Ανάπτυξη συντακτικού αναλυτή φυσικής γλώσσας με χρήση του φορμαλισμού LFG

Παναγιώτης Μίνος

18 Φεβρουαρίου 2014

Περίληψη

Η παρούσα μεταπτυχιακή διπλωματική εργασία αναφέρεται στον σχεδιασμό και την υλοποίηση ενός συντακτικού αναλυτή (parser), ο οποίος βασίζεται στον φορμαλισμό της Λεξικής-Λειτουργικής Γραμματικής (Lexical-Functional Grammar - LFG). Ο συντακτικός αναλυτής είναι δυνατόν να ενσωματωθεί σε συστήματα επεξεργασίας φυσικής γλώσσας (natural language processing - NLP). Επιπλέον, αναπτύχθηκε μία γραφική διεπαφή χρήστη (Graphical User Interface - GUI), η οποία επιτρέπει την άμεση προβολή της εξαγόμενης πληροφορίας.

Σύμφωνα με την προσέγγιση που υιοθετήσαμε, οι γραμματικές που χρησιμοποιεί ο συντακτικός αναλυτής εκλαμβάνονται ως ένα σύνολο κανόνων συμφραστικά ανεξάρτητης γραμματικής φραστικής δομής. Οι κανόνες είναι επισημειωμένοι με λειτουργικές εξισώσεις (functional equations). Η συντακτική ανάλυση μίας πρότασης φυσικής γλώσσας πραγματοποιείται σε δύο στάδια. Στο πρώτο στάδιο, εξάγονται οι συστατικές δομές (c-structures) της πρότασης με τη βοήθεια ενός συντακτικού αναλυτή για συμφραστικά ανεξάρτητες γραμματικές (context-free grammars), ο οποίος βασίζεται στον αλγόριθμο του Earley. Στο δεύτερο στάδιο, επιλύεται η λειτουργική περιγραφή (f-description) της πρότασης και δημιουργούνται οι λειτουργικές δομές (f-structures) για κάθε συστατικό της.

Για τη συντακτική ανάλυση των γραμματικών χρησιμοποιήθηκε το εργαλείο JavaCC (Java Compiler Compiler), ενώ οι αλγόριθμοι υλοποιήθηκαν στη γλώσσα προγραμματισμού Java, η οποία εξασφαλίζει ανεξαρτησία πλατφόρμας και εγγενή υποστήριξη του προτύπου κωδικοποίησης χαρακτήρων Unicode.

Λέξεις-Κλειδιά: επεξεργασία φυσικής γλώσσας (ΕΦΓ) / Natural Language Processing (NLP), συντακτική ανάλυση / parsing, συντακτικός αναλυτής / parser, Λεξική-Λειτουργική Γραμματική / Lexical-Functional Grammar (LFG)

Περιεχόμενα

Εισαγωγή **5**

Δομή εργασίας 7

1 Θεωρητικό υπόβαθρο **8**

1.1 Τυπικές γλώσσες και γραμματικές 8

1.1.1 Συντακτικός αναγνωριστής 11

1.1.2 Συντακτικός αναλυτής 11

1.2 Η Ιεραρχία του Chomsky 11

1.3 Ο αλγόριθμος του Earley 13

1.4 Γραμματικές βασισμένες σε περιορισμούς 16

1.4.1 Δομές ιδιοτήτων 16

1.4.2 Εγκλεισμός 18

1.4.3 Ενοποίηση και Γενίκευση 19

1.5 Η Λεξική-Λειτουργική Γραμματική 20

1.5.1 Συστατική δομή 20

1.5.2 Λειτουργική δομή 21

1.5.3 Λειτουργική περιγραφή 23

1.5.4 Η σχέση αντιστοίχισης συστατικής και λειτουργικής δομής 25

1.5.5 Λειτουργικά υποδείγματα 26

1.5.6 Λειτουργικές εξισώσεις 27

1.5.7 Λειτουργική αβεβαιότητα 30

2 Σχεδιασμός και υλοποίηση του συντακτικού αναλυτή 31
2.1 Εισαγωγή 31
2.2 Σχεδιασμός 32
2.2.1 Η αρχιτεκτονική του συστήματος 32
2.2.2 Το περιβάλλον ανάπτυξης 32
2.3 Υλοποίηση 35
2.4 Λειτουργία 39
2.4.1 Δημιουργία της συστατικής δομής 39
2.4.2 Δημιουργία της λειτουργικής δομής 39
2.5 Περιγραφή των αρχείων γραμματικής 41
2.5.1 Περιγραφή των κανόνων 41
2.5.2 Περιγραφή του λεξικού 43
2.5.3 Περιγραφή των λειτουργικών προτύπων 44
2.6 Η γραφική διεπαφή 46

3 Συμπεράσματα και μελλοντικές προοπτικές 49

Α΄ Γλωσσάρι 54

Β΄ Ακρωνύμια 56

Κατάλογος σχημάτων

1.1 Η ιεραρχία του Chomsky 13
1.2 πίνακας ιδιοτήτων-τιμών 17
1.3 απεικόνιση γλωσσικής πληροφορίας με πίνακα ιδιοτήτων-τιμών 17
1.4 Η αρχιτεκτονική παράλληλης προβολής της ΛΛΓ 20
1.5 Συστατική δομή σύμφωνα με τη ΛΛΓ 21
1.6 Λειτουργική δομή σύμφωνα με τη ΛΛΓ 22
1.7 Λειτουργική δομή σύμφωνα με τη ΛΛΓ 24
1.8 Λειτουργική δομή σύμφωνα με τη ΛΛΓ 25
1.9 Η σχέση αντιστοίχισης ϕ 26
1.10 Κανόνες και Λεξικό της ΛΛΓ 27
1.11 Συστατική δομή, επισημειωμένη με λειτουργικές εξισώσεις . . 28
1.12 Λειτουργική δομή σύμφωνα με τη ΛΛΓ 30

2.1 Η λογική αρχιτεκτονική του συντακτικού αναλυτή 33
2.2 Τα τμήματα και οι εξαρτήσεις του συντακτικού αναλυτή . . . 39
2.3 Η συστατική δομή 40
2.4 Συγγραφή γραμματικών 47
2.5 Συντακτική ανάλυση 48

Κατάλογος πινάκων

1.1 Οι περιορισμοί των γραμματικών 11

Εισαγωγή

Αντικείμενο της παρούσας εργασίας είναι η υλοποίηση ενός συντακτικού αναλυτή φυσικής γλώσσας σύμφωνα με τον φορμαλισμό της Λεξικής-Λειτουργικής Γραμματικής (LFG). Η έρευνά μας εντάσσεται στο επιστημονικό πεδίο της υπολογιστικής γλωσσολογίας.

Η υπολογιστική γλωσσολογία (computational linguistics – CL) είναι ο διεπιστημονικός τομέας που αντλεί τη θεωρία και τη μεθοδολογία του από τη γλωσσολογία και την πληροφορική, με σκοπό τη μοντελοποίηση της φυσικής γλώσσας από υπολογιστική σκοπιά. Εκτός από τη δημιουργία δομών δεδομένων και αλγορίθμων για την περιγραφή γλωσσικών φαινομένων, έχει και πρακτική συνεισφορά στην ανάπτυξη εφαρμογών και συστημάτων επεξεργασίας φυσικής γλώσσας (ΕΦΓ) (natural language processing - NLP) που επιτρέπουν την ανάλυση ή και παραγωγή γραπτού και προφορικού λόγου, τη μετάφραση από τη μία γλώσσα στην άλλη καθώς και την υλοποίηση διεπαφών για την επικοινωνία μεταξύ των ανθρώπων και των υπολογιστών.

Γενικά, τόσο από τους θεωρητικούς όσο και από τους υπολογιστικούς γλωσσολόγους αναγνωρίζονται πέντε επίπεδα ανάλυσης του γλωσσικού συστήματος:

Φωνητική-Φωνολογία

Η Φωνητική ασχολείται με την προφορά των λέξεων, καθεμιάς χωριστά ή συνδυασμένων μεταξύ τους μέσα σε προτάσεις. Η Φωνολογία ασχολείται με τη λειτουργία των φθόγγων μέσα σε ένα συγκεκριμένο γλωσσικό σύστημα.

Μορφολογία

Η Μορφολογία ασχολείται με την εσωτερική δομή των λέξεων και το σχηματισμό τους από τα συστατικά τους στοιχεία, τα μορφήματα.

Σύνταξη

Η Σύνταξη ασχολείται με τους τρόπους και τους κανόνες σύμφωνα με τους οποίους οι λέξεις συνδυάζονται μέσα σε μεγαλύτερες ενότητες, σχηματίζοντας φράσεις και προτάσεις.

Σημασιολογία

Η Σημασιολογία ασχολείται με τα νοήματα που εκφράζουν οι λέξεις και, πιθανόν και οι δομές, και τον τρόπο που αυτά τα νοήματα αλληλεπιδρούν για τη διαμόρφωση του νοήματος των προτάσεων.

Πραγματολογία

Η Πραγματολογία ασχολείται με την εξάρτηση του νοήματος μιας πρότασης από τα ευρύτερα (γλωσσικά και εξωγλωσσικά) συμφραζόμενα μέσα στα οποία εμφανίζεται.

Από τα πέντε παραπάνω επίπεδα ανάλυσης της φυσικής γλώσσας, η εργασία μας εστιάζει στο επίπεδο της σύνταξης. Η συντακτική ανάλυση είναι απαραίτητο στάδιο στην επεξεργασία φυσικής γλώσσας, ειδικά όταν πρόκειται για προηγμένες εφαρμογές όπως συστήματα μηχανικής μετάφρασης, εξαγωγής και ανάκτησης πληροφορίας, ερωτοαποκρίσεων κ.ά., τα οποία προϋποθέτουν την κατανόηση ή/και παραγωγή γραπτού ή προφορικού λόγου.

Για την συντακτική ανάλυση έχουν προταθεί και χρησιμοποιηθεί από τους υπολογιστικούς γλωσσολόγους διάφορες προσεγγίσεις με πιο διαδεδομένες αυτές που βασίζονται σε συμφραστικά ανεξάρτητες γραμματικές (CFG, PCFG) και σε γραμματικές που βασίζονται στη λειτουργία της ενοποίησης (LFG, HPSG, PATR).

Συγκεκριμένα, η δική μας προσέγγιση βασίζεται στον φορμαλισμό της Λεξικής-Λειτουργικής Γραμματικής (LFG). Στόχος μας είναι να υλοποιήσουμε έναν συντακτικό αναλυτή με βάση αυτόν τον φορμαλισμό, προκειμένου να συνεισφέρουμε στις, λιγοστές για την ώρα, υπάρχουσες υλοποιήσεις τέτοιου είδους. Επιπλέον παρέχουμε λογισμικό το οποίο είναι δυνατόν να εκτε-

λεστεί σε διάφορες πλατφόρμες και να χειριστεί αλφάβητα UNICODE και συνεπώς είναι δυνατόν να χρησιμοποιηθεί για όλες τις φυσικές γλώσσες.

Δομή εργασίας

Η εργασία αποτελείται από 3 κεφάλαια.

Στο κεφάλαιο 1 περιγράφονται βασικές θεωρητικές έννοιες και ορισμοί, καθώς και το θεωρητικό μοντέλο της Λεξικής-Λειτουργικής Γραμματικής (LFG).

Το κεφάλαιο 2 αναφέρεται στον σχεδιασμό και την υλοποίηση του συντακτικού αναλυτή.

Στο κεφάλαιο 3 παρουσιάζονται τα γενικά συμπεράσματα και οι τελικές διαπιστώσεις μας από αυτήν τη έρευνα, καθώς και προοπτικές για μελλοντική έρευνα.

Κεφάλαιο 1

Θεωρητικό υπόβαθρο

1.1 Τυπικές γλώσσες και γραμματικές

Οι τυπικές γλώσσες και γραμματικές έχουν χρησιμοποιηθεί στην υπολογιστική γλωσσολογία για την περιγραφή γλωσσικών φαινομένων που έχουν σχέση κυρίως με τη μορφολογία και τη σύνταξη. Θεωρούμε σκόπιμο να δώσουμε μερικούς βασικούς ορισμούς.

Αλφάβητο *Αλφάβητο* (alphabet) είναι ένα μη κενό, πεπερασμένο σύνολο Σ το οποίο περιέχει ένα ή περισσότερα μη περαιτέρω αναλύσιμα στοιχεία που ονομάζονται *σύμβολα* (symbols).

Συμβολοσειρά *Συμβολοσειρά* (string) ή *λέξη* (word) είναι μια πεπερασμένη ακολουθία από μηδέν, ένα ή περισσότερα σύμβολα που ανήκουν στο Σ. Μήκος n μίας συμβολοσειράς α είναι το πλήθος των συμβόλων της και το συμβολίζουμε με $|\alpha|$, $n \geq 0$. Η κενή συμβολοσειρά (empty string), όπου $n = 0$, συμβολίζεται ως ϵ. Το σύνολο όλων των συμβολοσειρών που μπορούν να παραχθούν από ένα αλφάβητο Σ συμβολίζεται ως Σ^*. Τέλος, ως Σ^+ συμβολίζεται το σύνολο των συμβολοσειρών που δεν περιέχει το ϵ, δηλαδή $\Sigma^+ = \Sigma^* - \{\epsilon\}$.

Τυπική γλώσσα Μια *τυπική γλώσσα* (formal language) L επί του αλφαβήτου Σ είναι ένα, γνήσιο ή μη, υποσύνολο του Σ^*.

Τυπική γραμματική Οι *τυπικές γραμματικές* (formal grammars) είναι αφηρημένες δομές που χρησιμοποιούνται για την ακριβή περιγραφή τυπικών γλωσσών. Μία τυπική γραμματική G ορίζεται ως μία διατεταγμένη τετράδα της μορφής $\langle N, T, P, S\rangle$ όπου:

N είναι ένα μη κενό, πεπερασμένο σύνολο που περιέχει τα *μη τερματικά* (non-terminal) σύμβολα της γραμματικής.

T είναι ένα μη κενό, πεπερασμένο σύνολο που περιέχει τα *τερματικά* (terminal) σύμβολα της γραμματικής. Τα σύνολα N και T είναι ξένα μεταξύ τους, δηλαδή $N \cap T = \varnothing$.

P είναι ένα μη κενό, πεπερασμένο σύνολο που περιέχει τους *κανόνες παραγωγής* (production rules) της γραμματικής. Οι κανόνες παραγωγής, που ονομάζονται και *κανόνες επανεγγραφής* (rewrite rules), είναι της μορφής $\alpha \to \beta$ όπου $\alpha \in (T \cup N)^+$ και $\beta \in (T \cup N)^*$.

S είναι ένα ένα διακεκριμένο σύμβολο που το ονομάζουμε *αρχικό σύμβολο* (start symbol) και το οποίο ανήκει στο σύνολο N, δηλαδή $S \in N$.

