ΚΕΦΑΛΑΙΟ 3

Γραμματικές - Τεχνολογητές

3.1. Γραμματικές Γενικά

Σ' αυτό το κεφάλαιο θα περιγράψουμε τις **γραμματικές** που θα χρησιμοποιήσουμε για να *τεχνολογήσουμε* την φυσική γλώσσα. Θα χρησιμοποιήσουμε τις ***Γραμματικές Ορισμένης Φράσης*** *(ΓΟΦ) (Definite Clause Grammars)* τις οποίες θα εμπλουτίσουμε με ***ορίσματα*** *(attributes).* Έτσι, θα περιγράψουμε τις πληροφορίες που είναι αναγκαίες για την αναπαράσταση της φυσικής γλώσσας, και μάλιστα αυτές που υπερβαίνουν τα όρια της *φραστικής δομής*.

Οι σύγχρονες γραμματικές έχουν **χαρακτηριστικά** που θεωρήθηκαν από αρκετούς πρωτοπόρους της γνωστικής επιστήμης ως εγγενή στοιχεία της γλώσσας (η γλώσσα νοείται εδώ ως χαρακτηριστική ικανότητα του ανθρωπίνου είδους). Σ' αυτό υπάρχουν βέβαια και εξαιρέσεις. Κάποια από αυτά τα χαρακτηριστικά έχουν αμφισβητηθεί αλλά τα χρησιμοποιούμε γιατί, αν και ανεπαρκή, απλοποιούν τον *φορμαλισμό* χωρίς οι γραμματικές να μειώνουν την κάλυψη των γλωσσικών δεδομένων. Τα χαρακτηριστικά που θα εξετάσουμε εδώ είναι: η ύπαρξη ***φραστικών συστατικών***, η ***αναδρομή*** και η ***ανεξαρτησία συμφραζομένων***.

3.1.1. Φραστικά Συστατικά

Όταν επεξεργαζόμαστε μια *συμβολοακολουθία* (ακολουθία συμβόλων ή χαρακτήρων) η μικρότερη δυνατή μονάδα (άτομο) σ' αυτή την επεξεργασία είναι το *γράμμα* που αντιστοιχεί σε ένα *σύμβολο*. Οι *συλλαβές* και τα *μορφήματα* είναι οι αμέσως επόμενες γλωσσικές μονάδες που συγκροτούνται από γράμματα και οι *λέξεις* που αποτελούνται από μορφήματα ή συλλαβές και τελικώς από γράμματα. Η *συντακτική δομή* μιας φράσης, που είναι σύνολο από λέξεις, αποτελεί την επόμενη, στη σειρά της ιεραρχίας, οργάνωση της γλωσσικής γνώσης.

Πιστεύεται ότι μπορούμε να αναγνωρίσουμε σύνολα λέξεων τα οποία αποτελούν λειτουργικά διακριτές ομάδες μέσα στην πρόταση. Τις ομάδες αυτές τις αποκαλούμε ***φραστικά συστατικά*** *(συστατικά της φράσης).* Χρησιμοποιούμε για την αναγνώρισή τους διάφορα διαγνωστικά κριτήρια. Σ' αυτά τα συστατικά δίνονται και διακριτικά ονόματα (πρβλ. με κεφάλαιο 2), όπως *Ονοματική Φράση*, *Ρηματική Φράση* κλπ

Παραδείγματα:

(α) Η ερωτηματική έκφραση αντικαθιστά ένα συστατικό.
Είδα τον Γιάννη. -- Ποιόν είδες;
(Το συστατικό εδώ ειναι η Ονοματική Φράση "τον Γιάννη".)
Ήρθα χθές με την βροχή. -- Πότε ήρθες;
(Το συστατικό εδώ είναι η Επιρρηματική Φράση "χθες με την βροχή".)
(β) Η αντωνυμική έκφραση *"το ίδιο"* αντικαθιστά συστατικό (ρηματική φράση).

Εγώ κάθομαι στα κάρβουνα. – Και εγώ το ίδιο.
(Το συστατικό εδώ είναι "κάθομαι στα κάρβουνα")

(γ) Σύζευξη: μόνο συστατικά του ιδίου τύπου μπορούν να συζευχθούν.
Ο Γιάννης και ο Κώστας
**Ο Γιάννης και τρέχει* [1]

3.1.2 Αναδρομή

Η **αναδρομή** παρουσιάζεται στη φυσική γλώσσα και, θεωρητικά, είναι δυνατόν να υπάρξουν προτάσεις απείρου μήκους. Αυτό δεν γίνεται στην πράξη γιατί ο άνθρωπος έχει πεπερασμένες δυνατότητες επεξεργασίας. Οι γραμματικές όμως που προσπαθούν να εκφράσουν τις γλωσσικές ικανότητες του ανθρώπου είναι σε θέση να παράγουν *συμβολοακολουθίες (strings)* απείρου μήκους, όπως:

Ο Κώστας είπε ότι ο Γιάννης πιστεύει ότι ο Σπύρος νόμισε ότι η Ελένη υπεστήριξε ότι ….

