ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συστήματα και
Τεχνολογίες Γνώσης
Σημασιολογία για τη γλώσσα των νηπίων

Μηλαίου Ειρήνη
ΑΜ 03109089

Γενικά γνωρίζουμε ότι τα νήπια χρησιμοποιούν μία κάπως ελλειπτική γλώσσα για να εκφραστούν, γλώσσα την οποία βεβαίως όλοι κατανοούμε λίγο πολύ. Συνήθως παρατηρούμε ότι υπάρχει έλλειψη του ρήματος ή υπονοείται μέσω κάποιας λέξης η οποία δεν είναι ρήμα αλλά περισσότερο κάποιο ουσιαστικό. Σύνηθες φαινόμενο είναι επίσης να χρησιμοποιούν ονοματοποιημένα ρήματα και καθόλου άρθρα στις προτάσεις τους.
Σκοπός της άσκησης είναι να αποδίδεται συντακτική δομή καθώς και η σημασία για κάθε πρόταση ,μορφής όπως αναφέρθηκε παραπάνω, που θα δίνει ο χρήστης.

Για την άσκηση χρησιμοποιήθηκε η Prolog σε συνεργασία με το εργαλείο Gulp που αποτελεί επέκταση της Prolog για ενοποιητικές γραμματικές και βασίστηκε σε ένα παράδειγμα που δόθηκε από το υλικό του μαθήματος. Οι ενοποιητικές γραμματικές αποτελούν έναν από τους πιο διαδεδομένους φορμαλισμούς για την παράσταση γλωσσικής πληροφορίας. Είναι ένα σύνολο Συμφραστικώς Ανεξαρτήτων Κανόνων σχολιασμένων με δομές ιδιοτήτων όπου τα σχόλια εκφράζουν περιορισμούς τους οποίους πρέπει να ικανοποιούν οι συμβολοακολουθίες που γίνονται αποδεκτές ως γραμματικές προτάσεις. Κύρια χαρακτηριστικά είναι η Ενοποίηση και η Γενίκευση. Η ενοποίηση στην Prolog παίρνει δύο δομές και τις αντικαθιστά με την ενοποίηση τους. Η βάση της λύσης είναι οι διάφοροι κανόνες φραστικής δομής οι οποίοι χρησιμοποιούνται για να τεχνολογήσουμε ή να συνθέσουμε φράσεις. Εξετάζοντας αυτούς τους κανόνες με καθοδική τεχνολόγηση (χαρακτηριστικό της γλώσσας Prolog) τους αναπτύσσει ψάχνοντας τα συστατικά τους μέρη, στη συνέχεια αναπτύσσει τα συστατικά μέρη σε επιμέρους μέρη και όταν καταλήξει σε απλά κατηγορήματα προσπαθεί να τα ταυτοποιήσει στα υπάρχοντα τερματικά σύμβολα. Αν αποτύχει θα δοκιμάσει τον επόμενο κανόνα και θα ακολουθηθεί η ίδια διαδικασία. Κάθε κανόνας έχει ορισμένα χαρακτηριστικά (features) τα οποία χρησιμοποιούνται ,όπως είναι η πτώση (case) και το πρόσωπο (per) και πρέπει να συμφωνούν κατάλληλα. Έχουμε έξι διαφορετικούς κανόνες παραγωγής φράσεων (s) και np , vp όπου τα np αποτελούν τερματικά σύμβολα με την αντίστοιχη ερμηνεία ενώ τα vp αποτελούνται από v και np τα οποία είναι τερματικά. Να σημειώσουμε ότι το λεξικό μας είναι μικρό σε έκταση και οι δυνατοί/εφικτοί συνδυασμοί είναι περιορισμένοι.

Για την εκτέλεση του προγράμματος χρειάζεται να λάβουμε υπόψιν μας το αρχείο για την επέκταση Gulp (αρχείο gulp4swi.pl) στη συνέχεια “φορτώνουμε” το αρχείο με τον κώδικα και αρκεί πλέον να πατήσουμε την εντολή -try([String]).- και θα πάρουμε το επιθυμητό αποτέλεσμα αν μπορεί με βάση τους ορισμένους κανόνες να γίνει ανάλυση της πρότασης. Να σημειώσουμε ότι οποιαδήποτε πρόταση πρέπει να δίνεται σε greeklish (πχ mama,mpampa,nini). Ακολουθεί ο κώδικας της άσκησης και screen shots από εκτέλεση διαφορετικών σεναρίων.

