)
                  …
   (slot/multislot <slot-namen> <constraint-form>*))
```

Πρότυπα γεγονότων (2)

Constraint forms

- Τύπου

  (type <type>) , όπου <type> ∈ {SYMBOL, STRING, LEXEME, INTEGER, FLOAT, NUMBER,?VARIABLE}
- Απαρίθμησης τιμών

  (allowed-<type> <value1> … <valuen>) , όπου <type> ∈ {symbols, strings, lexems, integers, floats, numbers, values} και τα <valuei> είναι αντίστοιχου τύπου.
- Περιοχής τιμών

  (range <init-val> <final-val>) , όπου <init-val>, <final-val> μπορεί να είναι ?VARIABLE (για ανοικτό κάτω, πάνω όριο)
- Εξ' ορισμού/Προκαθορισμένης τιμής

  (default <value>) , όπου <value> συγκεκριμένη τιμή ή ?DERIVE ή ?NONE (απαιτεί την εισαγωγή τιμής)
- Πλήθους τιμών

  (cardinality <min> <max>) , ελάχιστος και μέγιστος αριθμός τιμών μιας multislot.

Πρότυπα γεγονότων (3)

Παράδειγμα

```
(deftemplate student
   (slot name (type STRING) (default ?NONE))
   (slot sex (type SYMBOL) (allowed-symbols male female))
   (slot age (type INTEGER) (range 18 40))
   (multislot courses (type SYMBOL) (cardinality 1 4)))
```

Πρότυπα γεγονότων (4)

Εισαγωγή γεγονότων

Παραδείγματα

(assert (student (name “petros mixos”) (sex male) (age 19)))

(assert (student (name “giannis panou”) (sex male)

(courses ai db)))

(assert (student (sex female) (age 20))) → ERROR!!!

(λόγω name)

Επίσης και με deffacts κατά παρόμοιο τρόπο εισάγουμε πολλά γεγονότα μαζί.

Πρότυπα γεγονότων (5)

Τροποποίηση γεγονότων

(modify <fact-index> (<slot> <new-value>)*)

(αφαιρεί το παλιό γεγονός και εισάγει το νέο)

Παράδειγμα

?x <- (student (name ?y) (age 20))

(modify ?x (courses (math ai db)))

Κανόνες (1)

Σύνταξη

(defrule <rule-name>
"<comments>"
(condition-1)
(condition-2)
…
(condition-n)
LHS
=>
(command-1)
…
(command-m)
RHS

Παράδειγμα

(defrule pick-up-cube
"pick up a free cube"
(free hand)
(on table ?x)
?n1 <- (free hand)
?n2 <- (on table ?x)
=>
(assert (hand contains ?x))
(retract ?n1)
(retract ?n2)

Κανόνες (2)

Τύποι Συνθηκών

1. Προτύπου (pattern)
   - Το πρώτο στοιχείο είναι πάντα ένα σύμβολο που καθορίζει αν η συνθήκη προτύπου εφαρμόζεται σε διατεταγμένα γεγονότα ή γεγονότα προτύπου.
   - Στα υπόλοιπα μπορούν να χρησιμοποιηθούν
     - σταθερές τιμές
     - τα ?, $? σαν wildcards
     - μεταβλητές (μονότιμες ή πολλαπλών τιμών)
     - περιορισμοί λογικής σύνδεσης (&, |, ~)
     - περιορισμοί κατηγορήματος (:<κλήση-συνάρτησης>)
     - περιορισμοί επιστρεφόμενης τιμής (= <κλήση-συνάρτησης>)

Κανόνες (3)

2. Διεύθυνσης προτύπου

   Ανατίθεται η τιμή της διεύθυνσης ενός γεγονότος σε μια μεταβλητή:

   <μεταβλητή> <- <συνθήκη προτύπου>

3. Εκτιμήσιμες

   Για περιπτώσεις που έχουμε ως υποθέσεις συγκρίσεις αριθμητικών τιμών:

   (test <κλήση-συνάρτησης>)

4. Λογικά Συνδεόμενες

   Χρήση των λογικών συνδετικών (and, or, not) για δημιουργία σύνθετων συνθηκών.

Κανόνες (4)

