Κεφάλαιο 1

Εισαγωγή

Σε αυτό το κεφάλαιο αρχικά παρουσιάζονται ο προστακτικός και ο δηλωτικός προγραμματισμός. Μετά, ακολουθεί η παρουσίαση των κύριων χαρακτηριστικών των γλωσσών του Λογικού Προγραμματισμού. Στη συνέχεια, παρουσιάζονται τα βασικά χαρακτηριστικά και η εξέλιξη της Prolog. Τέλος, γίνεται μια σύντομη παρουσίαση των υπόλοιπων κεφαλαίων αυτού του βιβλίου.

1.1. Προστακτικός και Δηλωτικός Προγραμματισμός.

Προγραμματιστική μέθοδος ή προγραμματιστική προσέγγιση (programming paradigm) είναι ο τρόπος που προγραμματίζουμε. Δηλαδή, ο τρόπος που βλέπουμε ένα πρόβλημα, δηλαδή τον κόσμο του προβλήματος, όταν το γράφουμε σε πρόγραμμα. Μια μέθοδος προγραμματισμού τείνει να επιβάλει στους χρήστες της κάποια εικόνα, άποψη για το πρόβλημα, και στον τρόπο προσέγγισης της λύσης του προβλήματος. Κάθε μέθοδος ή προσέγγιση προγραμματισμού εμπερικλείει κάποια σημασιολογία η οποία οδηγεί τον προγραμματιστή να σκέφτεται και να βλέπει το πρόβλημα με βάση αυτή τη σημασιολογία.

Οι κυριότερες προγραμματιστικές προσεγγίσεις είναι οι εξής:

- *Προστακτική μέθοδος (Imperative paradigm)*. Με αυτή την προσέγγιση το πρόγραμμα είναι μια ακολουθία από εντολές οι οποίες εκτελούνται διαδοχικά. Το πρόγραμμα λέει στον υπολογιστή πώς να κάνει κάτι και ως συνέπεια της ακολουθίας αυτών των πράξεων τί αποτέλεσμα θα πάρει. Οι γλώσσες που ακολουθούν αυτή την προσέγγιση χαρακτηρίζονται ως *προστακτικές γλώσσες*.
- *Δηλωτική μέθοδος (Declarative paradigm)*. Το πρόγραμμα λέει στον υπολογιστή τί θέλει να κάνει το πρόγραμμα (δηλαδή τί πρόβλημα να λύσει), και ο υπολογιστής καθορίζει πώς θα το κάνει (δηλαδή καθορίζει πώς θα λύσει το πρόβλημα). Οι γλώσσες που ακολουθούν αυτή την προσέγγιση χαρακτηρίζονται ως *δηλωτικές γλώσσες*.

Το σημαντικότερο χαρακτηριστικό των προστακτικών γλωσσών (imperative languages) προγραμματισμού είναι *η σημαντικότητα της σειράς με την οποία γίνονται οι υπολογισμοί*, η σειρά εκτέλεσης των πράξεων και των εντολών. Η σημασιολογία των *προστακτικών γλωσσών* βασίζεται σε καταστάσεις. Οι υπολογισμοί θεωρούνται ως μια διαδικασία μετάβασης από μια κατάσταση σε μια άλλη. Κάθε βήμα υπολογισμών σε αυτές τις γλώσσες περιγράφει τη διαδικασία αλλαγής μιας κατάστασης. Οι κυριότερες κατηγορίες προστακτικών γλωσσών προγραμματισμού είναι οι διαδικαστικές γλώσσες όπως η Pascal, η C και άλλες καθώς και αντικειμενοστραφείς γλώσσες όπως η C++, η Java και άλλες.

Το σημαντικότερο χαρακτηριστικό των *δηλωτικών γλωσσών (declarative languages)* προγραμματισμού είναι *η εστίαση της σημασιολογίας τους στη λογική* (δηλαδή στο *τί κάνει το πρόγραμμα*) και όχι στον *έλεγχο* (δηλαδή στο *πώς θα το κάνει, πώς θα λύσει το πρόβλημα*). Οι κυριότερες κατηγορίες δηλωτικών γλωσσών είναι οι παρακάτω. *Οι γλώσσες του λογικού προγραμματισμού* όπως η Prolog, η Gödel, κτλ. *Οι γλώσσες του συναρτησιακού λογικού προγραμματισμού* όπως η Lisp, Haskell,, κτλ. *Οι γλώσσες του περιοριστικού προγραμματισμού (Constraint programming)* όπως η Prolog III, CHIP, κτλ. *Οι γλώσσες επερωτήσεων σε βάσεις δεδομένων* όπως η SQL, η XQuery και άλλες.

Κάθε γενεά γλωσσών προγραμματισμού στόχευε σε ένα υψηλότερο επίπεδο αφαίρεσης ώστε λεπτομέρειες που έχουν σχέση με το υλικό να μη φαίνονται καθιστώντας την γλώσσα περισσότερο φιλική στον προγραμματιστή, πιο ευέλικτη και με μεγαλύτερες προγραμματιστικές δυνατότητες. Οι γλώσσες 4^{ης} γενεάς υποστήριζαν διαχείριση βάσεων δεδομένων, ανάπτυξη γραφι-

κής διεπικοινωνίας και ανάπτυξη εφαρμογών διαδικτύου. Οι δημοφιλείς συμβατικές γλώσσες προγραμματισμού όπως C, Java, κτλ, είναι γλώσσες 3ης γενεάς οι οποίες επεκτάθηκαν με βιβλιοθήκες ώστε να υποστηρίζουν χαρακτηριστικά των γλωσσών 4η γενεάς. Τα κύρια χαρακτηριστικά αυτών των γλωσσών είναι η προστακτική περιγραφή των προβλημάτων και η περισσότερη προγραμματιστική ευελιξία στον προγραμματιστή. Ο προγραμματισμός σε αυτές τις γλώσσες αντανακλούσε την von Neumann αρχιτεκτονική των υπολογιστών. Δηλαδή, το πρόγραμμα αποτελείται από μια *ακολουθία από εντολές οι οποίες επεξεργάζονται τα δεδομένα του προβλήματος*. Σε αυτή την ακολουθία υπάρχουν κάποιες *εντολές ελέγχου* οι οποίες επηρεάζουν τις επόμενες προς εκτέλεση εντολές. Αυτός ο τρόπος προσέγγισης του προγραμματισμού ονομάζεται *προστακτικός προγραμματισμός (imperative programming)*. Στον *προστακτικό* προγραμματισμό η έμφαση δίνεται *στη σειρά εκτέλεσης των υπολογισμών,* δηλαδή στο «***πώς;***» (*πώς θα επιλυθεί το πρόβλημα;*). Ο προστακτικός προγραμματισμός ονομάζεται συχνά και *διαδικαστικός προγραμματισμός*, όμως ουσιαστικά ο *διαδικαστικός προγραμματισμός* είναι μια κατηγορία προστακτικού προγραμματισμού. Στις προστακτικές γλώσσες προγραμματισμού όπως Pascal, C, Java, κτλ., ο προγραμματισμός εκφράζεται από την γνωστή αποφθεγματική εξίσωση του Wirth [Wirth, 1976].

