Some Aspects Regarding Natural Language Processing

Drd. Nicolae MĂRGINEAN
Facultatea de Ştiinţe Economice, Universitatea "Babeş-Bolyai", Cluj-Napoca

It is known that the key properties of every computer system interface is to be very friendly. What is more natural than a communication between human and machine realized in natural language? Natural language is a medium for human-machine interaction that has several obvious and desirable properties: it provides a means of accessing information in the computer independently of its structure and encodings, it shields the user from the formal access language of the underlying system and it is available with a minimum of training. There are many approaches regarding natural language processing but in this study, I focus my attention to DCG grammar, and shortly, ATN grammar. I have tried to present the main analysis that must be performed while processing a text: syntactic analysis, morphologic analysis, semantic analysis and pragmatic analysis.
***Keywords*:** *grammar, analysis, syntactic, morphologic, semantic, pragmatic.*

Cunoaştem faptul că una dintre caracteristicile de bază ale interfeţei unui sistem suport de decizie este convivialitatea, sau, cu alte cuvinte, calitatea acesteia de a fi prietenoasă faţă de utilizator. Ce este mai natural într-o interfaţă om-calculator decât o comunicare realizată într-un limbaj cât mai apropiat de limbajul natural? Domeniul inteligenţei artificiale care se preocupă de acest aspect ar fi "prelucrarea limbajului natural", situat undeva la graniţa între lingvistică şi ştiinţele cognitive. Părintele acestui domeniu este considerat omul de ştiinţă Naom Chomsky, studiile sale începând cu 1950, influenţând dezvoltarea ulterioară a limbajelor de programare prin intermediul limbajelor formale. Este memorabilă teza acestuia potrivit căreia "ne cunoaştem cunoscând limbajul". Plecând de la această teză, Alan Turing, tot în 1950, propune ca una dintre cerinţele pe care un calculator trebuie să le îndeplinească pentru a fi considerat inteligent, să fie aceea de a fi capabil să înţeleagă si să răspundă într-un limbaj natural.

Limbajul natural este o modalitate de comunicare om-maşină care prezintă câteva avantaje majore de partea utilizatorului:

- Nu necesită instruire pentru folosirea lui;
- Scuteşte utilizatorul de familiarizarea cu limbajele formale;
- Este un mijloc direct de accesare a informaţiei, independent de structura şi codificarea acesteia.

Interfeţele legate de interogarea bazelor de date precum şi cele legate de comunicarea cu sistemele expert, care asistă sistemele suport de decizie, reprezintă doar două direcţii în care prelucrarea limbajului natural oferă rezultate remarcabile.

În construirea interfeţelor este utilă studierea principiilor care guvernează dialogurile interumane. În 1975, Grice a enunţat un grup de trei principii de dialogare interumană:

1. principiul calităţii: orice participant la dialog nu face afirmaţii pe care le consideră false, şi el previne inducerea unor concluzii false la parteneri;
2. principiul cantităţii: orice participant produce o cantitate de informaţie care nu este nici mai mare nici mai mică decât este necesară;
3. principiul relaţiei: orice contribuţie la dialog trebuie să fie relevantă.

O interfaţă inteligentă este o interfaţă cu comportament comparabil cu acela al unei persoane ce încearcă să înţeleagă şi să răspundă unei întrebări puse de o altă persoană si care respectă principiile anterioare.

O interfaţă inteligentă trebuie să fie flexibilă, să ofere răspunsuri cooperante ca reacţie la o interogare si să ofere posibilitatea alegerii dintre mai multe răspunsuri corecte, pe acela care este cel mai util în dialog. O interfaţă inteligentă flexibilă trebuie să aibă în vedere şi

propozițiile mai puțin gramaticale sau imperfecte care apar deseori în dialogurile umane.
O interfață inteligentă ajută utilizatorul în extragerea informațiilor din baza de date, în manevrarea cunoştințelor din bazele de cunoştințe, în procesele de diagnoză etc., prin furnizarea unor mesaje contextuale, a unor inferențe realizate independent de utilizator, facilitând astfel comunicarea om-maşină. Problema centrală a acestora este transformarea unei intrări formulate în limbaj natural, posibil ambiguă, într-o formă neambiguă, internă, a sistemului, care poate fi utilizată în lansarea unei operații de procesare a datelor. În funcție de aceste transformări distingem:

- interfețe bazate pe traducerea directă a limbajului natural într-un limbaj de comandă;
- interfețe care transformă intrarea într-o formă logică intermediară;
- interfețe bazate pe dialog, care determină ce acțiuni trebuie executate după prezentarea intrărilor.

Se disting două avantaje majore ale interfețelor în limbaj natural:

- flexibilitatea în formularea întrebărilor, utilizatorul netrebuind să învețe un nou limbaj şi să respecte restricțiile acestuia;
- independența interfeței de structurile de date, utilizatorul netrebuind să cunoască protocoalele de căutare, diferitele mnemonici şi proceduri.

