«Επεξεργασία Φυσικής Γλώσσας»

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Απαλλακτική εργασία

Ιουνίου – Σεπτεμβρίου 2021

Του Μαθητή Γεώργιου Σαμαρά με Αριθμό Μητρώου Π18134

Επιβλέπων Καθηγητής: Θεμιστοκλής Παναγιωτόπουλος

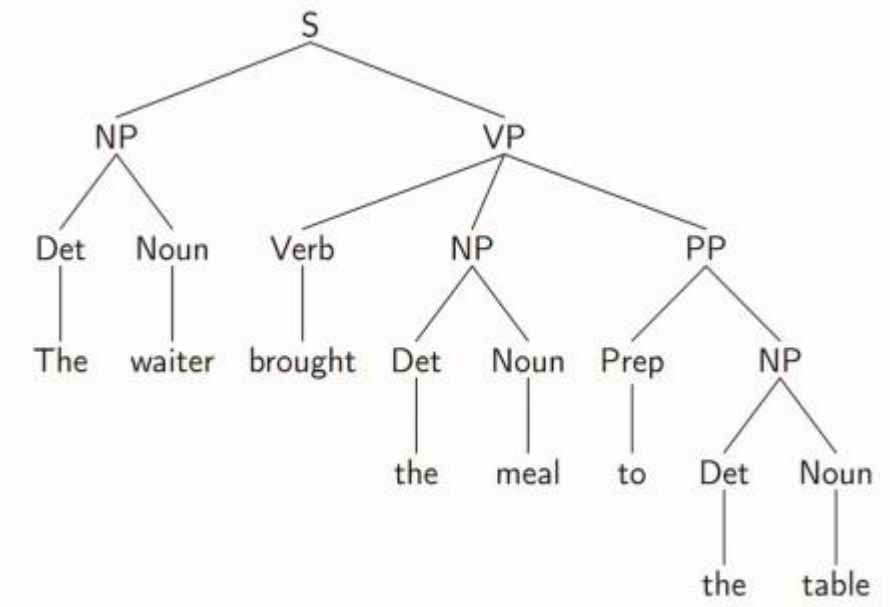
Περιεχόμενα

Θεμα 1ο

Α Ερώτημα

Εκφώνηση

Με ποιά γραμματική σε μορφή DCG μπορούμε να αναγνωρίσουμε την πρόταση : [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα;



(The waiter brought the meal to the table)

Κώδικας Prolog

```
/* Απαλακτική Εργασία Ιουνίου 2021 */

/* Θέμα 1ο */

/* α) Με ποιά γραμματική σε μορφή DCG μπορούμε να αναγνωρίσουμε την */

/* πρόταση : [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα;
*/



:- discontiguous noun/2.


s --> np, vp.

np --> det,noun.

pp --> prep,np.

vp --> verb,np,pp.

det-->[the].

noun-->[waiter].

noun-->[meal].

verb-->[brought].

prep-->[to].

noun-->[table].


/* Ερώτηση 1α */

/* s([the, waiter, brought, the, meal, to, the, table],[]). */
```

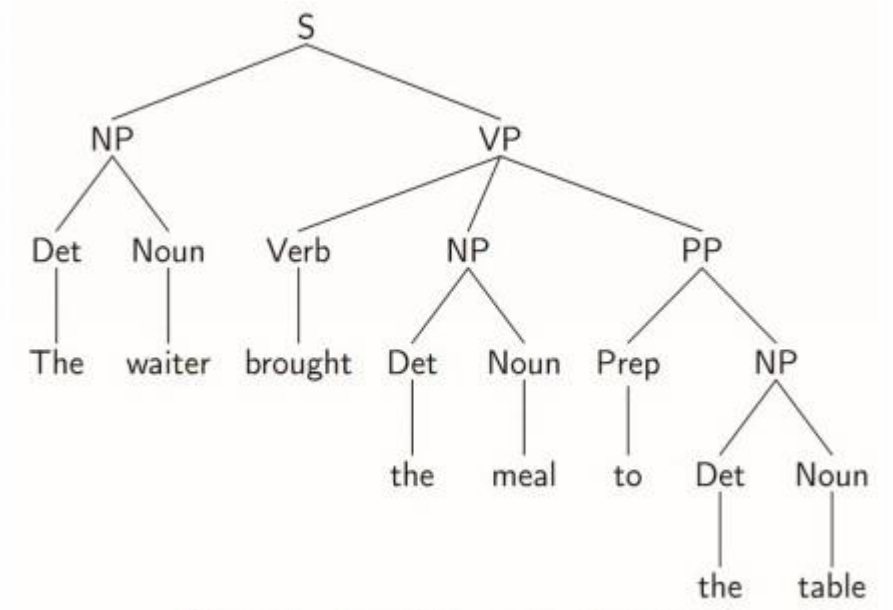
Αποτέλεσμα

```
?- s([the, waiter, brought, the, meal, to, the, table],[]).
true.
```