Προγράμματα = Αλγόριθμοι + Δομές Δεδομένων

Σύμφωνα με αυτή την εξίσωση, *«τα προγράμματα είναι συγκεκριμένες μορφοποιήσεις αφηρημένων αλγορίθμων με βάση κάποιες αναπαραστάσεις και δομές δεδομένων.»*.

Η άλλη προσέγγιση προγραμματισμού, εισάγεται με τον *δηλωτικό προγραμματισμό (declarative programming)* η οποία σε αντίθεση με τον προστακτικό προγραμματισμό διαχωρίζει τη *λογική περιγραφή του προβλήματος* από τον *τρόπο επεξεργασίας της περιγραφής από τον υπολογιστή* για να βρει τις λύσεις του προβλήματος. Η Prolog ανήκει στις *δηλωτικές* γλώσσες προγραμματισμού. Σε αυτές τις γλώσσες, προγραμματισμός σημαίνει περιγραφή του προβλήματος και των περιορισμών του και όχι μια ακολουθία εντολών, στην εξίσωση του Wirth αναφέρεται ως αλγόριθμος επίλυσής του όπως γίνεται στον *προστακτικό προγραμματισμό*. Στον *δηλωτικό προγραμματισμό (declarative programming)* η έμφαση είναι στην *τυπική περιγραφή του προβλήματος*, δηλαδή στο «***ποιο;***» (*ποιο είναι το προς επίλυση πρόβλημα;*). Στον ιδανικό δηλωτικό προγραμματισμό, ο προγραμματιστής χρειάζεται να δώσει μόνο το *λογικό μέρος του προβλήματος* ως ένα σύνολο από ορισμούς (σχέσεων και συναρτήσεων) και *ο έλεγχος της λύσης του προβλήματος* θα δίνεται

αυτόματα από το υπολογιστικό σύστημα, π.χ. την Prolog. Στην περίπτωση της Prolog ο προγραμματιστής περιγράφει το πρόβλημά του και τους περιορισμούς του σε προτάσεις της λογικής, δηλαδή σε προτάσεις γεγονότα και προτάσεις κανόνες. Η Prolog εφαρμόζει *συνεπαγωγική συλλογιστική (deductive reasoning)*, χρησιμοποιεί τον *μηχανισμό ελέγχου της* ο οποίος συστηματικά ερευνά τις προτάσεις του προγραμματιστή για να βρει τις λύσεις του προβλήματος. Ο προγραμματιστής ζητάει τις λύσεις του προβλήματός του με μια άλλη κατηγορία προτάσεων, τις *ερωτήσεις*. Η έρευνα από τη μηχανή αναζήτησης της Prolog γίνεται με σειρά από αριστερά προς τα δεξιά και από την κορυφή προς τη βάση. Η αρχή της αναζήτησης, δηλαδή η κορυφή του δένδρου αναζήτησης, ξεκινά από την ερώτηση του χρήστη. Αυτός ο τρόπος περιγραφής του προβλήματος και αναζήτησης των υπαρχόντων λύσεων είναι που κάνει διαφορετικό τον προγραμματισμό σε Prolog. Ο προγραμματισμός σε Prolog στηρίζεται στη φιλοσοφία της αποφθεγματικής εξίσωσης του Kowalski [Kowalski, 1979].

Αλγόριθμος = Λογική + Έλεγχος.

Σύμφωνα με αυτή την εξίσωση ισχύουν τα εξής. «Ένας *αλγόριθμος ή πρόγραμμα* θεωρείται ότι αποτελείται από το *τμήμα της λογικής* το οποίο καθορίζει τη *γνώση που απαιτείται για την επίλυση του προβλήματος* και το *τμήμα ελέγχου* το οποίο προσδιορίζει τις *στρατηγικές επίλυσης του προβλήματος* οι οποίες χρησιμοποιούν αυτή τη *γνώση*. Η αποτελεσματικότητα ενός αλγορίθμου μπορεί να βελτιωθεί απλά βελτιώνοντας το τμήμα ελέγχου χωρίς να αλλάξει η λογική του αλγορίθμου» [Kowalski, 1979]. Αυτή η αποφθεγματική εξίσωση είναι αρκετά διαφορετική από την αποφθεγματική εξίσωση του Wirth. Στην εξίσωση του Wirth υπάρχουν και οι «δομές δεδομένων» οι οποίες εμπεριέχονται στο τελικό πρόγραμμα. Οι δομές δεδομένων περιέχουν την αναπαράσταση των δεδομένων ενός προβλήματος. Η αναπαράσταση των δεδομένων ενός προβλήματος στην Prolog μπορεί να γίνει με δύο τρόπους. Ο ένας τρόπος αναπαράστασης είναι ως όροι (terms) σε ορίσματα κατηγορημάτων. Οι όροι είναι ο μοναδικός τρόπος δόμησης των δεδομένων σε Prolog. Οι όροι έχουν δυναμική αναπαράσταση στα ορίσματα των κατηγορημάτων και μπορούν να αλλάζουν ενώ το πρόγραμμα τρέχει. Αυτός ο τρόπος αναπαράστασης εμπεριέχεται στην αποφθεγματική εξίσωση του Kowalski στη «Λογική» διότι οι όροι είναι τμήμα της λογικής του προβλήματος. Ο δεύτερος τρόπος αναπαράστασης των δεδομένων είναι ως προτάσεις γεγονότα της Prolog που αντιστοιχούν σε προτάσεις της λογικής. Τα δεδομένα με αυτό τον

τρόπο αναπαράστασης μπορούν να έχουν δυναμική επεξεργασία αρκεί τα αντίστοιχα κατηγορήματα να δηλωθούν ως δυναμικά. Ο δεύτερος τρόπος αναπαράστασης εμπεριέχεται επίσης στην αποφθεγματική εξίσωση του Kowalski στη «Λογική». Συνεπώς, η «*αναπαράσταση των δεδομένων του προβλήματος*» εμπεριέχεται στη «Λογική» της αποφθεγματικής εξίσωσης του Kowalski. Η αποφθεγματική εξίσωση του Kowalski θα μπορούσε ισοδύναμα να γραφτεί και ως εξής.

