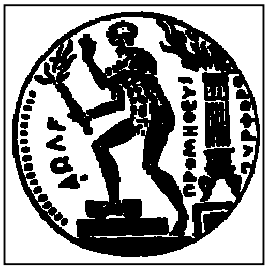

Εθνικό Μετσόβιο Πολυτεχνείο
Τμήμα Ηλεκτρολόγων Μηχανικών και
Μηχανικών Ηλεκτρονικών Υπολογιστών

Γλώσσες Προγραμματισμού 2
Άσκηση 6

Μάρτιος 2004

Τζάννες Αλέξανδρος

6. η γλώσσα των νηπίων - σημασιολογία

Ακούμε συχνά τα νήπια να μιλάνε με απλές προτάσεις όπου οι ονοματικές και ρηματικές φράσεις τους αποτελούνται από ονόματα και ονοματοποιημένα ρήματα, πριν μάθουν τα άρθρα (και τις συμφωνίες τους με το όνομα). Π.χ.:

μαμά κακά	**{η μαμά έκανε| κάνει κακά} | {θέλω κακά}**
μπαμπά ντα	**{ο μπαμπάς με έδειρε} | {ο μπαμπάς δέρνει}**
νινί άτα	**{θέλω βόλτα} | {πήγα βόλτα}**
τσίσα	**{θέλω (ενν. να κάνω) τσίσα} | {έκανα τσίσα}**
μαμ	**{θέλω να φάω} | {πεινάω} | {τρώω}**

Και σε επόμενη φάση να προσθέτουν στο συντακτικό τους ένα αντικείμενο. Π.χ.:

μπαμπά ντα νινί **{ο μπαμπάς έδειρε το νινί (ενν. εμένα)}**

Μπορούμε να τεχνολογήσουμε τέτοιες προτάσεις και να δώσουμε τη σημασία ή τις σημασίες των προτάσεων των νηπίων;

***Υπόδειξη**: Διαλέξτε εσείς φορμαλισμό (και υπολογιστικό εργαλείο).*

Αρχικά να σημειώσω ότι τα πρώτα στάδια της εργασίας ήταν μια συνεργασία μου με τον Μιχάλη Παπακυριάκου, ενώ στην συνέχεια συνεχίσαμε την ανάπτυξη ξεχωριστά καθότι ο τρόπος που σκεφτόταν ο καθένας ήταν αρκετά διαφορετικός (χωρίς κάποιος να είναι λάθος ή χειρότερος) με αποτέλεσμα να καθυστερεί η πρόοδος της εργασίας. Αυτό ήταν ένα πολύ ενδιαφέρον μάθημα για το πόσο ευέλικτος μπορεί να είναι ο τρόπος σκέψης σε θέματα στα οποία η γνώση δεν έχει τυποποιηθεί πλήρως.

Το υπολογιστικό εργαλείο που έχω διαλέξει είναι η prolog με την οποία έχω μεγαλύτερη εξοικείωση και χρησιμοποιώ τον φορμαλισμό DCG. Το μη τερματικό s είναι αυτό που κατασκευάζει στις προτάσεις (sentence) και εκτός από την πρόταση στη γλώσσα των νηπίων παίρνει και δύο ακόμα λίστες (λίστες διαφορών), Out και In, οι οποίες περιέχουν τη σημασία της πρότασης του νηπίου και την αρχική σημασία αντίστοιχα. Σε κανονική χρήση η s καλείται με την In κενή και την Out μια μεταβλητή αφού αυτό είναι το ζητούμενο.

Τα ρήματα (verb_nini) που είναι γνωστά στο πρόγραμμα είναι τα “μαμ”, “ντα”, “κακά”, “άτα”, “τσίσα” ενώ τα ουσιαστικά (noun_nini) που είναι γνωστά είναι τα “μαμά”, “μπαμπά”, “νινί”.

```
%nouns
noun_nini(mama) --> [mama].
noun_nini(mpampa) --> [mpampa].
noun_nini(nini) --> [nini].

%verbs
verb_nini(mam) --> [mam].
verb_nini(nta) --> [nta].
```

```
verb_nini(nta,2) --> [nta].
verb_nini(kaka) --> [kaka].
verb_nini(ata) --> [ata].
verb_nini(tsisa) --> [tsisa].
```

Παρατηρούμε ότι το "ντα" περιλαμβάνεται δύο φορές, την δεύτερη με ένα όρισμα '2'. Αυτό σημαίνει ότι είναι η εκδοχή του που λαμβάνει αντικείμενο (δέρνω κάποιον). Βλέπουμε επίσης ότι περιέχεται σαν όρισμα η ίδια η λέξη (π.χ. `verb_nini(`**`mam`**`) --> [mam].`). Αυτό γίνεται για τη μεταφορά της πληροφορίας στο παραπάνω επίπεδο του δένδρου για να γίνει εκεί η επεξεργασία.