Β Ερώτημα

Εκφώνηση

Με ποιά γραμματική σε μορφή DCG μπορούμε να παράγουμε σε μορφή functor το συντακτικό δένδρο για την αναγνώριση της πρότασης : [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα;



(The waiter brought the meal to the table)

Κώδικας Prolog

```
/* β) Με ποιά γραμματική σε μορφή DCG μπορούμε να παράγουμε σε μορφή */
/* functor το συντακτικό δένδρο για την αναγνώριση της πρότασης : */
/* [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα; */

:- discontiguous noun/3.

s(s(NP,VP))-->np(NP),vp(VP).
np(np(D,N))-->det(D),noun(N).
pp(pp(PREP,NP))-->prep(PREP),np(NP).
vp(vp(V,NP,PP))-->verb(V),np(NP),pp(PP).
det(det(the))-->[the].
prep(prep(to))-->[to].
noun(noun(waiter))-->[waiter].
verb(verb(brought))-->[brought].
noun(noun(meal))-->[meal].
noun(noun(table))-->[table].

/* Ερώτηση 1β */
/* s(S,[the, waiter, brought, the, meal, to, the, table],[]). */
```

Αποτέλεσμα

```
?- s(S,[the, waiter, brought, the, meal, to, the, table],[]).
S = s(np(det(the),noun(waiter)),vp(verb(brought),np(det(the),noun(meal)),pp(prep(to),np(det(the),noun(table))))).
```

Θεμα 2ο

Εκφώνηση

Το παρακάτω πρόγραμμα αναγνωρίζει και υπολογίζει αριθμητικές εκφράσεις όπως αναλύθηκε στο θεωρητικό μέρος. Να αναπτυχθεί ένα αντίστοιχο πρόγραμμα όπου οι αριθμοί είναι διαδικοί και οι αριθμητικές εκφράσεις είναι αντίστοιχα αριθμητικές εκφράσεις διαδικών αριθμών.

```
expression(Value) --> number(Value).
expression(Value) --> number(X), [+], expression(V), {Value is X+V}.
expression(Value) --> number(X), [-], expression(V), {Value is X-V}.
expression(Value) --> number(X), [*], expression(V), {Value is X*V}.
expression(Value) --> number(X), [/], expression(V), {V/=0, Value is X/V}.
expression(Value) --> left_parenthesis, expression(Value), right_parenthesis.
left_parenthesis --> ['('].
right_parenthesis --> [')'].
number(X) --> digit(X).
number(Value) --> digit(X), number(Y),{numberofdigits(Y,N), Value is X*10^N+Y}.
digit(0) --> [0].
digit(1) --> [1].
digit(2) --> [2].
digit(3) --> [3].
digit(4) --> [4].
digit(5) --> [5].
digit(6) --> [6].
digit(7) --> [7].
digit(8) --> [8].
digit(9) --> [9].
numberofdigits(Y,1) :- Z is Y/10, Z<1.
numberofdigits(Y,N) :-
 Z is (Y - mod(Y,10))/10,
 numberofdigits(Z,N1),
 N is N1+1.
```