Πρόγραμμα = Λογική + Έλεγχος

Οι δηλωτικές γλώσσες έχουν τα εξής πλεονεκτήματα σε σύγκριση με τις προστακτικές ή διαδικαστικές γλώσσες.

1. Ένα δηλωτικό πρόγραμμα έχει καθαρότερη σημασιολογία από ένα προστακτικό. Αυτό σημαίνει ότι η σχέση του δηλωτικού προγράμματος με τον κόσμο του προβλήματος είναι ευκρινής και υπάρχει λογική αμεσότητα σε αντίθεση με τον προστακτικό προγραμματισμό. Στο προστακτικό πρόγραμμα ο προγραμματιστής πρέπει να περιγράψει βήμα-βήμα πώς θα υπολογιστεί το αποτέλεσμα. Ενώ σε ένα δηλωτικό πρόγραμμα, ο προγραμματιστής περιγράφει το πρόβλημά του χωρίς να περιγράφει τον τρόπο λύσης του. Υπάρχει διαχωρισμός της περιγραφής του προβλήματος από τον τρόπο λύσης του. Στο δηλωτικό πρόγραμμα περιγράφονται οι σχέσεις που υπάρχουν μεταξύ των οντοτήτων του προβλήματος.
2. Ένα δηλωτικό πρόγραμμα μπορεί να έχει αυτόματη επεξεργασία από μετα-προγράμματα. Για τα δηλωτικά προγράμματα μπορεί να κατασκευαστούν μετα-προγράμματα με απλό τρόπο τα οποία θα τα μετασχηματίζουν αυτόματα. Αυτό σημαίνει ότι τα δηλωτικά προγράμματα μπορούν να συντηρούνται ευκολότερα και με πιο αυτόματο τρόπο από τα προστακτικά προγράμματα.
3. Οι αλλαγές στον κώδικα σε προστακτικά προγράμματα μπορεί να δημιουργήσουν παρενέργειες (side effects) λόγω αλληλεξαρτήσεων. Αυτό είναι δύσκολο να συμβεί στο δηλωτικό προγραμματισμό.

1.2. Κύρια Χαρακτηριστικά των Γλωσσών του Λογικού Προγραμματισμού.

Υπάρχουν πολλές απόψεις με επικαλύψεις, από τις οποίες μπορούμε να δούμε το Λογικό Προγραμματισμό (ΛΠ). Η κάθε άποψη τονίζει διαφορετικά

χαρακτηριστικά του ΛΠ που τον διακρίνουν από τα άλλα προγραμματιστικά παραδείγματα όπως ο προστακτικός προγραμματισμός. Αυτά τα χαρακτηριστικά τα οποία θα αναλύσουμε περισσότερο είναι τα παρακάτω.

1. Στο ΛΠ οι υπολογισμοί (computations) είναι μια διαδικασία παραγωγής (deduction).
2. Στο ΛΠ έχουμε απόδειξη θεωρημάτων.
3. Στο ΛΠ έχουμε δηλωτικό προγραμματισμό σε αντίθεση με τον προστακτικό προγραμματισμό.
4. Υπάρχει άλλη έννοια για το τι είναι πρόγραμμα στο ΛΠ.

Πρόγραμμα = Λογική + Έλεγχος.

5. Ο ΛΠ είναι μια πολύ υψηλού επιπέδου γλώσσα προγραμματισμού.
6. Στο ΛΠ έχουμε διαδικαστική ερμηνεία μιας δηλωτικής προδιαγραφής.

Στη συνέχεια θα αναλύσουμε κάθε μια από τις προαναφερθείσες απόψεις με τις οποίες μπορούμε να δούμε το ΛΠ.

1. Οι υπολογισμοί ως διαδικασία παραγωγής.

- Ο λογικός προγραμματισμός προσφέρει ένα διαφορετικό παράδειγμα για τους υπολογισμούς. Ο *υπολογισμός (computation)* είναι μια *λογική παραγωγή (deduction)*.
- Χρησιμοποιεί τη γλώσσα της λογικής για να εκφράσει δεδομένα και προγράμματα.

```
∀X ∀Y πατέρας(X ,Y) εάν γονέας(X ,Y) και άρρεν(X).
```

- Οι γλώσσες του ΛΠ χρησιμοποιούν τη Λογική Πρώτης Τάξης (ΛΠΤ) ή γνωστό ως Κατηγορηματικό Λογισμό Πρώτης Τάξης (ΚΛΠΤ) ή απλά Κατηγορηματικό Λογισμό (ΚΛ).
- Η έννοια της πρώτης τάξης αναφέρεται στο ότι μπορούμε να έχουμε ποσοδείκτες πάνω σε αντικείμενα (σταθερές, μεταβλητές) αλλά όχι πάνω σε σχέσεις (κατηγορήματα).

Μπορούμε να πούμε «Όλοι οι άνθρωποι είναι θνητοί»

```
∀X/Άνθρωπος θνητός(X)
```

αλλά δεν μπορούμε να πούμε «*Όλες οι ιδιότητες του Γιάννη*, π.χ. ψηλός, εργατικός, επιμελής, κτλ»,

```
∀P P(γιάννης)
```

2. Απόδειξη θεωρημάτων.

Ο λογικός προγραμματισμός χρησιμοποιεί την έννοια της *αυτόματης απόδειξης θεωρημάτων*. Η *απόδειξη θεωρημάτων (theorem proving)* παράγει μια επιθυμητή λύση από ένα *αρχικό σύνολο αξιωμάτων*. Δηλαδή στο ΛΠ η λύση ενός προβλήματος αντιστοιχεί σε μια διαδικασία απόδειξης θεωρήματος. Στην οποία διαδικασία η *θεωρία* είναι η *γνώση αναπαράστασης του προβλήματος* και το *θεώρημα* που αποδεικνύεται είναι *η λύση* του προβλήματος. Η λύση πρέπει να είναι *«δημιουργική» (constructive)* ώστε περισσότερες από μια αληθείς ή ψευδείς απαντήσεις να εξαχθούν.

3. Δηλωτικός προγραμματισμός.

