# Continuare Automate stiva. Analiza descendenta 1

April 9, 2019

# Automat finit - reamintire

Automat finit (finite automaton, finite state acceptor):

$$A = (T, Q, R, q_0, F)$$

- $Q$ set nevid - setul starilor interne
- $(T \cup Q, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $qt \to q'$, $q, q' \in Q, t \in T$

$$L(A) = \{\tau \in T^* | q_0\tau \Rightarrow^* q, q \in F\}$$

# Automat stiva - definitie sistem de rescriere -reamintire

Automat stiva

$$A = (T, Q, R, q_0, F, S, s_0)$$

, unde:

- $Q$ set nevid - setul starilor interne
- $(T \cup Q \cup S, R)$ sistem de rescriere; $T \cap Q = \emptyset$
- $q_0 \in Q$ - starea initiala
- $s_0 \in S \cup \{\varepsilon\}$ - simboluri stiva, $s_0$ continutul initial al stivei
- $F \subseteq Q$ - stari finale
- fiecare element din R are forma $\sigma q t \tau \rightarrow \sigma' q' \tau$, $\sigma, \sigma' \in S^*,\ q, q' \in Q, t \in T \cup \varepsilon, \tau \in T^*$

Daca automatul e la configuratia $s_1 ... s_n q \tau$ intr-o derivare, automatul e in starea $q$, $\tau$ este partea necitita din input, $s_1, ..., s_n$ este continutul pe stiva, $s_n$ in varf.

# Limbaj acceptat

Daca automatul e la configuratia $s_1 \ldots s_n q \tau$ intr-o derivare, automatul e in starea $q$, $\tau$ este partea necitita din input, $s_1, \ldots, s_n$ este continutul pe stiva, $s_n$ in varf.

$$L(A) = \{\tau | s_0 q_0 \tau \Rightarrow^* q, q \in F, \tau \in T^*\}$$

# CFG - PDA

Pentru fiecare gramatica independenta de context $G$ exista un automat stiva $A$ a.i. L(A)=L(G).

# Parsing

- Rolul parsarii: reconstruirea derivarii prin care o CFG poate genera un input string dat.
- Echivalent cu construirea arborelui de parsare care reprezinat derivarea
- Directii:
  - Top-Down - constructia incepe de la radacina; derivarea stanga
  - Bottom-up - constructia incepe de la frunze; mai greu de construit manual, dar pot fi generate

# Recursive descent - top down parse

Arborele de parsare e construit:

- de la simbolul de start
- de la stanga la dreapta
- se incearca regulile in ordinea in care apar
- revenire si incercare alternative

$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

Parse tree pt: ( int )

↑

E

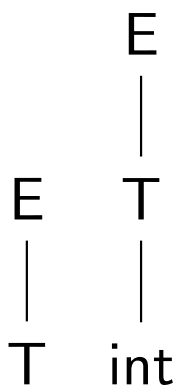
|

T

$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

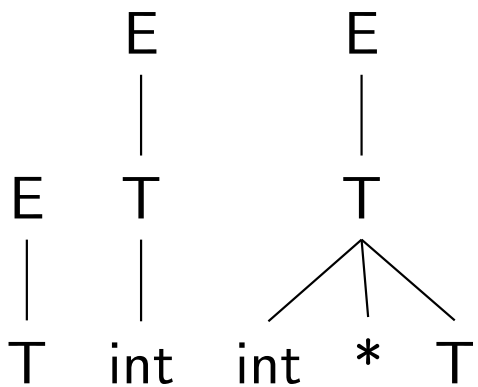
Parse tree pt: ( int )



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

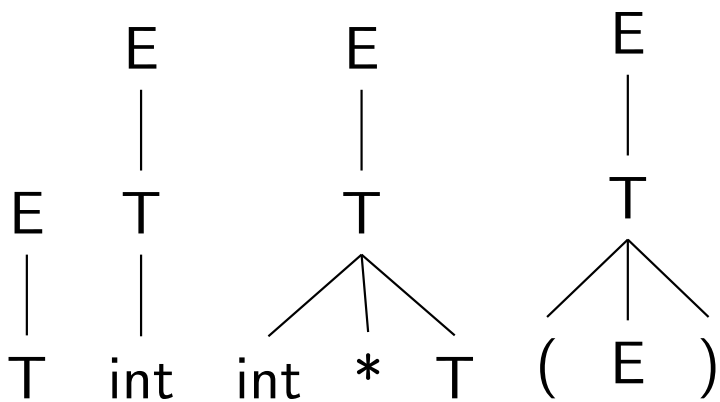
Parse tree pt: ( int )