Κώδικας Prolog

```
/* Απαλακτική Εργασία Ιουνίου 2021 */
/* Θέμα 2ο */
/* de doulevei me olous tous arithmous */

expression(Value) --> number(Value).
expression(Value) --> number(X), [+], expression(V), {Value is X+V}.
expression(Value) --> number(X), [-], expression(V),{Value is X-V}.
expression(Value) --> number(X), [*], expression(V), {Value is X*V}.
expression(Value) --> number(X), [/], expression(V),{V\=0, Value is X/V}.
expression(Value) --> left_parenthesis, expression(Value), right_parenthesis.
left_parenthesis --> ['('].
right_parenthesis --> [')'].
number(X) --> digit(X).
number(Value) --> digit(X), number(Y), {numberofdigits(Y,N), Value is X*2^N+Y}.
digit(0) --> [0].
digit(1) --> [1].
numberofdigits(Y,1) :- Z is Y/2, Z<1, !.
numberofdigits(Y,N) :-
Z is (Y - mod(Y,2))/2,
numberofdigits(Z,N1),
N is N1+1, !.

/* Ερώτηση 2 */
% expression(V,[1,1,1,+,1,0],[]).
% expression(V,[1,1,-,1,1,0],[]).
% expression(V,[1,1,1,1,*,1,1],[]).
% expression(V,[1,1,1,1,/,1,1],[]).
```

Αποτέλεσμα

```
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- expression(V,[1,1,1,+,1,0],[]).
V = 9 .

?- expression(V,[1,1,-,1,1,0],[]).
V = -3 .

?- expression(V,[1,1,1,1,*,1,1],[]).
V = 45 .

?- expression(V,[1,1,1,1,/,1,1],[]).
V = 5
```

Θεμα 3ο

Εκφώνηση

Χρησιμοποιείστε το σύνολο του κώδικα που σας δόθηκε σε Prolog για την «κατανόηση» μιας μικρής ιστορίας. Πρέπει να παράγετε τα περιεχόμενα της βάσης γνώσης και να μπορείτε να εισάγετε νέες πληροφορίες από το πληκτρολόγιο, να κάνετε ερωτήσεις στην βάση γνώσης, κ.λπ., παρόμοιες με αυτές της πρότυπης λύσης.

Ιστορία

'timmy comes quickly to the house. timmy finds tim. tim is confused. tim needs help. timmy helps tim. timmy gets confused. timmy is sad. tim hugs timmy. the problem is difficult. timmy tries again. timmy is angry. timmy solves the problem. timothy sees the problem. timothy laughs loudly at the kids.'.

Εκφώνηση

Για το θέμα αυτό χρησιμοποιήθηκε το πρόγραμμα σε prolog από την εκφώνηση της εργασίας, το οποίο τροποποιήθηκε για τις ανάγκες αυτού του ερωτήματος καθώς και διορθωθήκαν μερικά λάθη που ανακαλυφθήκαν.

Ο Λεκτικός Αναλυτής χρησιμοποιήθηκε ως είχε μιας και δεν χρειάζονταν αλλαγές για την λειτουργία του με μια άλλη συλλογή από προτάσεις.

Ο Συντακτικός Αναλυτής όμως όπως και ο Σημασιολογικός επεξεργάστηκαν καταλλήλως ώστε να μπορούν να λειτουργήσουν με τη δικιά μου ιστορία. Πιο συγκεκριμένα προστέθηκαν και τροποποιήθηκαν συντακτικοί και σημασιολογικοί κανόνες και τα λεξιλόγια των δυο αναλυτών.

Όσων αφορά την Βάση Γνώσης και τις ερωτοαπαντησεις σε αυτήν τροποποιήθηκαν κάποιοι από τους κανόνες που δόθηκαν στην πρότυπη λύση.

Παρακάτω θα παραθέσω τον κώδικα από τα κομμάτια που άλλαξα και όχι όλο τον κώδικα του προγράμματος επειδή είναι αρκετά μεγάλος και θα πάρει πολλές σελίδες.