3.1.3. Ανεξαρτησία Συμφραζομένων

Ορισμός

Γραμματική ***ανεξάρτητη συμφραζομένων*** *(context free)* είναι μία *πλειάδα (tuple)* η οποία ορίζεται ως εξής:

$$\mathbf{G = < V_T, V_N, S, R>}$$

όπου $\mathbf{V_T}$: το **σύνολο** των **τερματικών συμβόλων** της γλώσσας
(πχ ο, η, σκύλος, γάτος, τρώει, νιαουρίζει κλπ)
$\mathbf{V_N}$: το **σύνολο** των **μη τερματικών συμβόλων** της γλώσσας
(πχ. ΟΦ, ΡΦ, Ο, Ρ, ΑΡΘ κλπ)
S : το **αρχικό σύμβολο** της γραμματικής
(συνήθως στα Ελληνικά χρησιμοποιούμε το Π: 'Πρόταση')
R : το **σύνολο** των **κανόνων** της γλώσσας.
Οι κανόνες είναι της μορφής: $A = \psi$
όπου $A \in V_N$ *και* $\psi \in \{V_N \cup V_T\}^*$
Η κενή συμβολοακολουθία ***e*** *ανήκει στο σύνολο* V_T ($e \in V_T$)

Η κενή συμβολοακολουθία χρησιμεύει σε ορισμένες θεωρίες για να αναλύσουν φράσεις όπου υποτίθεται ότι κάποιο συστατικό έχει μεταφερθεί σε άλλο σημείο της φραστικής δομής (α) ή λείπει (β) και έχει αφήσει πίσω του, ας πούμε, μια *κενή υποδοχή*.

Παραδείγματα
(α) *Πρόσεχε.*
(β) *Ποιόν είπε η Ελένη ότι είδε ο Γιάννης;*

[1] Με αστερίσκο (*) σημειώνονται οι *μη γραμματικές φράσεις.*

Στην ακόλουθη μικρή *συμφραστικώς ανεξάρτητη* γραμματική με *κανόνες φραστικής δομής*, τα τερματικά σύμβολα δίνονται με μικρά γράμματα και τα μη τερματικά με κεφαλαία.

S	→	NP VP
NP	→	DET (A*) N
VP	→	V (NP)
DET	→	ο, τον
N	→	σκύλος, ποντικό
V	→	νυστάζει, κυνηγά

Συχνά χρησιμοποιούμε **παρενθέσεις** όταν γράφουμε γραμματικούς κανόνες. Η παρένθεση εισάγει ένα είδος συντομογραφίας. Ο αστερίσκος σημαίνει *καμία* ή *περισσότερες* φορές εμφάνιση. Για παράδειγμα, ο κανόνας (1) αναλύεται στους κανόνες (1.1) και (1.2).

(1)	ΟΦ	→	ΑΡΘ (Ε) Ο
(1.1)	ΟΦ	→	ΑΡΘ Ε Ο
(1.2)	ΟΦ	→	ΑΡΘ Ο

Ασκήσεις

Α *3.1.1.*
Μπορείτε να αναφέρετε δομές της Ελληνικής οι οποίες μπορούν να περιγραφούν με αναδρομικές γραμματικές; Γράψτε τους σχετικούς κανόνες.

Α. *3.1.2.*
Μπορείτε να γράψετε τη γραμματική που αναγνωρίζει τις προτάσεις;
ο σκύλος έδωσε ένα παξιμάδι στον ποντικό.
ο σκύλος ξέρει ότι ο ποντικός νυστάζει.

3.2. Σύγκριση Γραμματικών και Τεχνολογητών (Parsers)

Μια γραμματική (με κανόνες φραστικής δομής, λογουχάρη) χρησιμεύει για να περιγράφει απλώς (*περιγραφική* λειτουργία) τις λέξεις που ανήκουν στη γλώσσα που ορίζεται από τη γραμματική αυτή. Αυτή η λειτουργία αντιστοιχεί στη *δηλωτική ερμηνεία* της γραμματικής.