Σε έναν κανόνα παραγωγής της μορφής $\alpha \to \beta$, το αριστερό μέρος (α) ονομάζεται και *κεφαλή* (head) του κανόνα, και το δεξί μέρος (β) ονομάζεται και *σώμα* (body) του κανόνα.

Συντακτικά δέντρα Τα *συντακτικά δέντρα* (syntax trees, parse trees) ή *δέντρα παραγωγής* (derivation trees) είναι καταρχάς ένας γραφικός τρόπος αναπαράστασης της παραγωγής μίας συμβολοσειράς σύμφωνα με τους κανόνες μίας γραμματικής.

Ένα *δέντρο* είναι ένα πεπερασμένο σύνολο T από έναν ή περισσότερους κόμβους, τέτοιο ώστε:

- να περιλαμβάνει έναν ειδικό κόμβο που ονομάζεται *ρίζα* (root) του δέντρου, $root(T)$
- οι υπόλοιποι κόμβοι να μπορούν να διαμεριστούν σε ξένα μεταξύ τους σύνολα $T_1, \ldots, T_m$, όπου κάθε υποσύνολο είναι με τη σειρά του ένα δέντρο. Τα δέντρα $T_1, \ldots, T_m$ ονομάζονται υποδέντρα (subtrees) της ρίζας.

Ένας κόμβος που έχει 0 υποδέντρα ονομάζεται *τερματικός κόμβος* (terminal node) ή *φύλλο* (leaf), ενώ αν έχει τουλάχιστον ένα, ονομάζεται *μη τερματικός κόμβος* (non-terminal node) ή *κλαδί* (branch). Αν ένας κόμβος n_1 έχει ένα υποδέντρο με ρίζα τον κόμβο n_2, τότε ο n_1 είναι πρόγονος (ancestor) του n_2 και ο n_2 είναι απόγονος (descendant) ή παιδί (child) του n_1.

Ένα *διατεταγμένο δέντρο* (ordered tree) είναι ένα δέντρο με ρίζα στο οποίο η σειρά των παιδιών κάθε κόμβου είναι καθορισμένη.

Συντακτικό δέντρο σύμφωνα με μία γραμματική $G = \{N, T, P, S\}$ είναι ένα διατεταγμένο δέντρο όπου:

- η επιγραφή (label) της ρίζας του δέντρου περιέχει το αρχικό μη τερματικό σύμβολο S
- η επιγραφή κάθε κόμβου του δέντρου που δεν είναι τερματικός περιέχει ένα μη τερματικό σύμβολο της γραμματικής από το σύνολο N
- η επιγραφή κάθε φύλλου του δέντρου περιέχει ένα τερματικό σύμβολο της γραμματικής από το σύνολο T
- αν ο κόμβος n έχει επιγραφή A και οι κόμβοι $n_1, n_2, \ldots, n_k$ με επιγραφές $X_1, X_2, \ldots, X_k$ είναι παιδιά του n, σε διάταξη από τα αριστερά προς τα δεξιά, αντίστοιχα τότε ο $A \rightarrow X_1 X_2 \cdots X_k$ πρέπει να είναι κανόνας παραγωγής της γραμματικής από το σύνολο P

1.1.1 Συντακτικός αναγνωριστής

Συντακτικός αναγνωριστής (recognizer) για μία γραμματική G είναι ένας αλγόριθμος που δέχεται ως είσοδο μία συμβολοσειρά και αναγνωρίζει αν η συμβολοσειρά ανήκει ή όχι στην γλώσσα $L(G)$.

1.1.2 Συντακτικός αναλυτής

Συντακτικός αναλυτής (parser) για μία γραμματική G είναι ένας συντακτικός αναγνωριστής ο οποίος, εφόσον αναγνωρίσει οτι η συμβολοσειρά ανήκει στην γλώσσα $L(G)$, τότε κατασκευάζει τα συντακτικά δέντρα της συμβολοσειράς S.

1.2 Η Ιεραρχία του Chomsky

Ο γλωσσολόγος Noam Chomsky [2], [3] εισήγαγε μία μέθοδο κατάταξης των τυπικών γραμματικών με βάση ορισμένους περιορισμούς ως προς την μορφή των κανόνων παραγωγής τους (βλ. πίνακα 1.1).

Τύπος	Γραμματική	Κανόνες παραγωγής και περιορισμοί
0	χωρίς περιορισμούς	$\alpha \to \beta$, $\alpha \in (T \cup N)^+$, $\beta \in (T \cup N)^*$
1	συμφραστικά εξαρτημένη	$\alpha \to \beta$, $\alpha \in (T \cup N)^+$, $\beta \in (T \cup N)^*$ $\lvert\alpha\rvert \leq \lvert\beta\rvert$
2	συμφραστικά ανεξάρτητη	$A \to \alpha$, $A \in N$, $\alpha \in (T \cup N)^*$
3	κανονική (δεξιογραμμική)	$w \to x$ ή $w \to yz$, $w \in N$, $x \in T \cup \epsilon$, $y \in T$, $z \in N$
	κανονική (αριστερογραμμική)	$w \to x$ ή $w \to zy$, $w \in N$, $x \in T \cup \epsilon$, $y \in T$, $z \in N$

Πίνακας 1.1: Οι περιορισμοί των γραμματικών

Η παραπάνω κατάταξη περιλαμβάνει 4 τύπους τυπικών γραμματικών:

τύπος 0 ή γραμματική χωρίς περιορισμούς (unrestricted): πρόκειται για την κλάση με την μεγαλύτερη εκφραστικότητα. Οι γλώσσες που παράγει ονομάζονται και αναδρομικά απαριθμήσιμες (recursively enumerable). Ο μοναδικός περιορισμός είναι ότι η κεφαλή του κανόνα δεν μπορεί να είναι η κενή συμβολοσειρά (ϵ).

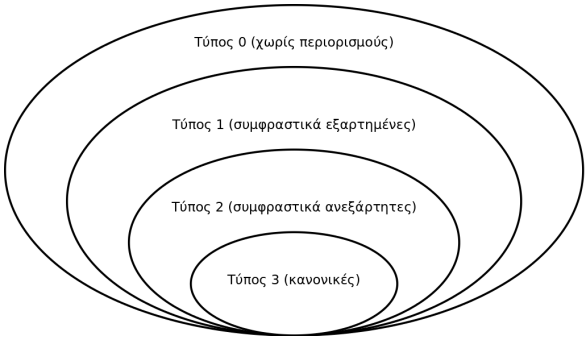
τύπος 1 ή συμφραστικά εξαρτημένη γραμματική (context-sensitive): πρόκειται για την κλάση που παράγει τις συμφραστικά εξαρτημένες γλώσσες. Οι περιορισμοί στους κανόνες είναι ότι η κεφαλή του κανόνα δεν μπορεί να είναι η κενή συμβολοσειρά και ότι το πλήθος των συμβόλων στο σώμα του κανόνα θα πρέπει να είναι μεγαλύτερο ή ίσο με το πλήθος των συμβόλων της κεφαλής.

τύπος 2 ή συμφραστικά ανεξάρτητή γραμματική (context-free): πρόκειται για την κλάση που παράγει τις συμφραστικά ανεξάρτητες γλώσσες. Η κεφαλή του κανόνα παραγωγής μπορεί να έχει μόνο ένα σύμβολο και αυτό θα πρέπει να είναι μη τερματικό.

τύπος 3 ή κανονική γραμματική (regular): πρόκειται για την κλάση με την μικρότερη εκφραστικότητα και παράγει τις κανονικές γλώσσες. Ανάλογα με την μορφή των κανόνων παραγωγής, οι κανονικές γραμματικές μπορεί να είναι δεξιογραμμικές (right-linear) ή αριστερογραμμικές (left-linear).

Κάθε γραμματική τύπου n είναι ένα γνήσιο υποσύνολο της γραμματικής τύπου $n-1$, συνθέτοντας μία γνήσια ιεραρχία όπου τύπος 3 $\subset$ τύπος 2 $\subset$ τύπος 1 $\subset$ τύπος 0, η οποία ονομάζεται *Ιεραρχία του Chomsky* (βλ. Εικόνα 1.1).

Από τους παραπάνω τύπους, οι συμφραστικά ανεξάρτητες γραμματικές έχουν χρησιμοποιηθεί αρκετά στην υπολογιστική γλωσσολογία για την συντακτική περιγραφή φυσικών γλωσσών αλλά και στην πληροφορική για την δημιουργία μεταγλωττιστών.



Σχήμα 1.1: Η ιεραρχία του Chomsky

1.3 Ο αλγόριθμος του Earley

Ο *αλγόριθμος του Earley* [6], [5] είναι ένας από τους πιο διαδεδομένους αλγόριθμους συντακτικής αναγνώρισης συμβολοσειρών για συμφραστικά ανεξάρτητες γλώσσες. Η ονομασία του προέρχεται από το όνομα του δημιουργού του Jay Earley. Ο συγκεκριμένος αλγόριθμος μπορεί να χειριστεί το σύνολο των συμφραστικά ανεξάρτητων γραμματικών και η χρονική πολυπλοκότητά του (time complexity) είναι της τάξης $O(n^3)$, αυτό σημαίνει ότι στην χείριστη περίπτωση (worst-case) ο χρόνος που χρειάζεται για να αναγνωρίσει μία συμβολοσειρά μεγέθους n είναι ανάλογος του n^3. Η πολυπλοκότητα χώρου του (space complexity) είναι της τάξης του $O(n^2)$, δηλαδή η μνήμη που χρειάζεται ο αλγόριθμος για να αναγνωρίσει μία συμβολοσειρά μεγέθους n είναι ανάλογη του n^2. Αν και ο αλγόριθμος είναι καταρχάς αλγόριθμος συντακτικής αναγνώρισης, ο Earley περιέγραψε τον τρόπο ώστε να μπορεί να πραγματοποιήσει και συντακτική ανάλυση. Σε αυτή την περίπτωση η χρονική πολυπλοκότητα παραμένει της τάξης $O(n^3)$, ενώ η πολυπλοκότητα χώρου γίνεται $O(n^3)$.

Ο αλγόριθμος ακολουθεί την προσέγγιση από πάνω προς τα κάτω (top-down) και χρησιμοποιεί την τεχνική του δυναμικού προγραμματισμού (dynamic programming). Ένας αλγόριθμος δυναμικού προγραμματισμού λύνει ένα πρόβλημα λύνοντας πρώτα απλούστερα, επικαλυπτόμενα (όχι ανεξάρτητα) υποπροβλήματα. Κάθε υποπρόβλημα μπορεί με την σειρά του να αναλυθεί σε

απλούστερα υποπροβλήματα. Όταν ένα υποπρόβλημα λύνεται για πρώτη φορά, η λύση του αποθηκεύεται συνήθως σε έναν πίνακα ώστε να μπορεί να επαναχρησιμοποιηθεί. Με αυτό τον τρόπο, η λύση κάθε υποπροβλήματος υπολογίζεται μόνο μια φορά και επαναχρησιμοποιείται όσες φορές χρειάζεται κατά τη διάρκεια της επίλυσης του αρχικού προβλήματος.

Περιγραφή του αλγόριθμου Για μία γραμματική $G = \langle N, T, P, S \rangle$ και για μία συμβολοσειρά εισόδου $w = X_1 \dots X_n$, ο αλγόριθμος δημιουργεί έναν πίνακα για την απομνημόνευση των λύσεων. Ο πίνακας αυτός ονομάζεται διάγραμμα (chart). Ουσιαστικά πρόκειται για $n+1$ σύνολα καταστάσεων (state set) $S_0 \dots S_n$. Μία κατάσταση (state) ορίζεται ως η τριάδα $\langle p, j, f \rangle$ όπου:

p ένας κανόνας παραγωγής της γραμματικής G, όπου $\bar{p}$ ο αριθμός των συμβόλων στο σώμα του κανόνα

j ένας ακέραιος με τιμή $0 \leqq j \leqq \bar{p}$, ο αριθμός των συμβόλων στο σώμα του κανόνα p που έχουν ήδη αναγνωριστεί

f ένας ακέραιος με τιμή $0 \leqq f \leqq n+1$, η θέση στην συμβολοσειρά w από την οποία ξεκίνησε η αναγνώριση γι' αυτόν τον κανόνα

Θα θέλαμε να αναφέρουμε ότι ο Earley, στην αρχική περιγραφή του αλγόριθμου, στον ορισμό της κατάστασης χρησιμοποιεί ένα προαιρετικό χαρακτηριστικό που ονομάζει συμβολοσειρά πρόβλεψης (lookahead string) και είναι μία συμβολοσειρά μεγέθους k. Στη συνέχεια θα θεωρήσουμε ότι $k = 0$ και θα αγνοήσουμε την συμβολοσειρά πρόβλεψης, με σκοπό να δώσουμε μία πιο απλή περιγραφή του αλγόριθμου.

Ένας διαφορετικός τρόπος αναπαράστασης μίας κατάστασης είναι η χρήση της τελείας ($\bullet$) στο σώμα του κανόνα για να δείξουμε τον αριθμό των συμβόλων που έχουν ήδη αναγνωριστεί. Μια κατάσταση της μορφής $[A \to \alpha \bullet \beta, j]$ σημαίνει ότι έχει ήδη αναγνωριστεί η συμβολοσειρά α και αναμένεται η αναγνώριση της συμβολοσειράς β.

Αρχικά, όλα τα σύνολα καταστάσεων είναι κενά. Το πρώτο βήμα είναι η προσθήκη μίας κατάστασης της μορφής $[\varnothing \to \bullet S, 0]$ στο σύνολο S_0. Το σύμβολο $\varnothing$ δεν ανήκει στα σύμβολα της γραμματικής.

Ύστερα, επεξεργαζόμαστε με την σειρά τα σύνολα $S_0 \dots S_n$. Για κάθε σύνολο καταστάσεων, έστω S_i, ελέγχουμε την μορφή του κανόνα παραγωγής κάθε κατάστασης και εκτελούμε μία από τις παρακάτω λειτουργίες, οι οποίες μπορεί να προσθέσουν μία νέα κατάσταση στο σύνολο S_i ή στο σύνολο S_{i+1}.

Πρόβλεψη (predict)

> Η λειτουργία πρόβλεψης εφαρμόζεται σε καταστάσεις στο σύνολο S_i που έχουν την μορφή $[A \to \dots \bullet B \dots, j]$, δηλαδή έχουν ένα μη τερματικό σύμβολο δεξιά της τελείας. Σε αυτή την περίπτωση, για κάθε κανόνα παραγωγής της γραμματικής, που έχει τη μορφή $B \to \alpha$, προσθέτει στο S_i μία νέα κατάσταση της μορφής $[B \to \bullet \alpha, i]$, προσθέτοντας ουσιαστικά την τελεία στην αρχή του δεξιού μέρους του κανόνα παραγωγής της νέας κατάστασης.