Συντακτικός Αναλυτής

Κώδικας Prolog

Γραμματική

```
/* Sentence (snt) */
 snt(s(NP,VP)) --> np(NP), vp(VP).
/* Noun Phrase (np) */
np(np(N)) --> pn(N).
 np(np(D,N)) --> det(D), n(N).
 np(np(N)) --> n(N).
 np(np(P,D,N)) --> pr(P), det(D), n(N).
/* Verb Phrase (vb) */
 % Intransitive verbs :
 vp(vp(V)) --> iv(V).
 vp(vp(V,ADV)) --> iv(V), adv(ADV).
 vp(vp(V,D,NP)) --> iv(V), adv(ADV), np(NP).
 % Auxiliary verbs
 vp(vp(AV,A)) --> av(AV), adj(A).
 % Transitive verbs :
 vp(vp(TV, PN, NP)) --> v(TV), np(PN), np(NP).
% verbs
 vp(vp(V,NP)) --> v(V), np(NP).
 vp(vp(V,D,NP)) --> v(V), det(D), np(NP).
 vp(vp(V,D,NP,NP)) --> v(V), det(D), np(NP), np(NP).
```

Κομμάτι του Λεξιλογίου

```
/*=========================================================*/
 /* VOCABULARY OF EXAMPLE */
 /*=========================================================*/
 /* Intransitive Verbs (iv) */
 % needed for example :
 iv(iv(comes))-->[comes].
 iv(iv(come))-->[come].
 iv(iv(coming))-->[coming].
 iv(iv(gets))-->[gets].
 iv(iv(get))-->[get].
 iv(iv(getting))-->[getting].
 iv(iv(laughs))-->[laughs].
 iv(iv(laugh))-->[laugh].
 iv(iv(laughing))-->[laughing].
 iv(iv(tries))-->[tries].
 iv(v(try))-->[try].
/* Auxiliary Verbs (av) */
% needed for example :
 av(av(is))-->[is].
 % extension of vocabulary :
 av(av(does))-->[does].
 av(av(are))-->[are].
 av(av(do))-->[do].
 /* Verbs (v) */
 % needed for example :
 v(v(comes))-->[comes].
 v(v(come))-->[come].
 v(v(finds))-->[finds].
 v(v(find))-->[find].
```

```
v(v(helps))-->[helps].
 v(v(help))-->[help].
 v(v(hugs))-->[hugs].
 v(v(hug))-->[hug].
 v(v(tries))-->[tries].
 v(v(try))-->[try].
 v(v(solves))-->[solves].
 v(v(solve))-->[solve].
 v(v(sees))-->[sees].
 v(v(see))-->[see].
 v(v(needs))-->[needs].
 v(v(need))-->[need].
 v(v(gets))-->[gets].
 v(v(get))-->[get]. n(n(apple))-->[apple].
n(n(apples))-->[apples].
n(n(park))-->[park].
n(n(parks))-->[parks].
n(n(floor))-->[floor].
/* Proper Nouns (pn) */
pn(pn(john))-->[john].
pn(pn(george))-->[george].
pn(pn(theo))-->[theo].
/* Determiner (det) */
det(det(the)) -->[the].
det(det(a)) -->[a].
det(det(an)) -->[an].
/* preposition (p) */
pr(pr(at))-->[at].
pr(pr(towards))-->[towards].
```

Αποτέλεσμα

Ερώτηση: understand('Ergasia3.txt').

