# NLP - C3

Analiză sintactică bazată pe constituenți:

- de jos în sus
- de sus în jos

## Strategia top-down

$S \rightarrow NP \cdot VP$

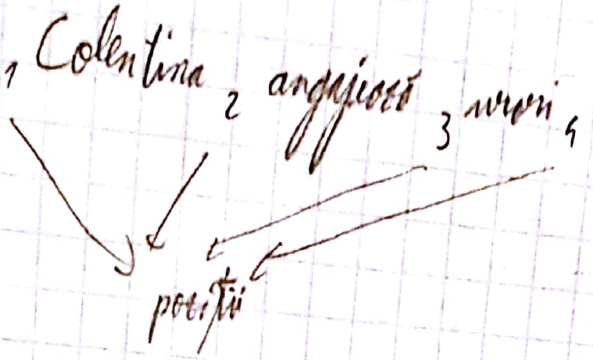
Algoritm de analiză al constituentului C (general).

Pas 1: Dacă C este un cuvânt individual:
este căutat în regulile lexicale
ale gramaticii și acceptat de
primul dispozitiv de intrare.

Pas 2: Altfel:
se caută în regulile gramaticii
pt. a-l rescrie pe C sub
forma unei liste de constituenți
și se analizează acei constituenți
unul câte unul.

7/9
17

ex.:




Stare a analizei = o pereche formată din lista simbolurilor găsite și nr. care marchează poz. curentă în cadrul șirului de intrare.

ex.:

S → NP · VP → nou Det

NP → ART · N

NP → ART · ADJ · N

VP → V

VP → V · NP

labeă : V

câine : N

un : ART

1 Un 2 câine 3 latră 4

~~ex de stare:~~

~~((N, VP), 2)~~ ~~poziția~~

~~stare: găsit până la acel moment.~~

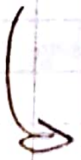
ex. de stare pt. poz 3:

S → NP · VP

((NP · VP), 3)

un câine

Alg. de analiză sintactică top-down:



Input: Starea inițială $((S), 1)$, fără stări back up.

P1: Selectează starea curentă:

- o scoate primul stare din lista de posibilități

Fie lista de posibilități $\Pi$ al lui C.

- dacă lista de posib. este $\emptyset$, at. alg. eșuează și STOP.

P2: Dacă C are o listă de simboluri $\emptyset$, iar conținul indică poziția de la sf. prop., at. alg. are succes și STOP.

P3: Altfel:

- generează următoarele stări posibile
- adaugă aceste stări în lista posibilităților în fct. de modul de organizare al listei (stivă sau coadă).

P3.1.

- dacă primul simbol din lista de simb. $\Pi$ al lui C este lexical, iar urm.

cuv. al șirului de intrare ~~apo~~ aparține acelei categ. lexicale, at. se creează o nouă stare prin înlocuirea primului simbol din lista de simboluri și repoziționarea conținutului de poziții. Starea nou creată se adaugă listei posib.

P 3.2: Altfel:

- se generează o nouă stare, corespunzător fiecărei reguli a gramaticii care poate rescrie acel neterminal.
- se adaugă listei posibilităților noile stări generate.

Ex.: $_1$Un $_2$câine $_3$latră $_4$.

| Pas | Stare analiză cuvânt | Stări backup | Comentarii |
|---|---|---|---|
| 1 | ((S), 1) | – | pâr de început |
| 2 | ((NP · VP), 1) | – | Rescriu S cu regula 1 |
| 3 | ((Art · N · VP), 1) | ((Art · Adj · N · VP), 1) | Se rescrie primul simbol NP cu două reguli |
| | ↓ Art. e simb. lexical | | |
| 4 | ((N, VP), 2) | —u— | Se reduce simb. lexical Art cu "Un" |
| 5 | ((VP), 3) | —u— | Se reduce simbolul N cu "câine" |
| 6 | ((V), 3) | ~~stiva se pune~~ ((V, NP), 3) ((Art · Adj · NP), 1) | Se rescrie VP cu două reguli; - starea nouă se pune sus (considerăm stiva) |
| 7 | ((), 4) | —u— | Se reduce V cu "latră". |
| | ↓ Acceptat | | |
| 8 | STOP | | |

6/9

categorii

Implementare prolog:

1. Dispozitiv de recunoaștere

?- recunoaște (s, ["Colentina", angajează, oameni], []).

Program:

recunoaște (Categ, S1, S2) :-

regula (Categ, Fii),

↳ Lista cu numb. nedeterm.

Imperechere (Fii, S1, S2).

Def.: O listă de categorii se împerechează cu un șir dacă:

1. Ambele sunt vide
2. Ambele conțin același unic cuvânt.
3. O porțiune inițială a șirului poate fi recunoscută ca reprezentând prima categ. din listă, iar restul listei de categorii se împerechează cu restul șirului

```
imperecheaza([],S,S).
imperecheaza([Cuvant],[Cuvant|S],S).
imperecheaza([Categorie|Categorii],S1,S3):-
        :- recunoaste(Categorie,S1,S2),
           imperecheaza(Categorii,S2,S3).
```

Reguli din gramatică:

```
regula(s,[np,vp]).
regula(v,["angajează"]).
```

## 2. Analizator sintactic:

Vă fi trei două predicate:

- parse: analiză sintactică a unui constituent.
- parse_lista: analizează o listă de constituenți

→ corp. lui recunoaște

→ corp. lui imperecheză.

categorie

```
parse(C, [Cuvânt | S], S) :- cuvânt(C, Cuvânt).

parse(C, S1, S) :- regula(C, Cs),
                   parse_lista(Cs, S1, S).
                   (lista de constituenți)

parse_lista([C | Cs], S1, S) :- parse(C, S1, S2),
                                parse_lista(Cs, S2, S).
parse_lista([], S, S).
```

Reguli:

```
regula(s, [np, vp]).
cuvânt(v, angajează).
```

(Un alg. top-down)
Nu ~~poate~~ se poate aplica pe gramatici recursive la stânga (ex.: $A \to A \cdot B$)

$NP \to NP \cdot Adj \cdot NP$

↓

pt. că pot cicla

↓

$2^n$

9/9