Σάρωση (scan)

> Η λειτουργία σάρωσης εφαρμόζεται σε καταστάσεις στο σύνολο S_i που έχουν την μορφή $[A \to \dots \bullet a \dots, j]$, δηλαδή δεξιά της τελείας υπάρχει ένα τερματικό σύμβολο. Ο σαρωτής συγκρίνει αυτό το σύμβολο με το σύμβολο X_{i+1}. Εάν $a = x_{i+1}$, τότε προσθέτει στο S_{i+1} μία κατάσταση $[A \to \dots a \bullet \dots, j]$, ουσιαστικά ένα αντίγραφο της κατάστασης στην οποία η τελεία έχει μετακινηθεί μια θέση δεξιότερα, προς υπόδειξη ότι αναγνωρίστηκε το τερματικό σύμβολο a.

Συμπλήρωση (complete)

> Η λειτουργία συμπλήρωσης εφαρμόζεται σε καταστάσεις στο σύνολο S_i που έχουν την μορφή $[A \to \dots \bullet, j]$, δηλαδή σε καταστάσεις στις οποίες η τελεία βρίσκεται στο τέλος του κανόνα. Σε αυτή την περίπτωση προσθέτει μία κατάσταση $[B \to \dots A \bullet \dots, k]$ στο σύνολο S_i για κάθε κατάσταση της μορφής $[B \to \dots \bullet A \dots, k]$ που ανήκει στο σύνολο S_j.

Ουσιαστικά για όλες τις καταστάσεις στο S_j που αναμένουν την αναγνώριση του μη τερματικού συμβόλου A, προσθέτει στο τρέχον σύνολο S_i ένα αντίγραφο της κατάστασης στην οποία η τελεία έχει μετακινηθεί μια θέση δεξιότερα, προς υπόδειξη ότι αναγνωρίστηκε το μη τερματικό σύμβολο A.

Οι παραπάνω λειτουργίες προσθέτουν μία κατάσταση σε ένα σύνολο καταστάσεων μόνο αν αυτό το σύνολο δεν περιέχει ήδη την κατάσταση. Αν επεξεργαστούμε όλες τις καταστάσεις που περιέχει το σύνολο S_i και το σύνολο S_{i+1} παραμείνει κενό, τότε η συμβολοσειρά δεν ανήκει στην γλώσσα $L(G)$. Αν στο τελευταίο σύνολο S_n υπάρχει μία κατάσταση $[\varnothing \rightarrow S\bullet, 0]$ τότε η συμβολοσειρά ανήκει στην γλώσσα $L(G)$.

1.4 Γραμματικές βασισμένες σε περιορισμούς

Οι *γραμματικές βασισμένες σε περιορισμούς* (constrained-based grammars) που συναντώνται στη βιβλιογραφία και ως *ενοποιητικές γραμματικές* (unification grammars - UG) είναι φορμαλισμοί που βασίζονται σε δομές ιδιοτήτων και στην λειτουργία της ενοποίησης για να περιγράψουν την σύνταξη της φυσικής γλώσσας. Σε σχέση με τις συμφραστικά ανεξάρτητες γραμματικές, έχουν μεγαλύτερη εκφραστική δύναμη και μπορούν να χειριστούν πιο σύνθετα φαινόμενα, όπως η συμφωνία (agreement) και η υποκατηγοριοποίηση (subcategorization). Συνήθως χρησιμοποιούν συμφραστικά ανεξάρτητες γραμματικές ως σκελετό, αν και αυτό δεν είναι απαραίτητο. Στη συνέχεια θα περιγράψουμε τις δομές ιδιοτήτων, την έννοια του εγκλεισμού και τις λειτουργίες της ενοποίησης και της γενίκευσης.

1.4.1 Δομές ιδιοτήτων

Οι *δομές ιδιοτήτων* (feature structures) είναι ένας τρόπος αναπαράστασης της γλωσσική πληροφορίας, με τον οποίο αντιστοιχίζονται *τιμές* (values) σε

ιδιότητες (features). Ο πιο συχνός τρόπος γραφικής αναπαράστασης των δομών ιδιοτήτων στην βιβλιογραφία είναι ως *πίνακες ιδιοτήτων-τιμών* (attribute-value matrices - AVMs), όπως φαίνεται στο Σχήμα 1.2, και σπανιότερα ως *γράφοι ιδιοτήτων* (feature graphs), δηλαδή συνεκτικοί, κατευθυνόμενοι γράφοι με ετικέτες (directed, connected, labelled graphs).

$$\begin{bmatrix} \text{feature}_1 & \text{value}_1 \\ \text{feature}_2 & \text{value}_2 \\ \vdots & \\ \text{feature}_n & \text{value}_n \end{bmatrix}$$

Σχήμα 1.2: πίνακας ιδιοτήτων-τιμών

Οι τιμές των δομών ιδιοτήτων μπορεί να είναι *ατομικές* (atomic) όπως η ιδιότητα ΜτΛ ή *σύνθετες* (complex) όπως η ιδιότητα συμφωνία (βλέπε Σχήμα 1.3).

$$\begin{bmatrix} \text{ΜτΛ} & \text{ουσιαστικό} \\ \text{συμφωνία} & \gamma\begin{bmatrix} \text{ένος} & \text{θηλυκό} \\ \text{αριθμός} & \text{ενικός} \\ \text{πτώση} & \text{ονομαστική} \end{bmatrix} \end{bmatrix}$$

Σχήμα 1.3: απεικόνιση γλωσσικής πληροφορίας με πίνακα ιδιοτήτων-τιμών

Οι δομές ιδιοτήτων είναι μερικές συναρτήσεις από το σύνολο των ιδιοτήτων στο σύνολο των τιμών. Ο συμβολισμός $D(f)$ δηλώνει την τιμή της ιδιότητας f στην δομή ιδιοτήτων D, π.χ. με βάση το Σχήμα 1.3, $D(\text{ΜτΛ}) = \text{ουσιαστικό}$. Το σύνολο των ιδιοτήτων μίας δομής ιδιοτήτων D, το λέμε και *πεδίο* (domain) της D και το συμβολίζουμε με $dom(D)$. Εάν $dom(D) = \varnothing$, τότε ονομάζεται *κενή δομή* ή *μεταβλητή* (variable) και συμβολίζεται ως [ ]. Για παράδειγμα, $dom(D) = \{\text{ΜτΛ}, \text{συμφωνία}\}$. *Μονοπάτι* (path) σε μια δομή είναι μια ακολουθία ιδιοτήτων που συμβολίζεται μέσα σε γωνιακές παρενθέσεις, π.χ.

⟨συμφωνία γένος⟩. Τα μονοπάτια είναι χρήσιμα για την εύρεση της τιμής σε μία ενθυλακωμένη δομή ιδιοτήτων, π.χ. $D(\langle\text{συμφωνία γένος}\rangle) = \text{θηλυκό}$.

1.4.2 Εγκλεισμός

Ο *εγκλεισμός* (subsumption) είναι μία σχέση μεταξύ δύο δομών ιδιοτήτων. Μία δομή ιδιοτήτων D εγκλείεται στην δομή ιδιοτήτων D' ($D \sqsubseteq D'$) εάν η D περιέχει ένα υποσύνολο της πληροφορίας της D', οπότε και περιγράφει ένα μεγαλύτερο σύνολο οντοτήτων (βλέπε (1.1)).

$$\begin{bmatrix}\text{γένος} & \text{αρσενικό}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{γένος} & \text{αρσενικό}\\ \text{αριθμός} & \text{ενικός}\end{bmatrix} \tag{1.1}$$

Πιο συγκεκριμένα, μία σύνθετη δομή ιδιοτήτων D εγκλείεται σε μία σύνθετη δομή ιδιοτήτων D' εάν: για κάθε $l \in dom(D)$ ισχύει $D(l) \sqsubseteq D'(l)$ και για κάθε μονοπάτι p και q της D τέτοιο ώστε $D(p) = D(q)$, θα πρέπει να ισχύει $D'(p) = D'(q)$.

Επίσης, ισχύει ότι μία ατομική τιμή δεν εγκλείει ούτε εγκλείεται από άλλη ατομική τιμή εάν οι δύο ατομικές τιμές είναι διαφορετικές (βλέπε (1.2)).

$$\text{αρσενικό}\neg \sqsubseteq \text{θηλυκό} \tag{1.2}$$

$$\text{αρσενικό} \sqsubseteq \text{αρσενικό} \tag{1.3}$$

Οι κενές δομές ιδιοτήτων εγκλείονται σε όλες τις άλλες δομές ιδιοτήτων, ατομικές και σύνθετες, επειδή αποτελούν την δομή με τη μικρότερη πληροφορία (βλέπε (1.4)).

$$\begin{bmatrix}\ \end{bmatrix} \sqsubseteq \begin{bmatrix}\text{γένος} & \text{αρσενικό}\end{bmatrix} \tag{1.4}$$

1.4.3 Ενοποίηση και Γενίκευση

Η πράξη της *ενοποίησης* (unification) είναι μία λειτουργία που επιτρέπει την συγχώνευση δύο δομών ιδιοτήτων, εφόσον περιέχουν συμβατά χαρακτηριστικά.

Το αποτέλεσμα της ενοποίησης δύο δομών ιδιοτήτων είναι η μικρότερη δομή ιδιοτήτων που εγκλείεται και στις δύο και συμβολίζεται ως $D = D' \sqcup D''$. Συγκεκριμένα, η ενοποίηση των δομών ιδιοτήτων D' και D'' είναι μία δομή D, τέτοια ώστε $D' \sqsubseteq D$ και $D'' \sqsubseteq D$ (βλέπε (1.5)).

$$\begin{bmatrix} \text{γένος} & \text{αρσ} \\ \text{αριθμός} & \text{εν} \end{bmatrix} \sqcup \begin{bmatrix} \text{γένος} & \text{αρσ} \\ \text{πτώση} & \text{ονομ} \end{bmatrix} = \begin{bmatrix} \text{γένος} & \text{αρσ} \\ \text{αριθμός} & \text{εν} \\ \text{πτώση} & \text{ονομ} \end{bmatrix} \tag{1.5}$$

Γενίκευση (generalization) δύο δομών ιδιοτήτων είναι η μεγαλύτερη δομή χαρακτηριστικών που εγκλείεται και από τις δύο και συμβολίζεται ως $D = D_1 \sqcap D_2$. Συγκεκριμένα, η γενίκευση των δομών ιδιοτήτων D' και D'' είναι μία δομή C, τέτοια ώστε $C \sqsubseteq D'$, $C \sqsubseteq D''$ και επιπλέον για κάθε C' να ισχύει $C' \sqsubseteq D'$, $C' \sqsubseteq D''$, $C' \sqsubseteq C$ (βλέπε (1.6)).

$$\begin{bmatrix} \text{γένος} & \text{αρσ} \\ \text{αριθμός} & \text{εν} \end{bmatrix} \sqcap \begin{bmatrix} \text{γένος} & \text{αρσ} \\ \text{πτώση} & \text{ονομ} \end{bmatrix} = \begin{bmatrix} \text{γένος} & \text{αρσ} \end{bmatrix} \tag{1.6}$$

Αντίθετα από την ενοποίηση, η πράξη της γενίκευσης είναι πάντα επιτυχής (βλέπε (1.7)).

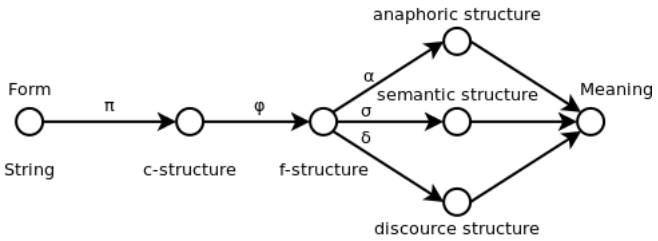
$$\begin{bmatrix} \text{γένος} & \text{αρσ} \\ \text{αριθμός} & \text{εν} \end{bmatrix} \sqcap \begin{bmatrix} \text{γένος} & \text{ουδ} \end{bmatrix} = \begin{bmatrix} \quad & \quad \end{bmatrix} \tag{1.7}$$

Η χρήση των δομών ιδιοτήτων και της ενοποίησης επιτρέπει την λεπτομερέστερη μοντελοποίηση της σύνταξης της φυσικής γλώσσας, σε αντίθεση με τις συμφραστικά ανεξάρτητες γραμματικές που περιορίζονται στην χρήση μίας μόνο ιδιότητας, της γραμματικής κατηγορίας.

1.5 Η Λεξική-Λειτουργική Γραμματική

Η *Λεξική-Λειτουργική Γραμματική* (Lexical-Functional Grammar - LFG), εφεξής ΛΛΓ, είναι ένα θεωρητικό πλαίσιο για την περιγραφή της σύνταξης φυσικών γλωσσών που αναπτύχθηκε στα τέλη της δεκαετίας του '70 από τους Joan Bresnan και Ron Kaplan [11]. Ανήκει στην κατηγορία των γενετικών (generative), μη μετασχηματιστικών (non-transformational) γραμματικών.

Κεντρικός άξονας της θεωρίας είναι η *αρχιτεκτονική παράλληλης προβολής* (parallel projection architecture) [10] ή *αρχιτεκτονική αντιστοίχισης* (correspondence architecture). Η αναπαράσταση της γλωσσικής πληροφορίας γίνεται με την παράλληλη χρήση πολλών, διαφορετικού είδους, δομών ή αλλιώς προβολών (projection), η κάθε μία από τις οποίες διέπεται από διαφορετικούς κανόνες και έχει τον δικό της τρόπο συμβολισμού. Μεταξύ των δομών αυτών, υπάρχουν σχέσεις που ονομάζονται συναρτήσεις αντιστοίχισης (correspondence functions) ή συναρτήσεις προβολών (projection functions) (βλέπε Σχήμα 1.4).



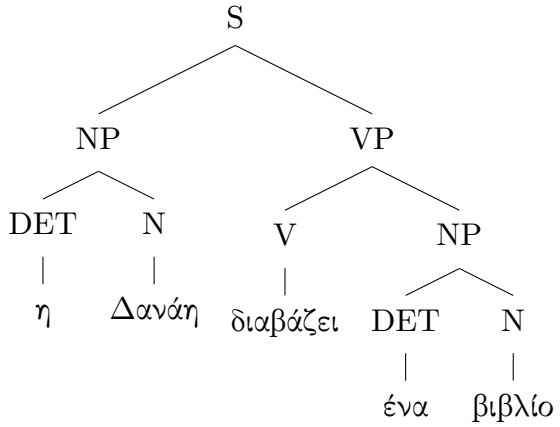
Σχήμα 1.4: Η αρχιτεκτονική παράλληλης προβολής της ΛΛΓ

Στην πιο απλή εκδοχή της θεωρίας υπάρχουν δύο δομές, η συστατική δομή και η λειτουργική δομή, καθώς και η μεταξύ τους συνάρτηση αντιστοίχισης ϕ.