Από την άλλη πλευρά, η ίδια γραμματική μπορεί να ερμηνευτεί είτε ως αλγόριθμος ανάλυσης *(συντακτική αναγνώριση)* είτε ως αλγόριθμος παραγωγής (*παραγωγική* λειτουργία). Έτσι, με τους κανόνες της γραμματικής μπορούμε να ελέγξουμε την 'νομιμότητα' μιας λέξης, δηλαδή, αν είναι σύμφωνη με τους γραμματικούς κανόνες, ή να παράγουμε όλες τις δυνατές 'νόμιμες' λέξεις. Αυτή η λειτουργία αντιστοιχεί στη *διαδικασιακή ερμηνεία*.

3.3. **Τεχνολογητές** (Parsers)

Οι **τεχνολογητές (parsers)** είναι υπολογιστικά συστήματα που επεξεργάζονται την φυσική γλώσσα, και προσδιορίζουν αν μια πρόταση παράγεται από μια συγκεκριμένη γραμματική. Σ' αυτό το κεφάλαιο θα εξετάσουμε τον συμβολισμό (notation) των ***Γραμματικών Ορισμένης Φράσης*** *(ΓΟΦ) (Definite Clause Grammars)*, οι οποίες μοιάζουν με *δηλωτικές γραμματικές*.

Είναι εύκολη η μετατροπή των *κανόνων φραστικής δομής* (phrase structure rules) σε κανόνες της Prolog. Πράγματι η Prolog έχει τη δυνατότητα αυτή. Γράφουμε κανόνες σε συμβολισμό *Γραμματικής Ορισμένης Φράσης* και η Prolog τους μετατρέπει σε δικούς της κανόνες.

Ο συμβολισμός αυτός είναι πολύ απλός και μοιάζει με τον συμβολισμό των κανόνων Φραστικής Δομής (πρβλ. και επόμενη παράγραφο). Παράδειγμα από απόσπασμα Γραμματικής σε συμβολισμό Γραμματικής Ορισμένης Φράσης (ΓΟΦ):

```
s     →   np, vp.
vp    →   v, np.
 . . .
det   →   [ο] ; [τον].
n     →   [σκύλος] ; [ποντικό].
```

Παίρνουμε τη μικρή γραμματική που αναφέραμε πιο πάνω και την διαμορφώνουμε έτσι που να δέχεται ως *ορίσματα (arguments) λίστες συμβολοακολουθιών* τις οποίες επεξεργάζεται με το κατηγόρημα της Prolog **append/3**. Έτσι, σχεδιάζεται ο τεχνολογητής:

Τεχνολογητής (Ι) με χρήση της append/3.

```
s(L1)   :-   np(L2), vp(L3), append(L2, L3, L1).
np(L1) :-    det(L2), n(L3), append(L2, L3, L1).
vp(L1)  :-   v(L2), np(L3), append(L2, L3, L1).
vp(L1)  :-   v(L1).

det([ο]).
det([τον]).
n([σκύλος]).
n([ποντικό]).
v([νυστάζει]).
v([κυνηγά]).
```

Στο παράδειγμα αυτό, το κατηγόρημα append/3 δηλώνει ότι η λίστα L2 παρατίθεται μετά την L1 κι έτσι (οι δύο μαζί) συναποτελούν τη λίστα L3. Η χρήση του κατηγορήματος append/3 κάνει τον τεχνολογητή εξαιρετικά αργό γιατί ελέγχει όλα τα επιμέρους στοιχεία των λιστών, ένα προς ένα. Ταχύτερος είναι ο τεχνολογητής που χρησιμοποιεί την τεχνική των ***λιστών διαφοράς*** *(difference lists)*.

Τεχνολογητής (ΙΙ) με χρήση λιστών διαφοράς.

```
s(L1, L)        :-      np(L1, L2),  vp(L2, L).
np(L1, L)       :-      det(L1, L2),  n(L2, L).
vp(L1, L)       :-      v(L1, L2),  np(L2, L).
vp(L1, L)       :-      v(L1, L).

det([ο|L], L).
det([τον|L], L).
n([σκύλος|L], L).
n([ποντικό|L], L).
v([νυστάζει|L], L).
v([κυνηγά|L], L).
```

Αυτός ο τεχνολογητής απαντά σε ερωτήσεις της μορφής:

```
?- s ( [ο, σκύλος, κυνηγά, τον, ποντικό] ).
```

Αν αφαιρεθούν οι παρενθέσεις αυτός ο τεχνολογητής, αντιστοιχεί ακριβώς στη γραμματική από την οποία ξεκινήσαμε. Στην πραγματικότητα ο τεχνολογητής (ΙΙ) είναι μία ΓΟΦ στην αναπτυγμένη της μορφή, δηλαδή στη μορφή που έχει η ΓΟΦ μέσα στην υπολογιστή. Εμείς, εδώ, θα γράφουμε γραμματικές ΓΟΦ με τον συμβολισμό που δίνεται στην επόμενη παράγραφο.