Ο κώδικας της άσκησης :

```
% A grammar in DCG notation, with GULP feature structures

s(sem~(arg1~X..sen~Y)) --> np2(sem~X..case~nom..per~t),np(sem~Y).

 s(sem~(arg1~X..sen~Y)) --> np2(sem~X..case~acc..per~t),np1(sem~Y).

 s(sem~(arg1~X..sen~Y)) --> np2(sem~X..case~nom..per~f),np1(sem~Y).

s(sem~X) -->np1(sem~X).

s(sem~(pred~X..arg1~Y..arg2~Z)) --> np2(sem~Y..case~nom..per~t),
vp(sem~(pred~X..arg2~Z)).

vp(sem~(pred~X1..arg2~Y1)) --> vf(sem~X1), np2(sem~Y1..case~acc..per~f).

vp(sem~(pred~X1..arg2~Y1)) --> v(sem~X1), np2(sem~Y1..case~acc..per~t).

s(sem~(pred~X..arg1~Y)) -->np2(sem~Y..case~nom..per~t),v(sem~X).

np(sem~(pred~'kanei'..arg2~'kaka')) --> [kaka].

np(sem~(pred~'ekane'..arg2~'kaka') )--> [kaka].

np(sem~(pred~'kanei'..arg2~'tsisa')) --> [tsisa].

np(sem~(pred~'ekane'..arg2~'tsisa') )--> [tsisa].

np1(sem~(pred~'thelw'..arg2~'kaka')) --> [kaka].

np1(sem~(pred~'thelw'..arg2~'tsisa')) --> [tsisa].

np1(sem~(pred~'ekana'..arg2~'tsisa')) --> [tsisa].

np1(sem~(pred~'ekana'..arg2~'kaka')) --> [kaka].
```

```
np1(sem~(pred~'thelw'..arg2~ 'faghto')) --> [mam].

np1(sem~(pred~'thelw'..arg2~ 'bolta')) --> [ata].

np1(sem~(pred~'phga'..arg2~ 'bolta')) --> [ata].

np1(sem~'peinaw') --> [mam].

np1(sem~'efaga') --> [mam].

v(sem~(pred~'edeire'..arg2~'emena')) --> [nta].

v(sem~(pred~'dernei'..arg2~'emena')) --> [nta].

vf(sem~'edeire') --> [nta].

vf(sem~'dernei') --> [nta].

v(sem~(pred~'paei'..arg3~'bolta')) --> [ata].

v(sem~(pred~'phge'..arg3~'bolta')) --> [ata].

v(sem~(pred~'dinei'..arg3~'faghto')) --> [mam].

v(sem~(pred~'edose'..arg3~'faghto')) --> [mam].

np2(sem~'H mama'..case~nom..per~t) --> [mama].

np2(sem~'O mpampas'..case~nom..per~t) --> [mpampa].

np2(sem~'mama'..case~acc..per~t) --> [mama].

np2(sem~'mpampa'..case~acc..per~t) --> [mpampa].

np2(sem~'emena'..case~acc..per~f) --> [nini].

np2(sem~'egw'..case~nom..per~f) --> [nini].

% Procedure to parse a sentence and display its features

try(String) :-  writeln([String]),
                                    phrase(s(Features),String),
```

display_feature_structure(Features).

Εκτελώντας μερικά σενάρια με διαφορετικές φράσεις έχουμε τα παρακάτω αποτελέσματα :

```
?- try([mama,kaka]).
[mama,kaka]
sem ~ arg1 ~ H mama
      sen ~ pred ~ kanei
            arg2 ~ kaka
true ;
sem ~ arg1 ~ H mama
      sen ~ pred ~ ekane
            arg2 ~ kaka
true ;
sem ~ arg1 ~ mama
      sen ~ pred ~ thelw
            arg2 ~ kaka
true ;
sem ~ arg1 ~ mama
      sen ~ pred ~ ekana
            arg2 ~ kaka
true ;
false.
```

Για τη φράση [μαμά ,κακά] υπάρχουν τέσσερις πιθανοί συνδυασμοί και φαίνονται παραπάνω.

```
?- try([mpampa,nta]).
[mpampa,nta]
sem ~ arg1 ~ O mpampas
      pred ~ pred ~ edeire
            arg2 ~ emena
true ;
sem ~ arg1 ~ O mpampas
      pred ~ pred ~ dernei
            arg2 ~ emena
true ;
false.
```

Για τη φράση [μπαμπά, ντα] έχουμε τις δύο παραπάνω ερμηνείες.

```
?- try([nini,ata]).
[nini,ata]
sem ~ arg1 ~ egw
      sen ~ pred ~ thelw
            arg2 ~ bolta
true ;
sem ~ arg1 ~ egw
      sen ~ pred ~ phga
            arg2 ~ bolta
true ;
false.
```