1.5.1 Συστατική δομή

Η *συστατική δομή* (c-structure ή constituent structure) παρέχει πληροφορίες για τα εξής: την σειρά των όρων της πρότασης (word order), τη φραστική κατηγορία στην οποία ανήκει το κάθε συστατικό και την ιεράρχηση των φραστικών συστατικών. Η δομή αυτή έχει την μορφή ενός δεντροδιαγράμματος.

Στο Σχήμα 1.5, βλέπουμε ένα δέντρο για την πρόταση *η Δανάη διαβάζει ένα βιβλίο.*



Σχήμα 1.5: Συστατική δομή σύμφωνα με τη ΛΛΓ

1.5.2 Λειτουργική δομή

Μία *λειτουργική δομή* (f-structure ή functional structure) είναι ένα πεπερασμένο σύνολο ζευγών ιδιότητας-τιμής και έχει την μορφή ενός πίνακα ιδιοτήτων-τιμών. Μία ιδιότητα είναι μία συμβολοσειρά (symbol), ενώ μία τιμή μπορεί να είναι είτε μία συμβολοσειρά είτε μία άλλη λειτουργική δομή. Ένα ζεύγος ιδιότητας-τιμής, όπου η τιμή είναι ένα σύμβολο ονομάζεται *χαρακτηριστικό* (feature). Τα χαρακτηριστικά χρησιμοποιούνται για την αναπαράσταση μορφοσυντακτικής πληροφορίας (π.χ. πτώση, χρόνος, γένος). Εάν η τιμή είναι μία λειτουργική δομή, τότε ονομάζεται *γραμματική συνάρτηση* (grammatical function) και αναπαριστά γραμματικές λειτουργίες (π.χ. υποκείμενο, αντικείμενο κ.λπ.). Επιπλέον, οι λειτουργικές δομές είναι δυνατόν να χρησιμοποιήσουν ως τιμές και σύνολα (sets) ή σημασιολογικές μορφές.

Μία *σημασιολογική μορφή* (semantic form) είναι μία συμβολοσειρά η οποία αναπαριστά τη σημασιολογική ερμηνεία μίας λέξης. Η συμβολοσειρά εμφανίζεται μέσα σε μονά εισαγωγικά (π.χ. 'Δανάη', 'βιβλίο'). Σε περιπτώσεις που είναι αναγκαίο, π.χ. σε ένα ρήμα, μία σημασιολογική μορφή μπορεί να περιέχει, μέσα

σε τριγωνικές αγκύλες, τα *ορίσματά* του (arguments), (π.χ. 'διαβάζω⟨SUBJ, OBJ⟩'). Ένα ζεύγος ιδιότητας-τιμής, όπου η τιμή είναι μία σημασιολογική μορφή ονομάζεται *σημασιολογικό χαρακτηριστικό* (semantic feature). Οι σημασιολογικές μορφές χρησιμοποιούνται συνήθως ως τιμές για την ιδιότητα $PRED$.

Στο Σχήμα 1.6 φαίνεται μία ενδεικτική λειτουργική δομή για το παράδειγμα στο Σχήμα 1.5.

$$\begin{bmatrix} PRED & \text{'διαβάζω}\langle SUBJ, OBJ\rangle\text{'} \\ TENSE & \text{NONPAST} \\ ASPECT & \text{IMPERFECTIVE} \\ PERSON & \text{THIRD} \\ NUMBER & \text{SING} \\ SUBJ & \begin{bmatrix} PRED & \text{'Δανάη'} \\ CASE & \text{NOM} \\ GENDER & \text{FEM} \\ NUMBER & \text{SING} \end{bmatrix} \\ OBJ & \begin{bmatrix} PRED & \text{'βιβλίο'} \\ CASE & \text{ACC} \\ GENDER & \text{NEUT} \\ NUMBER & \text{SING} \end{bmatrix} \end{bmatrix}$$

Σχήμα 1.6: Λειτουργική δομή σύμφωνα με τη ΛΛΓ

Κυβερνώμενες γραμματικές συναρτήσεις

Η ΛΛΓ διακρίνει τις γραμματικές συναρτήσεις σε κυβερνώμενες (governable) και μη κυβερνώμενες (non-governable). Οι κυβερνώμενες γραμματικές συναρτήσεις περιέχουν τις γραμματικές λειτουργίες οι οποίες είναι ορίσματα του

κατηγορήματος όπως το υποκείμενο και το αντικείμενο. Γραμματικές συναρτήσεις όπως τα προσαρτήματα είναι μη κυβερνώμενες γραμματικές συναρτήσεις.

Κριτήρια ορθού σχηματισμού λειτουργικών δομών

Μία λειτουργική δομή είναι έγκυρη αν πληροί ταυτόχρονα τα κριτήρια της *μοναδικότητας* (uniqueness ή consistency), της *πληρότητας* (completeness) και της *συνεκτικότητας* (coherence).

Μοναδικότητα

Σε μια λειτουργική δομή κάθε ιδιότητα έχει μία και μόνο μία τιμή.

Πληρότητα

Μια λειτουργική δομή είναι τοπικά πλήρης εάν και μόνον εάν περιέχει όλες τις κυβερνώμενες γραμματικές συναρτήσεις τις οποίες κυβερνά το κατηγόρημά της. Μία λειτουργική δομή είναι πλήρης εάν και μόνον εάν όλες οι επιμέρους λειτουργικές δομές είναι τοπικά πλήρεις.

Συνεκτικότητα

Μία λειτουργική δομή είναι τοπικά συνεκτική εάν και μόνον εάν όλες οι κυβερνώμενες γραμματικές συναρτήσεις τις οποίες περιέχει κυβερνώνται από ένα τοπικό κατηγόρημα. Μία λειτουργική δομή είναι συνεκτική εάν και μόνον εάν όλες οι επιμέρους λειτουργικές δομές είναι τοπικά συνεκτικές.

Σύμφωνα με την ΛΛΓ, μία συμβολοσειρά είναι γραμματικά ορθή αν πληροί τα εξής δύο κριτήρια. Αφενός, η συμβολοσειρά θα πρέπει να έχει μία έγκυρη συστατική δομή. Αφετέρου, πρέπει να συνοδεύεται από μία έγκυρη λειτουργική δομή, η οποία να ικανοποιεί τα παραπάνω τρία κριτήρια.

1.5.3 Λειτουργική περιγραφή

Η *λειτουργική περιγραφή* (f-description ή functional description) είναι ένας τρόπος αναπαράστασης μίας λειτουργικής δομής. Ουσιαστικά πρόκει-

ται για ένα σύνολο εξισώσεων, που στο πλαίσιο της ΛΛΓ ονομάζονται *λειτουργικές εξισώσεις* (functional equations). Μία λειτουργική δομή, όπως στο Σχήμα 1.7 μπορεί να θεωρηθεί ως συνάρτηση f με πεπερασμένο πεδίο ορισμού τις ιδιότητες της και με πεπερασμένο σύνολο τιμών τις τιμές της.

Σύμφωνα με την κλασική σημειογραφία, η τιμή της συνάρτησης f για όρισμα το $CASE$ συμβολίζεται ως $f(CASE) = NOM$. Ειδικότερα όμως στην ΛΛΓ χρησιμοποιείται ένας διαφορετικός τρόπος σημειογραφίας όπου η αριστερή παρένθεση έχει μεταφερθεί αριστερά, οπότε η εξίσωση γράφεται ως $(f\ CASE) = NOM$. Μία λειτουργική περιγραφή της μορφής (1.8) περιγράφει τη λειτουργική δομή f (Σχήμα 1.7).

$$f\begin{bmatrix} CASE & \text{NOM} \\ GENDER & \text{FEM} \\ NUMBER & \text{SING} \end{bmatrix}$$

Σχήμα 1.7: Λειτουργική δομή σύμφωνα με τη ΛΛΓ

$$\begin{aligned} &(f\ GENDER) = FEM \\ &(f\ NUM) = SING \\ &(f\ CASE) = NOM \end{aligned} \tag{1.8}$$

Μία λειτουργική περιγραφή μπορεί να χρησιμοποιηθεί και ως ένα σύστημα εξισώσεων το οποίο μπορούμε να επιλύσουμε με σκοπό να δημιουργήσουμε μία λειτουργική δομή. Το σύστημα (1.9) θα μας δώσει την λειτουργική δομή

που βλέπουμε στο Σχήμα 1.8.

$$
\begin{aligned}
&(f_1\ PRED) =' \text{κοιμάμαι}\langle SUBJ\rangle' \\
&(f_1\ SUBJ) = f_2 \\
&(f_1\ PERSON) = THIRD \\
&(f_2\ PRED) =' \Delta\text{ανάη}' \\
&(f_2\ GENDER) = FEM \\
&(f_2\ NUM) = SING \\
&(f_2\ CASE) = NOM
\end{aligned}
\tag{1.9}
$$

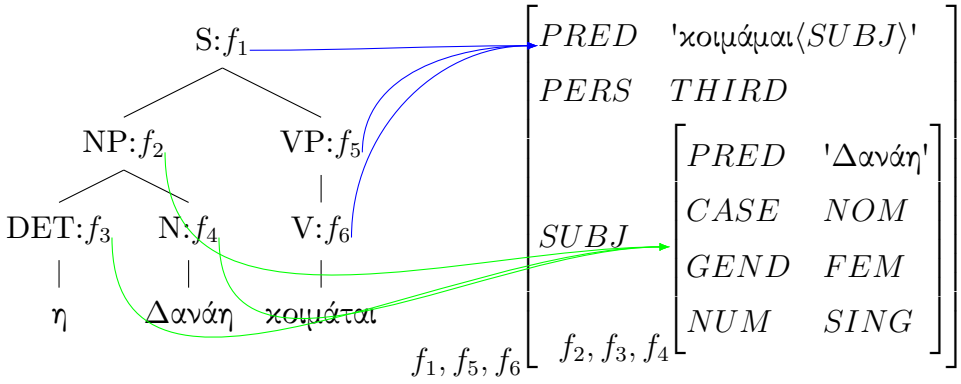
$$
\begin{bmatrix}
PRED & \text{'κοιμάμαι}\langle SUBJ\rangle\text{'} \\
PERSON & \text{THIRD} \\
SUBJ & \begin{bmatrix}
PRED & \text{'}\Delta\text{ανάη'} \\
CASE & \text{NOM} \\
GENDER & \text{FEM} \\
NUMBER & \text{SING}
\end{bmatrix}
\end{bmatrix}
$$

Σχήμα 1.8: Λειτουργική δομή σύμφωνα με τη ΛΛΓ

1.5.4 Η σχέση αντιστοίχισης συστατικής και λειτουργικής δομής

Όπως έχει αναφερθεί παραπάνω, μία πρόταση φυσικής γλώσσας περιγράφεται παράλληλα με μία συστατική και μία λειτουργική δομή. Μεταξύ τους υπάρχει μία σχέση αντιστοίχισης που συμβολίζεται με ϕ. Τυπικά, η ϕ ορίζεται ως συνάρτηση από το σύνολο των κόμβων της συστατικής δομής στο σύνολο των λειτουργικών δομών. Στο Σχήμα 1.9 βλέπουμε μία πλήρη ανάλυση της πρότασης *η Δανάη κοιμάται*.

Παρατηρούμε ότι η ϕ δεν είναι συνάρτηση *ένα προς ένα*, πολλαπλοί κόμβοι του δέντρου αντιστοιχούν σε μία λειτουργική δομή.



Σχήμα 1.9: Η σχέση αντιστοίχισης ϕ

Μία ολοκληρωμένη περιγραφή μίας πρότασης φυσικής γλώσσας περιλαμβάνει όχι μόνο τις δύο δομές, συστατική και λειτουργική, αλλά και την αντιστοίχιση μεταξύ τους.

1.5.5 Λειτουργικά υποδείγματα

Οι κανόνες της ΛΛΓ έχουν την μορφή κανόνων συμφραστικά ανεξάρτητων γραμματικών επισημειωμένων με *λειτουργικά υποδείγματα* (functional schemata). Τα λειτουργικά υποδείγματα είναι παρόμοια με τις λειτουργικές εξισώσεις, όμως αντί για ονόματα συναρτήσεων, περιέχουν τα σύμβολα $\uparrow$ (πάνω βέλος) και $\downarrow$ (κάτω βέλος). Τα δύο αυτά σύμβολα που ονομάζονται και μετα-μεταβλητές (metavariables) σχετίζουν έναν κόμβο της συστατικής δομής με μία λειτουργική δομή. Πιο συγκεκριμένα, η συνάρτηση M αντιστοιχεί έναν κόμβο στον γονικό του κόμβο. Για έναν κόμβο $*$, ο κόμβος $M(*)$ είναι ο γονικός κόμβος του κόμβου $*$. Αντί για $M(*)$, χρησιμοποιείται και ο συμβολισμός $\hat{*}$. Ισχύει ότι $\uparrow \equiv \phi(M(*))$, δηλαδή το βέλος δηλώνει την λειτουργική δομή του γονικού κόμβου, και $\downarrow \equiv \phi(*)$, όπου το βέλος υποδηλώνει την λειτουργική δομή του κόμβου. Στο Σχήμα 1.10 βλέπουμε μία γραμματική και ένα λεξικό της ΛΛΓ, το οποίο μπορεί να χρησιμοποιηθεί για την ανάλυση της πρότασης *η Δανάη κοιμάται*.

