Κεφάλαιο 1

Εισαγωγή

Η λέξη Prolog προκύπτει ως συντομογραφία από τις γαλλικές λέξεις «PROgrammation en LOGique» ή κατ' αντιστοιχία στην Αγγλική «PROgramming in 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 αυτό το κεφάλαιο θα κάνουμε μια σύντομη ιστορική αναδρομή στην εξέλιξη της Prolog. Η Prolog εισάγει μια νέα μορφή προγραμματισμού, το *δηλωτικό*, ενώ οι συμβατικές γλώσσες προγραμματισμού υψηλού επιπέδου, C, Java, κτλ, ακολουθούν τον *προστακτικό* προγραμματισμό. Μετά, θα μιλήσουμε για τις διαφορές του προστακτικού από το δηλωτικό προγραμματισμό, οι οποίες αντικατοπτρίζουν και τις διαφορές μεταξύ προστακτικών και δηλωτικών γλωσσών. Στη συνέχεια, θα παρουσιάσουμε τη δηλωτική και τη διαδικαστική έννοια ενός Prolog προγράμματος και τα χαρακτηριστικά της Prolog που την ισχυροποιούν προγραμματιστικά για εφαρμογές ΤΝ. Έπειτα, τα πεδία στα οποία η Prolog παίζει πρωτεύοντα ρόλο στην ανάπτυξη εφαρμογών καθώς και τα πλεονεκτήματά της στην ανάπτυξη λογισμικού. Ακολουθεί η εξέλιξη και οι επεκτάσεις της Prolog. Τέλος γίνεται μια σύντομη παρουσίαση των υπόλοιπων κεφαλαίων αυτού του βιβλίου.

1.1. 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 μεταφραστές 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 μεταφραστών 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 μεταφραστεί σε εκτελέσιμο κώδικα.

1.2. Προστακτικός και Δηλωτικός Προγραμματισμός.

Κάθε γενεά γλωσσών προγραμματισμού στόχευε σ' ένα υψηλότερο επίπεδο αφαίρεσης, ώστε λεπτομέρειες που έχουν σχέση με το υλικό να μη φαίνονται καθιστώντας τη γλώσσα περισσότερο φιλική στον προγραμματιστή, πιο ευέλικτη και με μεγαλύτερες προγραμματιστικές δυνατότητες. Οι γλώσσες $4^{ης}$ γενεάς υποστήριζαν διαχείριση βάσεων δεδομένων, ανάπτυξη γραφικής 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 $4^{η}$ γενεάς. Τα κύρια χαρακτηριστικά αυτών των γλωσσών είναι η αλγοριθμική περιγραφή των προβλημάτων και περισσότερη προγραμματιστική ευελιξία στον προγραμματιστή. Ο προγραμματισμός σε αυτές τις γλώσσες αντανακλούσε τη von Neumann αρχιτεκτονική των υπολογιστών. Δηλαδή, το πρόγραμμα αποτελείται από μια *ακολουθία από εντολές* οι οποίες εκτελούν κάποιες πράξεις και *ένα σύνολο από 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 στον *τρόπο εκτέλεσης των υπολογισμών*, δηλαδή στο «*πώς;*» (*πώς θα επιλυθεί το πρόβλημα;*). Ο προστακτικός προ-

γραμματισμός ονομάζεται και *διαδικαστικός προγραμματισμός*. Στις προστακτικές γλώσσες προγραμματισμού όπως Pascal, C, Java, κτλ., ο προγραμματισμός εκφράζεται από τη 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 δομών δεδομένων.».