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

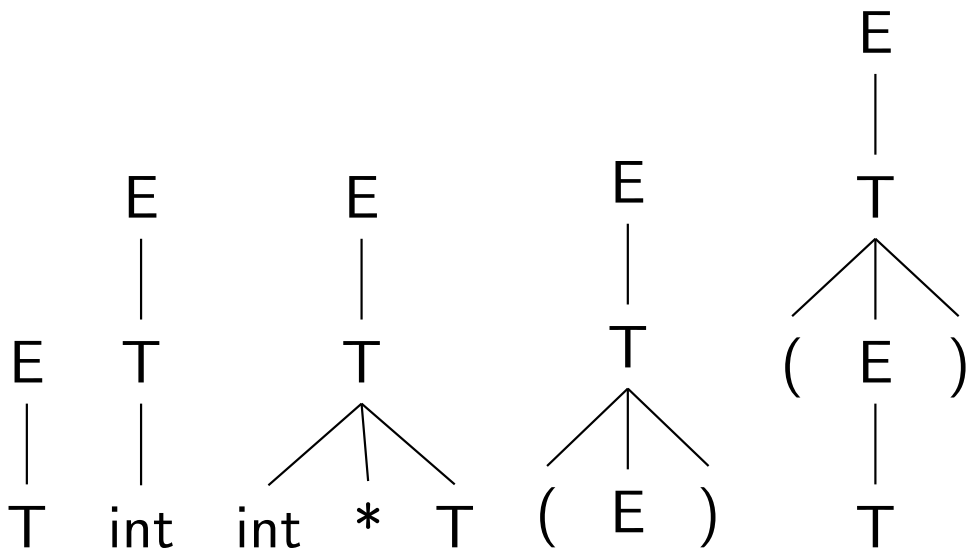
Parse tree pt: ( int )