Οι γλώσσες του ΛΠ είναι δηλωτικές γλώσσες προγραμματισμού. Σε μια δηλωτική γλώσσα καθορίζεται ***τί*** *χρειάζεται να υπολογιστεί* αλλά όχι ***πώς*** *θα υπολογιστεί*. Ο προγραμματιστής πρέπει να καθορίσει.

- Το *σύνολο των αντικειμένων* που εμπλέκονται στους υπολογισμούς.
- Τις *σχέσεις* που υπάρχουν μεταξύ αυτών των αντικειμένων.
- Οι *περιορισμοί* που πρέπει να ισχύουν στο πρόβλημα για να λυθεί.

Ο διερμηνέας ή ο μεταγλωττιστής της γλώσσας του λογικού προγραμματισμού αποφασίζει πώς θα ικανοποιηθούν οι περιορισμοί.

4. Άλλη έννοια για το τι είναι πρόγραμμα.

Ο *προστακτικός* προγραμματισμός εκφράζεται από την αποφθεγματική εξίσωση του Wirth

Προγράμματα = Αλγόριθμοι + Δομές Δεδομένων

Ο *δηλωτικός* προγραμματισμός σε γλώσσες Τεχνητής Νοημοσύνης όπως είναι η Prolog εκφράζεται από την αποφθεγματική εξίσωση του Kowalski

Αλγόριθμος = Λογική + Έλεγχος.

ή ισοδύναμα

Πρόγραμμα = Λογική + Έλεγχος.

Μπορούμε να δούμε το τμήμα της *λογικής* ως «*αναπαράσταση της γνώσης του προβλήματος*» που πρέπει να λυθεί. Το τμήμα της *λογικής* από μόνο του προσδιορίζει τις παραγόμενες λύσεις του προβλήματος. Μπορούμε να δούμε το τμήμα του *ελέγχου* ως «*ένα σύμβουλο (ένα μηχανισμό) σε μια υπολογιστική μηχανή (διερμηνέα ή μεταγλωττιστή) που την καθοδηγεί πώς πρέπει να προχωρήσει, η υπολογιστική μηχανή, για να λυθεί το πρόβλημα χρησιμοποιώντας την αναπαριστώμενη γνώση του προβλήματος*». Ουσιαστικά το τμήμα ελέγχου παριστάνει τις διαφορετικές στρατηγικές απόδειξης- θεωρημάτων.

5. Μια πολύ υψηλού επιπέδου γλώσσα προγραμματισμού.

Μια καλή γλώσσα προγραμματισμού δεν πρέπει να επιβαρύνει τον προγραμματιστή με μη ουσιαστικές λεπτομέρειες. Η εξέλιξη των γλωσσών προγραμματισμού ήταν προς την κατεύθυνση της απεμπλοκής του προγραμματιστή από λεπτομέρειες που δεν σχετιζόταν με το πρόβλημα αλλά αφορούσαν τη διαδικασία λύσης του προβλήματος και τον υπολογιστή (συμβολικές/assembly γλώσσες, Fortran, ALGOL, Pascal, C, Java, κτλ). Οι γλώσσες του ΛΠ είναι μια ομάδα γλωσσών οι οποίες προσπαθούν να απομπλέξουνε τον προγραμματιστή από το να εισάγει στο πρόγραμμά του και τον μηχανισμό ελέγχου των υπολογισμών.

6. Διαδικαστική ερμηνεία μια δηλωτικής προδιαγραφής.

- Ας θεωρήσουμε τη λογική έκφραση

```
∀X ∀Y πατέρας(X ,Y) εάν γονέας(X ,Y) και άρρεν(X).
```

- Μπορεί να γραφτεί σε Prolog ως εξής:

```
πατέρας(X ,Y) :- γονέας(X ,Y), άρρεν(X).
```

- Αυτή η έκφραση έχει δύο ερμηνείες, τη *δηλωτική ερμηνεία* και τη *διαδικαστική ερμηνεία.*
- Η δηλωτική ερμηνεία είναι η εξής:

  «Ο X είναι πατέρας του Y εάν ο X είναι γονέας του Y και ο X είναι άρρεν».

  Η δηλωτική ερμηνεία εκφράζεται ως μια πρόταση αληθών προϋποθέσεων, «*ο X είναι γονέας του Y και ο X είναι άρρεν*», η οποία πρέπει να είναι αληθής για να ικανοποιείται η σχέση «*ο X είναι πατέρας του Y*».

- Η διαδικαστική ερμηνεία είναι η εξής:

«*Για να αποδειχθεί ότι ο Χ είναι πατέρας του Υ πρέπει να αποδειχθεί ότι ο Χ είναι γονέας του Υ και ότι ο Χ είναι άρρεν*».

Δηλαδή μια περιγραφή τι πρέπει να αποδειχθεί για να είναι αληθής η σχέση «*ο Χ είναι πατέρας του Υ*».

Λογικός Προγραμματισμός (*Logic Programming*) είναι μια προγραμματιστική προσέγγιση η οποία βασίζεται στην τυπική λογική. Ένα πρόγραμμα σε μια γλώσσα του λογικού προγραμματισμού είναι ένα σύνολο από προτάσεις σε λογική οι οποίες αναπαριστούν το πρόβλημα με γεγονότα και κανόνες.

1.3. Βασικά Χαρακτηριστικά και Εξέλιξη της Prolog.

Η λέξη Prolog in προκύπτει ως συντομογραφία από τις Γαλλικές λέξεις «PROgrammation en LOGique» ή κατ' αντιστοιχία στην Αγγλική «PROgramming LOGic» που σημαίνει «προγραμματισμός σε λογική». Η Prolog έχει τις ρίζες της στη μαθηματική λογική και συγκεκριμένα στη λογική πρώτης τάξεως. Είναι μια γλώσσα δηλωτικού προγραμματισμού, γενικού σκοπού.

Σε αυτή την ενότητα θα κάνουμε μια σύντομη αναδρομή στην ιστορική εξέλιξη της Prolog. Μετά, θα παρουσιάσουμε τη δηλωτική και τη διαδικαστική έννοια ενός Prolog προγράμματος και τα χαρακτηριστικά της Prolog που την ισχυροποιούν προγραμματιστικά για εφαρμογές Τεχνητής Νοημοσύνης. Στη συνέχεια, θα αναφερθούμε στα πεδία στα οποία η Prolog παίζει πρωτεύοντα ρόλο στην ανάπτυξη εφαρμογών καθώς και τα πλεονεκτήματά της στην ανάπτυξη λογισμικού. Τέλος, ακολουθεί η σημερινή εξέλιξη και οι επεκτάσεις της Prolog.