Μια άλλη προσέγγιση προγραμματισμού, εισάγεται με το *δηλωτικό (declarative) προγραμματισμό* η οποία σε αντίθεση με τον προστακτικό προγραμματισμό, διαχωρίζει τη *λογική περιγραφή του προβλήματος* από τον *τρόπο επεξεργασίας της περιγραφής* για να βρει τις λύσεις του προβλήματος. Η Prolog ανήκει στις *δηλωτικές (declarative)* γλώσσες προγραμματισμού. Σε αυτές τις γλώσσες προγραμματισμός σημαίνει περιγραφή του προβλήματος και των περιορισμών του και όχι ο αλγόριθμος επίλυσής του όπως γίνεται στον *προστακτικό προγραμματισμό*. Στο *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 προγράμματος* σαν ένα σύνολο από ορισμούς (σχέσεων και συναρτήσεων) και *ο έλεγχος του προγράμματος* 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)* και χρησιμοποιεί το *μηχανισμό ελέγχου της* ο οποίος συστηματικά ερευνά τις προτάσεις του προγραμματιστή με ομοιόμορφη σειρά από αριστερά προς τα δεξιά και από την κορυφή προς τη βάση για να βρει τις λύσεις του προβλήματος. Ο προγραμματιστής ζητάει τις λύσεις του προβλήματος του με μια άλλη κατηγορία προτάσεων, τις *ερωτήσεις*. Η αρχή της αναζήτησης, δηλαδή η κορυφή του δέντρου αναζήτησης, ξεκινά από την ερώτηση του χρήστη. Αυτός ο τρόπος περιγραφής του προβλήματος και εύρεσης 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 χρησιμοποιείται στην επίλυση του προβλήματος και το *τμήμα ελέγχου* το οποίο προσδιορίζει τις *στρατηγικές επίλυσης του προβλήματος* μέσω των οποίων αυτή η γνώση χρησιμοποιείται. Η αποτελεσματικότητα ενός αλγορίθμου μπορεί να βελτιωθεί απλά βελτιώνοντας το τμήμα ελέγχου και χωρίς να αλλάξει η λογική του αλγορίθμου» [Kowalski, 1979]. Αυτή η αποφθεγματική εξίσωση είναι αρκετά διαφορική 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 σαν 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 μεταβλητές οι οποίες παριστούν τους όρους έχουν άλλη σημασιολογία και χρήση στην Prolog και στις άλλες γλώσσες του λογικού προγραμματισμού από αυτήν που έχει η μεταβλητή στις προστακτικές γλώσσες. Ο δεύτερος τρόπος αναπαράστασης των δεδομένων είναι σαν προτάσεις γεγονότα της Prolog. Τα δεδομένα με αυτόν τον τρόπο αναπαράστασης μπορούν να έχουν δυναμική επεξεργασία αρκεί τα αντίστοιχα κατηγορήματα να δηλωθούν σαν 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. Στο προστακτικό πρόγραμμα ο προγραμματιστής πρέπει να περιγράψει βήμα-βήμα πως θα υπολογιστεί το αποτέλεσμα. Ενώ σ' ένα δηλωτικό πρόγραμμα, ο προγραμματιστής περιγράφει το πρόβλημα του χωρίς να περιγράφει τον τρόπο λύσης του. Υπάρχει

διαχωρισμός 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. Μπορεί να υπάρχουν παρενέργειες (side effects) στο κώδικα σε προστακτικά προγράμματα λόγω αλληλεξαρτήσεων. Αυτό είναι δύσκολο να συμβεί στο δηλωτικό προγραμματισμό.

1.3. Δηλωτική και διαδικαστική έννοια ενός προγράμματος Prolog.