Σύμφωνα με αυτή την γραμματική, η συστατική δομή της πρότασης θα είναι αυτή στο Σχήμα 1.11. Παρατηρούμε ότι το δέντρο είναι επισημειωμένο με τα

$$
\begin{array}{lcll}
\text{S} & \rightarrow & \text{NP} & \text{VP} \\
 & & (\uparrow SUBJ) = \downarrow & \uparrow = \downarrow \\
\text{NP} & \rightarrow & \text{DET} & \text{N} \\
 & & \uparrow = \downarrow & \uparrow = \downarrow \\
\text{VP} & \rightarrow & \text{V} & \\
 & & \uparrow = \downarrow &
\end{array}
$$

(α′) Κανόνες ΛΛΓ

$$
\begin{array}{ll}
\eta, \text{ DET} & (\uparrow GEND) = FEM \\
 & (\uparrow NUM) = SING \\
 & (\uparrow CASE) = NOM \\
\Delta\alpha\nu\acute{\alpha}\eta, \text{ N} & (\uparrow GEND) = FEM \\
 & (\uparrow NUM) = SING \\
 & (\uparrow CASE) = NOM \\
 & (\uparrow PRED) =' \Delta\alpha\nu\acute{\alpha}\eta' \\
\kappa\omega\iota\mu\acute{\alpha}\tau\alpha\iota & (\uparrow PRED) =' \kappa\omega\iota\mu\acute{\alpha}\mu\alpha\iota\langle SUBJ\rangle' \\
 & (\uparrow PERS) = THIRD
\end{array}
$$

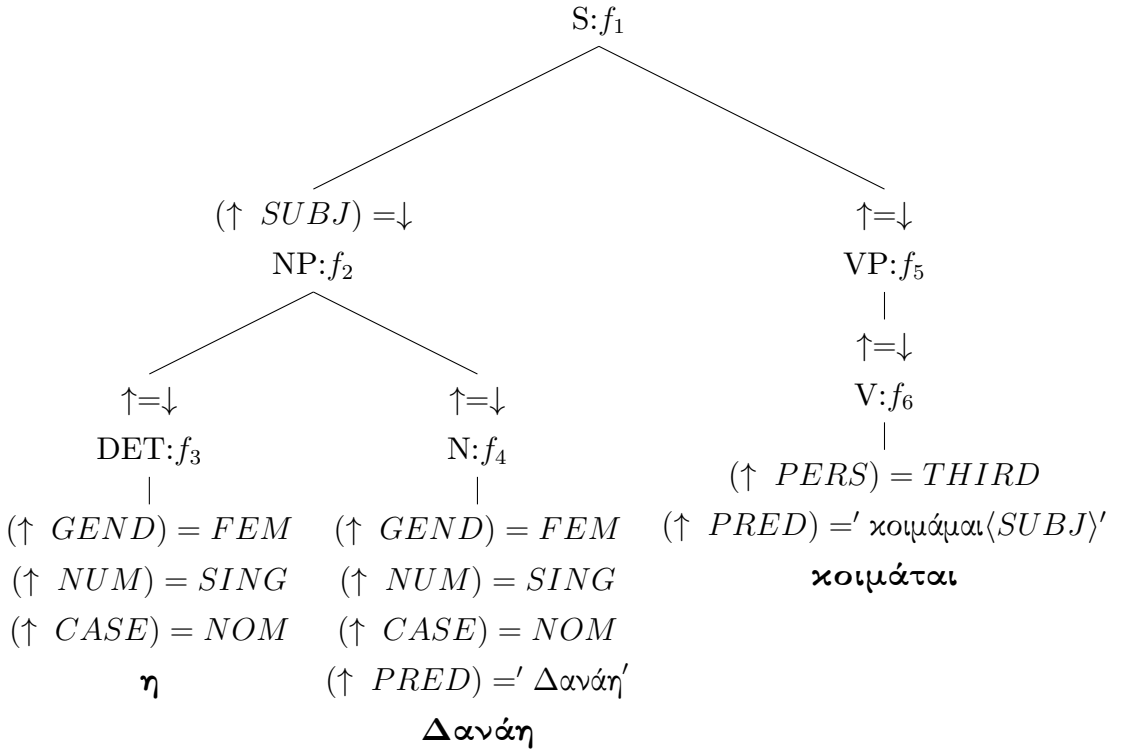
(β′) Λεξικό ΛΛΓ

Σχήμα 1.10: Κανόνες και Λεξικό της ΛΛΓ

λειτουργικά υποδείγματα. Εάν αντικαταστήσουμε τις μετα-μεταβλητές $\uparrow$ και $\downarrow$ με $\phi(M(*))$ και $\phi(*)$ αντίστοιχα, τότε το σύνολο τους είναι η λειτουργική περιγραφή της πρότασης, η οποία μπορεί να επιλυθεί ώστε να δημιουργηθούν οι λειτουργικές δομές της πρότασης. Η δομή που αντιστοιχεί στην ρίζα του δέντρου είναι η λειτουργική δομή της πρότασης.

1.5.6 Λειτουργικές εξισώσεις

Οι *λειτουργικές εξισώσεις* (functional equations) της ΛΛΓ χωρίζονται σε δύο κατηγορίες: τις εξισώσεις δήλωσης (defining equations) και τις εξισώσεις περιορισμού (constraining equations).



Σχήμα 1.11: Συστατική δομή, επισημειωμένη με λειτουργικές εξισώσεις

Εξισώσεις δήλωσης

Οι *εξισώσεις δήλωσης* (defining equations) χρησιμοποιούνται για την δημιουργία των λειτουργικών δομών. Η (1.10) θέτει την τιμή της ιδιότητας $CASE$ στην λειτουργική δομή f σε ACC. Εδώ, διευκρινίζεται ότι ο τελεστής $=$, στο πλαίσιο της ΛΛΓ, συμβολίζει την πράξη της ενοποίησης.

$$(f\ CASE) = ACC \tag{1.10}$$

Οι εξισώσεις δήλωσης μπορούν να χρησιμοποιηθούν και σε σύνολα. Η (1.11) προσθέτει τη λειτουργική δομή g ως μέλος στο σύνολο ADJ της λειτουργικής δομής f.

$$g \in (f\ ADJ) \tag{1.11}$$

Εξισώσεις περιορισμού

Οι *εξισώσεις περιορισμού* (constraining equations) είναι εξισώσεις που ελέγχουν για την ορθότητα της λύσης εφαρμόζοντας περιορισμούς. Σε αντίθεση με τις εξισώσεις δήλωσης, δεν συνεισφέρουν στην δημιουργία λειτουργικών δομών. Η (1.12) θα ικανοποιηθεί μόνο αν η λειτουργική δομή f περιέχει μία ιδιότητα με το όνομα $CASE$ και με τιμή ACC.

$$(f\ CASE) =_c ACC \tag{1.12}$$

Σε περίπτωση που θέλουμε να μην περιέχει κάποια συγκεκριμένη τιμή (π.χ. NOM), μπορούμε να χρησιμοποιήσουμε τον τελεστή μη ισότητας ($\neq$) όπως στην (1.13).

$$(f\ CASE) \neq NOM \tag{1.13}$$

Η (1.13) μπορεί να γραφτεί σε άλλη μορφή χρησιμοποιώντας τον τελεστή άρνησης ($\neg$), βλέπε (1.14).

$$\neg[(f\ CASE) = NOM] \tag{1.14}$$

Εκτός από τις εξισώσεις περιορισμού που ελέγχουν για την ύπαρξη ή απουσία συγκεκριμένων τιμών που μπορεί να έχει μία ιδιότητα, υπάρχουν και εξισώσεις υπαρξιακού περιορισμού (existential contrain) που ελέγχουν για την ύπαρξη ιδιοτήτων χωρίς όμως να λαμβάνουν υπόψη την τιμή της ιδιότητας. Η (1.15) ικανοποιείται αν η λειτουργική δομή f περιέχει μία ιδιότητα $CASE$ με οποιαδήποτε τιμή, ενώ η χρήση του τελεστή $\neg$ (1.16) υποδηλώνει ότι η λειτουργική δομή f δεν πρέπει να περιέχει μία ιδιότητα $CASE$.

$$(f\ CASE) \tag{1.15}$$

$$\neg(f\ CASE) \tag{1.16}$$

1.5.7 Λειτουργική αβεβαιότητα

Η *λειτουργική αβεβαιότητα* (functional uncertainty) είναι μία επέκταση του αρχικού θεωρητικού μοντέλου της ΛΛΓ, και πιο ειδικά των λειτουργικών εξισώσεων. Σκοπός της είναι η χρήση ιδιοτήτων ως ορίσματα σε λειτουργικές εξισώσεις, όταν δεν γνωρίζουμε εκ των προτέρων τις ιδιότητες αυτές. Είναι χρήσιμες για πιο σύνθετα φαινόμενα, π.χ. αναφορά ή εξαρτήσεις μεγάλης απόστασης (long distance dependencies).

Μία εξίσωση χωρίς λειτουργική αβεβαιότητα ορίζεται ως εξής: $(f\ \alpha) = u$ εάν και μόνο εάν f είναι μία λειτουργική δομή, α είναι μία ιδιότητα και το ζεύγος $\langle \alpha, u \rangle \in f$. Στο Σχήμα 1.12, ισχύει ότι η $(f\ E) = g$.

Η λειτουργική αβεβαιότητα ορίζεται ως εξής: Εάν το α είναι ένα σύνολο που αποτελείται από συμβολοσειρές, $(f\ \alpha) = u$ εάν και μόνο εάν υπάρχει $x \in \alpha$ τέτοιο ώστε να ισχύει $(f\ x) = u$. Στο Σχήμα 1.12, ισχύει ότι η $(f\ \{A|C|E\}) = g$.

Οι λειτουργικές εξισώσεις με μορφή από μέσα προς τα έξω (inside-out) είναι ένας άλλος τύπος λειτουργικής αβεβαιότητας, όπου για κάθε λειτουργική δομή f' και κάποια ιδιότητα α, η $(\alpha\ f')$ δείχνει μία λειτουργική δομή $(f\ \alpha) = f'$. Στο Σχήμα 1.12, ισχύει ότι η $(E\ g) = f$.

Η λειτουργική αβεβαιότητα σε εξισώσεις με μορφή από μέσα προς τα έξω ορίζεται ως εξής: Εάν το α είναι ένα σύνολο που αποτελείται από συμβολοσειρές, $(\alpha\ f) = u$ εάν και μόνο εάν υπάρχει $x \in \alpha$ τέτοιο ώστε να ισχύει $(x\ f) = u$. Στο Σχήμα 1.12, ισχύει ότι η $(\{A|C|E\}\ g) = f$.

$$f\left[\begin{array}{ll} A & \mathrm{B} \\ C & \mathrm{D} \\ E & g\left[\begin{array}{ll} J & \mathrm{K} \\ L & \mathrm{M} \end{array}\right] \end{array}\right]$$

Σχήμα 1.12: Λειτουργική δομή σύμφωνα με τη ΛΛΓ

Κεφάλαιο 2

Σχεδιασμός και υλοποίηση του συντακτικού αναλυτή

2.1 Εισαγωγή

Το προς επίλυση πρόβλημα είναι η ανάπτυξη ενός εργαλείου συντακτικής ανάλυσης για προτάσεις φυσικής γλώσσας, βασισμένου στον φορμαλισμό της Λεξικής-Λειτουργικής Γραμματικής (ΛΛΓ). Αρχική μας υπόθεση είναι ότι οι κανόνες της ΛΛΓ μπορούν να θεωρηθούν κανόνες συμφραστικά ανεξάρτητων γραμματικών, όπου τα σύμβολα στο σώμα των κανόνων είναι επισημειωμένα με λειτουργικά υποδείγματα. Με βάση την παραπάνω υπόθεση, το πρόβλημα της συντακτικής ανάλυσης είναι δυνατόν να επιμεριστεί σε δύο απλούστερα υποπροβλήματα:

α. την δημιουργία ενός δεντροδιαγράμματος επισημειωμένου με τα λειτουργικά υποδείγματα για την αναπαράσταση της συστατικής δομής της πρότασης,

β. την δημιουργία και επίλυση της λειτουργικής περιγραφής της πρότασης για την εξαγωγή της λειτουργικής δομής της.

2.2 Σχεδιασμός

2.2.1 Η αρχιτεκτονική του συστήματος

Πριν από την υλοποίηση του λογισμικού προηγήθηκε το απαραίτητο στάδιο του σχεδιασμού του. Σε αυτό το στάδιο, πραγματοποιήθηκε ο καταμερισμός της λειτουργικότητας σε επιμέρους *συστατικά* (components). Ένα συστατικό είναι μια αρθρωτή μονάδα με καλά καθορισμένες *διεπαφές* (interfaces). Τα οφέλη αυτής της προσέγγισης είναι πολλαπλά:

- διευκολύνει την ανάπτυξη, τον έλεγχο και την συντήρηση μικρότερων τμημάτων κώδικα
- παρέχει την δυνατότητα επαναχρησιμοποίησης υπαρχόντων συστατικών
- επιτρέπει την αντικατάσταση ενός συστατικού με μία νέα υλοποίηση, χωρίς να χρειάζεται να πραγματοποιήσουμε αλλαγές στον υπόλοιπο κώδικα της εφαρμογής.

Όπως φαίνεται στην λογική αρχιτεκτονική του συστήματος (Σχήμα 2.1), η εφαρμογή περιλαμβάνει τρία συστατικά:

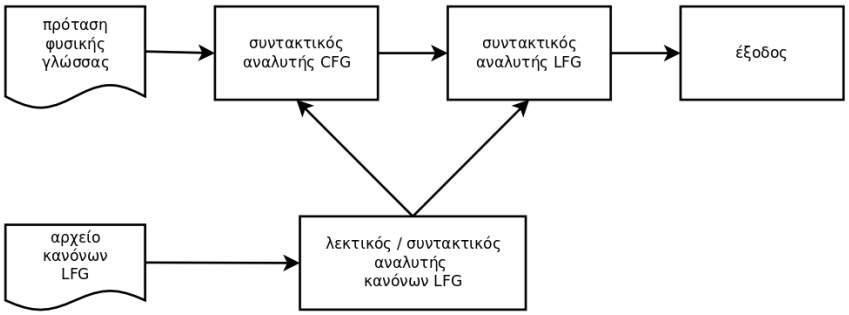
α. έναν συντακτικό αναλυτή για συμφραστικά ανεξάρτητες γραμματικές,

β. έναν επιλυτή λειτουργικών εξισώσεων της ΛΛΓ,

γ. έναν λεκτικό και συντακτικό αναλυτή για την ανάγνωση αρχείων με κανόνες της ΛΛΓ και για την μετατροπή τους σε δομές δεδομένων που μπορούν να χρησιμοποιηθούν από τα άλλα δύο συστατικά.

2.2.2 Το περιβάλλον ανάπτυξης

Η ανάπτυξη του λογισμικού έγινε με τη χρήση της γλώσσας προγραμματισμού *Java*. Η Java είναι μια *αντικειμενοστρεφής* (object oriented) γλώσσα προγραμματισμού, η οποία σχεδιάστηκε από την Sun Microsystems (πλέον Oracle Corporation). Ένα από τα βασικά πλεονεκτήματα της Java έναντι



Σχήμα 2.1: Η λογική αρχιτεκτονική του συντακτικού αναλυτή

των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία από το λειτουργικό σύστημα και την πλατφόρμα. Η μόνη προϋπόθεση για την εκτέλεση ενός προγράμματος γραμμένου σε Java είναι η δυνατότητα εκτέλεσης της εικονικής μηχανής Java (Java Virtual Machine - JVM). Επί του παρόντος, υποστηρίζονται τα λειτουργικά συστήματα Linux, OS X, Solaris και Windows. Ένα ακόμα χαρακτηριστικό της Java, σε σχέση με άλλες γλώσσες προγραμματισμού, είναι η εγγενής υποστήριξη του *Unicode*. Το πρότυπο Unicode είναι ένα διεθνές πρότυπο κωδικοποίησης χαρακτήρων που έχει την δυνατότητα αναπαράστασης χαρακτήρων για τις περισσότερες γλώσσες. Έχει καθιερωθεί ως η προτιμότερη κωδικοποίηση για χρήση σε πολυγλωσσικά υπολογιστικά συστήματα και εφαρμογές.