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

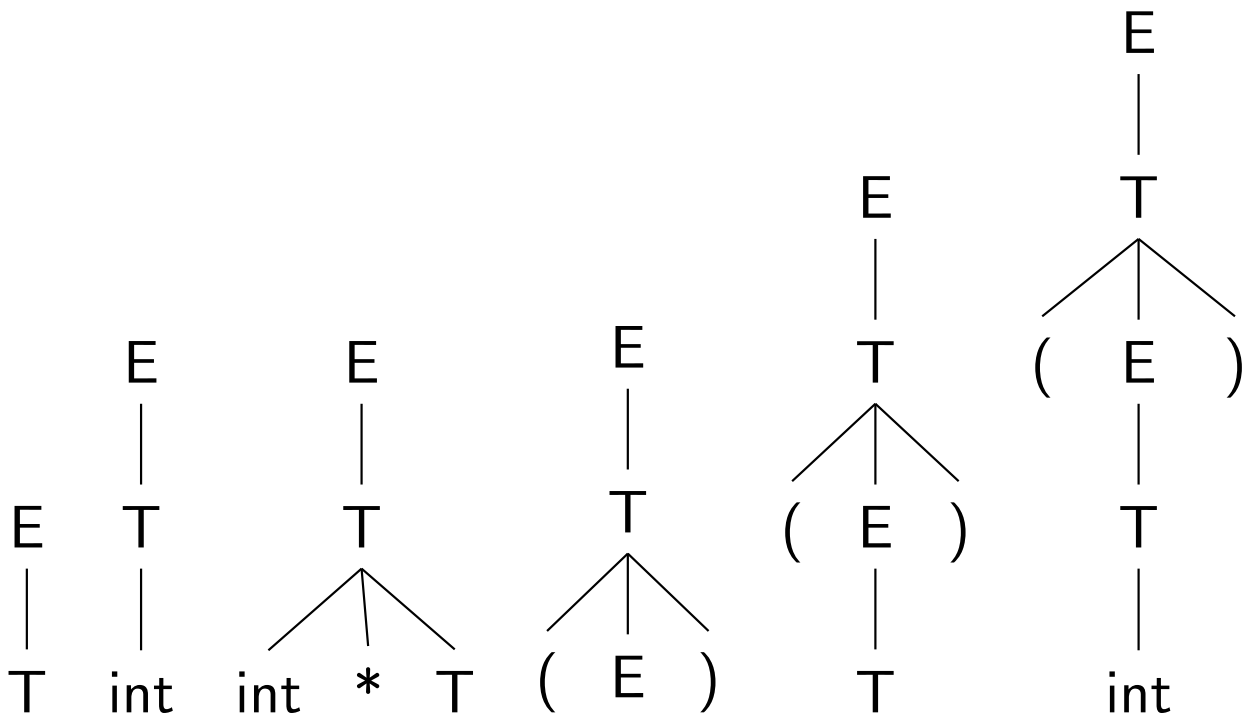
Parse tree pt: ( int ) ↑



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

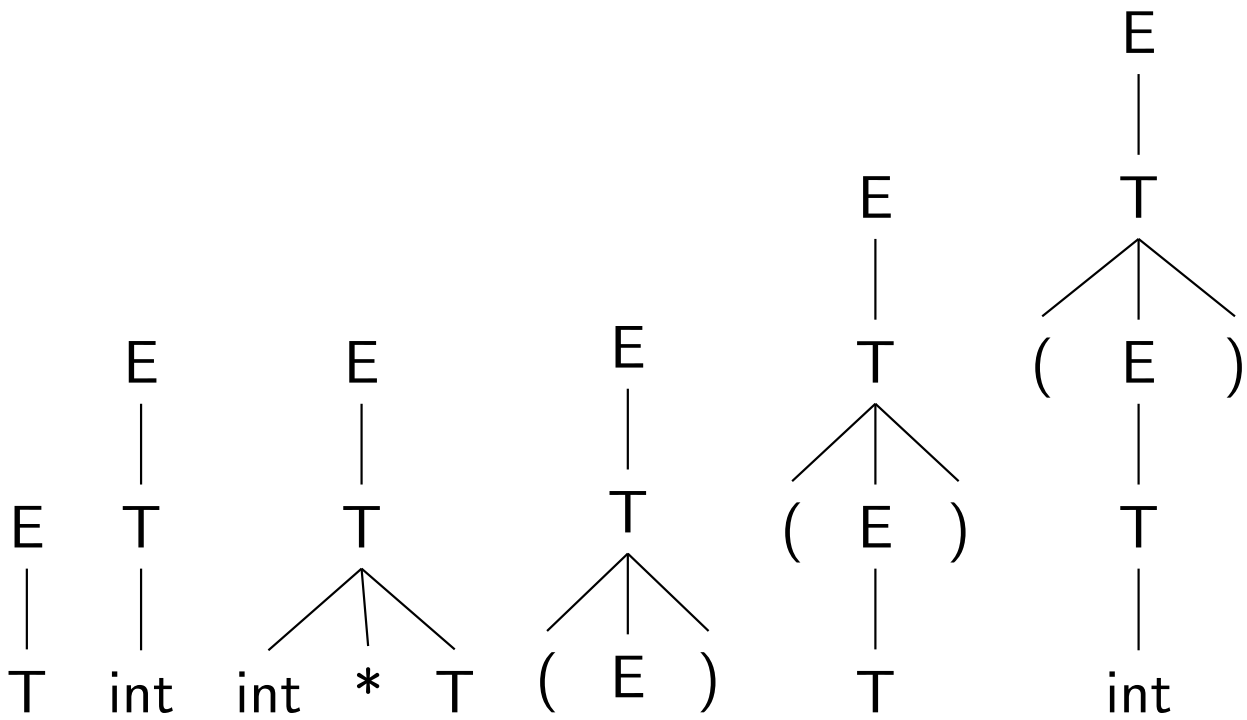
Parse tree pt: ( int ) ↑



$E \rightarrow T | T + E$

$T \rightarrow int | int * T | (E)$

Parse tree pt: ( int )



Recursive-descent parser - nestiind care dintre productiile alternative pt un nonterminal trebuie aplicata, exista posibilitatea de esec

- Predictive parser: dat fiind sirul de intrare $a$ (primul din sirul ramas) si nonterminalul $A$ care trebuie expandat, am putea determina care productie alternativa e cea care deriveaza stringul ramas dupa $a$
  idee: alternativa corecta trebuie detectata uitandu-ne inainte la $k$ simboluri din stringul care trebuie derivate: $LL(1)$ si $LR(1)$

Fie gramatica G=$(\{i, (, +, )\}, \{S, E\}, P, \{S\}$ cu productiile

- $S \to E$
- $E \to (E + E)$
- $E \to i$

Derivare stanga $((i + i) + i)$

$$S \Rightarrow E \to ?$$

# Analiza descendenta. Predictive parser. Gramatici LL(K)

Letia& Chifu 4.2
Fie $G = (T, N, P, Z)$ o CFG si automatul stiva
$A = (T, \{q\}, R, q, \{q\}, V, Z)$ cu $V = T \cup N$ si $R$:
(alfabet, stari, productii, stare initiala, stari finale, alfabet stiva, continut initial stiva)

$\{tqt \rightarrow q | t \in T\} \cup$
$\{Xq \rightarrow x_n....x_1 q | X \rightarrow x_1 x_2 ... x_n \in P, n \geq 0, X \in N, X_i \in V\}$

Automatul accepta un sir din L(G) prin

- construirea unei derivari cea mai din stanga a acelui sir si
- compararea simbolurilor generate (de la stanga la dreapta) cu simbolurile care apar in sir.

