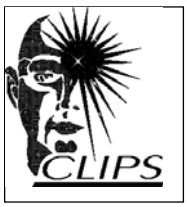
ΠΑΡΑΡΤΗΜΑ 2

Το Σύστημα Κανόνων CLIPS

Το CLIPS (C L*anguage* I*ntegrated* P*roduction* S*ystem*) είναι ένα περιβάλλον που προσφέρει δυνατότητες για προγραμματισμό με κανόνες, αντικείμενα και συναρτήσεις. Αναπτύχθηκε από τη NASA με σκοπό να αποτελέσει μια χαμηλού κόστους πλατφόρμα ανάπτυξης έμπειρων συστημάτων, αντικαθιστώντας τα ήδη υπάρχοντα συστήματα τα οποία βασίζονταν στη γλώσσα LISP.



Η ανάπτυξη του CLIPS άρχισε το 1984 και η πρώτη έκδοση ήταν έτοιμη την άνοιξη του 1985. Για την ανάπτυξή του χρησιμοποιήθηκε η γλώσσα C, έτσι ώστε το τελικό προϊόν να είναι μεταφέρσιμο σε όλα τα γνωστά λειτουργικά συστήματα (DOS, Windows, UNIX, VMS) και να είναι εύκολα επεκτάσιμο. Η πρώτη πλήρης έκδοση του CLIPS ήταν η 3.0, η οποία ήταν έτοιμη το καλοκαίρι του 1986. Την έκδοση αυτή ακολούθησαν οι εκδόσεις 4.0, 4.1 και 4.2, οι οποίες περιείχαν σημαντικές βελτιώσεις στον αρχικό κώδικα, χωρίς να προσθέτουν όμως σημαντικά στοιχεία. Η έκδοση 5.0, που παρουσιάστηκε το 1991, επέκτεινε το αρχικό σύστημα σημαντικά, υποστηρίζοντας εκτός από προγραμματισμό με κανόνες, διαδικαστικό (Procedural Programming) και αντικειμενοστραφή προγραμματισμό (Object Oriented Programming). Η αντικειμενοστραφής γλώσσα προγραμματισμού που παρέχεται με το CLIPS ονομάζεται COOL (CLIPS Object-Oriented Language). Η έκδοση 5.1 υποστήριζε καινούργια διεπαφή χρήστη (user interface) για τα νέα λειτουργικά συστήματα και εμφανίστηκε το 1991. Η τελευταία έκδοση είναι η 6.22 η οποία παρουσιάστηκε το καλοκαίρι του 2004. Η αρχική σχεδίαση του συστήματος έγινε έτσι ώστε να είναι συμβατό με το ART, ένα εμπορικό εργαλείο ανάπτυξης έμπειρων συστημάτων.

Το παράρτημα αυτό αποτελεί μια σύντομη παρουσίαση του προγραμματισμού με κανόνες και συναρτήσεις του CLIPS. Το παρόν κείμενο δε φιλοδοξεί να γίνει πλήρης οδηγός εκμάθησης, αλλά να χρησιμεύσει σα μια μικρή εισαγωγή στο περιβάλλον και τη γλώσσα προγραμματισμού. Πλήρης περιγραφή των δυνατοτήτων του συστήματος περιέχεται στα εγχειρίδια χρήσης. Τα παραδείγματα συστημάτων γνώσης που υπάρχουν στο τέλος του Παραρτήματος βρίσκονται στην ιστοσελίδα του βιβλίου **aibook.csd.auth.gr**.

Π2.1 Δομή του CLIPS

Το CLIPS είναι ένα *διερμηνευόμενο τυπικό σύστημα παραγωγής* (*interpreted production system*), το οποίο υποστηρίζει την *ορθή ακολουθία εκτέλεσης* (*forward chaining*). Τα κύρια μέρη του συστήματος είναι:

- Η *λίστα γεγονότων* (*facts list*), η οποία αντιστοιχεί στη *μνήμη εργασίας* (*working memory*) των συστημάτων παραγωγής. Όπως δηλώνει και το όνομά της, είναι ο χώρος στον οποίο αποθηκεύονται τα *γεγονότα* (*facts*), τόσο εκείνα που ορίζονται κατά την εκκίνηση του συστήματος, όσο και εκείνα που δημιουργούνται κατά την εκτέλεσή του.
- Η *βάση κανόνων* (*rule base / knowledge base*), όπου περιέχονται οι κανόνες. Αν και οι κανόνες μπορούν να ορισθούν μέσα από το περιβάλλον του συστήματος, συνήθως είναι αποθηκευμένοι σε κάποιο αρχείο απλού κειμένου (text document), το οποίο φορτώνεται στο σύστημα.
- Ο *μηχανισμός εξαγωγής συμπερασμάτων* (*inference engine*), ο οποίος ελέγχει τη λειτουργία ολόκληρου του συστήματος. Ο μηχανισμός αυτός προσφέρει ένα πλήθος από *στρατηγικές επίλυσης συγκρούσεων* (*conflict resolution strategies*) για την επιλογή του κανόνα που θα πυροδοτηθεί. Το σύνολο των υποψήφιων κανόνων για πυροδότηση αποτελεί το σύνολο σύγκρουσης ή την *ατζέντα* (*conflict set* ή *agenda*) του συστήματος.

Ένα πρόγραμμα στο CLIPS είναι ένα σύνολο από κανόνες και γεγονότα και η εκτέλεση του συνίσταται σε μια ακολουθία από πυροδοτήσεις κανόνων, των οποίων οι συνθήκες ικανοποιούνται. Η ικανοποίηση των συνθηκών γίνεται μέσω ταυτοποίησής τους με τα γεγονότα που υπάρχουν στη λίστα γεγονότων. Η εκτέλεση τερματίζεται όταν δεν υπάρχουν άλλοι κανόνες προς πυροδότηση ή όταν κληθεί συγκεκριμένη εντολή τερματισμού. Ο κύκλος λειτουργίας του συστήματος 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 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).
```

Π2.2 Σύνταξη του CLIPS

Η σύνταξη της γλώσσας που προσφέρει το σύστημα CLIPS είναι απλή και θυμίζει εκείνη της γλώσσας προγραμματισμού LISP. Στη συνέχεια, παρουσιάζεται η σύνταξη αυτή ξεκινώντας από τα βασικά δομικά στοιχεία της και συνεχίζοντας με την παρουσίαση των μεταβλητών, των γεγονότων και των κανόνων. Θα πρέπει να σημειωθεί ότι στο CLIPS υπάρχει διαχωρισμός κεφαλαίων και πεζών χαρακτήρων (case-sensitive).

Παράδειγμα:

Η εντολή

```
(create$ the day is (grey as it was))
```

επιστρέφει:

```
(the day is grey as it was)
```

explode$

Σύνταξη: **(explode$ <string>)**

Επιστρέφει μια πολλαπλή τιμή την οποία δημιουργεί από το αλφαριθμητικό. Η εντολή είναι όμοια με την προηγούμενη, διαφέροντας μόνο στο είδος των ορισμάτων που δέχεται.

Παράδειγμα:

Η εντολή

```
(explode$ "the night was blue")
```

θα επιστρέψει:

```
(the night was blue)
```

implode$

Σύνταξη: **(implode$ <multivalue>)**

Επιστρέφει το αντίστοιχο αλφαριθμητικό από μια πολλαπλή τιμή.

Παράδειγμα:

Η εντολή

```
(implode$ (explode$ "the night was blue"))
```

θα επιστρέψει **"the night was blue"**.

nth$

Σύνταξη: **(nth$ N <multivalue>)**

Επιστρέφει το N-οστό πεδίο μιας πολλαπλής τιμής.

Παράδειγμα:

Η εντολή

