Előző órai bizonyítás folytatása…

## Megszorítási tétel -ra

3 transzformáció: Alap 3→Megszorított 3

1. -mentesítés (2. típusra)
2. Láncmentesítés
3. Hosszredukció

## Hosszredukció

Rossz szabály:

 új nyelvtani jelek.

Szimuláció jó szabályokkal.

 hasonlóan.

## Normálforma tétel -ra

Elegendő megszorított 3-ból indulni.

Megszorított 3:

Normálforma 3:

Rossz szabály:

Szimuláció jó szabállyal.

Bevezetjük: új nyelvtani jel.

Minden szabályt átalakítunk -fé.

Plusz még új szabály.

# Szóprobléma vizsgálata

## Nyelvtanok fontossága

Nyelvekkel feladatokat megoldjunk.

PASCAL nyelven helyes-e egy adott program. Benne van-e a nyelvben.

## Szóprobléma

Adott nyelv, Eldöntendő:

Feltesszük, hogy valamilyen () nyelvtannal van adva.

### Szóprobléma nyelvtanokon

Adott nyelvtan és szó. Eldöntendő:

Létezik levezetése.

(Legfeljebb i lépésben előállítható.)

### 2 állítás

### Eldöntő algoritmus

A megadott rekurzív képlet alapján képezzük -ket! (Végtelen ciklusban)

Kilépés csak extra módon:

Ha , akkor kilépés válasszal.

Ha , akkor kilépés válasszal.

Ha benne van, biztos igen választ ad. Ha nincs benne, vagy nem választ ad vagy végtelen ideig fut. Parciálisan eldöntő algoritmus -re

## TÉTEL:

Parciálisan rekurzív nyelvek családjába tartoznak a nyelvtannal leírható nyelvek.

## TÉTEL: Nem létezik olyan totálisan eldöntő algoritmus, mely tetszőleges és esetén eldönti -t.

Így a -s típus majdnem használhatatlan gyakorlatban. Nézzük meg -et!

## i=1

### Emlékeztető: 1-es típusú nyelvtan

### Emlékeztető vége

 eldöntése egyszerű: Benne van-e a EKS vagy nem.

Ez bizony véges halmaz. Biztosan van olyan , melyre . Az algoritmus mint előbb, csak -t képzünk helyett és garantáltan megáll. Totálisan eldöntő algoritmus.

### TÉTEL:

Ezt láttuk be. (Ez -hez tartozik)

### Az előbbi totálisan eldöntő algoritmusnak mennyi a futási ideje?

###  esetén

Valamilyen konstans\*szabályok száma lineáris

###  esetén

legrosszabb futási idő:

 a lehető legtovább nő. elemszáma

-ban exponenciált.

Olyan algoritmus, ami exponenciális futásidejű, használhatatlan.

## i=2 (CYK algoritmus)

Emlékeztető: 2. típusú normálforma =

Adott és egy szó

Eldöntendő:

###  esetén

Triviális.

###  esetén

-t levezetjük -ból

-t levezetjük -ből

Ötlet:

Tehát:

-kre egy rekurzív előállítást adunk. (Jelölés másképp: )

 növekvő sorrendjében megcsináljuk a további -ket:

Kettévágás:

Szemléletesebb módon:

Ez a CYK algoritmus.

Kitöltése: idő alatt (szabályok száma) alulról fölfelé. Két átlón levőt vizsgálva, mindig csak a legalsókat.

Műveletigénye: Egy elem kiszámolása a következőképp:

 legfeljebb db

 legfeljebb

 legfeljebb

Szabálykeresés legfeljebb

Legrosszabb eset:

Elemek száma:

Teljes költség:

Teljes műveleti igény:

Legrosszabb esetben a CYK algoritmus műveleti igénye

Összefoglalva: -re találtunk eldöntő algoritmust, háromszög mátrix kitöltéséhez kapcsolódik. Műveletigénye a szóhossz köbe. Az input hosszában polinomiális, tehát gyakorlatban is használható.

De ha , azért elég sokáig tartana, mire lefut a CYK ()

Sokkal jobb eredmény -re nincs. Van , de nem elég.

Nagy bemenetre lassú az algoritmus. Vagy áttérünk -ra (legközelebb), vagy megszorításokat teszünk -re úgy, hogy a szó hosszával arányosan fusson le algoritmus.