Υπάρχουν διάφορες στρατηγικές τεχνολόγησης (parsing). Οι δύο πιο βασικές είναι η ***καθοδική τεχνολόγηση*** *(top-down parsing)* και η ***ανοδική τεχνολόγηση*** *(bottom-up parsing)*. Η γλώσσα Prolog έχει ενσωματωμένη την στρατηγική της *καθοδικής τεχνολόγησης* και αυτήν χρησιμοποιούν και οι ΓΟΦ. Για παράδειγμα, ο τεχνολογητής (ΙΙ) για να αναλύσει τη φράση:

```
ο σκύλος νυστάζει
```

1. Παίρνει τον (πρώτο) κανόνα για το s, τον αναπτύσσει, ψάχνοντας να βρει από ποια συστατικά αποτελείται το s. (Στο παράδειγμά μας np και vp.)
2. Μετά παίρνει τον κανόνα που αναπτύσσει το np. Αναπτύσσει το np και βρίσκει ότι αποτελείται από τα det και n.
3. Μετά παίρνει τον κανόνα που έχει ως αριστερό σύμβολο το det και προσπαθεί να ταυτοποιήσει τα τερματικά σύμβολα (οντότητες του λεξικού) που ταιριάζουν στο det.
4. Ακολούθως, συνεχίζει αναπτύσσοντας το vp, *(πρβλ. με βήμα 1, πιο πάνω)* κ.ο.κ.

Δηλαδή, ξεκινάει από την υπόθεση ότι έχει μία δομή συγκεκριμένη, στο παράδειγμά μας μία πρόταση (s), την χτίζει βήμα προς βήμα και επαληθεύει τις υποθέσεις του.

Αντιθέτως, ο τεχνολογητής που ακολουθεί την στρατηγική *ανοδικής τεχνολόγησης*, ξεκινά από κάτω, δηλαδή ταυτοποιεί τα τερματικά σύμβολα (λεξικές οντότητες) και χτίζει προοδευτικά τις ανώτερες δομές. Πιο αναλυτικά:

1. Ταυτοποιεί τα τερματικά σύμβολα (στο παράδειγμά μας 'ο', 'σκύλος', 'νυστάζει') ως det, n, και v, αντιστοίχως.
2. Μετά χτίζει τη δομή np που περιέχει τα det και n.
3. Ακολούθως, τη δομή v που περιέχει το v.
4. Τέλος, δημιουργεί την δομή s (πρόταση στο παράδειγμα)

Και οι δύο στρατηγικές (ανοδική και καθοδική) έχουν ανάγκη από την λειτουργία της ***οπισθοδρόμησης*** σε περίπτωση αποτυχίας κάποιου κανόνα. Οπισθοδρομούν για να πάρουν τον επόμενο στη σειρά κανόνα. Οι τεχνολογητές που 'οπισθοδρομούν' είναι **μη ντετερμινιστικοί**.

Άσκηση

Α *3.3.1*
*Ποιός από τους δύο τεχνολογητές μπορεί να χρησιμοποιηθεί στη **σύνθεση** (ή παραγωγή) **κειμένου** (text generation*);

3.4 Γραμματικές Ορισμένης Φράσης (**D**efinite **C**lause **G**rammars)

Η γλώσσα Prolog (οι περισσότερες εκδόσεις της) διαθέτει ενσωματωμένο φορμαλισμό για την συγγραφή ***Γραμματικών Ορισμένης Γράσης (ΓΟΦ)***. Ένα παράδειγμα σε ΓΟΦ είναι το ακόλουθο.

s	→	np, vp.
np	→	det, n.
vp	→	v, np.
vp	→	v.
det	→	[ο], [τον].
n	→	[σκύλος], [ποντικό].
v	→	[νυστάζει], [κυνηγά].