```
(nth$ 2 (create$ 3 4 5))
```

θα επιστρέψει την τιμή **4**.

member$

Σύνταξη: **(member$ <symbol> <multivalue>)**

Π2.8 Η Γλώσσα COOL

Το CLIPS υποστηρίζει τη δυνατότητα αντικειμενοστραφούς προγραμματισμού μέσω της γλώσσας COOL (CLIPS Object-Oriented Language). Η COOL έχει πέντε βασικά χαρακτηριστικά των αντικειμενοστραφών γλωσσών προγραμματισμού: *αφαίρεση* (*abstraction*), *εγκλεισμό* (*encapsulation*), *κληρονομικότητα* (*inheritance*), *πολυμορφισμό* (*polymorphism*) και *δυναμική δέσμευση* (*dynamic binding*).

Η *αφαίρεση* είναι μία περισσότερο διαισθητική, υψηλού επιπέδου αναπαράσταση μίας σύνθετης έννοιας. Στη γλώσσα COOL, ο ορισμός νέων κλάσεων επιτρέπει την αφαίρεση νέων τύπων δεδομένων. Οι ιδιότητες (slots) και οι μέθοδοι (message-handlers) αυτών των κλάσεων περιγράφουν τις ιδιότητες (properties) και τη συμπεριφορά (behavior) μίας καινούριας ομάδας αντικειμένων.

Τα χαρακτηριστικά ενός αντικειμένου δεν είναι απευθείας προσβάσιμα στον υπόλοιπο κόσμο, δηλαδή στο υπόλοιπο πρόγραμμα. Συνήθως η εσωτερική κατάσταση του αντικειμένου αποκρύπτεται. Αυτή η ιδιότητα ονομάζεται *εγκλεισμός* των ιδιοτήτων του αντικειμένου. Η COOL υποστηρίζει την ιδιότητα του εγκλεισμού, απαιτώντας την αποστολή μηνυμάτων (messages) για το χειρισμό στιγμιοτύπων (instances) των κλάσεων, ορισμένων από το χρήστη. Ένα στιγμιότυπο δε μπορεί να αποκριθεί σε ένα μήνυμα για το οποίο δεν έχει καθορισμένη μέθοδο.

Οι κλάσεις είναι συνήθως οργανωμένες σε ιεραρχίες. Οι πιο γενικές κλάσεις είναι τοποθετημένες ψηλά στην ιεραρχία, ενώ οι πιο συγκεκριμένες χαμηλότερα. Οι κλάσεις που βρίσκονται ψηλά στην ιεραρχία έχουν κάποια γενικά χαρακτηριστικά και μεθόδους τα οποία είναι κοινά για όλες τις κλάσεις που βρίσκονται χαμηλότερα στην ιεραρχία. Για την αποφυγή της επανάληψης ορισμού κοινών χαρακτηριστικών και μεθόδων υπάρχει η *κληρονομικότητα*, σύμφωνα με την οποία η δομή και η συμπεριφορά μιας γενικότερης κλάσης κληρονομείται στις περισσότερο συγκεκριμένες.

Ακόμη, η COOL υποστηρίζει την *πολλαπλή κληρονομικότητα* (*multiple inheritance*), όπου μία κλάση δε συνδέεται ιεραρχικά μόνο με μία γενικότερη κλάση αλλά με περισσότερες. Η κλάση που συνδέεται με πολλαπλές γενικότερες κλάσεις κληρονομεί χαρακτηριστικά και μεθόδους από όλες. Η COOL, χρησιμοποιώντας την υπάρχουσα ιεραρχία των κλάσεων, ορίζει μία λίστα, τη *λίστα προτεραιότητας κλάσεων* (*class precedence list*), για μία νέα κλάση. Αντικείμενα, τα οποία είναι στιγμιότυπα της νέας κλάσης, μπορούν να κληρονομήσουν ιδιότητες και συμπεριφορά από κάθε μία από τις κλάσεις της λίστας αυτής. Η λέξη *προτεραιότητα* υποδηλώνει ότι μία μέθοδος μίας κλάσης, που είναι πρώτη στη λίστα, υπερισχύει της ίδιας μεθόδου άλλης κλάσης που βρίσκεται πιο μετά στη λίστα.