Trebuie avut totuşi în vedere faptul că limbajul natural este atât de complex încât construirea unor modele care să rezolve în totalitate problemele conexe este deocamdată un deziderat greu de realizat.
O aplicație de prelucrare a limbajului natural poate prezenta următoarele trei componente:
1. scanner-ul care realizează descompunerea unei propoziții/fraze într-o listă a cuvintelor sale componente;
2. parser-ul care realizează analiza propozițiilor/frazelor conform regulilor specificate de gramatica adoptată precum şi obținerea semanticii aferente acestora. Pentru aceasta, se apelează la analiza sintactică şi morfologică, la analiza semantică şi la analiza pragmatică;
3. executivul care preia semantica obținută în pasul anterior, transformându-o în comenzi sau cunoştințe.

Unul din limbajele cele mai vechi şi mai eficiente, folosite în prelucrarea limbajului natural este Prolog-ul. Pe parcursul acestei lucrări vom reîntâlni secvențe de program specifice TurboProlog şi Sicstus Prolog.
Asupra scannerului nu voi insista, acesta fiind de fapt o procedură clasică de "spargere" a propozițiilor în cuvintele componente, folosind limbajul Prolog.

```
scanner(Text,[Token|Tokenlist]):-
      fronttoken(Text,Token, Resttext),
scanner(Resttext, Tokenlist).
scanner(_,[]).
```

Pentru construirea parser-ului vom apela la mai multe tipuri de analiză. Prima analiză avută în vedere este cea sintactică.
Analiza sintactică ne ajută să înțelegem modalitatea prin care cuvintele se grupează în vederea construirii unei propoziții sau fraze. În acest sens, o propoziție este formată din constituenți sintactici, unii dintre ei fiind, la rândul lor, compuşi din constituenți. Constituenții care la rândul lor sunt formați din constituenți se numesc non-lexicali în timp ce toți ceilalți se numesc lexicali.
Constituenții lexicali (cuvintele vocabularului) împreună cu caracteristicile lor morfologice (număr, timp, caz etc.) formează lexiconul. Constituenții lexicali pot fi: substantive, adjective, adverbe, verbe, aşa numiții determinanți în care intră articolele, adjectivele demonstrative, adjectivele relative, adjectivele posesive etc., apoi conjuncțiile, prepozițiile şi pronumele. De regula, determinanții, pronumele, conjuncțiile şi prepozițiile se cunosc şi sunt fixate formând o clasa închisă de cuvinte. În schimb, restul constituenților lexicali formează o clasă de cuvinte deschisă, deoarece se pot adăuga mereu noi elemente.
Constituenții non-lexicali pot fi centrați în jurul unui anumit tip de constituent. Astfel putem distinge:

- NP (noun phrase) – sunt centrați în jurul unui substantiv; sunt formați dintr-un substantiv sau pronume plus elementele care îl modifică (determinant, adjectiv, alt substantiv etc);
- VP (verb phrase) – sunt centrați în jurul unui verb; sunt formați dintr-un verb şi complementele sale directe şi indirecte;

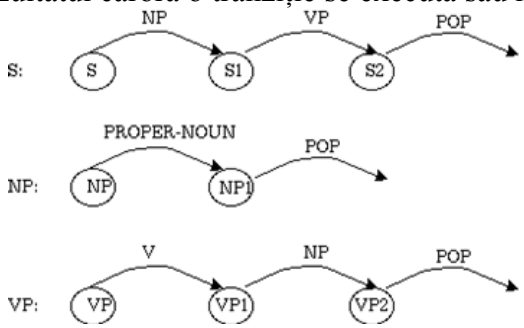
- ADJP (adjective phrase) – sunt centraţi în jurul unui adjectiv; sunt formaţi dintr-un adjectiv şi modificatori;
- ADVP (adverbial phrase) – sunt centraţi în jurul unui adverb; sunt formaţi dintr-un adverb şi modificatori.;
- PP (prepositional phrase) – sunt centraţi în jurul unei prepoziţii; sunt formaţi dintr-o prepoziţie şi un NP;
- RelClause (relative clause) – sunt centraţi în jurul unui pronume relativ; sunt formaţi dintr-un pronume relativ(care, ca) şi unVP sau chiar o altă propoziţie;
- S (sentence) – este formată din constituenţii de mai sus, în principal un NP şi un VP.

Regulile de sintaxă care specifică organizarea posibilă a cuvintelor într-o propoziţie poartă denumirea de gramatică. Aceasta încearcă să specifice cât mai multe posibilităţi de structurare a propoziţiilor. Ea se defineşte într-o maniera similară cu cea din cadrul limbajelor de programare. Totuşi, în contextul limbajului natural, aceasta este mult mai complexa. Din acest motiv, este puţin probabil ca o gramatică să acopere toate cazurile posibile de structurare a propoziţiilor.

De-a lungul timpul s-au dezvoltat mai multe gramatici, dintre care eu voi aminti doar două dintre acestea.

Gramatica ATN (Augmented Transition Networks). Gramatica ATN seamănă foarte mult cu reţelele semantice, propoziţia fiind reprezentată sub forma unui set de grafuri de tranziţie. Fiecare graf corespunde unui constituent neterminal al gramaticii. Arcurile grafurilor sunt etichetate cu simbolurile constituenţilor terminali sau neterminali. Fiecare cale străbătută prin reţea, de la starea iniţiala până la starea finală, corespunde părţii de premisă a regulii aferente acelui nonterminal. Când există mai multe reguli pentru un nonterminal, între starea iniţială şi finală a unui graf se pot identifica mai multe căi de parcurgere. Parcurgerea cu succes a grafului de la un capăt la celălalt presupune validitatea acelui nonterminal în cadrul regulii din care face parte, aşa încât se poate trece la stabilirea validităţii următorului constituent al regulii. Putem afirma astfel că o propoziţie satisface o gramatică ATN în cazul în care structurii acesteia îi corespunde o cale de parcurgere a grafurilor de tranziţie, de la un capăt la celălalt.