Χρησιμοποιούμε τις αγκύλες "**[**" και "**]**" για να συμβολίσουμε τα *άτομα* και την *κενή* συμβολοακολουθία και άγκιστρα "**{**" και "**}**" για να εισάγουμε εντολές της Prolog παρενθέτοντάς τες στο σώμα προτάσεων ΓΟΦ. Το σύμβολο '**;**' παριστάνει τη ***διάζευξη*** *(disjunction)*.

Παράδειγμα:

vp → v, [], {write('I am a vp with an empty NP-object string')}.

Ασκήσεις

Α *3.4.1*
Δουλεύει η πιό κάτω εντολή;

```
s → np, { write(L2) }, vp.
```

Α *3.4.2*
Πώς δουλεύει το πιό κάτω πρόγραμμα.

```
snoop(X,Y,X,Y).
s → np, snoop(A, B), {write(A), A=B}, vp.
```

3.5. Αριστερή αναδρομή

Όπως είπαμε, με τις Γραμματικές Ορισμένης Φράσης (ΓΟΦ) σχεδιάζουμε τεχνολογητές που διαφέρουν από τις δηλωτικές (περιγραφικές) γραμματικές. Ένα καλό παράδειγμα αυτής της διαφοράς προσφέρεται με το πρόβλημα της αριστερής αναδρομής. Προσέξτε τα παραδείγματα (2) και (3), που παραθέτουμε , καθώς και τους γραμματικούς κανόνες που τα ακολουθούν. Οι κανόνες αυτοί δεν είναι προβληματικοί αν θεωρηθούν ότι περιγράφουν την γλώσσα, δηλαδή υποδεικνύουν **ποιά** συστατικά πρέπει να βρίσκονται σε ποιά θέση για να είναι νόμιμη μία συμβολοακολουθία (σύμφωνη μ' αυτούς). Αν όμως θεωρηθούν διαδικασιακοί, δηλαδή ότι υποδεικνύουν **πώς** αναλύεται μια δομή, τότε αντιμετωπίζουμε ένα πρακτικό πρόβλημα γιατί αυτοί οι κανόνες οδηγούν τον υπολογιστή σε ατέρμονες βρόχους (δεν σταματά).

(2) *ο λύκος και ο σκύλος*
ΟΦ → ΟΦ και ΟΦ
ΟΦ → ΑΡΘ Ο

(3) *η πιπίλα του μωρού*
ΟΦ → ΟΦ ΟΦ
ΟΦ → ΑΡΘ Ο

Η κλασική λύση σ' αυτή την περίπτωση είναι η ακόλουθη. Δεν χρησιμοποιούμε ποτέ ως πρώτο σύμβολο του σώματος, ενός κανόνα, το σύμβολο της κεφαλής του ίδιου κανόνα. Τα παραδείγματα (4) και (5) δίνουν τις μη προβληματικές εκδόσεις του τεχνολογητή για τα (2) και (3) αντιστοίχως.

(4) ΟΦ → ΟΦΧ και ΟΦ
ΟΦ → ΟΦΧ
ΟΦΧ → ΑΡΘ Ο

(5) ΟΦ → ΟΦΧ ΟΦ
ΟΦ → ΟΦΧ
ΟΦΧ → ΑΡΘ Ο

<u>Ασκήσεις</u>

Α *3.5.1.*
Πώς δουλεύει η λύση που προτείναμε;

Α *3.5.2.*
Τι δομές (δέντρα) δίνει αυτή η λύση; Είναι επιθυμητά; Τί άλλα χαρακτηριστικά έχει αυτή η λύση που από δηλωτική άποψη είναι ανεπιθύμητα;

3.6. Ορίσματα Γραμματικής Ορισμένης Φράσης (ΓΟΦ)

Οι ΓΟΦ, έτσι όπως τις παρουσιάσαμε μέχρι τώρα, δεν έχουν αρκετή εκφραστική δύναμη για να κωδικοποιήσουν τα πάρα πολλά αλληλεπιδρώντα χαρακτηριστικά της φυσικής γλώσσας. Για παράδειγμα, η ΓΟΦ της Παραγράφου 3.4 θα δεχόταν ως σωστή την δομή

τον σκύλος νυστάζει

η οποία δεν υπακούει στον κανόνα συμφωνίας άρθρου-ουσιαστικού. Ευτυχώς η Prolog μας επιτρέπει να μετατρέψουμε τα μη τερματικά σύμβολα σε κατηγορήματα και να τους προσθέσουμε ορίσματα. Η εκφραστική δύναμη των ΓΟΦ αυξάνεται εντυπωσιακά με αυτόν τον τρόπο. Στην συνέχεια, όλες οι ΓΟΦ που θα χρησιμοποιήσουμε θα έχουν αυξηθεί με διάφορα, ανά περίπτωση, ορίσματα. Η κωδικοποίηση των φαινομένων συμφωνίας είναι τώρα απλή:

3.6.1 Συμφωνία άρθρου, επιθέτου, ονόματος στην ΟΦ

np(Gen,Num,Case) → det(Gen,Num,Case), adj(Gen,Num,Case), n(Gen,Num,Case).