# exemplu 1

Fie $G_1 = (T, N, S, P)$

- $T = \{+, (, ), i\}$, $N = \{S, F\}$
- cu productiile P
  - $S \to F$
  - $S \to (S + F)$
  - $F \to i$

Care e automatul pentru analiza descendenta?
Care e derivarea stanga pentru $(i + i)$?
Automatul accepta $(i + i)$?

# exemplu 2

Fie $G_1 = (T, N, E, P)$

- $T = \{+, *, (, ), i\}$, $N = \{E, T, F\}$
- cu productiile P
  - $(1, 2) E \to T | E + T$
  - $(3, 4) T \to F | T * F$
  - $(5, 6) F \to i | (E)$

Automatul stiva:

- $T = \{+, *, (, ), i\}$, $Q = \{q\}$,
  $q_0 = q, F = \{q\}, S = \{+, -, *, (, ), i, E, T, F\}, s_0 = E$
- cu productiile R
  1. $Eq \to Tq, Eq \to T + Eq$,
  2. $Tq \to Fq, Tq \to F * Tq$,
  3. $Fq \to iq, Fq \to )E(q$,
  4. $+q+ \to q, *q* \to q$, $(q(\to q, )q) \to q, iqi \to q\}$

# Derivarea gasita: i+i*i

| stiva | stare | intrare | derivarea cea mai din stanga |
|---|---|---|---|
| E | q | $i + i * i$ | E |
| T+E | q | $i + i * i$ | E+T |
| T+T | q | $i + i * i$ | T+T |
| T+F | q | $i + i * i$ | F+T |
| T+i | q | $i + i * i$ | i+T |
| T+ | q | $+i * i$ | |
| T | q | $i * i$ | |
| F*T | q | $i * i$ | i+T*F |
| F*F | q | $i * i$ | i+F*F |
| F*i | q | $i * i$ | i+i*F |
| F* | q | $*i$ | |
| F | q | $i$ | |
| i | q | $i$ | i+i*i |
| | q | | |

# No backtracking

Analiza descendenta sau predictiva - traseaza derivarea de la simbolul de start la propozitie, prezicand simbolurile care trebuie sa fie prezente.

- stiva precizeaza sirul din $V^*$ utilizat pentru derivarea restului sirului de la intrare
- automat stiva determinist: pentru gramatici $LL(k)$

# Asumptii si structuri ajutatoare: CFG

Presupunem ca CFG $(T, N, P, Z)$ contin

- $Z \to S$ singura in care apare Z
- fiecare propozitie se termina cu $\#$ - indica finalul propozitiei
- productia $i$ are forma

$$X_i \to \chi_i, \ \textit{unde} \ \chi_i = x_{i,1}x_{i,2}...x_{i,m}$$

- $k:\omega$ primele $min(k, |\omega|+1)$ simboluri din $\omega\#$

$$k:\omega = \begin{cases} \omega\#, & daca\ |\omega| < k \\ \alpha, & daca\ \omega = \alpha\gamma\ si\ |\alpha| = k \end{cases}$$

- $FIRST_k(\omega)$ setul tuturor capetelor $k:\omega$ terminale ale sirurilor derivabile din $\omega$

$$FIRST_k(\omega) = \{\tau | \exists \nu \in T^*\ a.i.\ \omega \Rightarrow^* \nu, \tau = k:\nu\}$$

- $EFF_k(\omega)$ ($\varepsilon - free\ first$, primul fara $\varepsilon$) - toate sirurile din $FIRST_k(\omega)$ pentru care nu s-a aplicat nicio productie $\varepsilon$ in ultimul pas din derivarea cea mai din dreapta

$$EFF_k(\omega) = \{\tau \in FIRST_k(\omega) | \nexists A \in N, \nu \in T^*\ a.i.\ \omega \Rightarrow^R A\tau\nu \Rightarrow \tau\nu\}$$