Δύο αντικείμενα διαφορετικής κλάσης μπορεί να απαντήσουν στο ίδιο μήνυμα με ένα εντελώς διαφορετικό τρόπο. Αυτή η ιδιότητα ονομάζεται *πολυμορφισμός*. Ο πολυμορφισμός επιτυγχάνεται επισυνάπτοντας μεθόδους που έχουν το ίδιο όνομα αλλά εκτελούν διαφορετικές ενέργειες, στις κλάσεις των δύο αντικειμένων αντίστοιχα.

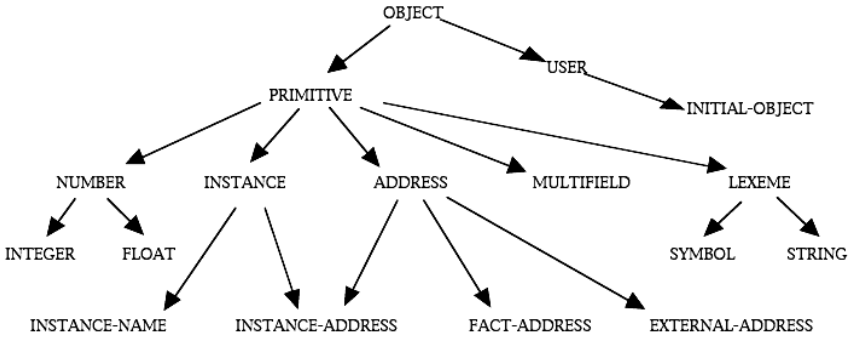
Η ιδιότητα της *δυναμικής δέσμευσης* υποστηρίζεται στην COOL με την έννοια ότι μία *αναφορά αντικειμένου* (*object reference*) κατά την κλήση μίας συνάρτησης δεν έχει τιμή μέχρι τη στιγμή της εκτέλεσης. Για παράδειγμα, ένα όνομα στιγμιότυπου (instance-name) ή μία μεταβλητή μπορεί να αναφέρεται σε ένα αντικείμενο όταν

στέλνεται ένα μήνυμα και να αναφέρεται σε ένα άλλο αντικείμενο κάποια άλλη στιγμή αργότερα.

Π2.8.1 Προκαθορισμένες Κλάσεις Συστήματος

Η COOL ορίζει δεκαεπτά κλάσεις συστήματος: **OBJECT**, **USER**, **INITIAL-OBJECT**, **PRIMITIVE**, **NUMBER**, **INTEGER**, **FLOAT**, **INSTANCE**, **INSTANCE-NAME**, **INSTANCE-ADDRESS**, **ADDRESS**, **FACT-ADDRESS**, **EXTERNAL-ADDRESS**, **MULTIFIELD**, **LEXEME**, **SYMBOL** και **STRING**. Αυτές οι κλάσεις δεν πρέπει να διαγραφούν ή να τροποποιηθούν. Στο Σχήμα Π2.1 απεικονίζονται οι σχέσεις κληρονομικότητας μεταξύ αυτών των κλάσεων.

Όλες οι κλάσεις συστήματος, εκτός της **INITIAL-OBJECT**, είναι *αφηρημένες* (*abstract*) κλάσεις, δηλαδή χρησιμοποιούνται μόνο για λόγους κληρονομικότητας. Άμεσα αντικείμενα αυτών των κλάσεων δεν επιτρέπονται. Καμία από αυτές τις κλάσεις δεν έχει ιδιότητες και, εκτός από την **USER**, καμία δεν έχει μεθόδους. Ωστόσο, ο χρήστης μπορεί να επισυνάψει μεθόδους σε όλες τις κλάσεις συστήματος εκτός από τις κλάσεις **INSTANCE**, **INSTANCE-ADDRESS** και **INSTANCE-NAME**. Η κλάση **OBJECT** είναι η υπερκλάση όλων των άλλων κλάσεων, συμπεριλαμβανομένων και των ορισμένων από το χρήστη κλάσεων.



Σχήμα Π2.1: Ιεραρχία των προκαθορισμένων κλάσεων συστήματος.

Όλες οι κλάσεις, που ορίζονται από το χρήστη, θα έπρεπε, αλλά δεν απαιτείται, να κληρονομούν άμεσα ή έμμεσα από την κλάση **USER**, γιατί αυτή η κλάση έχει όλες τις βασικές μεθόδους του συστήματος, όπως *αρχικοποίηση* (*initialization*) και *διαγραφή* (*deletion*).

Π2.8.2 Ορισμός Κλάσεων Χρήστη

Στην COOL οι κλάσεις ορίζονται μέσω της ειδικής συνάρτησης **defclass**. Με τη **defclass** καθορίζονται οι ιδιότητες και η συμπεριφορά μίας κλάσης αντικειμένων.

defclass

Σύνταξη:

```
(defclass <name> [<comment>]
        (is-a <superclass-name>+)
```

τερη από αυτή του **?b1**. Αν ισχύει αυτό, το **?b1** δεν είναι επιλέξιμο γιατί απλά δεν έχει τη μέγιστη ηλικία. Αν χρησιμοποιηθεί η **do-for-all-instances** στο παράδειγμα αυτό, τότε κάθε φορά που βρίσκεται κάποιο αγόρι με τη μέγιστη ηλικία αυτό θα διαγράφεται με αποτέλεσμα η ερώτηση να συνεχίζεται για τα υπόλοιπα αγόρια όπου η μέγιστη ηλικία θα αντιστοιχεί πλέον σε άλλη χαμηλότερη τιμή! Άρα τελικά θα διαγραφούν πολύ περισσότερα αντικείμενα από όσα θα έπρεπε.

Π2.9 Παραδείγματα

Στη συνέχεια παρουσιάζονται τρία παραδείγματα πλήρων συστημάτων γνώσης σε CLIPS. Το πρώτο χρησιμοποιεί αδόμητα γεγονότα, το δεύτερο χρησιμοποιεί πρότυπα γεγονότων, ενώ το τρίτο χρησιμοποιεί αντικείμενα της COOL.

Π2.9.1 Κίνηση Ρομπότ

Το παράδειγμα αυτό αποτελεί υλοποίηση σε CLIPS του παραδείγματος κίνησης ρομπότ που παρουσιάστηκε στο κεφάλαιο των *Συστημάτων Κανόνων*. Η υλοποίηση διαφέρει σε μερικά σημεία από τη θεωρητική παρουσίαση του παραδείγματος εφόσον οι στρατηγικές επίλυσης συγκρούσεων του συστήματος CLIPS δε συμπίπτουν με αυτές του θεωρητικού παραδείγματος. Συγκεκριμένα, στο CLIPS υπάρχουν επτά προκαθορισμένες στρατηγικές οι οποίες δεν μπορούν να συνυπάρχουν την ίδια χρονική στιγμή, με αποτέλεσμα να μην μπορεί να υπάρξει ο συνδυασμός των στρατηγικών που απαιτεί το θεωρητικό παράδειγμα, δηλαδή *αποφυγή επανάληψης*, *επιλογή του πιο ειδικού* και *τυχαία επιλογή*, με αυτήν τη σειρά.

Το πρόβλημα έγκειται κυρίως στο ότι δεν μπορούν να συνυπάρξουν οι στρατηγικές της επιλογής του πιο ειδικού με την τυχαία επιλογή. Στο συγκεκριμένο παράδειγμα, η τυχαία επιλογή είναι απαραίτητη για την επιλογή τυχαίας κατεύθυνσης όταν το ρομπότ πέφτει πάνω σε εμπόδια, ενώ η επιλογή του πιο ειδικού χρειάζεται για να δώσει προτεραιότητα στους κανόνες αποφυγής εμποδίων έναντι αυτών της κίνησης. Για το δεύτερο υπάρχει εναλλακτική λύση μέσω της χρήσης των προτεραιοτήτων κανόνων του CLIPS (salience), έτσι η στρατηγική που χρησιμοποιείται τελικά είναι η **random**.

Άλλες διαφοροποιήσεις σε σχέση με το θεωρητικό παράδειγμα είναι ο κανόνας αρχικοποίησης ο οποίος εισάγει τα δυναμικά γεγονότα, αν και η χρήση του δεν είναι απαραίτητη, η αποφυγή της εξόδου του ρομπότ από το πλέγμα, ανάγοντάς το σε αποφυγή εμποδίων, και ο τερματισμός της εκτέλεσης όταν βρεθεί κάποιο αντικείμενο.

```
(deffacts static-facts
   ;;; Εμπόδια
   (obstacle_at 4 2)  (obstacle_at 5 2)  (obstacle_at 9 2)  (obstacle_at 2 3)
   (obstacle_at 7 4)  (obstacle_at 5 6)  (obstacle_at 7 7)  (obstacle_at 3 8)
   (obstacle_at 6 8)
   ;;; Δυνατές κατευθύνσεις κίνησης
   (choice w)  (choice e)  (choice n)  (choice s)
```