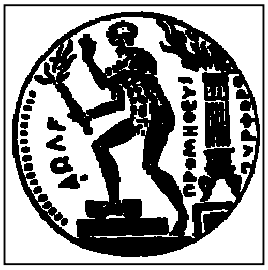

Εθνικό Μετσόβιο Πολυτεχνείο
Τμήμα Ηλεκτρολόγων Μηχανικών και
Μηχανικών Ηλεκτρονικών Υπολογιστών

Γλώσσες Προγραμματισμού 2
Άσκηση 2

Μάρτιος 2004

Τζάννες Αλέξανδρος

2. κανόνες φραστικής δομής (phrase structure rules)

α. Χρησιμοποιώντας τον φορμαλισμό των κανόνων φραστικής δομής (PS) σχεδιάστε μια γραμματική με την οποία να παράγονται οι αρχικοί χρόνοι των Ελληνικών ρημάτων με βάση τους τύπους του "γράφω" (γράφω, έγραφα, θα γράψω, έγραψα, έχω γράψει, θα έχω γράψει).

Υπόδειξη: *Να χρησιμοποιήσετε όσο το δυνατόν λιγότερη πληροφορία ώστε να εξασφαλίσετε όσο γίνεται μεγαλύτερη γενικότητα στη γραμματική σας. Για παράδειγμα, η δημιουργία του Παρατατικού θα μπορούσε να δημιουργείται με κανόνα του τύπου:*

Παρατατικός → ε Ενεστωτικό_Θέμα α

β. Το μοντέλο της γραμματικής σας αντιμετωπίζει τα ρήματα "τρώω", "λέω"; Αν όχι πώς μπορεί να τα αντιμετωπίσει; Υποδείξτε πιθανές αλλαγές.

Θα χρησιμοποιήσουμε για κάθε ρήμα ως μόνη πληροφορία το "ατελές θέμα" (πιο γνωστό ως "ενεστωτικό θέμα") και το "τέλειο θέμα" (πιο γνωστό ως "θέμα αορίστου"). Οι κανόνες για το σχηματισμό των αρχικών χρόνων με χρήση των δύο αυτών θεμάτων ακολουθούν σε φορμαλισμό DCG σε prolog:

```
enestwtas --> ateles_thema, [w].
paratatikos --> [e], ateles thema, [a].
mellontas_eksakolou8itikos --> [tha], [-], ateles_thema, [w].
mellontas_stigmiaios --> [tha], [-], teleio_thema, [w].
aoristos --> [e], teleio_thema, [a].
parakeimenos --> [exw], [-], teleio_thema, [ei].
ypersintelikos --> [eixa], [-], teleio_thema, [ei].
syntelesmenos_mellontas --> [tha], [-], [exw], [-], teleio_thema, [ei].
```

Αυτοί οι κανόνες δεν περιλαμβάνουν τους διάφορους φωνολογικούς κανόνες, οπότε δεν μπορούν να περιλάβουν όλες τις περιπτώσεις ρημάτων. Μια κατηγορία που δεν αντιμετωπίζεται έτσι είναι τα συνηρημένα ρήματα.

Για κάθε ρήμα πρέπει να προσθέσουμε κανόνες της ακόλουθης μορφής :

```
%Grafw
rima(grafw, graf, graps).
ateles_thema --> [graf].
teleio_thema --> [graps].

%Lynw
rima(lynw, lyn, lys).
ateles_thema --> [lyn].
teleio_thema --> [lys].

%Klebw
rima(klebw, kleb, kleps).
ateles_thema --> [kleb].
teleio_thema --> [kleps].
```

Σε κάθε τριάδα κανόνων, ο πρώτος συνδέει το ρήμα με τα δύο θέματά του και οι άλλοι δύο προσδιορίζουν τι είδους θέμα είναι το καθένα, σε φορμαλισμό DCG. Έχουμε τρία ρήματα για το παράδειγμά μας, τα "γράφω" "λύνω" και "κλέβω".