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

1. Η *δηλωτική έννοια* ενδιαφέρεται μόνο με τις σχέσεις οι οποίες ορίζονται σ' ένα λογικό πρόγραμμα. Δηλαδή, προσδιορίζει τι κάνει το *πρόγραμμα* αλλά όχι πώς το κάνει.
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) :- father(Y,X), male(X).
```

η οποία αντιστοιχεί στην ακόλουθη πρόταση σε λογική.

$$son(X,Y) \leftarrow father(Y,X) \wedge male(X).$$

Η δηλωτική έννοια της πρότασης της Prolog είναι η εξής: «Για όλα τα Χ και Υ, ο Χ είναι γιός του Υ, εάν ο Υ είναι πατέρας του Χ και ο Χ είναι άρρεν». Η *διαδικαστική έννοια* της πρότασης είναι η εξής: «Για να απαντηθεί η ερώτηση, εάν ο Χ είναι γιός του Υ, πρέπει πρώτα να απαντηθεί η σύζευξη των ερωτήσεων ότι ο Υ είναι πατέρας του Χ και ο Χ είναι άρρεν.».

1.4. Χαρακτηριστικά της Prolog που την ισχυροποιούν για εφαρμογές ΤΝ.

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 προγραμματισμό και χρησιμοποιούνται για ανάπτυξη εφαρμογών ΤΝ και όχι μόνο. Η Prolog βασίζεται σε ένα σύνολο μηχανισμών οι οποίοι την κάνουν αρκετά δυνατή και ευέλικτη γλώσσα προγραμματισμού. Οι μηχανισμοί στους οποίους βασίζεται είναι οι 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* σαν προτάσεις εκφρασμένες στη λογική πρώτης τάξης, και να προσαρτήσει σε αυτή τη *βάση γνώσης* μια αποτελεσματική *διαδικασία απόδειξης*. Ένα τέτοιο υπολογιστικό σύστημα μπορεί να λύνει προβλήματα εξάγοντας με τυπικό τρόπο θεωρήματα από τις προτάσεις της βάσης γνώσης. Η Prolog είναι ένα τέτοιο ευφυές σύστημα, δηλαδή είναι ένα σύστημα απόδειξης θεωρημάτων. Η *θεωρία* είναι το πρόγραμμα και το *θεώρημα* είναι η ερώτηση ή ο στόχος τον οποίο έχει να αποδείξει. Για την *απόδειξη* της ερώτησης ή του στόχου χρησιμοποιούνται τεχνικές ΤΝ όπως οι εξής. Γίνεται έρευνα χώρου καταστάσεων του προβλήματος για την εύρεση των λύσεων. Η έρευνα του χώρου καταστάσεων στην Prolog ξεκινάει από το στόχο (goal-driven) και προχωρά σε βάθος-πρώτα (depth-first). Εφαρμόζεται οπισθοδρόμηση σε περίπτωση είτε αποτυχίας είτε 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.

1.5. Prolog και Ανάπτυξη Λογισμικού.

Η Prolog είναι μια γλώσσα η βάση της οποία είναι η κατηγορηματική λογική. Έχει υλοποιηθεί με τεχνικές Τεχνητής Νοημοσύνης, δηλαδή είναι ένα ευφυές σύστημα. Οι εφαρμογές της είναι σε πεδία της ΤΝ όπως τα συστήματα γνώσης, η επεξεργασία φυσικής γλώσσας, οι εφαρμογές του σημασιολογικού ιστού, 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 σύνταξής της 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 σαν 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 σαν συνέπεια η κατασκευή μετα-προγραμμάτων σε Prolog να είναι απλή.

1.6. Η εξέλιξη της Prolog.

Η βιβλιοθήκη της Prolog συνεχώς επεκτείνεται με νέα χαρακτηριστικά, νέα πακέτα κατηγορημάτων, για να έχει τη δυνατότητα ανάπτυξης εφαρμογών σε περισσότερες περιοχές. Γι' αυτό γίνεται συνεχώς πιο δημοφιλής. Οι βιβλιοθήκες των περισσότερων υλοποιήσεων Prolog έχουν επεκταθεί με

ειδικά τμήματα (modules) κατηγορημάτων, που παρέχουν τη δυνατότητα προγραμματισμού με περιορισμούς σε διάφορα πεδία [Carlsson, 2012], [Wielemaker, 2012]. Έτσι παρέχουν τη δυνατότητα για ανάπτυξη εφαρμογών σε χρονοπρογραμματισμό (scheduling), σχεδιασμό ενεργειών (planning), μοντελοποίηση ψηφιακών κυκλωμάτων, έλεγχο μοντέλων (model checking), κτλ.