Η προσέγγιση που δώσαμε στην μηχανιστική ερμηνεία της γλώσσας των νηπίων ήταν να δίνουμε διάφορες ερμηνείες (meanings) στα τερματικά της γλώσσας. Οι δύο κύριες κατηγορίες ερμηνειών ήταν η ερμηνεία σε ουσιαστικό (noun_meaning), και η ερμηνεία σε ρήμα (verb_meaning).

Τέλος οι τύποι των προτάσεων που υποστηρίζει το πρόγραμμα είναι οι εξής :
s → verb
s → noun verb
s → noun verb noun
s → verb noun
s → noun s

Παρατηρούμε ότι ο τελευταίος κανόνας περιέχει δεξιά αναδρομή. Χρησιμοποιείται ως κλητική προσφώνηση, δηλαδή "noun ! s". Αυτό θα γίνει πιο κατανοητό με ένα παράδειγμα :

```
?- s(Out,[],[mama,kaka],[]).

Out = [mama, '!', kanw, kaka] ;
Out = [mama, '!', ekana, kaka] ;
Out = [mama, '!', thelo, na, kanw, kaka] ;
Out = [i_mama, kanei, kaka] ;
Out = [i_mama, ekane, kaka] ;
Out = [i_mama, thelei_na_kanei, kaka] ;
no
```

Βλέπουμε ότι παρέχονται 6 ερμηνείες, οι τρεις από τον κανόνα s → noun s και οι άλλες τρεις από τον κανόνα s → noun s. Ας δούμε ακόμα μερικά παραδείγματα :

```
?- s(Out,[],[ata],[]).

Out = [kanw, bolta] ;
Out = [ekana, bolta] ;
Out = [thelo, na, kanw, bolta] ;
no
```

```
?- s(Out,[],[nta],[]).

Out = [kanw, kati_kako] ;
Out = [ekana, kati_kako] ;
Out = [thelo, na, kanw, kati_kako] ;
Out = [dernw] ;
```

```
Out = [edeira] ;
no
```

```
?- s(Out,[],[nini,tsisa],[]).

Out = [kanw, tsisa] ;
Out = [ekana, tsisa] ;
Out = [thelo, na, kanw, tsisa] ;
Out = [to_mwro, kanei, tsisa] ;
Out = [to_mwro, ekane, tsisa] ;
Out = [to_mwro, thelei_na_kanei, tsisa] ;
no
```

```
?- s(Out,[],[mama,nta],[]).

Out = [mama, '!', kanw, kati_kako] ;
Out = [mama, '!', ekana, kati_kako] ;
Out = [mama, '!', thelo, na, kanw, kati_kako] ;
Out = [mama, '!', dernw] ;
Out = [mama, '!', edeira] ;
Out = [i_mama, kanei, kati_kako] ;
Out = [i_mama, ekane, kati_kako] ;
Out = [i_mama, thelei_na_kanei, kati_kako] ;
Out = [i_mama, me, dernei] ;
Out = [i_mama, me, edeire] ;
Out = [i_mama, dernei] ;
Out = [i_mama, edeire] ;
no
```

```
?- s(Out,[],[mama,nta,mpampa],[]).

Out = [mama, '!', dernw, ton_mpampa] ;
Out = [mama, '!', edeira, ton_mpampa] ;
Out = [i_mama, dernei, ton_mpampa] ;
Out = [i_mama, edeire, ton_mpampa] ;
no
```

```
?- s(Out,[],[nta,mpampa],[]).

Out = [dernw, ton_mpampa] ;
Out = [edeira, ton_mpampa] ;
no
```

```
?- s(Out,[],[mama, mpampa, ata],[]).

Out = [mama, '!', mpampa, '!', kanw, bolta] ;
Out = [mama, '!', mpampa, '!', ekana, bolta] ;
Out = [mama, '!', mpampa, '!', thelo, na, kanw, bolta] ;
Out = [mama, '!', o_mpampas, kanei, bolta] ;
Out = [mama, '!', o_mpampas, ekane, bolta] ;
Out = [mama, '!', o_mpampas, thelei_na_kanei, bolta] ;
no
```

```
?- s(Out,[],[mama, mpampa, mam],[]).

Out = [mama, '!', mpampa, '!', trww] ;
```

```
Out = [mama, '!', mpampa, '!', peinaw] ;
Out = [mama, '!', mpampa, '!', thelw_na_faw] ;
Out = [mama, '!', o_mpampas, me, trwei] ;
Out = [mama, '!', o_mpampas, me, peinaei] ;
Out = [mama, '!', o_mpampas, me, thelei_na_faei] ;
no
```