Gramatica ATN extinde gramatica TN (transition network) prin faptul că permite ataşarea unor proceduri arcelor constituente ale grafurilor de tranziţie. Aceste proceduri se execută în momentul în care se parcurg arcele grafurilor. Aceste proceduri pot asocia valori caracteristicilor gramaticale ale constituenţilor sau pot lansa nişte teste în funcţie de rezultatul cărora o tranziţie se execută sau nu.



Cele trei grafuri le putem reuni într-un singur graf. Astfel, tranziţiile de mai sus pot fi reprezentate foarte uşor în Prolog sub forma unor termeni de genul:

trans(stare1, stare2, stare_finala, functie_morfologica, functie_sintactica)

ordonaţi după stare1 cu ajutorul b-arborilor.

Lexiconul îl putem reprezenta sub forma unor termeni de genul *word(cuvant, functie morfologica)*, ordonându-l alfabetic cu ajutorul b-arborilor.

În cele ce urmează vom prezenta un analizor sintactic

```
gramatica:-
     open_databases,
     set_state(s0),
     write("Introdu propozitia:"),
     readln(S),
     analyze(S),
     afisez_sintaxa(s0),
     clear_sintax,
     close_databases,
     clear_state.

set_state(S):-assertz(stare(S)).
get_state(S):-stare(S).

analyze("."):-
    get_state(S),
    key_search(net,netarb,S,Adr),
    ref_term(net,transition,Adr,Arc),
    Arc=trans(S,_,"yes",_,_).
analyze(Sentence):-
    get_state(S),
    key_search(net, netarb,S,Adr),
    ref_term(net,transition, Adr, Arc),
```

```
    Arc=trans(S,S1,State_f,         Type,
Function),
    fronttoken(Sentence, Token, Rest),
    key_search(lex,lexarb,Token,Adr1),
    ref_term(lex,word,Adr1,Word),
    Word=word(Token,Type),!,
    set_state(S1),
    assertz(syntax(S,S11,Toke,
Type,Function)),
    analyze(Rest);
analyze(_):-fail.

afisez_sintaxa(S):-
      syntax(S,S1,Token,Type,Function),
      write(Token, Type, Function),
      afisez_sintaxa(S1),
afisez_sintaxa(_).
```

Gramaticile CFG (context free grammars) şi DCG(definite clause grammars)

O gramatică CFG este o specificare formală a structurilor admise într-un limbaj, sub forma unor reguli de descriere. Este larg utilizată deoarece formalismul acesteia este suficient de puternic pentru a descrie cea mai mare parte a structurilor unui limbaj natural. După cum observăm şi din denumire, aceasta este o gramatica ce nu ține seama de contextul în care este folosit acel limbaj.

De exemplu, pentru reprezentarea propozițiilor:

a) Ion mănâncă un biscuit.
b) Acest leu mănâncă o căprioară.

obținem următoarea gramatică CFG:

```
sentence →noun_phrase, verb_phrase.
noun_phrase → proper_noun.
noun_phrase→determiner, noun.
verb_phrase →verb, noun_phrase.
proper_noun →[ion].
noun →[caprioara].
Noun→[biscuit].
noun →[leu].
verb →[mananca].
determiner →[un,o, acest].
```

Se observă cum o propoziție satisface gramatica enunțată dacă este formată dintr-un noun_phrase şi un verb_phrase. La rândul său, un noun_phrase poate avea două structuri, una dată de un substantiv propriu, şi una dată de un determinant plus un substantiv comun. În fine, un verb_phrase este format dintr-un verb şi un noun_phrase, apelându-se în mod recursiv definițiile enunțate mai sus.

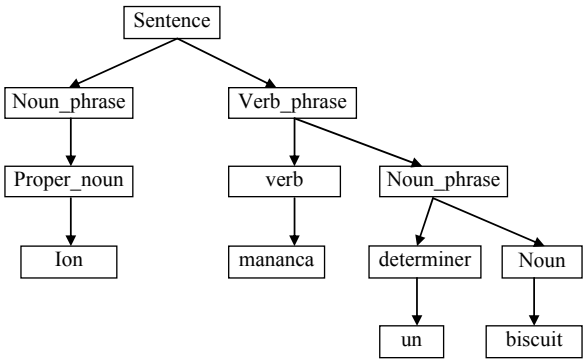
Rolul parser-ului este de a identifica regulile gramaticii care se potrivesc cu structura propoziției analizate. Acțiunea lui se concretizează într-un proces de căutare în toate structurile sintactice posibile, identificându-le pe cele care se potrivesc cu propoziția de analizat. După cum ştim, există mai multe strategii de căutare (depth first, breath first), dar cea mai la îndemână este depth first (căutarea în adâncime cu revenire prin backtracking) care este implementată direct în Prolog. Rezultatul oferit de parser ne confirmă sau ne infirmă corectitudinea unei propoziții în raport cu gramatica enunțată în prealabil.