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]. Στο σημασιολογικό ιστό η λογική χρησιμοποιείται για την επεξεργασία των πληροφοριών που ανακτώνται και στην εξαγωγή συμπερασμάτων. Από τις ανακτώμενες πληροφορίες φτιάχνεται η βάση γνώσης από την οποία εξάγονται τα αποτελέσματα με βάση τις ερωτήσεις του χρήστη. Οι περισσότερες Prolog όπως η Sicstus και η SWI έχουν επεκτείνει τις βιβλιοθήκες τους με τμήματα, για να είναι δυνατή η κατασκευή διαδικτυακών εφαρμογών σε περιβάλλον Prolog [Carlsson, 2012], [Wielemaker, 2012]. Όμως, παράλληλα γίνονται και προτείνονται επεκτάσεις της Prolog ώστε να εξελιχθεί η Prolog σε γλώσσα προγραμματισμού διαδικτύου με ενσωμάτωση κατάλληλων χαρακτηριστικών [Wielemaker, Huang, van Der Meij, 2008].

1.7. Σύντομη παρουσίαση των υπόλοιπων κεφαλαίων του βιβλίου.

Στο Κεφάλαιο 2 παρουσιάζονται τα βασικά μέρη ενός προγράμματος Prolog. Επίσης, παρουσιάζεται με λεπτομέρεια η ενοποίηση. Στο Κεφάλαιο 3 αρχικά παρουσιάζονται οι έννοιες της επαγωγής και της αναδρομής. Ακολουθεί η παρουσίαση του ειδικού όρου της λίστας καθώς και δύο τεχνικών κατασκευής δομής, στην κεφαλή και στο σώμα μιας πρότασης. Η μόνη εντολή επανάληψης που διαθέτει η Prolog είναι η αναδρομή. Αυτές οι τεχνικές θα βοηθήσουν τον αναγνώστη στην κατασκευή 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 βασικοί ορισμοί της *SLD-εξαγωγής*, *της SLD-απόρριψης* και *του δέντρου αναζήτησης*, ακολουθούν παραδείγματα επίδει-

ξης των ορισμών. 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 Prolog με επίδειξη παραδειγμάτων. Τέλος, παρουσιάζονται οι διαφορές στην εκτέλεση μιας διαδικασίας σε Prolog και σε συμβατικές γλώσσες υψηλού επιπέδου. Στο Κεφάλαιο 6 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 η αναλυτική μεθοδολογία για κατασκευή Prolog προγραμμάτων. Στη συνέχεια παρουσιάζονται διάφορες προγραμματιστικές τεχνικές, όπως τα σχήματα προγραμμάτων, τα δυαδικά δέντρα, 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. Οι δομές δεδομένων που παρουσιάζονται είναι οι εξής: οι ακολουθίες, τα σύνολα, τα πολυσύνολα, οι στοίβες, οι δυαδικές σχέσεις και οι γράφοι. Ο αναγνώστης που με επιτυχία ολοκληρώνει 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 μεταφραστών 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 έρευνα σε χώρο καταστάσεων. Αρχικά παρουσιάζεται το θέμα της έρευνας σε θεωρητική βάση. Ακολουθούν υλοποιήσεις έρευνας σε κατευθυνόμενους μη κυκλικούς και κυκλικούς γράφους. Στη συνέχεια παρουσιάζεται αναλυτικά η υλοποίηση ενός σύνθετου προβλήματος έρευνας χώρου καταστάσεων. Στο Κεφάλαιο 11 παρουσιάζονται τα συστήματα γνώσης. Αρχικά γίνεται μια θεωρητική παρουσίαση των

συστημάτων γνώσης και παρουσιάζεται 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 επεξεργασία 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.

1.8. Ασκήσεις

Άσκηση 1

Θεωρείστε την εξής πρόταση κανόνα της Prolog,

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

η οποία αντιστοιχεί στην ακόλουθη πρόταση σε λογική.

$$son(X,Y) \leftarrow mother(Y,X) \wedge male(X).$$

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

Άσκηση 2

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

Άσκηση 3

Ποιες είναι οι διαφορές των αποφθεγματικών εξισώσεων του Wirth και του Kowalski;

Άσκηση 4

Σε ποιους μηχανισμούς βασίζεται η Prolog;

Άσκηση 5

Ποια είναι τα πλεονεκτήματα της Prolog στην ανάπτυξη λογισμικού;