Βασικές έννοιες της Java είναι το *αντικείμενο* (object) και η *κλάση* (class). Η Java χρησιμοποιεί κλάσεις για να οργανώσει τον κώδικα σε λογικές ενότητες, ουσιαστικά πρόκειται για κάποιο αρχείο *πηγαίου κώδικα* (source code) ή *εκτελέσιμου κώδικα* (object code). Ένα αντικείμενο είναι *στιγμιότυπο* (instance) μίας κλάσης, υπάρχει δε η δυνατότητα να υπάρχουν περισσότερα του ενός αντικείμενα που να είναι στιγμιότυπα της ίδιας κλάσης. Μία κλάση μπορεί να περιέχει *μεθόδους* (methods) και *πεδία* (fields). Οι μέθοδοι καθορίζουν την συμπεριφορά ενός αντικειμένου. Τα πεδία περιέχουν δεδομένα σχετικά με την κατάσταση αυτού του αντικειμένου. Μία έννοια που θα συναντήσουμε στην συνέχεια είναι οι διασυνδέσεις. Μια *διασύνδεση* (interface) ορίζει έναν τρόπο συμπεριφοράς που μπορεί να υλοποιηθεί από οποιαδήποτε κλάση. Δηλώνει

ένα σύνολο μεθόδων αλλά δεν προσφέρει την υλοποίηση τους. Οι διασυνδέσεις χρησιμοποιούνται για να περιγράψουν την λειτουργικότητα, αλλά όχι την πιθανή υλοποίηση. Διαφορετικές κλάσεις μπορούν να υλοποιήσουν μία διασύνδεση, κάθε μία με διαφορετικό τρόπο.

Όπως αναφέρθηκε παραπάνω, η παρούσα εργασία απαιτεί ένα συστατικό για την ανάγνωση αρχείων με κανόνες της ΛΛΓ. Το συστατικό αυτό θα πρέπει να περιέχει έναν λεκτικό αναλυτή και έναν συντακτικό αναλυτή τυπικής γλώσσας. Ένας *λεκτικός αναλυτής* (lexical analyzer, lexer, tokenizer ή scanner) είναι ένα πρόγραμμα που μετατρέπει μια ακολουθία από χαρακτήρες (string) σε μια ακολουθία από *λεκτικές μονάδες* (tokens). Αν και ως προγραμματιστές μπορούμε να δημιουργήσουμε λεκτικούς αναλυτές με συγγραφή κώδικα, συνήθως προτιμάται η χρήση μίας *γεννήτριας λεκτικών αναλυτών* (lexical analyzer generator) για την παραγωγή τους. Οι γεννήτριες λεκτικών αναλυτών είναι προγράμματα που δέχονται ένα σύνολο από κανόνες περιγραφής λεκτικών μονάδων και παράγουν αυτόματα τον κώδικα για την λεκτική ανάλυση τους. Η περιγραφή γίνεται συνήθως με χρήση κανονικών εκφράσεων ή με κάποια μορφή κανονικής γραμματικής. Αντίστοιχα, υπάρχουν και γεννήτριες συντακτικών αναλυτών, δηλαδή εφαρμογές που δημιουργούν αυτόματα τον κώδικα για την συντακτική ανάλυση μίας τυπικής γλώσσας, σύμφωνα πάντα με κάποιους κανόνες. Επειδή η είσοδος αυτών των συντακτικών αναλυτών είναι λεκτικές μονάδες, η λεκτική ανάλυση είναι ένα απαραίτητο στάδιο πριν την συντακτική ανάλυση. Για την παρούσα εργασία χρησιμοποιήσαμε την JavaCC™.

Η εφαρμογή JavaCC™(Java Compiler Compiler™) είναι μία γεννήτρια λεκτικών και συντακτικών αναλυτών τυπικής γλώσσας σε γλώσσα Java. Οι συντακτικοί αναλυτές που δημιουργεί είναι τύπου $LL(k)$. Πρόκειται για έναν τύπο συντακτικού αναλυτή από πάνω προς τα κάτω με δυνατότητα χρήσης k συμβόλων για πρόβλεψη. Οι συντακτικοί αναλυτές αυτού του τύπου μπορούν να αναλύσουν *ντετερμινιστικές συμφραστικά ανεξάρτητες γλώσσες* (deterministic context-free languages - DCFGs), ένα υποσύνολο των συμφραστικά ανεξάρτητων γλωσσών. Η συντακτική ανάλυση πραγματοποιείται σε γραμμικό χρόνο. Η δημιουργία ενός αναλυτή τυπικής γλώσσας γίνεται με βάση μία περιγραφή,

γραμμένη σε *Εκτεταμένη Μορφή Μπάκους-Νάουρ* (Extended Backus–Naur Form - EBNF). Εφαρμογές όπως η JavaCC χρησιμοποιούνται κυρίως από *μεταγλωττιστές* (compilers) και *διερμηνείς* (interpreters) για να διαβάζουν τα αρχεία πηγαίου κώδικα προγράμματος που πρέπει να μεταγλωττιστεί ή να εκτελεστεί. Ωστόσο, μπορούν να χρησιμοποιηθούν και σε άλλες εφαρμογές, όταν υπάρχει ανάγκη επεξεργασίας ή ερμηνείας δεδομένων που ακολουθούν τον φορμαλισμό κάποιας τυπικής γλώσσας, όπως στην περίπτωσή μας.

2.3 Υλοποίηση

Η υλοποίηση του συντακτικού αναλυτή αποτελείται από 6 έργα (projects), τα οποία φαίνονται, μαζί με τις εξαρτήσεις τους, στο Σχήμα 2.2. Ακολουθεί μία περιγραφή του κάθε έργου:

interfaces

Το έργο `interfaces` περιέχει μία σειρά διασυνδέσεων που περιγράφουν λειτουργικότητα που έχει σχέση με τις συμφραστικά ανεξάρτητες γραμματικές. Οι διασυνδέσεις είναι οι εξής:

- IParser: περιγράφει την λειτουργικότητα συντακτικών αναλυτών για συμφραστικά ανεξάρτητες γλώσσες και γραμματικές
- IGrammar: περιγράφει την λειτουργικότητα που πρέπει να παρέχει ένα αντικείμενο συμφραστικά ανεξάρτητης γραμματικής
- IRule: περιγράφει την λειτουργικότητα κανόνων μεταγραφής συμφραστικά ανεξάρτητης γραμματικής
- ISymbol: περιγράφει την λειτουργικότητα συμβόλων συμφραστικά ανεξάρτητης γραμματικής
- ILemma: περιγράφει την λειτουργικότητα λημμάτων
- Tree: περιγράφει δεντροδιαγράμματα

- Node: περιγράφει κόμβους δεντροδιαγραμμάτων

earley

Το έργο `earley` περιέχει την υλοποίηση ενός συντακτικού αναλυτή για συμφραστικά ανεξάρτητες γραμματικές που βασίζεται στον αλγόριθμο του Earley. Το μεγαλύτερο μέρος της λειτουργικότητας βρίσκεται στην κλάση `EarleyParser` που υλοποιεί την λειτουργικότητα που περιγράφεται από την διασύνδεση `IParser`. Δέχεται ως είσοδο μία λίστα από μεταβλητές τύπου συμβολοσειρά (String) και ένα αντικείμενο που υλοποιεί την διασύνδεση `IGrammar`. Η έξοδος του αναλυτή είναι ένα αντικείμενο που υλοποιεί την διασύνδεση Tree.

lfg-common

Το έργο `lfg-common` περιέχει διασυνδέσεις και κλάσεις που σχετίζονται με την ΛΛΓ. Ενδεικτικά αναφέρουμε τις κλάσεις `LFGrammar`, `LFGRule`, `LFGSymbol` και `LFGLemma` που υλοποιούν τις διασυνδέσεις `IGrammar`, `IRule`, `ISymbol` και `ILemma` αντίστοιχα. Επίσης περιέχονται οι κλάσεις για την περιγραφή λειτουργικών εξισώσεων και προσδιοριστών.

lfg-grammar-parser

Σε αυτό το έργο περιέχεται ο λεκτικός και συντακτικός αναλυτής για την ανάγνωση γραμματικών της ΛΛΓ. Περιέχει την κλάση `LfgGrammarParser` καθώς και κάποιες άλλες βοηθητικές κλάσεις που χρησιμοποιούνται από την πρώτη. Οι βασικές μέθοδοι που παρέχει η κλάση `LfgGrammarParser` είναι οι εξής:

> `public static LFGrammar evaluate(String file);`
> Ανάγνωση ενός αρχείου με το όνομα `file` και δημιουργία ενός αντικειμένου της κλάσης `LFGrammar`.
>
> `public static LFGrammar evaluateString(String grammar);`

Ανάγνωση της συμβολοσειράς `grammar` που περιέχει μία γραμματική και δημιουργία ενός αντικειμένου της κλάσης `LFGrammar`.

Επίσης σε αυτό το έργο βρίσκεται και το αρχείο `LfgGrammarParser.jj`. Το αρχείο αυτό περιέχει την περιγραφή του φορμαλισμού που θα πρέπει να διαθέτουν τα αρχεία γραμματικής που δέχεται ως είσοδο το σύστημά μας. Σε περίπτωση επέκτασης του φορμαλισμού μας, οι αλλαγές θα πρέπει να πραγματοποιηθούν σε αυτό το αρχείο και στη συνέχεια θα πρέπει να ξαναδημιουργήσουμε την κλάση `LfgGrammarParser` με χρήση του εργαλείου JavaCC όπως παρακάτω:

```
javacc -STATIC=false LfgGrammarParser.jj
```

lfg-solver

Το έργο `lfg-solver` περιέχει τον κώδικα για την επίλυση λειτουργικών περιγραφών. Οι πιο βασικές κλάσεις είναι οι `LFGAnalysis`, `LfgSolution` και `LfgSolver`.

Η κλάση `LFGAnalysis` αντιπροσωπεύει το αποτέλεσμα της συντακτικής ανάλυσης μίας πρότασης φυσικής γλώσσας. Περιέχει μηδέν, ένα ή περισσότερα αντικείμενα (σε περίπτωση αμφισημίας) του τύπου `LfgSolution`. Η κλάση `LFGAnalysis` περιέχει μεθόδους για εξαγωγή της πληροφορίας της συντακτικής ανάλυσης σε μορφή XML (Extensible Markup Language). Επίσης, μπορεί να αναπαραστήσει την πληροφορία χρησιμοποιώντας το πρότυπο SVG (Scalable Vector Graphics) για τις συστατικές δομές και το πρότυπο MathML (Mathematical Markup Language) για τις λειτουργικές δομές.

Η κλάση `LfgSolver` είναι η κλάση εκείνη που δέχεται ως είσοδο ένα δέντρο και από αυτό δημιουργεί και επιλύει μία λειτουργική περιγραφή. Χρησιμοποιεί αντικείμενα τύπου `LFGAnalysis` για την αναπαράσταση του αποτελέσματος.

lfg-parser

Το έργο `lfg-parser` περιέχει την κλάση `LFGParser` η οποία είναι και ο κυρίως συντακτικός αναλυτής της ΛΛΓ. Η βασική μέθοδος της κλάσης αυτής

είναι η μέθοδος `public LFGAnalysis analyse(String[] tokens);`. Η μέθοδος `analyse` δέχεται ως παράμετρο έναν πίνακα με τις λέξεις της προς ανάλυση πρότασης και επιστρέφει ένα αντικείμενο τύπου `LFGAnalysis` το οποίο περιέχει το αποτέλεσμα της ανάλυσης.

Ακολουθεί ένα παράδειγμα χρήσης της κλάσης `LFGParser`.

Listing 2.1: Παράδειγμα χρήσης της κλάσης `LFGParser`

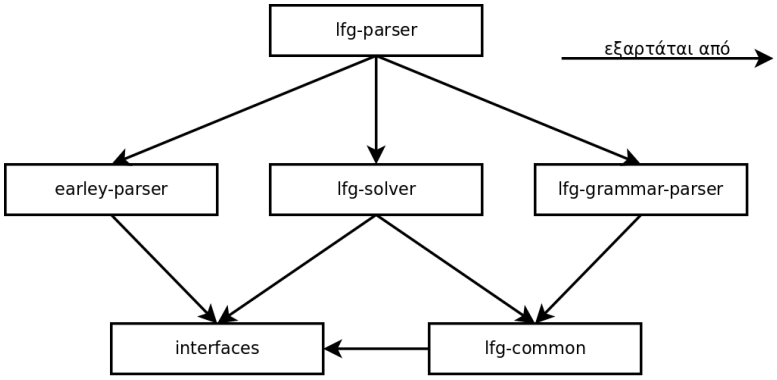
```
//read the grammar file
String filename = ...;
LFGrammar grammar = LfgGrammarParser.evaluate(filename);
LFGParser parser = new LFGParser(grammar);

//the sentence to be analyzed
String[] tokens = {"η", "Δανάη", "κοιμάται"};

//create the analysis
LFGAnalysis analysis = parser.analyse(tokens);

//get number of solutions
int solutionCount = analysis.getSolutionsCount();

for (int i = 0; i < solutionCount; i++) {
  //check if solution is valid
  boolean correct = analysis.getSolution(i).isCorrect();
  if (correct) {
    //print an xml represantation of the solution to screen
    String xml = analysis.getSolution(i).createXML(true);
    System.out.println(xml);
  }
}
```



Σχήμα 2.2: Τα τμήματα και οι εξαρτήσεις του συντακτικού αναλυτή

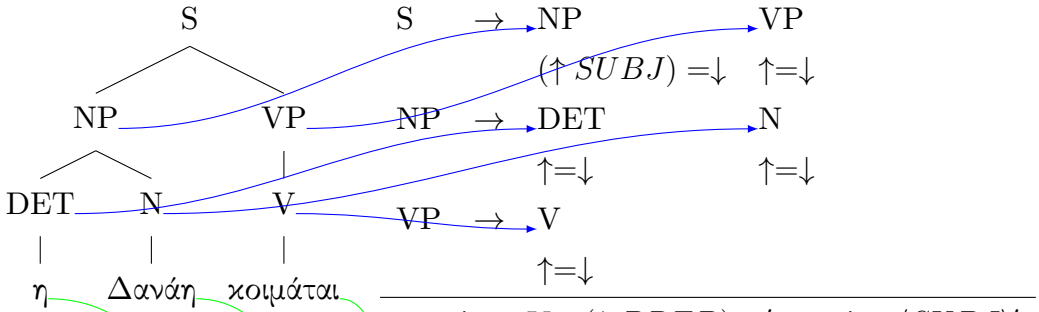
2.4 Λειτουργία

2.4.1 Δημιουργία της συστατικής δομής

Η δημιουργία της συστατικής δομής είναι το πρώτο βήμα για την συντακτική ανάλυση μίας πρότασης φυσικής γλώσσας σύμφωνα με την δοσμένη γραμματική και γίνεται με την βοήθεια ενός συντακτικού αναλυτή για συμφραστικά ανεξάρτητες γραμματικές. Ο συντακτικός αναλυτής που υλοποιήθηκε είναι μία τροποποιημένη έκδοση του αλγόριθμου του Earley. Δέχεται ως είσοδο μία πρόταση φυσικής γλώσσας και δημιουργεί μία δομή δεδομένων, ένα δέντρο, που αντιστοιχεί στο δεντροδιάγραμμα της πρότασης αυτής. Κατά την δημιουργία του δέντρου, αποθηκεύουμε στους κόμβους του αναφορές στους κανόνες της γραμματικής και στις καταχωρίσεις του λεξικού ώστε τα υπόλοιπα τμήματα του αναλυτή να έχουν την δυνατότητα πρόσβασης σε αυτή την πληροφορία που είναι απαραίτητη για την περαιτέρω ανάλυση (Σχήμα 2.3).