Ορίζουμε και κάποιες βοηθητικές συναρτήσεις για τα παραδείγματά μας:

```
printList(List) : Εκτυπώνει μια λίστα στην οθόνη
enestwtas(X): Εκτύπωση του ενεστώτα του ρήματος X
paratatikos(X): Εκτύπωση του παρατατικού του ρήματος X
mellontas eksakolou8itikos(X): Εκτύπωση του εξ. μέλλοντα του ρήματος X
mellontas_stigmiaios(X): Εκτύπωση του στ. μέλλοντα του ρήματος X
aoristos(X): Εκτύπωση του αορίστου του ρήματος X
parakeimenos(X): Εκτύπωση του παρακειμένου του ρήματος X
ypersintelikos(X): Εκτύπωση του υπερσυντέλικου του ρήματος X
syntelesmenos_mellontas(X): Εκτύπωση του συντ. μέλλοντα του ρήματος X

all(R): εκτύπωση των αρχικών χρόνων του ρήματος R

member(H,List): Απαντά στην ερώτηση αν το H είναι μέλος της λίστας List
```

Παράδειγμα εκτέλεσης:

```
?- all(grafw).
grafw
egrafa
tha grafw
tha grapsw
egrapsa
exw grapsei
eixa grapsei
tha exw grapsei

yes
?- all(klebw).
klebw
ekleba
tha klebw
tha klepsw
eklepsa
exw klepsei
eixa klepsei
tha exw klepsei

yes
?- all(lynw).
lynw
elyna
tha lynw
tha lysw
elysa
exw lysei
eixa lysei
tha exw lysei

yes
```

Ακολουθεί ο κώδικας :

```
enestwtas --> ateles_thema, [w].
paratatikos --> [e], ateles_thema, [a].
mellontas_eksakolou8itikos --> [tha], [-], ateles_thema, [w].
mellontas_stigmiaios --> [tha], [-], teleio_thema, [w].
aoristos --> [e], teleio_thema, [a].
parakeimenos --> [exw], [-], teleio_thema, [ei].
ypersintelikos --> [eixa], [-], teleio_thema, [ei].
syntelesmenos_mellontas --> [tha], [-], [exw], [-], teleio_thema, [ei].

%Grafw
rima(grafw, graf, graps).
ateles_thema --> [graf].
teleio_thema --> [graps].

%Lynw
rima(lynw, lyn, lys).
ateles_thema --> [lyn].
teleio_thema --> [lys].

%Klebw
rima(klebw, kleb, kleps).
ateles_thema --> [kleb].
teleio_thema --> [kleps].

printList([]):- nl.
printList([H|T]):- H='-', write(' '), printList(T), !.
printList([H|T]):- write(H), printList(T).

enestwtas(X):- enestwtas(X, []), printList(X).
paratatikos(X):- paratatikos(X, []), printList(X).
mellontas_eksakolou8itikos(X):- mellontas_eksakolou8itikos(X, []), printList(X).
mellontas_stigmiaios(X):- mellontas_stigmiaios(X, []), printList(X).
aoristos(X):- aoristos(X, []), printList(X).
parakeimenos(X):- parakeimenos(X, []), printList(X).
ypersintelikos(X):- ypersintelikos(X, []), printList(X).
syntelesmenos_mellontas(X):- syntelesmenos_mellontas(X, []), printList(X).

all(R):- rima(R, X, Y),
                  enestwtas(X1, []), member(X, X1), printList(X1),
                  paratatikos(X2, []), member(X, X2), printList(X2),
                  mellontas_eksakolou8itikos(X3, []), member(X, X3), printList(X3),
                  mellontas_stigmiaios(Y1, []), member(Y, Y1), printList(Y1),
                  aoristos(Y2, []), member(Y, Y2), printList(Y2),
                  parakeimenos(Y3, []), member(Y, Y3), printList(Y3),
                  ypersintelikos(Y4, []), member(Y, Y4), printList(Y4),
                  syntelesmenos_mellontas(Y5, []), member(Y, Y5), printList(Y5).

member(H,[H|T]).
member(X,[H|T]) :- member(X,T).
```