Ο κανόνας αυτός λέει ότι τα τρία συστατικά της ΟΦ (det, adj, n) πρέπει να συμφωνούν κατά γένος, αριθμό και πτώση, δηλαδή να έχουν τα ίδια γένος, αριθμό και πτώση. Προφανώς, τα λήμματα για μια τέτοια γραμματική έχουν την μορφή:

```
det(masc,sg,nom)    → [o].
adj(masc,sg,nom)    → [koimismenos].
n(masc,sg,nom)      → [skilos].
```

Ασκήσεις

Α. *3.6.1.*
Γιατί ο επόμενος κανόνας δεν είναι σωστός
vp(Num) → v(Num), np

Α. *3.6.2.*
Ποιά γραμματική απορρίπτει προτάσεις σαν τις επόμενες;
**ο γάτος κυνηγά ο ποντικός*
**Ο Γιάννης ξεκίνησε η νύχτα.*

Μπορούμε να χρησιμοποιήσουμε τα ορίσματα για να χτίσουμε συντακτικά δέντρα.

```
s(s(NP, VP))     →   np(NP),  vp(VP).
np(np(DET,N))    →   det(DET), n(N).
vp(vp(V,NP))     →   v(V), np(NP).
n(n(skilos))     →   [skilos].
det(det(o))      →   [o].
v(v(nistazi))    →   [nistazi].
```

Ασκήσεις

Α *3.6.3.*
Ποιά είναι η βασική δομική σχέση την οποία κωδικοποιούμε με δέντρα ή με αγκύλες αλλά δεν μπορούμε να κωδικοποιήσουμε με γραμμικές δομές του τύπου:

αρχή πρότασης αρχή ΟΦ αρχή άρθρου ……

Α *3.6.4.*
(Προχωρημένη) Γράψτε ένα πρόγραμμα Prolog το οποίο παίρνει ΓΟΦ χωρίς ορίσματα και τις μετατρέπει αυτόματα σε κανόνες που χτίζουν συντακτικά δέντρα και γράφουν την μετάφραση σε ένα άλλο αρχείο.

3.6.2. Υποκατηγοριοποίηση

Το ακόλουθο παράδειγμα δεν είναι γραμματικό γιατί δεν υπακούει στους περιορισμούς (συντακτικής και σημασιολογικής) υποκατηγοριοποίησης του ρήματος *"φορώ"*.

**Ο Πέτρος φοράει*

Κάθε ρήμα (και γενικότερα, κάθε κατηγόρημα) καθορίζει πόσα συντακτικά ορίσματα πρέπει να έχει ώστε η πρόταση (ή γενικότερα η φράση) της οποίας αποτελεί την **κεφαλή** να είναι γραμματική. Επίσης, σύμφωνα με μια αρκετά διαδεδομένη άποψη, καθορίζει και άλλα σημασιολογικά και συντακτικά χαρακτηριστικά των ορισμάτων, πχ έμψυχο, πτώση γενική για το αντικείμενο. Παράβαλε με τη φράση: '*έτυχε της αγάπης των συμπολιτών του'*, κλπ. Χρησιμοποιώντας ορίσματα μπορούμε να εκφράσουμε τους περιορισμούς υποκατηγοριοποίησης με τις ΓΟΦ.

```
vp    →  v(1).
vp    →  v(2), np.
vp    →  v(3), np, np.

v(1)  →  [rohalizi].
v(2)  →  [vlepi].
v(3)  →  [dini].
```

Έτσι, χειριζόμαστε τα ρήματα με δυο τρόπους: και όλα μαζί γιατί ανήκουν στην ίδια *γραμματική κατηγορία* (ρήματα) και ξεχωριστά, ανάλογα με την *υποκατηγορία* στην οποία ανήκει το καθένα.

Άσκηση

Α *3.6.5.*
Τι δεν εξασφαλίζει η γραμματική του παραδείγματος της παραγράφου 3.6.2; Η επιδιόρθωση να λαμβάνει υπόψη της και προτάσεις όπως

Η πρόταση δεν έτυχε της απαιτούμενης προσοχής
ο Πέτρος έδωσε της γάτας φαγητό

Γέμισαν τα αμπάρια ποντικούς.