1.3.1. Η Ιστορική Εξέλιξη της Prolog.

Το πρώτο σύστημα της Prolog αναπτύχθηκε το 1972 στα πλαίσια ενός ερευνητικού έργου για επεξεργασία φυσικής γλώσσας στο Πανεπιστήμιο της Μασσαλίας από τον Alain Colmerauer και τον Philippe Roussel [Colmerauer, Roussel, 1993]. Ο Robert Kowalski από το πανεπιστήμιο του Εδιμβούργου συνεργάστηκε μαζί τους για την ανάπτυξη του πρώτου συστήματος της Prolog. Ο Kowalski ασχολήθηκε με το τμήμα που αφορούσε τη λογική του συστήματος. Οι διερμηνείς της Prolog ήταν πολύ αργοί πριν το 1983. Το 1983 ο David Warren πρότεινε ένα μοντέλο υλοποίησης της Prolog, το Warren Abstract Machine (WAM), το οποίο έχει γίνει στάνταρντ τεχνική υλοποίησης μεταγλωττιστών και διερμηνέων Prolog [Warren, 1983]. Το WAM ορίζει ένα

σύνολο εντολών υψηλού επιπέδου το οποίο απεικονίζεται, πολύ κοντά στον πηγαίο κώδικα της Prolog. Ο κώδικας της Prolog μεταγλωττίζεται σε κώδικα WAM ο οποίος στη συνέχεια μπορεί πολύ αποτελεσματικά να διερμηνευτεί σε εκτελέσιμο κώδικα.

1.3.2. Δηλωτική και Διαδικαστική Έννοια ενός Προγράμματος Prolog.

Σε ένα Prolog πρόγραμμα διακρίνουμε δύο επίπεδα εννοιών, τη *δηλωτική έννοια (declarative meaning)* και τη *διαδικαστική έννοια (procedural meaning)*.

1. Η *δηλωτική έννοια* ενδιαφέρεται μόνο για τις σχέσεις οι οποίες ορίζονται σε ένα λογικό πρόγραμμα. Δηλαδή, προσδιορίζει τι κάνει το πρόγραμμα αλλά όχι πώς κάνει τους υπολογισμούς.
2. Η *διαδικαστική έννοια* ενδιαφέρεται για τον υπολογισμό των σχέσεων, καθορίζει *πώς υπολογίζονται οι σχέσεις των αντικειμένων*. Δηλαδή προσδιορίζει πώς θα υπολογιστεί η έξοδος του προγράμματος.

Ακολουθεί ένα παράδειγμα που επεξηγεί τη δηλωτική και διαδικαστική έννοια.

❑ Παράδειγμα 1.1

Θεωρούμε την εξής πρόταση κανόνα της Prolog

```
son(X,Y) :- parent(Y,X), male(X).
```

η οποία αντιστοιχεί στην επόμενη πρόταση σε λογική.

```
son(X,Y) ← parent(Y,X) ∧ male(X).
```

Η δηλωτική έννοια της πρότασης της Prolog είναι η εξής: «Για όλα τα X και Y, ο X είναι γιός του Y εάν ο Y είναι γονέας του X και ο X είναι άρρεν.». Η *διαδικαστική έννοια* της πρότασης είναι η εξής: «Για να αποδειχθεί ότι ο X είναι γιός του Y, πρέπει πρώτα να αποδειχθεί ότι ο Y είναι γονέας του X και μετά να αποδειχθεί ότι ο X είναι άρρεν.». Σημείωση: Τα σύμβολα της λογικής που χρησιμοποιούμε σε αυτό το βιβλίο και η έννοιά τους παρουσιάζονται στο Κεφάλαιο 14.

1.3.3. Βασικά Χαρακτηριστικά της Prolog.

Η Prolog είναι ο κυριότερος εκπρόσωπος του *σχεσιακού λογικού προγραμματισμού* και η Lisp είναι ο κυριότερος εκπρόσωπος του *συναρτησιακού λογικού προγραμματισμού*. Αυτές οι δύο οικογένειες γλωσσών υποστηρίζουν δηλωτικό λογικό προγραμματισμό και χρησιμοποιούνται για ανάπτυξη εφαρμογών Τεχνητής Νοημοσύνης (ΤΝ) και όχι μόνο. Η Prolog βασίζεται σε ένα σύνολο μηχανισμών οι οποίοι την κάνουν αρκετά δυνατή και ευέλικτη γλώσσα προγραμματισμού. Τα βασικά χαρακτηριστικά που διακρίνουμε στην Prolog και στις άλλες γλώσσες ΤΝ είναι τα εξής:

1. Η ενοποίηση (unification).

2. Οι επαγωγικές δομές δεδομένων.

3. Η οπισθοδρόμηση (backtracking).

Για να γραφτεί ένα πρόγραμμα το οποίο επιτυχώς επιδεικνύει τεχνητή νοημοσύνη θα πρέπει να γίνει *αναπαράσταση* της σχετικής *γνώσης* ως προτάσεις εκφρασμένες στη λογική πρώτης τάξης σε μια *βάση γνώσης*, και μια αποτελεσματική *διαδικασία απόδειξης (proof procedure)* να επεξεργαστεί αυτή τη *βάση γνώσης*. Ένα τέτοιο υπολογιστικό σύστημα μπορεί να λύνει προβλήματα εξάγοντας τις λύσεις με τυπικό τρόπο ως θεωρήματα από τις προτάσεις της βάση γνώσης. Η Prolog είναι ένα τέτοιο ευφυές σύστημα, δηλαδή είναι ένα σύστημα απόδειξης θεωρημάτων. Η *θεωρία* είναι το πρόγραμμα Prolog, δηλαδή τα γεγονότα και οι κανόνες, και το *θεώρημα* είναι η ερώτηση ή ο στόχος ο οποίος έχει δοθεί προς απόδειξη. Για την *απόδειξη* της ερώτησης ή του στόχου χρησιμοποιούνται τεχνικές ΤΝ όπως οι εξής. Γίνεται αναζήτηση χώρου καταστάσεων του προβλήματος για την εύρεση των λύσεων. Η αναζήτηση του χώρου καταστάσεων στην Prolog ξεκινάει, από το στόχο (goal-driven) και προχωρά σε βάθος-πρώτα (depth-first). Εφαρμόζεται οπισθοδρόμηση σε περίπτωση είτε αποτυχίας ή για εύρεση περισσότερων λύσεων. Κάθε βήμα απόδειξης γίνεται με την εφαρμογή του συμπερασματικού κανόνα της επίλυσης (resolution). Η βασική πράξη που εφαρμόζεται για την υλοποίηση των βημάτων απόδειξης είναι η ενοποίηση. Συνεπώς, ένα πρόγραμμα σε Prolog είναι η βάση γνώση ενός ευφυούς συστήματος από την οποία εξάγονται οι λύσεις του προβλήματος με τη μηχανή συλλογισμών της Prolog.