(Ίσως εμφανιστούν προβλήματα με τα παραπάνω μονά αυτάκια σε διαφορετικό υπολογιστή, δεν γνωρίζω γιατί συμβαίνει.)

```
?- understand('Ergasia3.txt').
[timmy,comes,quickly,to,the,house]
[timmy,finds,tim]
[tim,is,confused]
[tim,needs,help]
[timmy,helps,tim]
[timmy,gets,confused]
[timmy,is,sad]
[tim,hugs,timmy]
[the,problem,is,difficult]
[timmy,tries,again]
[timmy,is,angry]
[timmy,solves,the,problem]
[timothy,sees,the,problem]
[timothy,laughs,loudly,at,the,kids]

the lexical analysis completed!

s(np(pn(timmy)),vp(iv(comes),_13604,np(pr(to),det(the),n(house))))
s(np(pn(timmy)),vp(v(finds),np(pn(tim))))
s(np(pn(tim)),vp(av(is),adj(confused)))
s(np(pn(tim)),vp(v(needs),np(n(help))))
s(np(pn(timmy)),vp(v(helps),np(pn(tim))))
s(np(pn(timmy)),vp(iv(gets),adv(confused)))
s(np(pn(timmy)),vp(av(is),adj(sad)))
s(np(pn(tim)),vp(v(hugs),np(pn(timmy))))
s(np(det(the),n(problem)),vp(av(is),adj(difficult)))
s(np(pn(timmy)),vp(iv(tries),adv(again)))
s(np(pn(timmy)),vp(av(is),adj(angry)))
s(np(pn(timmy)),vp(v(solves),np(det(the),n(problem))))
s(np(pn(timothy)),vp(v(sees),np(det(the),n(problem))))
s(np(pn(timothy)),vp(iv(laughs),_14110,np(pr(at),det(the),n(kids))))

the syntactic analysis completed!
```

Σημασιολογικός Αναλυτής
Κώδικας Prolog

Γραμματική

```
/* SEMANTICS CREATION RULES */

sem(1,Sem) --> sem_np(N), sem_vp(1,V,N1), {Sem=..[V,N,N1]}.

sem(2,Sem) --> sem_np(N), sem_vp(2,_,A), {Sem=..[A,N]}.

sem(3,Sem) --> sem_np(N), sem_iv(V,s), {Sem=..[V,N]}.

sem(4,Sem) --> sem_np(N), sem_iv(V,s), sem_adv(A), {Sem=..[V,N,A]}.

sem(5,Sem) --> sem_np(N), sem_tv(V,s), sem_np(N1), sem_np(N2), {Sem=..[V,N,N1,N2]}.

sem(6,Sem) --> sem_np(N), sem_vp(1,V,N1), sem_pp(P), sem_np(N2), {Sem=..[V,N,N1,P,N2]}.

sem(7,Sem) --> sem_np(N), sem_iv(V,s), sem_adv(A), sem_pp(P), sem_np(N1), {Sem=..[V,N,A,P,N1]}.

/* noun phrase */

sem_np(N) --> sem_pn(N).

sem_np(N) --> sem_det(_), sem_n(N).

sem_np(N) --> sem_n(N).

%sem_np(P,N) --> sem_pr(P), sem_det(_), sem_n(N).

sem_pp(P) --> sem_pr(P).

/* verb phrase */

sem_vp(1,V,N) --> sem_v(V,s), sem_np(N).

sem_vp(2,is,A) --> sem_av(is), sem_adj(A).
```

Κομμάτι του Λεξιλογίου

```
/* SEMANTICS VOCABULARY */
/* Intransitive Verbs (sem_iv) */
% needed for example :
sem_iv(walks,s) -->[walks].
sem_iv(comes,s)-->[comes].
sem_iv(comes,q)-->[coming].
sem_iv(gets,s)-->[gets].
sem_iv(gets,q)-->[getting].
sem_iv(laughs,s)-->[laughs].
sem_iv(laughs,q)-->[laughing].
sem_iv(tries,s)-->[tries].
sem_iv(tries,q)-->[trying].
% extension of vocabulary :
sem_iv(runs,s) -->[runs].
sem_iv(runs,q) -->[running].
sem_iv(hurts,s) -->[hurts].
sem_iv(hurts,q) -->[hurting].
sem_iv(jumps,s) -->[jumps].
sem_iv(jumps,q) -->[jumping].
sem_iv(shoots,s) -->[shoots].
sem_iv(shoots,q) -->[shooting].
/* Auxiliary Verbs (sem_av) */
% needed for example :
sem_av(is) -->[is].
% extension of vocabulary :
sem_av(does) -->[does].
sem_av(do) -->[do].
```

Αποτέλεσμα

Ερώτηση: understand('Ergasia3.txt').