<u>Σημείωση</u>: Οι δύο τρόποι που υπάρχουν για να επιδιορθώσετε την πιο πάνω γραμματική εκφράζουν την υπαρκτή διελκυστίνδα ανάμεσα στην σύνταξη και στην λεξική πληροφορία. Ποιός τρόπος είναι πιο οικονομικός από υπολογιστική άποψη;

Προσθέτοντας ορίσματα στις ΓΟΦ μας δίνεται η δυνατότητα να δημιουργήσουμε ***υποκαθορισμένες*** *(underspecified)* δομές οι οποίες συγκεκριμενοποιούνται καθώς προχωρεί η τεχνολόγηση. Οι δομές αυτές λειτουργούν ως *περιορισμοί (constraints)* στην τεχνολόγηση και περιορίζουν την ***υπερπαραγωγή*** των ΓΟΦ που κωδικοποιούν την φραστική δομή μόνο. Για να κάνουμε καθαρό τί εννοούμε κοιτάξτε πάλι το απλό παράδειγμα της συμφωνίας κατά γένος, αριθμό και πτώση στην ΟΦ.

```
np(Gen,Num,Case) → det(Gen,Num,Case), adj(Gen,Num,Case), n(Gen,Num,Case).
```

Είναι οικονομικότερο να μην καθορίσουμε το γένος του τύπου "*του*". Αυτός θα καθοριστεί μόλις ο τεχνολογητής προχωρήσει και βρεί το επίθετο. Τότε ο κανόνας της ΟΦ θα αναγκάσει την μεταβλητή G να πάρει την τιμή masc, χάρη στην ***ενοποίηση*** της Prolog.

```
det(G,sg,gen)        →   [tou].
adj(masc,sg,gen)     →   [koimismenou].
n(masc,sg,gen)       →   [skilou].
```

Άσκηση

Α *3.6.6.*
Είναι δυνατόν να υπάρξουν υποκαθορισμένες δομές στο τέλος μιας επιτυχούς τεχνολόγησης; Φέρτε παραδείγματα από την φυσική γλώσσα.

Ένα πρώτο παράδειγμα: "Χαιρέτισε το φίλο του", ή "I eat fish"

3.7. Τεχνολόγηση δομών με εξαρτήσεις μεγάλης απόστασης *(long distance dependencies)*

Στίς ακόλουθες δομές της Αγγλικής, το συστατικό το οποίο συναντάται στην αρχή της πρότασης λογικά προέρχεται από εσωτερικές θέσεις. Χρησιμοποιούμε το σύμβολο ∪ για να δείξουμε αυτές τις θέσεις. Λέμε ότι έχουμε δομές με εξαρτήσεις μεγάλης απόστασης.

Bob said Bill claimed Doug announced Mary went to The States.
Who ∪ said Bill claimed Doug announced Mary went to The States.
Who did Bob say ∪claimed Doug announced Mary went to The States.
Who did Bob say Bill claimed ∪announced Mary went to The States.

Η ακόλουθη γραμματική μπορεί να τεχνολογήσει αυτές τις δομές. Ο πιο κάτω κανόνας (α) εισάγει την πληροφορία ότι υπάρχει ένα κενό αλφαριθμητικό το οποίο μεταφέρεται με τις λίστες IN, OUT μέχρις ότου βρεθεί η κατάλληλη θέση όπου η πληροφορία αυτή επαληθεύεται και χρησιμοποιείται ο κανόνας (β).

(α)
s(In,Out) → [who, did], np([who|In],Out1), vp(Out1,Out).
s(In,Out) → np(In,Out1),vp(Out1,Out).

(β)
np([who|Out],Out) → [].
np(X,X) → [bob];[bill];[doug];[mary];[the_states].

vp(X,X) → v.
vp(In,Out) → v,np(In,Out).
vp(In,Out) → v,s(In,Out).

v → [said];[claimed];[announced];[went].
v → [say].

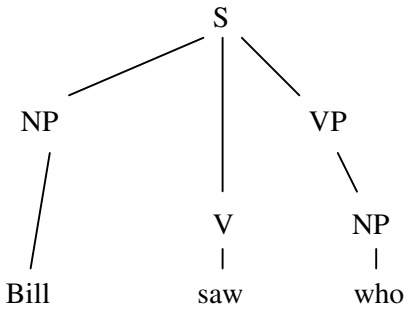
Η γραμματική δέχεται ερωτήσεις σαν την επόμενη:

s ([],[],[who,did,bob,say,bill,claimed,felix,barked],[]).