Traseul rezolutiv al unui algoritm de analiză sintactică (sau al parser-ului) al unei propoziții poate fi reprezentat sub forma unui arbore (parser tree).

De exemplu, pentru propoziția "Ion mănâncă un biscuit." obținem următoarele reprezentări:

1.în formă liniară *sentence(noun_phrase(proper_noun(Ion)), verb_phrase(verb(mananca), noun_phrase(determiner(un), noun(biscuit))))*

2. în formă arborescentă



Totuşi, analizând gramatica de mai sus, observăm că ea este incompletă. De exemplu, forma verbului este aceeaşi pentru toate formele subiectului, indiferent de număr sau persoană. În realitate, situația este alta, forma verbului depinzând de numărul subiectului. De aceea se impune introducerea unor parametrii care să permită şi evidențierea acestor aspecte ale gramaticii. În plus, tot gramatica de mai sus suferă de aşa numitul fenomen de supragenerare, care permite validarea unor propoziții eronate semantic. De exemplu, propoziția "un biscuit mănâncă Ion" este corectă din punct de vedere al sintacticii gramaticii enunțate dar incorectă din punct de vedere semantic. De aceea, se impune introduce-

rea aşa numitelor constrângeri semantice.
Gramatica CFG în care se introduc variabile poartă denumirea de gramatica DCG(definite clause grammar). Ea se mai numeşte şi augmented grammar, de la operaţia de adăugare la neterminale a unor variabile. Formalismul DCG este atractiv deoarece ne permite descrierea unei gramatici în termenii logicii de ordinul întâi.
Gramatica CFG de mai sus poate fi transformată într-o gramatică DCG în care se va introduce un parametru ce reprezintă numărul subiectului.

```
sentence → noun_phrase(Num),
verb_phrase(Num).
noun_phras(Num)e → proper_noun(Num.
noun_phrase(Num)→determiner(Num),
noun(Num).
verb_phrase →verb, noun_phrase(_).
proper_noun(s) →[ion].
noun(s) →[caprioara].
noun(s) →[leu].
noun→[biscuit].
verb(s) →[mananca].
Determiner(s) →[un,o, acest].
```

Operaţiunea de acord subiect-verb de mai sus face parte dintr-o analiză mai amplă a formării cuvintelor în cadrul unei propoziţii, numită **analiza morfologică**. Sistemele de prelucrare a limbajului natural trebuie să conţină reguli explicite de formare a diferitelor inflexiuni ale unei intrări lexicale: singularul/pluralul substantivelor, genul masculin/feminin/neutru, conjugarea verbelor la toate persoanele, acordurile adjectivelor cu substantivele etc. Alte probleme importante în procesarea morfologică ar fi cele privitoare la fenomenul de aglutinare, (adică un acelaşi cuvânt suportă mai multe sufixe, ex naţion, naţional, naţionalism) şi la formarea cuvintelor compuse utilizând cratima (weekend).
În continuare, pentru exemplificarea unei analize morfologice, să analizăm formarea pluralului substantivelor limbii engleze. De exemplu, se ştie că substantivele care se termină în y, la plural y se transformă în *i* şi se adaugă sufixul *es*. Excepţie fac substantivele care înaintea lui y au o vocală, situaţie în care, pentru plural, se adaugă doar un s.

```
plural(S,P):-convert(S,List),
//converteste un cuvant intr-o lista de
litere
append(Base,[C,y], List), //imparte lis-
ta in doua subliste
not(vocala (C),
append(Base,[C,i,e,s],List1), //reuneste
primele doua liste in a treia
convert(P,List1). //transforma lista de
litere in cuvant
vocala(C):-member(C,[a,e,i,o,u]).
//verifica daca C este vocala
```

Revenind la **constrângerile semantice**, spuneam că scopul lor este de împiedicare a generării unor propoziţii cu înţeles eronat. Formele acestora sunt din cele mai variate.
O constrângere semantică simplu de realizat este aceea care statuează că subiectul şi complementul unei propoziţii trebuie să fie diferite.
O altă constrângere simplă ar fi introducerea caracterului de tranzitivitate a verbelor. Verbele intranzitive acceptă doar un singur parametru dat de subiectul acţiunii pe când verbele tranzitive acceptă mai mulţi parametrii, pe lângă subiect fiind prezente şi complementele.

```
Verb tranzitiv mananca(X,Y)
Verb intranzitiv canta(X)
```

O adăugire la constrângerea de mai sus ar fi aşa numita subcategorizare a verbelor. Prin aceasta înţelegem că fiecărui verb tranzitiv îi ataşăm o listă cu acele categorii de constituienţi pe care le acceptă ca şi complemente. De exemplu, în propoziţia "John gives the book to me" verbul "give" are complementul (de tip NP) "the book" şi complementul (de tip PP) "to me", deci lista de subcategorizare este[NP,PP]. Acelaşi verb give poate avea lista de subcategorizare [NP,NP], ca in exemplul "John give me the book". Pentru a integra subcategorizarea verbelor într-o gramatică DCG va trebui să adăugam categoriei VP un argument de tip listă, reprezentând complementele necesare pentru a forma un VP complet.

```
VP(S,Subcat):-VP1(S1,[np|Subcat]),NP(S2),append(S1,S2,S)
                          |VP1(S1,[pp|Subcat]),PP(S2),append(S1,S2,S).
VP1([S],[np]):-verb1(S).
VP1([S],[pp]):-verb2(S).
verb1(mananca).
verb2(canta).
```

unde
```
Sentence(S)-
>NP(S1),VP(S2,[]),append(S1,S2,S).
```
iar `verb1` reprezintă verbele care acceptă NP; `verb2` reprezintă verbele care accepta PP

O altă constrângere pe care o putem introduce este aceea a ataşării unor numere de ordine constituenţilor, astfel încât constituentul cu numărul de ordine mai mic să se găsească, în structura propoziţiei, înaintea constituentului cu numărul de ordine mai mare.