```
(deffacts data-facts
   (data 1.0 blue “red”)
   (data 1 green)
   (data 1 blue red)
   (data 1 gray RED)
   (data 1 blue red 6.9 “05”))

(deftemplate person
   (slot name)
   (slot age)
   (multislot friends))

(deffacts people
   (person (name paul) (age 20))
   (person (name john) (age 20))
   (person (name paul) (age 30))
   (person (name niki) (age 35)
           (friends maria helen)))
```

```
(data 1 green)
(person (name paul))
(person (name paul) (age 20))
(data ? blue red $?)
(person (name ?) (age 20))
(person)
(data ?x blue red $?y)
(data 1 ~blue&~green $?)
(data 1 ?x~blue&~green $?y)
(data 1 gray|green $?y)
(person (name ?y&~paul) (age 20))
(data ?x&:(floatp ?x) blue $?y)
(data 1 blue $?x&:(> (length $?x) 2))
(person (name paul) (age ?z&:(> ?z 20)))
```

Κανόνες (4)

```
(deffacts data-facts
   (data 1.0 blue “red”)
   (data 1 green)
   (data 1 blue red)
   (data 1 gray RED)
   (data 1 blue red 6.9 “05”))

(deftemplate person
   (slot name)
   (slot age)
   (multislot friends))

(deffacts people
   (person (name paul) (age 20))
   (person (name john) (age 20))
   (person (name paul) (age 30))
   (person (name niki) (age 35)))
```

```
(defrule example-1
   (person (name paul) (age ?y))
   (test (> ?y 20))
   =>
   (printout t “this paul is “ ?y crlf))

(defrule example-2
   (person (name paul) (age ?x))
   (person (name ?y) (age ?z&:(> ?z ?x)))
   =>
   (printout t ?y “is older than pauls“ crlf))

(defrule example-3
   (or (and (temp high) (valve closed))
       (and (temp low) (valve open)))
   =>
   (printout t “there is a problem“ crlf))
```

Οι Συναρτήσεις του CLIPS (1)

Η κλήση μιας συνάρτησης γίνεται με την δήλωση:

(<όνομα-συνάρτ.> όρισμα1 όρισμα2 .. όρισμαΝ)

Για παράδειγμα αν μέσα σε ένα πρόγραμμα CLIPS υπάρχει η εντολή (assert (The number is (+ 2 3))) τότε το γεγονός το οποίο θα αποθηκευτεί στη μνήμη θα είναι το (The number is 5).

Οι Συναρτήσεις του CLIPS (2)

- Αριθμητικές
  - +, -, *, /
- Σύγκρισης
  - <, >, =, >=, <=, <>
- Λογικές
  - and, or, not, eq, neq
- Ελέγχου τύπου
  - numberp, symbolp, floatp, integerp, stringp

Οι Συναρτήσεις του CLIPS (3)

- Χειρισμού Πολλαπλών Τιμών
  - (create$ <values>) : Π.χ. (create$ a b c) → (a b c)
  - (explode$ <string>)
    Π.χ. (explode$ “give the ball”) → (give the ball)
  - (implode$ <multivalue>)
    Π.χ. (implode$ (give the ball)) → “give the ball”
  - (nth$ N <multivalue>) : Π.χ. (nth$ 2 (a b c)) → b
  - (member$ <symbol> <multivalue>)
    Π.χ. (member$ c (a b c)) → 3
  - (first$ <multivalue>) : (first$ (a b c)) → a
  - (rest$ <multivalue>) : (rest$ (a b c)) → (b c)

Οι Συναρτήσεις του CLIPS (4)

- Εξόδου
  - (printout <συσκευή> <έκφραση>)
  - Π.χ. (printout t “The day is” ?type crlf)→
    The day is sunny
- Εισόδου
  - (read)
- Ανάθεσης
  - (bind <μεταβλητή> <τιμή>)
  - Π.χ. (bind ?name (read))

Οι Συναρτήσεις του CLIPS (5)

- Ελέγχου Ροής
  - (while (συνθήκη) do
    (εντολή 1)
    …
    (εντολή ν))
  - (if (συνθήκη)
    then (εντολή 1) … (εντολή ν)
    else (εντολή 1) … (εντολή μ))

Οι Συναρτήσεις του CLIPS (6)

- Ορισμός Συνάρτησης
  - (deffunction <όνομα-συν> (<μεταβλητές>)
    (εντολή 1)
    …
    (εντολή ν))

Οι Συναρτήσεις του CLIPS (7)

- Παράδειγμα Ορισμού Συνάρτησης