β. Το μοντέλο της γραμματικής μας δεν αντιμετωπίζει ρήματα όπως το τρώ(γ)ω και το λέ(γ)ω γιατί όπως είπαμε δεν αντιμετωπίζει γενικά την περίπτωση συναιρέσεων. Επειδή είναι αρκετά δύσκολο να οριστούν φωνολογικοί κανόνες το πιο απλό που μπορούμε να κάνουμε είναι να έχουμε ένα θέμα για κάθε χρόνο. Αυτό προσθέτει βέβαια πολύ πληροφορία στη βάση γνώσης του προγράμματος (τώρα για κάθε ρήμα έχουμε 9 αντί για 3 γραμμές που είχαμε προηγουμένως → δηλαδή τριπλασιάζεται το μέγεθος του αρχείου!) αλλά ασήμαντες αλλαγές σε όλα τα άλλα μέρη του προγράμματος όπως φαίνεται από τον κώδικα που ακολουθεί :

```
enestwtas --> thema_enest, [w].
paratatikos --> [e], thema_parat, [a].
mellontas_eksakolou8itikos --> [tha], [-], thema_mel_eks, [w].
mellontas_stigmiaios --> [tha], [-], thema_mel_stig, [w].
aoristos --> [e], thema_aor, [a].
parakeimenos --> [exw], [-], thema_parak, [ei].
ypersintelikos --> [eixa], [-], thema_ypers, [ei].
syntelesmenos_mellontas --> [tha], [-], [exw], [-], thema_synt_mel,
[ei].

%Grafw
rima(grafw, graf, graf, graf, graps, graps, graps, graps, graps).
thema_enest --> [graf].
thema_parat --> [graf].
thema_mel_eks --> [graf].
thema_mel_stig --> [graps].
thema_aor --> [graps].
thema_parak --> [graps].
thema_ypers --> [graps].
thema_synt_mel --> [graps].

%Lynw
rima(lynw, lyn, lyn, lyn, lys, lys, lys, lys, lys).
thema_enest --> [lyn].
thema_parat --> [lyn].
thema_mel_eks --> [lyn].
thema_mel_stig --> [lys].
thema_aor --> [lys].
thema_parak --> [lys].
thema_ypers --> [lys].
thema_synt_mel --> [lys].

%Klebw
rima(klebw, kleb, kleb, kleb, kleps, kleps, kleps, kleps, kleps).
thema_enest --> [kleb].
thema_parat --> [kleb].
thema_mel_eks --> [kleb].
thema_mel_stig --> [kleps].
thema_aor --> [kleps].
thema_parak --> [kleps].
thema_ypers --> [kleps].
thema_synt_mel --> [kleps].

%Trww
rima(trww, trw, trwg, trw, fa, fag, fa, fa, fa).
thema_enest --> [trw].
```

```
thema_parat --> [trwg].
thema_mel_eks --> [trw].
thema_mel_stig --> [fa].
thema_aor --> [fag].
thema_parak --> [fa].
thema_ypers --> [fa].
thema_synt_mel --> [fa].

%lew
rima(lew, le, leg, le, p, ip, p, p, p).
thema_enest --> [le].
thema_parat --> [leg].
thema_mel_eks --> [le].
thema_mel_stig --> [p].
thema_aor --> [ip].
thema_parak --> [p].
thema ypers --> [p].
thema_synt_mel --> [p].

%
printList([]):- nl.
printList([H|T]):- H='-', write(' '), printList(T), !.
printList([H|T]):- write(H), printList(T).

enestwtas(X):- enestwtas(X, []), printList(X).
paratatikos(X):- paratatikos(X, []), printList(X).
mellontas_eksakolou8itikos(X):- mellontas_eksakolou8itikos(X, []),
printList(X).
mellontas_stigmiaios(X):- mellontas_stigmiaios(X, []), printList(X).
aoristos(X):- aoristos(X, []), printList(X).
parakeimenos(X):- parakeimenos(X, []), printList(X).
ypersintelikos(X):- ypersintelikos(X, []), printList(X).
syntelesmenos_mellontas(X):- syntelesmenos_mellontas(X, []),
printList(X).

all(R):- rima(R, Xa, Xb, Xc, Ya, Yb, Yc, Yd, Ye),
                  enestwtas(X1, []), member(Xa, X1), printList(X1),
                  paratatikos(X2, []), member(Xb, X2), printList(X2),
                  mellontas_eksakolou8itikos(X3, []), member(Xc, X3),
printList(X3),
                  mellontas_stigmiaios(Y1, []), member(Ya, Y1),
printList(Y1),
                  aoristos(Y2, []), member(Yb, Y2), printList(Y2),
                  parakeimenos(Y3, []), member(Yc, Y3), printList(Y3),
                  ypersintelikos(Y4, []), member(Yd, Y4),
printList(Y4),
                  syntelesmenos_mellontas(Y5, []), member(Ye, Y5),
printList(Y5).

member(H,[H|T]).
member(X,[H|T]) :- member(X,T).
```

Παραδείγματα εκτέλεσης για τα δύο νέα ρήματα (τρώω και λέω) :

```
?- all(lew).
lew
elega
tha lew
tha pw
eipa
exw pei
eixa pei
tha exw pei

yes
?- all(trww).
trww
etrwga
tha trww
tha faw
efaga
exw faei
eixa faei
tha exw faei

yes
```