(Ίσως εμφανιστούν προβλήματα με τα παραπάνω μονά αυτάκια σε διαφορετικό υπολογιστή, δεν γνωρίζω γιατί συμβαίνει.)

```
comes(timmy,quickly,to,house)
finds(timmy,tim)
confused(tim)
needs(tim,help)
helps(timmy,tim)
gets(timmy,confused)
sad(timmy)
hugs(tim,timmy)
difficult(problem)
tries(timmy,again)
angry(timmy)
solves(timmy,problem)
sees(timothy,problem)
laughs(timothy,loudly,at,kids)

the semantic analysis completed!
```

Προσθήκη Γνώσης στη Βάση

Κώδικας Prolog

```
/* update knowledge base */
update_knowledge_base([]) :- !.
update_knowledge_base([S|Sem]) :-
 assert(kb_fact(S)),
 write(kb_fact(S)),write(' asserted'),nl,
 update_knowledge_base(Sem), !.
/* all facts of knowledge base */
show_kb :- listing(kb_fact/1).
/* insert additional information to knowledge base */
/* examples : */
/* ?- tell([john,loves,icecream]). */
tell(Sentence):-
 sem(_, Sem, Sentence, []),
 assert(kb_fact(Sem)),
 nl,write(kb_fact(Sem)), nl, write(' added to knowledge base.'),nl, !.
/* ask knowledge base */
% Yes-No questions
/* examples : */
/* ?- ask([is,the,problem,difficult]). */
/* ?- ask([does,timmy,solve,the,problem]). */
/* ?- ask([is,tim,sad]). */
```

Αποτέλεσμα

Κάποια παραδείγματα που έχω δοκιμάσει είναι τα εξής αλλά πέρα από αυτά μπορούν να προστεθούν πολλές ακόμα πληροφορίες αρκεί α υπάρχουν οι λέξεις στο λεξιλόγιο και η δομή της πρότασης στους παραπάνω κανόνες.

Ερώτηση: understand('Ergasia3.txt').

(Ίσως εμφανιστούν προβλήματα με τα παραπάνω μονά αυτάκια σε διαφορετικό υπολογιστή, δεν γνωρίζω γιατί συμβαίνει.)

Εισαγωγή των προτάσεων στην βάση.

```
kb_fact(comes(timmy,quickly,to,house)) asserted
kb_fact(finds(timmy,tim)) asserted
kb_fact(confused(tim)) asserted
kb_fact(needs(tim,help)) asserted
kb_fact(helps(timmy,tim)) asserted
kb_fact(gets(timmy,confused)) asserted
kb_fact(sad(timmy)) asserted
kb_fact(hugs(tim,timmy)) asserted
kb_fact(difficult(problem)) asserted
kb_fact(tries(timmy,again)) asserted
kb_fact(angry(timmy)) asserted
kb_fact(solves(timmy,problem)) asserted
kb_fact(sees(timothy,problem)) asserted
kb_fact(laughs(timothy,loudly,at,kids)) asserted

Knowledge Base Updated!
```

Example:

tell([john,loves,icecream]).

```
?- tell([john,loves,icecream]).

kb_fact(loves(john,icecream))
 added to knowledge base.
true.
```

Ερωτήσεις στη Βάση

Υπάρχουν 2 ήδη ερωτήσεων στη βάση. Αυτές με απάντηση το ναι και το όχι και αυτές με απάντηση ενός ουσιαστικού.

