

Logică și structuri discrete

Gramatici

Marius Minea

marius@cs.upt.ro

<http://www.cs.upt.ro/~marius/curs/lsd/>

9 decembrie 2013

Limbaje care nu sunt regulate

Există limbaje foarte simple care nu sunt regulate:

$\{a^n b^n \mid n \geq 0\}$ paranteze echilibrate

$\{ww \mid w \in \{a, b\}^*\}$ cuvânt, apoi repetat

$\{ww^R \mid w \in \{a, b\}^*\}$ cuvânt, apoi inversat

Limbajele regulate au proprietatea că în fiecare cuvânt suficient de lung există un subșir care poate fi repetat (*pumping lemma*):

$\exists p \in \mathbb{N}$ astfel ca fiecare cuvânt cu $|w| \geq p$ are forma $w = xyz$ cu

$$|y| \geq 1$$

$$|xy| \leq p$$

$$\forall k \geq 0 . xy^kz \in L$$

\Rightarrow așa demonstrăm prin reducere la absurd că o expresie nu e regulată

Gramatică

Un limbaj e descris prin mulțimea *simbolurilor* sale, și prin *regulile* prin care pot fi combinate acele simboluri: *sintaxa*.

Sintaxa unei limbi: modul în care pot fi formate *fraze* cu *cuvintele* limbii (cuvintele sunt aici simbolurile)

O *gramatică* descrie cum se obțin șirurile dintr-un limbaj prin *reguli de producție* (*reguli de rescriere*) pornind de la un *simbol de start*

Exemplu: propoziții limba engleză (mult simplificat)

$S \rightarrow N P$	parte substantivală + predicat
$N \rightarrow subst$	substantiv
$N \rightarrow adj N$	cu adjectiv înainte
$P \rightarrow verb$	predicat: doar verb
$P \rightarrow verb subst$	predicat: cu complement direct

Remarcăm:

simboluri care apar în stânga (sunt înlocuite): *neternale*

simboluri care apar numai în dreapta: *terminale*

Exemple de limbaje

șirurile de paranteze echilibrate

$$S \rightarrow \epsilon$$

$$S \rightarrow (S)S$$

$\{ww^R \mid w \in \{a, b\}^*\}$

$$S \rightarrow \epsilon$$

$$S \rightarrow 0S0$$

$$S \rightarrow 1S1$$

cuvânt, apoi inversat

Ierarhia Chomsky [după Noam Chomsky]

Notăm:

litere mari: neterminale; mici: terminale; grecești: șiruri arbitrare

0) gramatici nerestricționate (orice reguli de rescriere)

limbaje *recursiv enumerabile* (recunoscute de o mașină Turing)

1) gramatici *dependente de context*

reguli: $\alpha A \beta \rightarrow \alpha \gamma \beta$

A rescris doar când e între α și β

$\gamma \neq \epsilon$ (nevid), sau $S \rightarrow \epsilon$ doar dacă S nu apare în dreapta

2) gramatici *independente de context*

reguli: $A \rightarrow \gamma$ stânga: neterminal; dreapta: orice

3) gramatici *regulate*: generează *limbajele regulate*:

reguli de forma $A \rightarrow a$ și $A \rightarrow aB$ (regulate la dreapta)

alternativ: $A \rightarrow a$ și $A \rightarrow Ba$ (regulate la stânga)

dar nu amândouă combinat!

Gramatică formală

O gramatică formală G e formată din:

Σ : o mulțime de simboluri *terminale*

N : o mulțime de simboluri *neternale*, $N \cap \Sigma = \emptyset$

P : o mulțime de *reguli de producție*, de forma

$$(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)$$

un *simbol de start* S

Limbajul definit de G e format din toate șirurile de *terminale* care se pot obține din S aplicând oricâte reguli

Forma Backus-Naur (BNF)

dupa John Backus (dezvoltatorul limbajului FORTRAN)
și Peter Naur (ALGOL 60) (fiecare: premiul *Turing*)

Notăție frecvent folosită pentru gramatici independente de context

folosește ::= pentru definiție și | pentru alternativă

Nonterminal ::= rescriere1 | rescriere2 | ... | rescriereN

uneori folosite cu extensii:

[element-opțional]

simbol* (steaua Kleene) pentru repetiție

simbol⁺ pentru repetiție cel puțin odată

paranteze pentru gruparea elementelor

Exemple: instrucțiuni în C (simplificat)

Stmt ::= ExpStmt | IfStmt | WhileStmt | Block

ExpStmt ::= expr ;

IfStmt ::= if (expr) Stmt else Stmt
 | if (expr) Stmt

WhileStmt ::= while (expr) Stmt

Block ::= { Stmt* }

Problema: cu care **if** se potrivește **else** ?

if (x > 0) **if** (y > 0) x = 0; **else** y = 0;

Gramatica e *ambiguă*: există șiruri cu mai mulți *arbori de derivare* (arbori sintactici)

Dezambiguarea gramaticii

Pentru a dezambigua gramatica, trebuie rescrisă
în acest caz, “echilibrând” ramurile unui `if`

`Stmt ::= BalStmt | UnBallf`

`BalStmt ::= ExpStmt | WhileStmt | Block | Ballf`

`ExpStmt ::= expr ;`

`WhileStmt ::= while (expr) Stmt`

`Block ::= { Stmt* }`

`Ballf ::= if (expr) BalStmt else Stmt`

`UnBallf ::= if (expr) Stmt`

Expresii aritmetice

$E ::= \text{num} \mid E + E \mid E - E \mid E * E \mid E / E \mid (E)$

și aici avem ambiguitate: nu e precizată precedența operatorilor
rescriem pe 3 nivele de precedență:

$E ::= T \mid E + T \mid E - T$

$T ::= F \mid T * F \mid T / F$

$F ::= \text{int} \mid (E)$

Eliminăm și recursivitatea la stânga

(E pe prima poziție în producțiile lui E , la fel T)

$E ::= T \text{ Rest}E$

$\text{Rest}E ::= \epsilon \mid + T \text{ Rest}E \mid - T \text{ Rest}E$

$T ::= F \text{ Rest}T$

$\text{Rest}T ::= \epsilon \mid * F \text{ Rest}T \mid / F \text{ Rest}T$

$F ::= \text{int} \mid (E)$