1.3.4. Prolog και Ανάπτυξη Λογισμικού.

Η Prolog είναι μια γλώσσα η βάση της οποίας είναι η κατηγορηματική λογική. Έχει υλοποιηθεί με τεχνικές Τεχνητής Νοημοσύνης. Οι εφαρμογές της είναι σε πεδία της ΤΝ όπως τα συστήματα γνώσης, η επεξεργασία φυσικής γλώσσας, εφαρμογές του σημασιολογικού ιστού, η μηχανική μάθηση, η αυτοματοποίηση της ανάπτυξης λογισμικού αλλά όχι μόνο. Η Prolog έχει επίσης ευρεία χρήση στην ανάπτυξη λογισμικού γενικότερα όπως σε βάσεις δεδομένων (επαγωγικές βάσεις δεδομένων), σε προγραμματισμό με περιορισμούς, (χρονοπρογραμματισμός, σχεδιασμός ενεργειών κτλ), γρήγορη κατασκευή πρωτοτύπων και άλλες. Τα πλεονεκτήματά της για ανάπτυξη λογισμικού είναι τα εξής:

1. Η Prolog είναι μια γλώσσα υψηλού επιπέδου βασισμένη στη λογική η οποία υποστηρίζει τυπική συλλογιστική (formal reasoning).
2. Η Prolog είναι κατάλληλη για την κατασκευή γρήγορων πρωτοτύπων (rapid prototyping).
3. Λόγω της απλότητας της σύνταξης τα προγράμματα Prolog συντηρούνται και επαναχρησιμοποιούνται εύκολα.
4. Η Prolog μπορεί να χρησιμοποιηθεί ως γλώσσα υλοποίησης εκτελέσιμων προδιαγραφών εφόσον οι προδιαγραφές έχουν εκφραστεί σε λογική.
5. Η Prolog θεωρείται κατάλληλη για κατασκευή μετα-προγραμμάτων. Η Prolog χειρίζεται τα προγράμματα ως δεδομένα επειδή δεν κάνει διάκριση μεταξύ δεδομένων και προγραμμάτων. Τα δεδομένα και τα προγράμματα μπορούν να παρασταθούν με τον ίδιο τρόπο σε Prolog όπως θα δούμε σε επόμενα κεφάλαια. Αυτό έχει ως συνέπεια η κατασκευή μετα-προγραμμάτων σε Prolog να είναι απλή.

Η Prolog εκτός από γλώσσα προγραμματισμού είναι ένα ευφυές σύστημα με όλα τα χαρακτηριστικά και τους μηχανισμούς ενός ευφυούς συστήματος. Προστέθηκε το Κεφάλαιο 14 για την καλύτερη και σε βάθος κατανόηση από τον αναγνώστη της θεωρίας που αφορά την αναπαράσταση γνώσης σε λογική και τους σχετικούς μηχανισμούς συλλογιστικής που είναι ενσωματωμένοι στην Prolog και την κατατάσσουν στα ευφυή συστήματα.

1.3.5. Η Εξέλιξη της Prolog Σήμερα.

Η βιβλιοθήκη της Prolog συνεχώς επεκτείνεται με νέα χαρακτηριστικά, νέα πακέτα κατηγορημάτων, για να έχει τη δυνατότητα ανάπτυξης εφαρμογών σε περισσότερες περιοχές με συνέπεια να γίνεται συνεχώς πιο δημοφιλής. Οι βιβλιοθήκες των περισσότερων υλοποιήσεων Prolog έχουν επεκταθεί με ειδικά τμήματα (modules) κατηγορημάτων, που παρέχουν την δυνατότητα προγραμματισμού με περιορισμούς σε διάφορα πεδία [Carlsson, 2012], [Wielemaker, 2012], [Wielemaker, Schrijvers, Triska, Lager, 2012]. Έτσι παρέχουν τη δυνατότητα για ανάπτυξη εφαρμογών σε χρονοπρογραμματισμό (scheduling), σχεδιασμό ενεργειών (planning), μοντελοποίηση ψηφιακών κυκλωμάτων, έλεγχος μοντέλων (model checking), κτλ.

Ο (παγκόσμιος) ιστός δίνει τη δυνατότητα για ανάπτυξη εφαρμογών σε τεχνητή νοημοσύνη. Η επεξεργασία φυσικής γλώσσας, η εξαγωγή πληροφορίας και γνώσης από τον ιστό καθώς και η ανάπτυξη εφαρμογών ιστού με συλλογιστική είναι μερικά παραδείγματα εφαρμογών τεχνητής νοημοσύνης στον ιστό. Η λογική, ο λογικός προγραμματισμός και η Prolog έχουν σημαντικό ρόλο και στο σημασιολογικό ιστό. Ο στόχος του σημασιολογικού ιστού είναι να παρέχει τη δυνατότητα για πολύ εξελιγμένα συστήματα διαχείρισης γνώσης [Antoniou, van Harmelen 2009]. Η Prolog έχει κάποια ελκυστικά χαρακτηριστικά όπως ασφαλή σημασιολογία και αυτόματη διαχείριση της μνήμης που την κάνουν κύρια υποψήφια για ανάπτυξη εύρωστων υπηρεσιών ιστού. Οι περισσότερες Prolog όπως η Sicstus και η SWI-Prolog έχουν επεκτείνει τις βιβλιοθήκες τους με τμήματα, για να είναι δυνατή η κατασκευή διαδικτυακών εφαρμογών σε περιβάλλον Prolog [Carlsson,2012], [Wielemaker, 2012], [Wielemaker, Schrijvers, Triska, Lager, 2012]. Υπάρχουν δύο προσεγγίσεις στη χρήση της Prolog για διαδικτυακές εφαρμογές. Στη μία προσέγγιση η Prolog είναι ένα τμήμα της διαδικτυακής εφαρμογής. Σε αυτή την περίπτωση, το ευφυές μέρος της διαδικτυακής εφαρμογής έχει υλοποιηθεί σε Prolog και το υπόλοιπο τμήμα σε άλλες τεχνολογίες. Στην άλλη προσέγγιση, η ίδια η Prolog λειτουργεί ως αυτόνομος HTTP διακομιστής. Έτσι όλη η διαδικτυακή εφαρμογή μπορεί να είναι μόνο σε Prolog. Αυτή η προσέγγιση την οποία έχει ακολουθήσει η SWI–Prolog έχει αρκετά πλεονεκτήματα που αφορούν την αποτελεσματικότητα στην ανάπτυξη μιας διαδικτυακής εφαρμογής, στην καταλληλόλητα για ανάπτυξη διαδικτυακών εφαρμογών που απαιτούν ασφάλεια, στην επεκτασιμότητα και στον σχετικά εύκολο εντοπισμό λαθών στην εφαρμογή [Wielemaker, Huang, van Der Meij, 2008].