Κώδικας Prolog

Ναι/Όχι

```
ask(X):- q(_, tf, Sem, X, []),
 if_then_else(kb_fact(Sem), write('Yes.'), write('No.') ), !.
/*-------------------------------------------------------------------*/
/* yes/no queries */
/*-------------------------------------------------------------------*/
q(1,tf,Sem) --> sem_av(does), sem_pn(N), sem_v(V,q), sem_np(N1), {Sem=..[V,N,N1]}.
q(1,tf,Sem) --> sem_av(did), sem_pn(N), sem_v(V,q), sem_np(N1), {Sem=..[V,N,N1]}.
q(2,tf,Sem) --> sem_av(is), sem_np(N), sem_adj(A),
 {Sem=..[A,N]}.
q(3,tf,Sem) --> sem_av(does), sem_pn(N), sem_v(have), sem_n(N1),
 {Sem=..[have,N,N1]}.
q(4,tf,Sem) --> sem_av(is), sem_pn(N), sem_iv(V,q),
 {Sem=..[V,N]}.
q(5,tf,Sem) --> sem_av(is), sem_pn(N), sem_iv(V,q), sem_adv(A),
 {Sem=..[V,N,A]}.
q(6,tf,Sem) --> sem_av(does), sem_pn(N), sem_tv(V,q), sem_pn(N1),
sem_np(N2), {Sem=..[V,N,N1,N2]}.
q(6,tf,Sem) --> sem_av(does), sem_pn(N), sem_tv(V,q), sem_pn(N1), sem_np(N2),
 {Sem=..[V,N,N1,N2]}.
```

Fact

```
ask(X):- q(_,fact, Fact, X, []), write(Fact), !.


/* fact queries */
q(1,fact,F) --> [who], sem_v(V,s),sem_n(N1), {Sem=..[V,F,N1], kb_fact(Sem)}.
q(1,fact,F) --> [what], sem_av(does),sem_pn(N),sem_v(V,q),{Sem=..[V,N,F], kb_fact(Sem)}.
q(2,fact,F) --> [who], sem_av(is), sem_adj(A),{Sem=..[A,F],kb_fact(Sem)}.
q(3,fact,F) --> [who], sem_vp(1,V,N1), {Sem=..[V,F,N1],kb_fact(Sem)}.
q(3,fact,F) --> [who], sem_av(does),sem_pn(N),sem_v(V,q),{Sem=..[V,N,F],kb_fact(Sem)}.
q(4,fact,F) --> [who], sem_av(is),sem_iv(V,q),{Sem=..[V,F],kb_fact(Sem)}.
q(5,fact,F) --> [how], sem_av(does),sem_pn(N),sem_iv(V,s),{Sem=..[V,N,F],kb_fact(Sem)}.
q(5,fact,F) --> [how], sem_av(is),sem_pn(N),sem_iv(V,q),{Sem=..[V,N,F],kb_fact(Sem)}.
q(5,fact,F) --> [who], sem_iv(V,s),sem_adv(A),{Sem=..[V,F,A],kb_fact(Sem)}.
q(6,fact,F) --> [who], sem_tv(V,s),sem_pn(N1),sem_np(N2),{Sem=..[V,F,N1,N2],kb_fact(Sem)}.
q(6,fact,F) --> [who], sem_av(is),sem_pn(N),sem_tv(V,q2),sem_np(N2),[to],
{Sem=..[V,N,F,N2] ,kb_fact(Sem)}.
q(6,fact,F) --> [what],[is],sem_pn(N),sem_tv(V,q2),[to],sem_pn(N1),
 {Sem=..[V,N,N1,F] ,kb_fact(Sem)}.
```

Αποτέλεσμα

Κάποια παραδείγματα που έχω δοκιμάσει είναι τα εξής αλλά πέρα από αυτά μπορούν να προστεθούν πολλές ακόμα πληροφορίες αρκεί α υπάρχουν οι λέξεις στο λεξιλόγιο και η δομή της πρότασης στους παραπάνω κανόνες.

Ερωτήσεις Ναι/Όχι

ask([is,the,problem,difficult]).

ask([does,timmy,solve,the,problem]).

ask([is,tim,sad]).

```
?- ask([is,the,problem,difficult]).
Yes.
true.

?- ask([does,timmy,solve,the,problem]).
Yes.
true.

?- ask([is,tim,sad]).
No.
true.
```

Ερωτήσεις Γεγονοτος

ask([who,is,difficult]).

ask([who,needs,help]).

ask([who,is,confused]).

```
?- ask([who,is,difficult]).
problem
true.

?- ask([who,needs,help]).
tim
true.

?- ask([who,is,confused]).
tim
true.
```