Ασκήσεις

Α *3.7.1.*
Η πιο πάνω γραμματική τεχνολογεί δομές σαν την επόμενη;
who did bob say felix bit

Α *3.7.1.*
Προσθέστε ορίσματα στην πιο πάνω γραμματική έτσι ώστε να γεννά δέντρα. Φροντίστε ώστε το κενό αλφαριθμητικό να εμφανίζεται στην λογική του θέση, πχ.



3.8 Λεξικοί κανόνες (lexical rules)

Με τους λεξικούς κανόνες θέτουμε όρια ανάμεσα στην σύνταξη (ο τομέας του "κανονικού") και το Λεξικό (ο τομέας του "ιδιόμορφου"). Αυτός ο διαχωρισμός υπήρξε σημαντικός για την θεωρία της γλωσσολογίας κατά τις δύο τελευταίες δεκαετίες.

Στις ΓΟΦ που έχουμε δεί μέχρι τώρα κάθε λέξη οριζόταν, όπως στο παράδειγμα που ακολουθεί, όχι σαν γεγονός που συντελείται έξω από τον χώρο της Σύνταξης αλλά ως συντακτικός κανόνας. Συνεπώς δεν υπήρχε σαφές όριο ανάμεσα στο "Λεξικό" και την "Σύνταξη".

```
n(singular, nominative, neutral, n(dog)) → [dog].
```

Με τον πιο κάτω συμβολισμό οι λέξεις δηλώνονται ως γεγονότα τα οποία τροφοδοτούν τους κατάλληλους κανόνες -- τους λεγόμενους Λεξικούς Κανόνες. Με τους Λεξικούς Κανόνες ένα γεγονός ανάγεται σε γραμματική πληροφορία.

```
n(Number, Case, Gender, n(Noun)) → [Noun], {is_noun(Noun,Number,Case,Gender)}.
is_noun(dog).
```

Μπορούμε τώρα να χτίσουμε πιο οικονομικές βάσεις λεξικών δεδομένων και να παράγουμε λέξεις μόνον όταν τις χρειαζόμαστε. Για παράδειγμα, δεν είναι πλέον ανάγκη να περιέχεται στο Λεξικό όλο το κλιτικό παράδειγμα ενός ονόματος. Οι διάφορες μορφές μπορούν να σχηματίζονται με κατάλληλους κανόνες όταν χρειάζονται για τις ανάγκες της τεχνολόγησης. Έτσι, ο πιο κάτω Λεξικός Κανόνας δίνει τον πληθυντικό των ομαλών ονομάτων της Αγγλικής.

```
noun(Plural, N) : -
        is_noun(N, Sg, _ , _ ),
        name(N,Sgnoun),
        append(Sgnoun,"s",Plnoun),
        name(Plural, Plnoun).
```

Ασκήσεις

Α *3.7.1.*
Δώστε τον λεξικό κανόνα που δίνει τα Ελληνικά σύνθετα μαυρόασπρος, ηλιοκαμμένος, λιγόλογος.

Α *3.7.2.*
Μπορεί να χρησιμοποιηθεί ο κανόνας του παραδείγματος για παραγωγή κειμένου; Γιατί; Δώστε τον κανόνα που μπορεί να χρησιμοποιηθεί και για σύνθεση και για τεχνολόγηση.

Α *3.7.2*
Δώστε τον κανόνα πού δίνει τον αόριστο (λύνω, έλυσα) των ομαλών ρημάτων της Νέας Ελληνικής. Πώς θα αντιμετωπίσετε την περίπτωση των τρώω, λέω;

Βιβλιογραφία

Το βασικό βιβλίο αναφοράς είναι ο Covington (1994). Για περαιτέρω προγράμματα δείτε τους Gazdar and Mellish (1989). Για μια εισαγωγή στις τυπικές γλώσσες χρησιμοποιείστε τους Partee et al. (1990).

Barbara Partee, Alice ter Meylen and Robert Wall. 1991. *Mathematical Methods in Linguistics.* Dordrecht: Kluwer Academic Publishers

Michael A. Covington. 1994. *Natural Language Processing for Prolog Programmers.* Prentice Hall: New Jersey, USA

Gerald Gazdar and Chris Mellish. 1989. *Natural Language Processing in Prolog: an introduction to computational linguistics.* Wokingham: Addison-Wesley.