Dacă până în acest moment am încercat să identificăm structurile propoziţiilor, următoarea fază ar fi aceea de relevare a înţelesului acesteia. Cu această problema se ocupă semantica şi pragmatica. Dacă semantica încearcă să încropească o reprezentare parţială a înţelesului unei propoziţii, pornind de la înţelesul cuvintelor şi de la structurile sintactice ale propoziţiei, pragmatica vine şi întregeşte înţelesul propoziţiei cu ajutorul unor cunoştinţe legate de context.

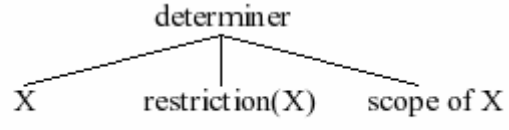
În ceea ce priveşte semantica unei propoziţii, una din strategiile des adoptate în aplicaţii este cea a semanticii compoziţionale, care prevede că semantica unei propoziţii se obţine prin reunirea semanticii fiecărui constituent în parte al propoziţiei. Această reunire a înţelesurilor fiecărui cuvânt se poate realiza în acelaşi timp cu analiza sintactică a propoziţiei în cauză. Pentru aceasta, fiecare constituent lexical din lexicon trebuie să aibă ataşat într-un argument înţelesul aferent. Putem spune astfel că funcţiile principale ale parserului sunt de analiză sintactică şi morfologică a propoziţiei la care, se adaugă, în caz de validare a acesteia, analiza semanticii aferente propoziţiei.

Semantica compoziţională presupune găsirea unei modalităţi de reprezentare adecvată a fiecărui cuvânt aşa încât să poată fi combinate într-o manieră sistematică pentru obţinerea semanticii globale a propoziţiei. Modalitatea cea mai uzitată pentru reprezentarea semanticii o reprezintă logica predicatelor.

De exemplu, în logica predicatelor vom reprezenta propoziţia "John loves Mary" cu ajutorul predicatului loves(john,mary). John şi Mary referă entităţi specifice. Pentru o propoziţie de tipul "The man likes Mary" nu ştim cine este bărbatul care o place pe Mary, de aceea îl reprezentăm printr-o variabilă X. Prin faptul ca substantivul "man" are în faţă articolul hotărât "the", înseamnă că există un bărbat bine determinat care o place pe Mary. Pentru aceasta trebuie să introducem cuantificatorul existenţial. Astfel semantica ataşată propoziţiei va avea următoarea formă: *exista (X, man(X) si likes(X,mary))*. Dacă mai adăugăm un atribut bărbatului "The tall man likes Mary", semantica propoziţiei devine *exista(X,man(X) si tall(X) si likes(X,mary))*.

Reprezentarea semantică de mai sus este foarte populară pentru mânuirea semanticii în Prolog şi poarta denumirea de reprezentarea 3BQ(3 branch quantifier). Astfel un determinant are in structura sa un X care reprezintă elementul determinat, o restricţie îndeplinită de X, şi un scop al lui X. Determinantul "a" din engleza poate fi reprezentat astfel:

A 3BQ tree for the determiner "a"



sau în Prolog
determiner(X, man(X), sleeps(X), exista(X, man(X), sleeps(X)))→[a].

Dacă în locul determinatului 'a' avem determinantul 'every" atunci vom avea
determiner(X, man(X), sleeps(X), all(X, man(X)=>sleeps(X)))→[every].

În exemplul următor putem observa o gramatică în care sunt introduse şi elementele semantice.
```
:-  op( 100, xfy, and).
:-  op( 150, xfy, =>).

sentence( S)  -->
  noun_phrase( X, P, S), verb_phrase( X,
P).
```

```
noun_phrase( X, P, S)  -->
  determiner( X, P12, P, S), noun( X,
P1), rel_clause( X, P1, P12).

noun_phrase( X, P, P)  -->
  proper_noun( X).

verb_phrase( X, P)  -->
  trans_verb( X, Y, P1), noun_phrase( Y,
P1, P).

verb_phrase( X, P)  -->
  intrans_verb( X, P).

rel_clause( X, P1, P1 and P2)  -->
  [that], verb_phrase( X, P2).
rel_clause( X, P1, P1)  --> [].

determiner( X, P1, P, all( X, P1 => P))
--> [every].
determiner( X, P1, P, exists( X, P1 and
P)) --> [a].

noun( X, man(X))  -->  [man].
noun( X, woman(X))  -->  [woman].
proper_noun( john)  -->  [john].
proper_noun( annie)  -->  [annie].
proper_noun( monet)  -->  [monet].
trans_verb( X, Y, likes( X, Y))  -->  [
likes].
trans_verb( X, Y, admires( X, Y))  -->
[admires].
intrans_verb(  X,  paints(X))  -->
[paints].
```

Semantica unei propoziții folosind maniera de reprezentare de mai sus va duce la obținerea unor arbori 3BQ liniarizați. Folosind codul de mai sus, pentru propoziția "Every woman that admires a man that paints likes Monet" obținem următorul înțeles:
all(X,woman(X) and exists(Y,(man(Y) and paints(Y)) and admires(X,Y) =>likes(X,monet)).

