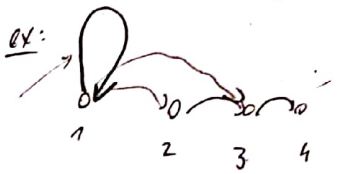
NLP - C5

Harta activă (TSBF):
↳ pt. că se modifică

1. Se permite un singur tip de cicluri, care dau naștere la „arce active vide"

revin în vf. din care au plecat.

ex:



2. Etichetele: sunt date de reguli, scrise ca reguli punctate.

ex.:

arce active (pt. că nu au fost confirmate ipotezele în totalitate).

S → NP . VP — ce e în stânga pct. a fost verificat

S → . NP VP

S → NP VP . — încă nu a fost verificată ipoteza

a fost verificat tot.

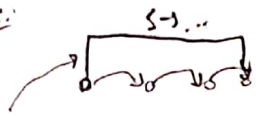
corespunde arcelor active vide.

Harta = struct. de date care conțin dinăuntru o mulțime de arce etichetate, numite „muchii".

muchii: ↳ active: corespunde unei ipoteze de structură.
↳ inactive (f): reprezintă un rezultat, constituent complet.

Analiza cu succes = în hartă apare o muchie inactivă acoperitoare (de tot șirul), etichetată cu regula de structură S→...

ex:



Regula fundamentală:

Analiza sintactică este interacțiunea dintre muchii active și inactive.

Dacă harta conține muchiile de la i, la j, cu eticheta $A \to W_1 . B W_2$

$\langle i, j, A \to W_1 . B W_2 \rangle$

și $\langle j, k, B \to W_3 . \rangle$, (posibil vide)

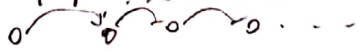
unde A, B sunt categorii, iar W_1, W_2, W_3 reprezintă secvențe de categorii sau cuvinte (posibil vide),

atunci se adaugă hărții muchia $\langle i, k, A \to W_1 B . W_2 \rangle$

Inițializarea hărții:

- Asigurarea existenței unor muchii inactive.
- Se adaugă în hartă cuvintele din șirul de intrare, cu părțile de vorbire corresp.:

ex: N→profesorul.



Regula Bottom-Up:

Dacă se adaugă hărții muchia $\langle i, j, C \rightarrow w_1 . \rangle$,
atunci, pt. fiecare regulă a gramaticii de forma $B \rightarrow C\, W_2$,
se adaugă hărții muchia $\langle i, i, B \rightarrow . C\, W_2 \rangle$

muchie (START, FINAL, ETICHETĂ, DEGĂSIT, GĂSIT)

interogare:

?- parse(vp, [iubește, o, carte]).

categorie — șirul de input

↳ output: [vp, [v, iubește], [np, [det, o], [n, carte]]]

Algoritm:

init_harta (V0, Vn, [])

plecăm de la poziția V0 → ajungi la poz. V0

(în Action: ex: cuvânt(V, substr).

init_harta (V0, Vn, [Cuvânt | Cuvinte]) :- V1 is V0 + 1,

pentru_fiecare (cuvânt (Categorie, Cuvânt),

adaugă_muchie (V0, V1, Categorie, [],

[Cuvânt, Categorii])),

(pentru_fiecare → găsește toate categoriile pt. primul cuvânt)

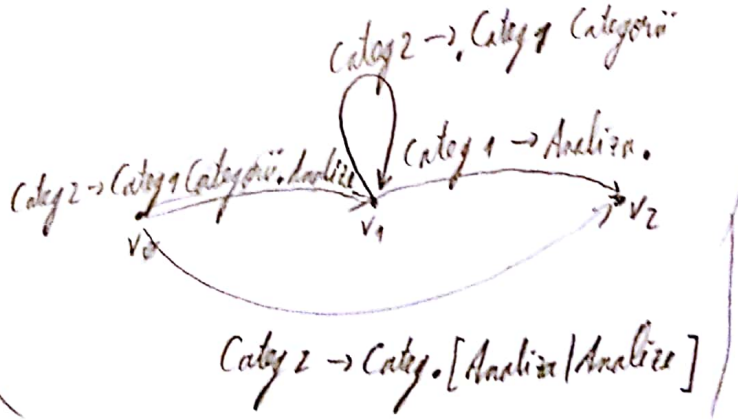
init_harta (V1, Vn, Cuvinte).

(adaugă muchie : → pt. muchii inactive : → regula button-up
→ regula fundamentală
→ pt. muchii active : → doar regula fundamentală)

adaugă_muchii (V0, V1, Categorie, Categorii, Analiza) :-

muchie (V0, V1, Categorie, Categorii, Analiza), !. → verifică dacă muchia deja există

ex: muchii inactive V1 - V2:



Categ 2 → Categ . [Analiza | Analize]

cu muchii inactive.

↳ adaugă_muchie (V1, V2, Categ1, [], Analiza) :-

asserta(muchie(V1, V2, Categ1, [], Analiza),

- predefinit;
- adaugă în baza de cunoștințe.

pentru_fiecare(regula(Categ2, [Categ1 | Categ]), ← Categorii
adaugă_muchie(V1, V1, Categ2, [Categ1 | Categ],
[Categ2])),
pentru_fiecare(muchie(V0, V1, Categ2, [Categorie1 | Categorii], Analize),
adaugă_muchie(V0, V2, Categ2, Categorii, [Analiza | Analize])).

categoria

parse(Cat, Șir) :- V0 is 1,
init_tocta(V0, Vn, Șir),
((\+ muchie(V0, Vn, Cat, [], _),
non
retract_all(muchie(_,_,_,_,_)), !, fail); ← cat aparținând (nu există analiză).
(pentru_fiecare(muchie(V0, Vn, Cat, [], Analiza), ← muchie acoperitoare
write(Analiza)),
retract_all(muchie(_,_,_,_,_)))).

cât în care a fost găsită o analiză