2.4.2 Δημιουργία της λειτουργικής δομής

Η λειτουργική περιγραφή μίας πρότασης φυσικής γλώσσας κατασκευάζεται διατρέχοντας το δέντρο και συλλέγοντας τα λειτουργικά υποδείγματα από τους κανόνες της γραμματικής. Για κάθε κόμβο n με πατρικό κόμβο τον p, δημιουργούμε μία κενή λειτουργική δομή f_n και προσθέτουμε στη λειτουρ-



κοιμάται, V	$(\uparrow PRED) =' \text{κοιμάμαι}\langle SUBJ\rangle'$
	$(\uparrow PERS) = THIRD$
Δανάη, N	$(\uparrow GEND) = FEM$
	$(\uparrow NUM) = SING$
	$(\uparrow CASE) = NOM$
	$(\uparrow PRED) =' \text{Δανάη}'$
η, DET	$(\uparrow GEND) = FEM$
	$(\uparrow NUM) = SING$
	$(\uparrow CASE) = NOM$

Σχήμα 2.3: Η συστατική δομή

γική περιγραφή τις λειτουργικές εξισώσεις που υπάρχουν στο σύμβολο του σώματος του κανόνα που αντιστοιχεί στον κόμβο, αντικαθιστώντας τα προσδιοριστικά $\downarrow$ και $\uparrow$ των εξισώσεων με αναφορές στις λειτουργικές δομές f_n και f_p αντίστοιχα. Στη συνέχεια επιλύουμε την λειτουργική περιγραφή, αρχικά τις εξισώσεις δήλωσης και στη συνέχεια τις εξισώσεις περιορισμού. Η λειτουργική δομή που αντιστοιχεί στην ρίζα του δέντρου είναι η λειτουργική δομή της πρότασης. Αν η δομή αυτή ικανοποιεί τα κριτήρια ορθότητας της ΛΛΓ και όλες τις εξισώσεις περιορισμού, τότε η πρόταση φυσικής γλώσσας ανήκει στην γλώσσα που περιγράφει η γραμματική.

2.5 Περιγραφή των αρχείων γραμματικής

Ένα αρχείο γραμματικής περιέχει έναν ή περισσότερους κανόνες μεταγραφής. Επίσης περιέχει και το λεξικό.

2.5.1 Περιγραφή των κανόνων

Ένα σύμβολο μπορεί να αποτελείται από συγκεκριμένους χαρακτήρες. Οι επιτρεπτοί χαρακτήρες είναι όλα τα γράμματα του αγγλικού και ελληνικού αλφάβητου, τα αριθμητικά ψηφία (0-9), ο χαρακτήρας underscore (_) και το μονό εισαγωγικό (').

Ένας κανόνας έχει την μορφή $A \rightarrow X;$, όπου:

A ένα μη τερματικό σύμβολο της γραμματικής

→ ο χαρακτήρας → (βέλος δεξιά). Εναλλακτικά μπορούμε να χρησιμοποιήσουμε -> (ο χαρακτήρας παύλα ακολουθούμενος από τον χαρακτήρα μεγαλύτερο από)

X ένα ή περισσότερα επισημειωμένα σύμβολα

; ο χαρακτήρας **;** (ελληνικό ερωτηματικό) που σηματοδοτεί το τέλος του κανόνα

Ένα επισημειωμένο σύμβολο αποτελείται από:

α. ένα μη τερματικό σύμβολο της γραμματικής

β. προαιρετικά έναν ποσοδείκτη, δηλαδή κανένα ή έναν χαρακτήρας από το σύνολο {?,*,+}

γ. προαιρετικά το σώμα του συμβόλου, δηλαδή οι χαρακτήρες **{ }** (αριστερή και δεξιά αγκύλη) μεταξύ των οποίων μπορεί να υπάρχει κανένα, ένα ή περισσότερα λειτουργικά πρότυπα

Οι ποσοδείκτες (quantifiers) χρησιμοποιούνται για να δηλώσουν επανάληψη. Πιο συγκεκριμένα

? ο χαρακτήρας ? (αγγλικό ερωτηματικό) δηλώνει ότι το σύμβολο μπορεί να υπάρχει μηδέν ή μία φορά.

Ο κανόνας μεταγραφής `NP → DET ADJ? N;` σημαίνει ότι μία ονοματική φράση αποτελείται από έναν προσδιοριστή, κανένα ή ένα επίθετο και ένα ουσιαστικό.

* ο χαρακτήρας * (αστερίσκος) δηλώνει ότι το σύμβολο μπορεί να υπάρχει μηδέν, μία ή περισσότερες φορές.

Ο κανόνας μεταγραφής `NP → DET ADJ* N;` σημαίνει ότι μία ονοματική φράση αποτελείται από έναν προσδιοριστή, κανένα, ένα ή περισσότερα επίθετα και ένα ουσιαστικό

\+ ο χαρακτήρας + (συν) δηλώνει ότι το σύμβολο μπορεί να υπάρχει μία ή περισσότερες φορές.

Ο κανόνας μεταγραφής `NP → DET ADJ+ N;` σημαίνει ότι μία ονοματική φράση αποτελείται από έναν προσδιοριστή, ένα ή περισσότερα επίθετα και ένα ουσιαστικό.

Το σώμα του συμβόλου, εάν υπάρχει, περιέχει τα λειτουργικά πρότυπα της ΛΛΓ.

Το σώμα μπορεί να είναι άδειο (2.1) ή και να μην υπάρχει (2.2). Οι δύο τρόποι είναι ισοδύναμοι μεταξύ τους.

$$\mathrm{S} \rightarrow \mathrm{VP}\ \{\}; \tag{2.1}$$

$$\mathrm{S} \rightarrow \mathrm{VP}; \tag{2.2}$$

Στο (2.3) δίνεται ένας κανόνας όπου το σύμβολο VP περιέχει το λειτουργικό πρότυπο $\uparrow = \downarrow$. Κάθε λειτουργικό πρότυπο πρέπει να τερματίζεται με τον χαρακτήρα ; (ελληνικό ερωτηματικό)

$$\mathrm{S} \rightarrow \mathrm{VP}\ \{\uparrow = \downarrow;\}; \tag{2.3}$$

Οι αλλαγές γραμμής και τα κενά δεν επηρεάζουν την σημασία του κανόνα. Συνεπώς, τα παραδείγματα (2.4) και (2.5) είναι ισοδύναμα.

$$\text{S} \rightarrow \text{NP } \{(\uparrow \text{number})=(\downarrow \text{number}); (\uparrow \text{case})=(\downarrow \text{case});\}; \tag{2.4}$$

$$\begin{aligned} &S \rightarrow NP \\ &\quad \{ \\ &\quad (\uparrow number) = (\downarrow number); \\ &\quad (\uparrow case) = (\downarrow case); \\ &\quad \}; \end{aligned} \tag{2.5}$$

2.5.2 Περιγραφή του λεξικού

Το λεξικό είναι ένα σύνολο από καταχωρίσεις που αφορούν λεκτικούς τύπους. Η δομή μίας καταχώρισης στο λεξικό αποτελείται από:

α. τον λεκτικό τύπο

β. το λήμμα στο οποίο ανήκει ο λεκτικός τύπος

γ. τη συντακτική κατηγορία του λεκτικού τύπου, ένα μη τερματικό σύμβολο της γραμματικής

δ. προαιρετικά, το σώμα του λεκτικού τύπου, δηλαδή οι χαρακτήρες **{ }** (αριστερή και δεξιά αγκύλη) μεταξύ των οποίων μπορεί να υπάρχει κανένα, ένα ή περισσότερα λειτουργικά πρότυπα

ε. ο χαρακτήρας **;** (ελληνικό ερωτηματικό) που σηματοδοτεί το τέλος της καταχώρισης

Στη συνέχεια μπορούμε να δούμε ένα δείγμα λεξικού που περιέχει δύο καταχωρίσεις.

```
παιδί παιδί N {
    (↑gender)=neut;
    (↑number)=sing;
    (↑case)=nom;
    (↑PRED)='παιδί';
};

κοιμάται κοιμάμαι V
{
    (↑tense)=nonpast;
    (↑aspect)=imperf;
    (↑person)=third;
    (↑PRED)='κοιμάμαι<SUBJ>';
};
```

2.5.3 Περιγραφή των λειτουργικών προτύπων

Στην παρούσα έκδοση υποστηρίζεται η χρήση των παρακάτω τελεστών:

Ο τελεστής =

Τα λειτουργικά πρότυπα που περιέχουν τον τελεστή = (ίσον) χρησιμοποιούνται για την λειτουργία της ενοποίησης (unification) (2.6). Ο τελεστής ενοποιεί το δεξί μέλος με το αριστερό και το αποτέλεσμα της ενοποίησης αποθηκεύεται και στα δύο μέλη.

$$\uparrow=\downarrow \tag{2.6}$$

Ο τελεστής $\in$

Ο τελεστής $\in$ (ανήκει σε) προσθέτει το αριστερό μέλος στο σύνολο που δηλώνεται από το δεξί μέλος (2.7).

$$\downarrow = (\uparrow\ ADJ) \tag{2.7}$$

Ο τελεστής $==$

Τα λειτουργικά πρότυπα που περιέχουν τον τελεστή $==$ (ίσον ίσον) χρησιμοποιούνται για να περιγράψουν εξισώσεις περιορισμού. Στην (2.8), το χαρακτηριστικό case της δομής που δηλώνεται από το αριστερό μέλος πρέπει να έχει την τιμή nom.

$$(\uparrow\ case) = nom \tag{2.8}$$

Ο τελεστής $!=$

Τα λειτουργικά πρότυπα που περιέχουν τον τελεστή $!=$ (θαυμαστικό ίσον) ή $\neq$ (διάφορο από) χρησιμοποιούνται για να περιγράψουν επίσης εξισώσεις περιορισμού. Στην (2.9), το χαρακτηριστικό case της δομής που δηλώνεται από το αριστερό μέρος δεν πρέπει να έχει την τιμή nom.

$$(\uparrow\ case) != nom \tag{2.9}$$

Ο τελεστής $\neg$

Ο τελεστής $\neg$ μπορεί να χρησιμοποιηθεί σε περιπτώσεις που μας ενδιαφέρει να μην ικανοποιείται ένας περιορισμός. Η (2.10) είναι ισοδύναμη με την (2.9).

$$\neg((\uparrow\ case) == nom) \tag{2.10}$$

Λειτουργικά πρότυπα υπαρξιακού περιορισμού

Τέλος, η εφαρμογή υποστηρίζει τη χρήση λειτουργικών προτύπων υπαρξιακού περιορισμού που ελέγχουν μόνο για την ύπαρξη μίας ιδιότητας. Η (2.11) θα ικανοποιηθεί μόνο εάν υπάρχει χαρακτηριστικό case στη δομή που δηλώνεται από το πρότυπο. Μπορούμε να χρησιμοποιήσουμε τον τελεστή ¬ εάν ο περιορισμός απαιτεί την απουσία του χαρακτηριστικού case όπως στην (2.12).

$$(\uparrow\ case) \tag{2.11}$$

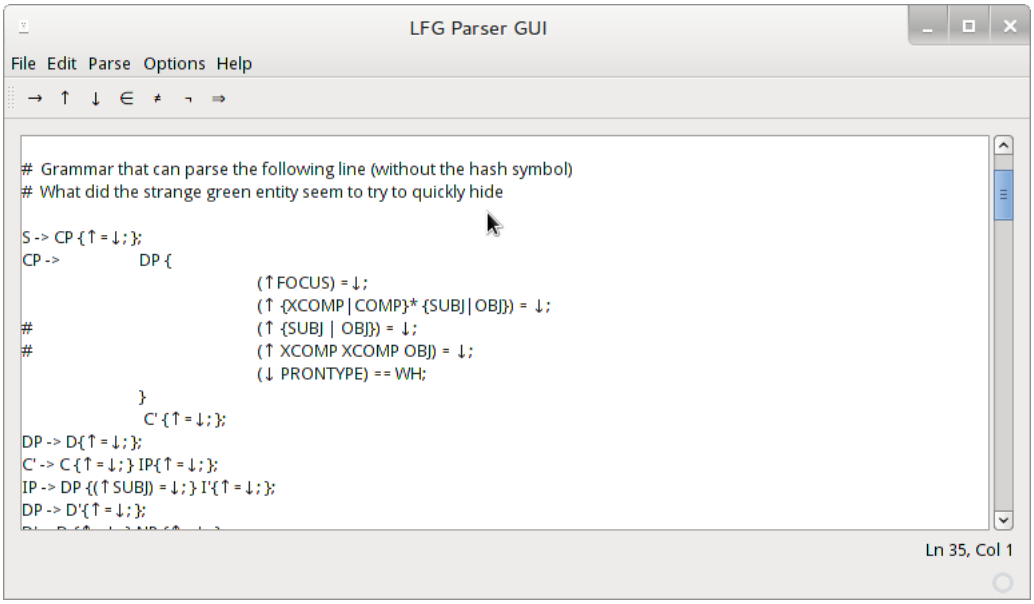
$$\neg(\uparrow\ case) \tag{2.12}$$

2.6 Η γραφική διεπαφή

Οι συντακτικοί αναλυτές είναι προγράμματα/υποσυστήματα που χρησιμοποιούνται σε μεγαλύτερα συστήματα και δεν υπάρχει η ανάγκη άμεσης αλληλεπίδρασης με τον χρήστη. Για ανάγκες επίδειξης αλλά και ελέγχου του αποτελέσματος της ανάλυσης δημιουργήθηκε μία γραφική διεπαφή χρήστη (GUI) για την συγγραφή γραμματικών και την εκτέλεση συντακτικών αναλύσεων, καθώς και την άμεση προβολή της εξαγόμενης πληροφορίας. Η εφαρμογή δεν έχει δυνατότητες όπως μορφολογική ανάλυση με χρήση μορφολογικού λεξικού, χωρισμό προτάσεων (sentence splitting) ή ανάλυση σε συλλογή κειμένων (corpus/document collection).