Ακολουθεί ο κώδικας του προγράμματος:

```
%nouns
noun_nini(mama) --> [mama].
noun_nini(mpampa) --> [mpampa].
noun_nini(nini) --> [nini].
%verbs
verb_nini(mam) --> [mam].
verb_nini(nta) --> [nta].
verb_nini(nta,2) --> [nta].
verb_nini(kaka) --> [kaka].
verb_nini(ata) --> [ata].
verb_nini(tsisa) --> [tsisa].

%noun meanings
noun_meaning(ata, bolta).
noun_meaning(kaka, kaka).
noun_meaning(tsisa, tsisa).
noun_meaning(nta, kati_kako).

noun_meaning(nini, nom, to_mwro).
noun_meaning(nini, acc, to_mwro).

noun_meaning(mama, nom, i_mama).
noun_meaning(mama, acc, ti_mama).

noun_meaning(mpampa, nom, o_mpampas).
noun_meaning(mpampa, acc, ton_mpampa).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% verb meanings
verb_meaning(mam, s1, trww).   %s1 : singular, 1st person
verb_meaning(mam, s1, peinaw).
verb_meaning(mam, s1, thelw_na_faw).
verb_meaning(mam, s3, trwei).   %s3 : singular, 1st person
verb_meaning(mam, s3, peinaei).
verb_meaning(mam, s3, thelei_na_faei).

verb_meaning(nta, s1, dernw).
verb_meaning(nta, s1, edeira).
verb_meaning(nta, s3, dernei).
verb_meaning(nta, s3, edeire).


debug(X).
%debug(X):-write(X),nl.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%--- s --> verb

s(Out, In) --> verb_nini(X),{ debug(1-1), noun_meaning(X, Y),
append(In, [kanw, Y], Out)}.
s(Out, In) --> verb_nini(X),{ debug(1-2), noun_meaning(X, Y),
append(In, [ekana, Y], Out)}.
s(Out, In) --> verb_nini(X),{ debug(1-3), noun_meaning(X, Y),
append(In, [thelo, na, kanw, Y], Out)}.

s(Out, In) --> verb_nini(X),{ debug(1-4), verb_meaning(X, s1, Y),
append(In, [Y], Out)}.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--- s --> noun s (recursion)

s(Out, In) --> noun_nini(X), {not(member(nini,[X])),
               append(In,[X,!],In1)}, s(Out, In1). %klitiki

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--- s --> noun verb
s(Out, In) --> noun_nini(nini), verb_nini(V), { s(Out, In, [V], []) }.

s(Out, In) --> noun nini(N), verb nini(V),
            { debug(2-2-1), noun_meaning(N,nom,Nm), noun_meaning(V,Vm),
              append(In, [Nm, kanei, Vm], Out)}.
s(Out, In) --> noun_nini(N), verb_nini(V),
            { debug(2-2-2), noun_meaning(N,nom,Nm), noun_meaning(V,Vm),
              append(In, [Nm, ekane, Vm], Out)}.
s(Out, In) --> noun_nini(N), verb_nini(V),
            { debug(2-2-3), noun_meaning(N,nom,Nm), noun_meaning(V,Vm),
              append(In, [Nm, thelei_na_kanei, Vm], Out)}.

s(Out, In) --> noun_nini(N), verb_nini(V,2),
            { debug(2-2-4), noun_meaning(N,nom,Nm), verb_meaning(V, s3,
              Vm),append(In, [Nm, me, Vm], Out)}.
s(Out, In) --> noun_nini(N), verb_nini(V),
            { debug(2-2-5), noun_meaning(N,nom,Nm), verb_meaning(V, s3,
              Vm),append(In, [Nm, Vm], Out)}.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--- s --> noun verb noun
s(Out, In) --> noun_nini(N1), verb_nini(V,2), noun_nini(N2),
            { debug(3-1), noun_meaning(N1, nom, N1m),
              noun meaning(N2, acc, N2m), verb meaning(V, s3, Vm),
              append(In, [N1m, Vm, N2m], Out)}.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--- s --> verb noun
s(Out, In) --> verb nini(V,2), noun nini(N2),
            { debug(4-1), noun_meaning(N2, acc, N2m),
              verb_meaning(V, s1, Vm), append(In, [Vm, N2m], Out)}.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
member(H,[H|T]).
member(X,[H|T]) :- member(X,T).


append([],X,X).
append([H|T1],X,[H|T2]) :- append(T1,X,T2).
```