Ca o paranteza, semantica de mai sus o obținem în parametrul S prin lansarea:
phrase(sentence(S),[every, woman, that, admires, a, man, that, paints,likes,monet]).
De asemenea, să nu uităm că în regulile gramaticale de mai sus lipsesc doi parametrii, unul care reprezintă şirul de intrare iar celălalt un sir intermediar de ieşire De exemplu, regula gramaticală p(X)→ q(X) devine în Prolog p(X,S0,S1):-q(X,S0,S1).
Dacă analizăm semantica obținută în exemplul de mai sus, aceasta nu este altceva decât o formulă cu cuantificatori a predicatelor de ordinul întâi. Observăm rolul deosebit de important pe care îl au determinanții în structura semanticii unei propoziții. Putem spune că semantica unei propoziții este formată din structurile imbricate ale determinanților (de tip existențial sau universal), în interiorul cărora regăsim semantica celorlalți constituenți.

În ceea ce priveşte semantica celorlalți constituenți, pentru substantive şi adjective descriptive (roşu, rotund etc.) se folosesc predicate cu o singură variabilă iar pentru verbe, în funcție de tranzitivitate, predicate cu una sau mai multe variabile. În cazul în care subiectul este format din mai multe categorii sintactice, (ex "fata care cântă"), semantica este formată din conjuncția categoriilor sintactice. Pentru cazul nostru, fata(x) şi cântă(x). La fel se pune problema şi pentru construcțiile la genitiv, cu precizarea că se introduce un predicat "posedă". De exemplu, semantica pentru "cartea studentului" se reprezintă astfel: *cartea(X) si poseda(X, student)*. În cazul complementelor unui verb, acestea se împart în două categorii: modificatorii care corespund complementelor verbelor tranzitive şi adjuncții care sunt dați de complementele de timp, mod şi loc. Modificatorii unui verb, X şi Y se reprezintă ca argumente ale verbului tranzitiv, foloseşte(X,Y). Numărul adjuncților nefiind în general cunoscut, ei se leagă printr-o conjuncție "si" de verbul la care se referă. De exemplu " Ion merge la film" se reprezintă film(X) şi merge(X,Y).
Pentru a putea fi mai uşor interpretată semantica unei propoziții în cadrul componentei executive, aceasta este bine să fie adusă la forma clauzală fără cuantificatori. Pentru aceasta se parcurg următorii trei paşi:
1. forma inițială se transformă în formă standard prenex;
2. forma standard prenex se aduce la forma Skolem;
3. se suprimă cuantificatorii universali ai formei Skolem obținând forma clauzală.

Trebuie făcute câteva precizări: Forma normala prenex este cea în care toți cuantificatorii se găsesc în fața formulei. Forma normala Skolem este o formă normală prenex în care nu regăsim decât cuantificatori universali.

Algoritmul pentru obținerea formei normale prenex:
1. se elimină conectivele de implicație şi echivalență;
2. se redenumesc anumite variabile legate pentru a nu avea variabile cuantificate de două ori. Motivul acestei redenumiri este faptul că

$$\forall xA(x) \vee \forall xB(x)\ not = \forall x(A(x) \vee B(x))$$

$$\exists xA(x) \wedge \exists xB(x)\ not = \exists x(A(x) \wedge B(x))$$

3. prin redenumirea variabilei x se poate scrie de exemplu

$$\forall xA(x) \vee \forall xB(x) = \forall xA(x) \vee \forall zB(z) =$$
$$\forall x \forall z(A(x) \vee B(z))$$

4. se aduc semnele negației în fața atomilor, aplicând:

$$\neg(\neg A) = A$$
$$\neg(A \vee B) = \neg A \wedge \neg B$$
$$\neg(A \wedge B) = \neg A \vee \neg B$$
$$\neg(\forall xA(x)) = \exists x(\neg A(x))$$
$$\neg(\exists xA(x)) = \forall x(\neg A(x))$$

5. se mută cuantificatorii la stânga întregii formule, folosind următoarele echivalențe, (x nu este variabilă liberă a formulei B)

$$(\forall xA(x) \wedge B) = \forall x(A(x) \wedge B)$$
$$(\forall xA(x) \vee B) = \forall x(A(x) \vee B)$$
$$(\exists xA(x) \wedge B) = \exists x(A(x) \wedge B)$$
$$(\exists xA(x) \vee B) = \exists x(A(x) \vee B)$$

6. Pentru a limita numărul cuantificatorilor formulei finale, uneori este util să nu se facă toate redenumirile posibile, ci să se utilizeze formulele:

$$\forall xA(x) \wedge \forall xC(x) = \forall x(A(x) \wedge C(x))$$
$$\exists xA(x) \vee \exists xC(x) = \exists x(A(x) \vee C(x))$$