Επιπλέον, εκτός από το HTTP πρωτόκολλο, οι επεκτάσεις της Prolog υποστηρίζουν την ανάλυση, την αναπαράσταση και τη δημιουργία εγγράφων του ιστού όπως HTML, XML και RDF έγγραφα. Με συνέπεια, να αναπτύσσονται εφαρμογές στο σημασιολογικό ιστό πλήρως σε περιβάλλον Prolog [Schreiber, etal, 2008].

1.4. Παρουσίαση των Υπόλοιπων Κεφαλαίων του Βιβλίου.

Στο Κεφάλαιο 2 παρουσιάζονται τα βασικά μέρη ενός προγράμματος Prolog. Επίσης, παρουσιάζεται με λεπτομέρεια η βασική πράξη του συμβολικού προγραμματισμού, δηλαδή η ενοποίηση.

Στο Κεφάλαιο 3 αρχικά παρουσιάζονται οι έννοιες της επαγωγής και της αναδρομής ώστε ο αναγνώστης να γνωρίζει τις βασικές έννοιες για δημιουργία αναδρομικών προγραμμάτων. Ακολουθεί η παρουσίαση του ειδικού όρου της λίστας καθώς και δύο τεχνικές κατασκευής αναδρομικών προγραμμάτων, κατασκευή δομής στην κεφαλή και κατασκευή δομής στο σώμα μιας πρότασης. Η μόνη εντολή επανάληψης που διαθέτει η Prolog είναι η αναδρομή. Αυτές οι τεχνικές θα βοηθήσουν τον αναγνώστη στην κατασκευή σύνθετων προγραμμάτων με αναδρομή. Στη συνέχεια παρουσιάζεται η αριθμητική σε Prolog που περιλαμβάνει τις βασικές αριθμητικές πράξεις και τους αντίστοιχους τελεστές. Για περισσότερες αριθμητικές πράξεις και τελεστές ο χρήστης μπορεί να συμβουλευτεί το εγχειρίδιο της γλώσσας Prolog που χρησιμοποιεί. Τέλος, ακολουθεί το πρώτο μεγάλο παράδειγμα με τις «8 βασίλισσες».

Στο Κεφάλαιο 4 αρχικά δίνονται οι βασικές έννοιες της SLD-παραγωγής, της SLD-απόρριψης και του δένδρου αναζήτησης, ακολουθούν παραδείγματα επίδειξης των εννοιών. Για περισσότερες λεπτομέρειες σχετικά με την SLD-παραγωγή, την SLD-απόρριψη και το δένδρο αναζήτησης λύσεων ο αναγνώστης μπορεί να συμβουλευτεί το Κεφάλαιο 14. Στη συνέχεια, παρουσιάζεται η οπισθοδρόμηση και η αποκοπή (!) με πολλά παραδείγματα γραφικής επίδειξης της αποκοπής και των συνεπειών της. Μετά, ακολουθεί η παρουσίαση της άρνησης που υποστηρίζει η Prolog. Τέλος, ακολουθεί το κατηγόρημα fail και παραδείγματα με τις διάφορες χρήσεις του.

Στο Κεφάλαιο 5 παρουσιάζεται το μοντέλο ελέγχου ροής σε προγράμματα Prolog με επίδειξη παραδειγμάτων. Τέλος, παρουσιάζονται οι διαφορές στην εκτέλεση μιας διαδικασίας σε Prolog και σε συμβατικές προστακτικές γλώσσες υψηλού επιπέδου.

Στο Κεφάλαιο 6 παρουσιάζεται η έννοια των τελεστών και πώς ο προγραμματιστής μπορεί να ορίσει τους δικούς του τελεστές. Παρουσιάζεται η οδηγία «:-op(Προτεραιότητα, Τύπος, Όνομα)» με την οποία δίνεται η δυνατότητα στον προγραμματιστή να ορίσει τους δικούς του τελεστές.

Στο Κεφάλαιο 7 παρουσιάζονται τα πιο σημαντικά ενσωματωμένα κατηγορήματα της Prolog ταξινομημένα σε ομάδες με βάση τις λειτουργίες τους.

Στο Κεφάλαιο 8 αρχικά παρουσιάζεται αναλυτική μεθοδολογία για κατασκευή μεγάλων Prolog προγραμμάτων. Στη συνέχεια παρουσιάζονται διάφορες προγραμματιστικές τεχνικές, όπως τα σχήματα προγραμμάτων, τα δυαδικά δένδρα, οι ανοικτές λίστες και οι λίστες διαφοράς. Τέλος, παρουσιάζεται ένα σύνολο από προχωρημένες δομές δεδομένων με πολλά παραδείγματα, ώστε ο αναγνώστης να μπορεί να υλοποιήσει σύνθετες και προχωρημένες εφαρμογές ΤΝ σε Prolog. Οι δομές δεδομένων που παρουσιάζονται είναι οι εξής: οι ακολουθίες, τα σύνολα, τα πολυσύνολα, οι πλειάδες, οι δυαδικές σχέσεις, τα γραφήματα, οι στοίβες και οι ουρές. Ο αναγνώστης που ολοκληρώνει με επιτυχία το Κεφάλαιο 8 θα είναι σε θέση να κατασκευάσει προχωρημένα και σύνθετα προγράμματα Prolog.