Για τη φράση [νινί,άτα] έχουμε τις δύο παραπάνω περιπτώσεις ενώ αν δοκιμάσουμε τη φράση [μαμά,άτα] έχουμε τα παρακάτω :

```
?- try([mama,ata]).
[mama,ata]
sem ~ arg1 ~ mama
      sen ~ pred ~ thelw
            arg2 ~ bolta
true ;
sem ~ arg1 ~ mama
      sen ~ pred ~ phga
            arg2 ~ bolta
true ;
sem ~ arg1 ~ H mama
      pred ~ pred ~ paei
             arg3 ~ bolta
true ;
sem ~ arg1 ~ H mama
      pred ~ pred ~ phge
             arg3 ~ bolta
true ;
false.
```

Για τη φράση [τσίσα] έχουμε το παρακάτω ενώ πχ για τη φράση [μπαμπά, τσίσα] έχουμε τις παρακάτω ερμηνείες :

```
?- try([tsisa]).
[tsisa]
sem ~ pred ~ thelw
      arg2 ~ tsisa
true ;
sem ~ pred ~ ekana
      arg2 ~ tsisa
true ;
false.
```

```
?- try([mpampa,tsisa]).
[mpampa,tsisa]
sem ~ arg1 ~ O mpampas
      sen ~ pred ~ kanei
            arg2 ~ tsisa
true ;
sem ~ arg1 ~ O mpampas
      sen ~ pred ~ ekane
            arg2 ~ tsisa
true ;
sem ~ arg1 ~ mpampa
      sen ~ pred ~ thelw
            arg2 ~ tsisa
true ;
sem ~ arg1 ~ mpampa
      sen ~ pred ~ ekana
            arg2 ~ tsisa
true ;
false.
```

Για τη φράση [μαμ] έχουμε τρεις ερμηνείες ενώ για τη φράση [μπαμπά,ντα,νινί] έχουμε δυο διαφορετικές ερμηνείες.

```
?- try([mam]).
[mam]
sem ~ pred ~ thelw
      arg2 ~ faghto
true ;
sem ~ peinaw
true ;
sem ~ efaga
true ;
false.
```

```
?- try([mpampa,nta,nini]).
[mpampa,nta,nini]
sem ~ arg1 ~ O mpampas
      pred ~ edeire
      arg2 ~ emena
true ;
sem ~ arg1 ~ O mpampas
      pred ~ dernei
      arg2 ~ emena
true ;
false.
```

Αυτά είναι λίγα παραδείγματα, ενώ ο χρήστης μπορεί να δοκιμάσει διάφορες φράσεις που θα μπορούσε να χρησιμοποιήσει ένα νήπιο για να δει τη συντακτική του δομή και τη σημασία του. Υπάρχουν βέβαια και φράσεις που δεν μπορούν να αναλυθούν καθώς είναι εκτός κανόνων. (πχ ντα μπαμπά). Στα παραπάνω παραδείγματα φαίνεται και η θέση που έχει κάθε άτομο στην πρόταση , το ρήμα , το υποκείμενο και το αντικείμενο αλλά και ελλειπτικές προτάσεις όπου λείπει το υποκείμενο (εννοείται) ή το αντικείμενο. Παρακάτω είναι μερικά παραδείγματα σχετικά με τη παραγωγή προτάσεων σε “νηπιακό” λεξιλόγιο.

```
?- try(X).
_G231
sem ~ arg1 ~ H mama
      sen ~ pred ~ kanei
            arg2 ~ kaka
X = [mama, kaka] ;
sem ~ arg1 ~ H mama
      sen ~ pred ~ ekane
            arg2 ~ kaka
X = [mama, kaka] ;
sem ~ arg1 ~ H mama
      sen ~ pred ~ kanei
            arg2 ~ tsisa
X = [mama, tsisa] ;
sem ~ arg1 ~ H mama
      sen ~ pred ~ ekane
            arg2 ~ tsisa
X = [mama, tsisa] ;
sem ~ arg1 ~ O mpampas
      sen ~ pred ~ kanei
            arg2 ~ kaka
X = [mpampa, kaka] ;
sem ~ arg1 ~ O mpampas
      sen ~ pred ~ ekane
            arg2 ~ kaka
X = [mpampa, kaka] ;
sem ~ arg1 ~ O mpampas
      sen ~ pred ~ kanei
            arg2 ~ tsisa
X = [mpampa, tsisa] ;
```