Algoritmul pentru determinarea formei normale Skolem:
1. partea formei normale prenex de după grupul cuantificatorilor se transformă în forma normală conjunctivă, aplicând formulele logicii predicatelor.
2. Se analizează fiecare cuantificator existențial din prefixul formei normale prenex;
a. Dacă înaintea unui cuantificator existențial nu se află nici un cuantificator universal, atunci se alege o constantă care diferă de toate celelalte şi se înlocuiesc toate aparițiile variabilei cuantificate cu aceasta şi se şterge cuantificatorul existențial din prefixul formei;
b. Dacă înaintea unui cuantificator existențial se află cel puțin un cuantificator universal, atunci alegem o funcție cu un număr de n variabile identic cu numărul cuantificatorilor universali aflați în fața celui existențial în cauză. În final ştergem cuantificatorul existențial din prefixul formulei.
Constantele şi funcțiile folosite pentru a înlocui variabilele existențiale se numesc funcții Skolem.

Pentru a finaliza cu adevărat înțelesul real al unei afirmații, pe lângă identificarea semanticii propoziției obținută prin reunirea semanticii individuale a cuvintelor, trebuie să ținem seama şi de contextul în care aceasta este făcută: unde este spusă, de către cine şi de ce, şi ce a fost spus anterior. După cum am mai amintit, ne situăm deja la nivelul **analizei pragmatice**. Vom aborda două dintre aspectele amintite.
Pentru a sublinia de ce nu este suficient să ne oprim după analiza semantică, să aruncăm o privire asupra rolului limbajului în comunicare şi acțiune. Când dialogăm cu cineva, fiecare afirmație are un scop. Uneori acest scop este de a comunica anumite fapte. Astfel, dacă noi spunem "Ion o iubeşte pe Maria" e posibil ca singurul scop a acestei afirmații să fie comunicarea acelui fapt. În consecință, ascultătorul adaugă faptul *iubeşte(ion,maria)* în baza sa de cunoştințe. Totuşi, uneori, scopurile afirmațiilor noastre sunt mult mai profunde. De exemplu, în propoziția "poți închide fereastra?" scopul poate fi, fie o rugăminte de a închide fereastra, fie o întrebare cu privire la capacitatea persoanei intervievate de a închide fereastra. La fel, în propoziția "ai întârziat", scopul poate fi, fie acela de a aduce la cunoştință cuiva că a întârziat, fie acela de a critica ascultătorul pentru lipsa de punctualitate. Exemplele pot continua. Actul vorbirii este folosit pentru a referi acțiuni ce pot fi realizate prin intermediul limbajului. Actul vorbirii poate include informarea, cererea şi promisiunea. În spatele fiecărui act de vorbire se află un scop, adică ceva ce vorbitorul

vrea sa realizeze. Să încercăm să exprimăm câte un scop posibil al propozițiilor de mai sus. Dacă S este vorbitorul şi H este ascultătorul, atunci putem exprima scopurile astfel: "S vrea ca H să închidă fereastra" sau "S vrea ca H să ştie ca a întârziat". Ascultătorul trebuie să fie capabil să identifice aceste scopuri pentru a putea răspunde corespunzător. Concluzia este ca nu este de ajuns să cunoaştem semantica unei afirmații, ci trebuie cunoscut şi scopul acesteia.

Aceasta analiză a scopurilor din spatele unei afirmații este doar un aspect al pragmaticii. Ideea de planificare a vorbirii este un alt aspect important al pragmaticii. Dacă limbajul este folosit pentru a atinge scopuri, ca şi celelalte acțiuni, decizia a ceea ce urmează să spunem poate fi considerat ca un proces de planificare. În timp ce identificarea scopurilor este o problemă de recunoaştere a unui plan, decizia de a vorbi are la baza un proces de întocmire a planului. Iar pentru a putea întocmi acest plan avem nevoie de anumite cunoştințe ale momentului. De exemplu, dacă întrebarea "ştii cât este ceasul?" este pusă într-o situație în care ascultătorul a întârziat la o întâlnire, atunci afirmația poate fi interpretată ca o critică. Cealaltă variantă este cea în care vorbitorul vrea să se informeze asupra orei exacte.

O altă problemă ce merită a fi amintită este folosirea pronumelor a căror semnificație este strâns legată de context. Să luăm următoarea afirmație:

"Ion o sărută pe Maria. El o iubeşte."

Este clar că "El" se referă la Ion în timp ce "o" face referire la Maria. Astfel semantica va avea următoarea formă: *iubeste(ion,maria*). Dar pentru a obține aceasta interpretare este clar că trebuie să analizăm propoziția precedentă. Aceasta oferă contextul în care propoziția ultimă este interpretată. Sunt situații în care semnificația pronumelor o regăsim cu mult înaintea propoziției în cauză.

De altfel, problema găsirii obiectelor de referință nu se pune numai în cazul pronumelor. Până şi "noun_phrase"-urile pot referi anumite obiecte regăsite anterior. De exemplu, "Ion a văzut o maşină albastră Nissan şi una roşie Porsche în garaj. El s-a decis să o cumpere maşina roşie." "Noun phrase"-ul "maşina roşie" se referă la maşina Porsche.