Το κυρίως παράθυρο της εφαρμογής, όπως βλέπουμε στο σχήμα 2.4, έχει ένα μενού, μία σειρά κουμπιών με χαρακτήρες που χρησιμοποιεί ο φορμαλισμός και δεν υπάρχουν στο πληκτρολόγιο, την περιοχή επεξεργασίας του κειμένου και, τέλος, κάτω δεξιά την τρέχουσα θέση του κέρσορα.

Στη συνέχεια μπορούμε να χρησιμοποιήσουμε αυτή την γραμματική για να πραγματοποιήσουμε μία συντακτική ανάλυση σε μία πρόταση μέσω του



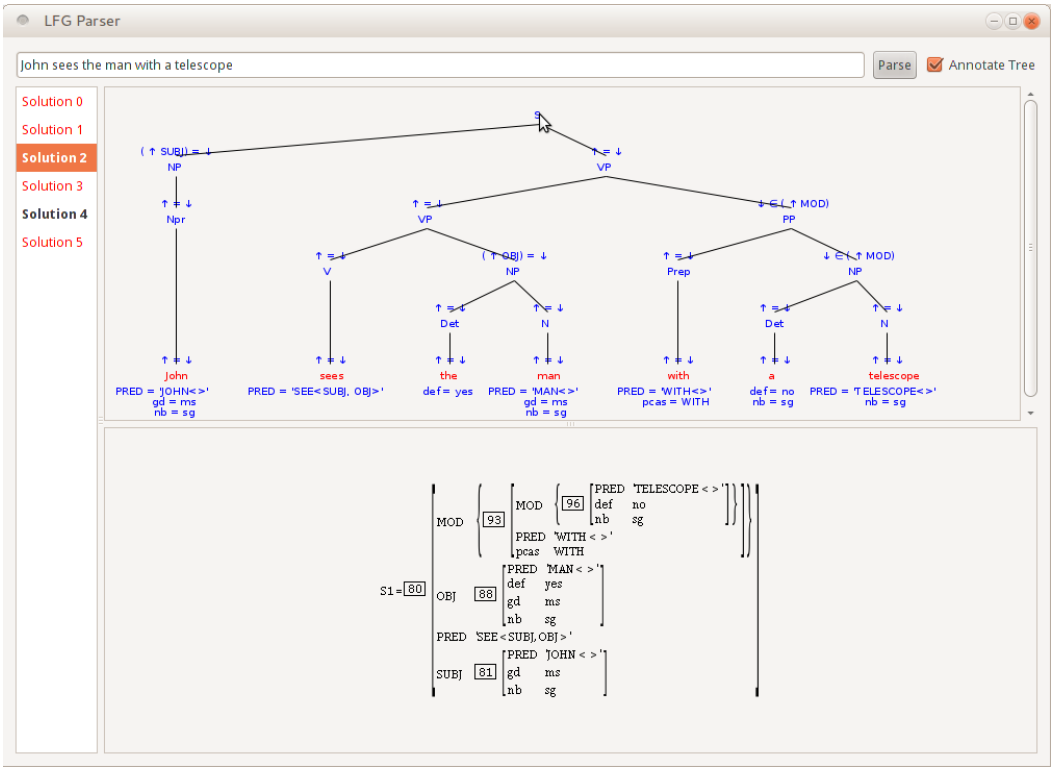
Σχήμα 2.4: Συγγραφή γραμματικών

μενού Misc → Parse. Θα εμφανιστεί ένα νέο παράθυρο που μπορούμε να χρησιμοποιήσουμε για την συντακτική ανάλυση, σχήμα 2.5.

Πάνω αριστερά γράφουμε την πρόταση στην οποία θέλουμε να πραγματοποιήσουμε συντακτική ανάλυση και πατάμε το κουμπί Parse. Στην περιοχή αριστερά θα εμφανιστούν τα αποτελέσματα της συντακτικής ανάλυσης, μία λίστα με τις πιθανές αναλύσεις: με κόκκινα γράμματα εμφανίζονται οι λύσεις που είναι λάθος, ενώ με μαύρα γράμματα εμφανίζονται οι λύσεις που είναι σωστές σύμφωνα με την γραμματική.

Μόλις επιλέξουμε μία λύση, είτε σωστή είτε λάθος, στην πάνω δεξιά περιοχή θα εμφανιστεί η συστατική δομή της πρότασης. Αν δεν θέλουμε να εμφανίζονται τα λειτουργικά πρότυπα (επισημειώσεις) της συστατικής δομής, τότε μπορούμε να αποεπιλέξουμε το Annotate Tree.

Εάν μετακινήσουμε τον κέρσορα πάνω από κάποιο σύμβολο ενός κόμβου της συστατική δομής, ο κέρσορας θα αλλάξει μορφή και με το κλικ του ποντικιού θα εμφανιστεί η λειτουργική δομή του συγκεκριμένου κόμβου στην περιοχή κάτω δεξιά. Αν την στιγμή που κάνουμε κλικ στο σύμβολο έχουμε πατημένο το πλήκτρο Shift, τότε η λειτουργική δομή θα εμφανιστεί σε νέο



Σχήμα 2.5: Συντακτική ανάλυση

παράθυρο. Με αυτό τον τρόπο μπορούμε να δούμε ταυτόχρονα τις λειτουργικές δομές για όλες τις λύσεις. Επίσης με δεξί κλικ πάνω στο δέντρο ή στη λειτουργική δομή εμφανίζεται ένα αναδυόμενο μενού (popup menu) με το οποίο μπορούμε να σώσουμε το αποτέλεσμα σαν μορφή εικόνας ή σαν αρχείο XML.

Κεφάλαιο 3

Συμπεράσματα και μελλοντικές προοπτικές

Στην παρούσα διπλωματική εργασία περιγράψαμε τον φορμαλισμό της Λεξικής-Λειτουργικής Γραμματικής (ΛΛΓ) και στη συνέχεια δείξαμε πώς μπορεί να σχεδιαστεί και να υλοποιηθεί ένας αρθρωτός συντακτικός αναλυτής για τον παραπάνω φορμαλισμό.

Η υλοποίηση αποτελείται από τρία μέρη. Το πρώτο μέρος αφορά την ανάπτυξη ενός συντακτικού αναλυτή για συμφραστικά ανεξάρτητες γραμματικές που βασίστηκε στον αλγόριθμο του Earley, έναν από τους πιο αποδοτικούς αλγόριθμους γενικής χρήσης, χωρίς περιορισμούς ως προς την μορφή των κανόνων, ο οποίος επιτρέπει την χρήση του συνόλου των συμφραστικά ανεξάρτητων γραμματικών.

Το δεύτερο και πιο απαιτητικό μέρος αφορά την υλοποίηση του κώδικα για τον χειρισμό και την επίλυση των λειτουργικών εξισώσεων. Η μεγαλύτερη δυσκολία σε αυτό το κομμάτι έγκειται στην λειτουργική αβεβαιότητα (functional uncertainty) και στις εξισώσεις που έχουν την μορφή από μέσα προς τα έξω (inside-out).

Τρίτο και τελευταίο μέρος είναι η δημιουργία ενός λεκτικού και συντακτικού αναλυτή για την ανάγνωση των αρχείων γραμματικής. Η χρήση της εφαρμογής JavaCC συμβάλει στην γρήγορη και αποτελεσματική υλοποίηση

του κώδικα, στην ελαχιστοποίηση της πιθανότητας δημιουργίας προγραμματιστικών λαθών (bugs) ενώ επιτρέπει την εύκολη πραγματοποίηση αλλαγών ή και επεκτάσεων της μορφής της γραμματικής.

Η αρθρωτή δομή σε τρία μέρη βοήθησε στην γρηγορότερη υλοποίηση και στον έλεγχο της λειτουργικότητας, ενώ ταυτόχρονα επιτρέπει μελλοντικά την εύκολη επέκταση με νέες δυνατότητες. Ως γλώσσα προγραμματισμού επιλέξαμε την Java η οποία παρέχει την δυνατότητα χρήσης σε μεγάλο αριθμό συστημάτων, ενώ η εγγενής υποστήριξή της για το πρότυπο UNICODE επιτρέπει την χρήση του συντακτικού αναλυτή για ανάλυση οποιασδήποτε φυσικής γλώσσας. Για την αποθήκευση της εξαγόμενης πληροφορίας κάναμε χρήση της τεχνολογίας XML (Extensible Markup Language), που αποτελεί ένα κοινό πρότυπο για την ανταλλαγή δεδομένων μεταξύ προγραμμάτων και υπολογιστικών συστημάτων, καθώς επίσης και των προτύπων SVG (Scalable Vector Graphics) και MathML (Mathematical Markup Language) που βασίζονται σε XML. Εκτός από την χρήση του συντακτικού αναλυτή ως υποσυστήματος σε συστήματα ΕΦΓ, αναπτύχθηκε μία γραφική διεπαφή χρήστη, η οποία επιτρέπει την άμεση προβολή της εξαγόμενης πληροφορίας, ώστε να μπορεί να χρησιμοποιηθεί ως αυτόνομο εργαλείο για γλωσσολογική ανάλυση ή ακόμα και για παιδαγωγική χρήση.

Όπως κάθε λογισμικό, ο συντακτικός αναλυτής επιδέχεται βελτιώσεις όπως βελτιστοποίηση του κώδικα και μείωση του χρόνου εκτέλεσης που απαιτείται για την συντακτική ανάλυση. Επιπλέον, ως μελλοντικές βελτιώσεις προτείνουμε τη προσθήκη λειτουργικότητας για την δημιουργία και χρήση μορφολογικών λεξικών βασισμένων σε πεπερασμένα αυτόματα για πιο γρήγορη μορφολογική ανάλυση της εισόδου. Παρόλο που ο αναλυτής υλοποιεί ήδη ένα μεγάλο μέρος της λειτουργικότητας σύμφωνα με την θεωρία της ΛΛΓ, ωστόσο θα μπορούσε να επεκταθεί με επιπρόσθετες λειτουργίες για την υποστήριξη επιπλέον επιπέδων γλωσσολογικής ανάλυσης όπως την παραγωγή αναφορικών ή σημασιολογικών δομών. Ο καταμερισμός της λειτουργικότητας του αναλυτή σε επιμέρους αρθρώματα επιτρέπει την εύκολη αντικατάστασή τους με άλλες υλοποιήσεις. Ενδιαφέρον θα είχε η υλοποίηση συντακτικών αναλυ-

τών για συμφραστικά ανεξάρτητες γραμματικές που βασίζονται σε άλλους αλγόριθμους (όπως του CYK) καθώς και η δυνατότητα ανάγνωσης γραμματικών άλλων συστημάτων (όπως του XLE). Τέλος, οι σύγχρονοι υπολογιστές έχουν την δυνατότητα πολυεπεξεργασίας (multiprocessing), αφού διαθέτουν περισσότερους από έναν επεξεργαστές. Η προσθήκη υποστήριξης για πολυνημάτωση (multithreading) θα αξιοποιούσε την δυνατότητα αυτή ώστε να μειώσουμε ακόμα περισσότερο τον χρόνο εκτέλεσης.

Βιβλιογραφία

[1] Joan Bresnan. *Lexical-Functional Syntax*. Blackwell Publishers, Oxford, 2001.

[2] N. Chomsky. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113--124, September 1956.

[3] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137--167, June 1959.

[4] Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell, and Annie Zaenen, editors. *Formal Issues in Lexical-Functional Grammar*. CSLI Publications, Stanford, CA, 1995.

[5] Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94--102, February 1970.

[6] Jay Clark Earley. *An efficient context-free parsing algorithm*. PhD thesis, Pittsburgh, PA, USA, 1968. AAI6907901.

[7] Yehuda N. Falk. *Lexical-Functional Grammar: An Introduction to Parallel Constraint-Based Syntax*. CSLI Publications, Stanford, CA, 2001.

[8] Nissim Francez and Shuly Wintner. *Unification Grammars*. Cambridge University Press, New York, NY, USA, 2011.

[9] John E. Hopcroft, Rajeev Motwani, Rotwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computability*.

Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2000.

[10] Ronald M. Kaplan. The formal architecture of Lexical-Functional Grammar. In Chu-Ren Huang and Keh-Jiann Chen, editors, *Proceedings of the Republic of China Computational Linguistics Conference (ROCLING II)*, pages 3--18, Taipei, 1989. Academia Sinica. Reprinted in Mary Dalrymple, Ronald M. Kaplan, John Maxwell, and Annie Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 7--27. Stanford: CSLI Publications. 1995.

[11] Ronald M. Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173--281. The MIT Press, Cambridge, MA, 1982. Reprinted in Mary Dalrymple, Ronald M. Kaplan, John Maxwell, and Annie Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 29--130. Stanford: CSLI Publications. 1995.

[12] Peter Linz. *An Introduction to Formal Languages and Automata.* Jones and Bartlett Publishers, Inc., USA, 3rd edition, 2001.

[13] Alan P. Parkes. *A Concise Introduction to Languages and Machines.* Springer Publishing Company, Incorporated, 1 edition, 2008.

[14] Stuart M. Shieber. *Constraint-based Grammar Formalisms: Parsing and Type Inference for Natural and Computer Languages.* MIT Press, Cambridge, MA, USA, 1992.

Παράρτημα Α΄

Γλωσσάρι

γραμματική βασισμένη σε περιορισμούς constraint based grammar

γραμματική φραστικής δομής phrase structure grammar

γραφική διεπαφή χρήστη graphical user interface

δομή ιδιοτήτων feature structure

δομή χαρακτηριστικών-τιμών feature structure

ενοποίηση unification

ενοποιητική γραμματική unification grammar

εξίσωση δήλωσης defining equation

εξίσωση περιορισμού constraining equation

επεξεργασία φυσικής γλώσσας natural language processing

κανόνας φραστικής δομής phrase structure rule

λειτουργική δομή functional structure, f-structure

λειτουργική εξίσωση functional equation

λειτουργική περιγραφή functional description, f-description

Λεξική-Λειτουργική Γραμματική lexical-functional grammar

λειτουργικό πρότυπο functional schema

συμβολοσειρά string

συμφραστικά ανεξάρτητη γλώσσα context-free language

συμφραστικά ανεξάρτητη γραμματική context-free grammar

συντακτική ανάλυση parsing

συντακτικός αναλυτής parser

συστατική δομή constituent structure, c-structure

Παράρτημα Β΄

Ακρωνύμια

AVM attribute-value matrix

CFG context-free grammar

GUI graphical user interface

LFG lexical-functional grammar

NLP natural language processing

UG unification grammar

ΕΦΓ επεξεργασία φυσικής γλώσσας

ΛΛΓ Λεξική-Λειτουργική Γραμματική