```
(deffunction findmax3 (?x ?y ?z)
      (if (> ?x ?y)
          then if (> ?x ?z)
                   then (return ?x)
                   else (return ?z)
          else if (> ?y ?z)
                   then (return ?y)
                   else (return ?z)))
```

- Κλήση Συνάρτησης

(findmax3 23 14 56) → 56

Επίλυση Συγκρούσεων (1)

Αναφέρεται στην επιλογή ενός κανόνα από το σύνολο των κανόνων (σύνολο σύγκρουσης: conflict set) που βρίσκεται σε κάθε βήμα στη στοίβα υποψηφίων για πυροδότηση κανόνων (agenda).

Η επιλογή στηρίζεται (α) στην προτεραιότητα κάθε κανόνα και (β) στη στρατηγική επίλυσης σύγκρουσης που έχει επιλεγεί.

Στην πραγματικότητα, ο κανόνας αυτός είναι πάντα ο πρώτος στην agenda.

Προτεραιότητα

Σύνταξη: (declare (salience <number>))

Επίλυση Συγκρούσεων (2)

Προτεραιότητα

Παράδειγμα

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

Στρατηγικές Επίλυσης Συγκρούσεων (1)

(1) depth: Οι «νέοι» κανόνες πάνω από τους «παλιούς». Π.χ. αν fact-1 ενεργοποιεί τους r1, r2 και fact-2 ενεργοποιεί τους r3 και r4 και fact-1 έχει εισαχθεί πριν από το fact-2 (δηλ. το fact-2 είναι πιο πρόσφατο) τότε οι r3, r4 τοποθετούνται πάνω από τους r1, r2. Μεταξύ τους οι r1, r2 και r3, r4 τοποθετούνται αυθαίρετα.

(2) breadth: Οι «παλιοί» κανόνες πάνω από τους «νέους». (το αντίστροφο από την depth).

Στρατηγικές Επίλυσης Συγκρούσεων (2)

(3) simplicity: ένας νεοεισερχόμενος κανόνας τοποθετείται υψηλότερα από όλους με ίση ή μεγαλύτερη εξειδίκευση (specificity). Εξειδίκευση = αριθμός συγκρίσεων ή/και κλήσεων συναρτήσεων στο LHS. Π.χ. ο κανόνας

```
(defrule r-example
   (item ?x ?y ?x)
   (test (and (numberp ?x) (> ?x (+ 10 ?y)) (< ?x 100)))
=>)
```

έχει εξειδίκευση = 5.

Στρατηγικές Επίλυσης Συγκρούσεων (3)

(4) complexity: ένας νεοεισερχόμενος κανόνας τοποθετείται υψηλότερα από όλους με ίση ή μικρότερη εξειδίκευση. (το αντίστροφο της simplicity)

(5) LEX: ένας νεοεισερχόμενος κανόνας με μεγαλύτερη επικαιρότητα (recency) τοποθετείται υψηλότερα. Σε περίπτωση ίδιας επικαιρότητας χρησιμοποιείται ως κριτήριο η εξειδίκευση.

(6) MEA: με βάση την επικαιρότητα της πρώτης συνθήκης μόνο. Σε περίπτωση σύμπτωσης χρησιμοποιείται η LEX.

(7) random: Ένας τυχαίος αριθμός χρησιμοποιείται για να ξεχωρίσουν κανόνες της ίδιας προτεραιότητας.

Στρατηγικές Επίλυσης Συγκρούσεων (4)

Ορισμός στρατηγικής:

(set-strategy <strategy>)

Ανίχνευση τρέχουσας στρατηγικής:

(get-strategy)

Οι Βασικές Εντολές Περιβάλλοντος

load

Σύνταξη:(load "<File_Name>")

reset

Σύνταξη:(reset)

run

Σύνταξη: (run)

clear

Σύνταξη:(clear)