În toate fazele tratării limbajului natural nu trebuie neglijată nici aşa numita problemă a **ambiguității propozițiilor,** potrivit căreia putem regăsi mai multe interpretări pentru o aceeaşi propoziție. Vom aminti în continuare câteva tipuri de ambiguități:

- Ambiguitatea sintactică. Aici distingem două probleme: ambiguitatea categorială şi ambiguitatea structurală. Prima se referă la posibilitatea existenței a două categorii sintactice pentru un acelaşi cuvânt. De exemplu cuvântul "flies" poate îmbrăca două forme: de substantiv – muşte sau de verb – zboară. A doua se referă la posibilitatea de a identifica mai multe structuri pentru o aceeaşi propoziție. Exemplu "John saw Mary on the hill with a telescope" este clasic în literatura de specialitate. Aici putem avea două situații: 1.)verbul saw are un singur complement modificator "Mary with a telescope" unde "with a telescope" se referă la Mary sau 2.) verbul saw are două complemente: complementul modificator Mary şi complementul adjunct "with a telescope". Eliminarea ambiguității categoriale se face de obicei prin analiza sintactică iar cea structurală prin folosirea unor elemente de punctuație (de exemplu virgula).
- Ambiguitatea semantică apare atunci când pentru un cuvânt avem aceeaşi funcție sintactică dar înțelesuri diferite. De exemplu, "bank" poate fi "river bank" (adica malul unui rau) sau o instituție bancară. În propoziția "I saw her run to the bank" nu putem identifica sensul cuvântului bank, deoarece ambele se potrivesc. Un alt exemplu ar fi cel al cuvântului charged din următoarele propoziții: "The battery was charged with jump leads" "Thief was charged by PC Smith" "The lecturer was charged with student recruitement". Această ambiguitate mai poartă numele de ambiguitate lexicală şi se poate elimina în urma analizei semantice.
- Ambiguitatea pragmatică sau referința anaforică apare atunci când în propoziție se află un pronume pentru care nu se ştie la ce sau la cine face referire. De exemplu, refe-

rința they în fraza "After they finished the exam the students and lecturers left". Se poate referi fie numai la studenti, fie numai la lectori, fie la ambii. De cele mai multe ori, referința pronumelui o găsim în propozițiile anterioare

- O ultimă situație de ambiguitate ar fi așa numitele "ellipsis" care constau în omiterea unor părți a propozițiilor. În exemplul "Peter worked hard and passed the exam. Kevin too" pot fi identificate trei interpretări: Kevin a muncit din greu, Kevin a trecut examenul sau le-a realizat pe amândouă.

După finalizarea analizei pragmatice, parserul își încheie misiunea, oferind componentei executive controlul şi semantica obținută. Executivul preia semantica sub forma unui parametru, o prelucrează şi o transforma în acțiunea dorită.
Înainte de a încheia, voi aminti atât doar că nu este suficient a înțelege ceea ce se transmite prin limbaj natural, ci există şi cealaltă fațetă, cea a posibilității formulării unor răspunsuri tot în limbaj natural. Foarte pe scurt, generarea în limbaj natural poate fi împărțită şi ea în două etape. Prima dată se stabileşte ce trebuie să se spună (plan amintit mai sus a ceea ce trebuie să se spună) după care se stabileşte cum să se spune, sau în ce formă. Operațiunea de generare a unui propoziții este oarecum inversă operației de analiză a unei propoziții. Dacă în procesul de analiză a unei propoziții, se pleca de la aceasta si se obținea semantica aferentă, în generare se pleacă de la semantica a ceea ce se doreşte a se transmite obținând ca rezultat propoziția dorită.
Ex: pentru sentence(likes(john, annie),S,[]) obținem *S=[john, likes, annie]*.
pentru *sentence(exist(X, man(X) and paints(X)), S, [])* obținem *S=[a ,man,paints]*.

Numărul aplicațiilor informatice din orice domeniu creşte pe zi ce trece, exigențele sunt din ce în ce mai mari iar concurența din ce în ce mai acerbă. Integrarea prelucrării limbajului natural în interfețele aplicațiilor informatice ale viitorului va constitui unul din elementele critice de succes ai unei aplicații informatice, în contextul în care comunicarea om-maşina va trebui simplificată cât mai mult posibil.

Bibliografie

1. Dumitrescu D, *Principiile inteligenței artificiale*, Editura Albastra, Cluj-Napoca. 1999
2. Ken Barker, *Natural language processing*, Fall, 1999
3. Meszaros Judith, *TurboProlog 2.0, Ghid de utilizare*, Editura Albastră, Cluj-Napoca, 1996
4. Russel, Norvig, *Artificial Inteligence, A modern approach,* Prentice-Hall International, 1995
5. Tacu Alecsandru, Vancea Romul, *Inteligența artificială, Teorie şi aplicații în economie.* Editura Economică, 1998
6. Tatar Doina, *Inteligența artificială, Demonstrarea automată a teoremelor, Prelucrarea limbajului natural*, Editura Albastră, Cluj-Napoca, 2001
7. Tatar Doina, *Inteligența artificială, Aplicații în prelucrarea limbajului natural,* Editura Albastră, Cluj-Napoca, 2003