Στο Κεφάλαιο 9 παρουσιάζεται η κατασκευή μετα-προγραμμάτων σε Prolog. Αρχικά παρουσιάζεται το κύριο πρόβλημα του μετα-προγραμματισμού που είναι η αναπαράσταση του προγράμματος αντικείμενο (object program). Το πρόγραμμα αντικείμενο είναι τα δεδομένα ενός μετα-προγράμματος. Στη συνέχεια παρουσιάζονται τρόποι αναπαράστασης του προγράμματος αντικείμενο σε βασική και μη-βασική μορφή. Ακολουθεί η παρουσίαση μιας κατηγορίας μετα-προγραμμάτων, των διερμηνέων και η υλοποίησή τους σε Prolog. Τέλος, παρουσιάζονται οι υλοποιήσεις δύο βασικών πράξεων, οι οποίες χρειάζονται στο μετα-προγραμματισμό εφόσον το πρόγραμμα αντικείμενο αναπαρίσταται σε βασικούς όρους (ground representation).

Στο Κεφάλαιο 10 παρουσιάζεται η αναζήτηση σε χώρο καταστάσεων. Αρχικά παρουσιάζεται το θέμα της αναζήτησης σε θεωρητική βάση. Ακολουθούν υλοποιήσεις αναζήτησης σε κατευθυνόμενα μη κυκλικά και κυκλικά γραφήματα. Στη συνέχεια παρουσιάζονται αναλυτικά οι υλοποιήσεις σύνθετων προβλημάτων αναζήτησης χώρου καταστάσεων.

Στο Κεφάλαιο 11 παρουσιάζονται τα συστήματα γνώσης. Αρχικά γίνεται μια θεωρητική παρουσίαση των συστημάτων γνώσης στην οποία παρουσιά-

ζεται η αρχιτεκτονική τους. Ακολουθεί η παρουσίαση των συστημάτων γνώσης που στηρίζονται σε if-then κανόνες. Τέλος ακολουθούν παραδείγματα και η λεπτομερής υλοποίηση ενός συστήματος γνώσης.

Στο Κεφάλαιο 12 παρουσιάζεται η επεξεργασία φυσικής γλώσσας. Η Prolog διαθέτει ειδικές δυνατότητες για ανάπτυξη συστημάτων επεξεργασίας φυσικής γλώσσας. Αρχικά γίνεται θεωρητική παρουσίαση του θέματος. Στη συνέχεια παρουσιάζονται οι Γραμματικές Οριστικών Προτάσεων (Definite Clause Grammars), για τις οποίες οι Prolog διαθέτουν ένα βολικό συμβολισμό για έκφρασή τους. Μετά ακολουθεί η παρουσίαση συντακτικών αναλυτών της Αγγλικής και της Ελληνικής γλώσσας υλοποιημένες σε Γραμματικές Οριστικών Προτάσεων. Στη συνέχεια παρουσιάζεται η σημασιολογική αναπαράσταση των προτάσεων στο λ-λογισμό. Τέλος ακολουθούν παραδείγματα επεξεργασίας φυσικής γλώσσας.

Σε μια εποχή που το διαδίκτυο άλλαξε τον τρόπο που επικοινωνούμε, μελετάμε και εργαζόμαστε, η δυνατότητα για κατασκευή προγραμμάτων που τρέχουν στο διαδίκτυο ανήκει πλέον στις βασικές απαιτήσεις που πρέπει να υποστηρίζει μια σύγχρονη γλώσσα προγραμματισμού. Γι' αυτό προσθέσαμε το Κεφάλαιο 13 που αφορά τη δυνατότητα κατασκευής προγραμμάτων σε Prolog που να τρέχουν στο διαδίκτυο.

Στο Κεφάλαιο 14 παρουσιάζεται η μετάβαση από τη λογική στο λογικό προγραμματισμό. Η Prolog είναι μια γλώσσα του Λογικού Προγραμματισμού που έχει ενσωματωμένα τα κύρια χαρακτηριστικά των γλωσσών του Λογικού Προγραμματισμού που αναφέρθηκαν παραπάνω. Για να κατανοήσει ο αναγνώστης σε βάθος αυτά τα χαρακτηριστικά έχει προστεθεί το Κεφάλαιο 14. Δηλαδή, μελετώντας το Κεφάλαιο 14 ο αναγνώστης θα κατανοήσει σε βάθος έννοιες όπως οι εξής: α) γιατί οι υπολογισμοί είναι μια διαδικασία παραγωγής, β) γιατί έχουμε απόδειξη θεωρημάτων, γ) γιατί ένα πρόγραμμα αποτελείται από τη λογική και τον έλεγχο. Στο κεφάλαιο 14 παρουσιάζονται αρχικά οι έννοιες στο προτασιακό λογισμό που είναι πιο απλός από τον κατηγορηματικό λογισμό ώστε να διευκολυνθεί ο αναγνώστης στη κατανόηση του κατηγορηματικού λογισμού που ακολουθεί.

1.5. Ασκήσεις.

Άσκηση 1

Ποια είναι τα πλεονεκτήματα των δηλωτικών γλωσσών σε σύγκριση με τις προστακτικές γλώσσες.

Άσκηση 2

Ποιες είναι οι διαφορές των αποφθεγματικών εξισώσεων του Wirth και του Kowalski. Υπάρχουν ομοιότητες, αν ναι, ποιες;

Άσκηση 3

Ποια χαρακτηριστικά διακρίνουν το Λογικό Προγραμματισμό από τα άλλα προγραμματιστικά παραδείγματα;

Άσκηση 4

Να δώσετε ένα παράδειγμα το οποίο να εξηγεί την πρόταση/έκφραση «οι *υπολογισμοί (computations)* στο Λογικό Προγραμματισμό είναι μια *διαδικασία παραγωγής (deduction).».*

Άσκηση 5

Θεωρήσετε την εξής πρόταση κανόνα της Prolog,

sister(X,Y) :- parent(X,Z), parent(Y,Z), \= (X,Y), female(X).

η οποία αντιστοιχεί στην ακόλουθη πρόταση σε λογική.

sister(X,Y) ← parent(X,Z) ∧ parent(Y,Z) ∧ X≠Y ∧ female(X).

Να γράψετε τη δηλωτική και τη διαδικαστική έννοια της πρότασης της Prolog.

Άσκηση 6

Θεωρήσετε τη λογική έκφραση

∀X ∀Y παππούς(X ,Y) ***εάν*** *∃Z (γονέας(X ,Z)* ***και*** *γονέας(Z, Y)* ***και*** *άρρεν(X)).*

Να τη γράψετε ως πρόταση σε Prolog και να δώσετε τη δηλωτική και τη διαδικαστική ερμηνεία της.