- $FOLLOW(\omega)$ captele $k$ care ar putea urma lui $\omega$; $FOLLOW_k(Z) = \{\#\}$

$$FOLLOW_k(\omega) = \{\tau | \exists \nu \in V^*\ a.i.\ Z \Rightarrow^* \mu\omega\nu, \tau \in FIRST_k(\nu)\}$$

# Exemplu de valori FIRST, FOLLOW pt $k = 1$

- $T = \{id, *, +, (, )\}$, $N = \{E, E', T, T', F\}$
- cu productiile P
  - $Z \to E$
  - $E \to TE'$
  - $E' \to +TE'|\varepsilon$
  - $T \to FT'$
  - $T' \to *FT'|\varepsilon$
  - $F \to (E)|id$

| simbol | $FIRST_1(X)$ | $FOLLOW_1(X)$ |
|---|---|---|
| $E$ | $\{(, id\}$ | $\{), \#\}$ |
| $E'$ | $\{+, \varepsilon\}$ | $\{), \#\}$ |
| $T$ | $\{(, id\}$ | $\{+, \#, )\}$ |
| $T'$ | $\{*, \varepsilon\}$ | $\{+, \#, )\}$ |
| $F$ | $\{(, id\}$ | $\{*, +, \#, )\}$ |

Exemplu

$$E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow (E)T'E' \Rightarrow^{+} (id) * FT'E' \Rightarrow$$

$$(id) * F * T' + TE' \Rightarrow (id) * id * id + id$$

# Exemplu de valori FIRST, FOLLOW pt $k = 1$

- $T = \{id, *, +, (, )\}$, $N = \{E, E', T, T', F\}$
- cu productiile P
  - $Z \rightarrow E$
  - $E \rightarrow TE'$
  - $E' \rightarrow +TE'|\varepsilon$
  - $T \rightarrow FT'$
  - $T' \rightarrow *FT'|\varepsilon$
  - $F \rightarrow (E)|id$

| simbol | $FIRST_1(X)$ | $FOLLOW_1(X)$ |
|---|---|---|
| $E$ | $\{(, id\}$ | $\{), \#\}$ |
| $E'$ | $\{+, \varepsilon\}$ | $\{), \#\}$ |
| $T$ | $\{(, id\}$ | $\{+, \#, )\}$ |
| $T'$ | $\{*, \varepsilon\}$ | $\{+, \#, )\}$ |
| $F$ | $\{(, id\}$ | $\{*, +, \#, )\}$ |

Exemplu

$$E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow (E)T'E' \Rightarrow^{+} (id) * FT'E' \Rightarrow$$

$$(id) * F * T' + TE' \Rightarrow (id) * id * id + id$$

# Intuitie gramatici LL(k)

Fie gramatica G=$(\{i, (, +, )\}, \{Z, E, F\}, P, Z)$ cu productiile

- $Z \to E$
- $E \to F$
- $E \to (E + F)$
- $F \to i$

Care e derivarea pt (i+i)? e de ajuns un lookahead de 1?

# Gramatici $LL(k)$

O gramatica independenta de context $G = (T, N, P, Z)$ este $LL(k)$ pentru un $k \geq 0$ daca pentru derivari arbitrare

$$Z \Rightarrow^L \mu X \chi \Rightarrow \mu \nu \chi \Rightarrow^* \mu \gamma$$

$$Z \Rightarrow^L \mu X \chi \Rightarrow \mu \omega \chi \Rightarrow^* \mu \gamma'$$

$$\textit{unde } \mu, \gamma, \gamma' \in T^*, \nu, \chi, \omega \in V^*, X \in N$$

avem urmatoarea proprietate: $k : \gamma = k : \gamma'$ implica $\nu = \omega$

Fie gramatica G=$(\{i,(,+,)\},\{Z,E,F\},P,Z)$ cu productiile

- $Z \to E$
- $E \to F$
- $E \to (E+F)$
- $F \to i$

Care e derivarea pt (i+i)?

$Z \Rightarrow E \Rightarrow \quad (E+F) \quad \Rightarrow (F+F) \Rightarrow^* \quad (i+i)$

$Z \Rightarrow E \Rightarrow \quad (E+F) \quad \Rightarrow ((E+F)+F) \Rightarrow^* \quad ((i+i)+i)$

$Z \Rightarrow^L \quad \mu X \chi \quad \Rightarrow \mu\nu\chi \Rightarrow^* \quad \mu\gamma$

$Z \Rightarrow^L \quad \mu X \chi \quad \Rightarrow \mu\omega\chi \Rightarrow^* \quad \mu\gamma'$

*unde* $\mu,\gamma,\gamma' \in T^*, \nu,\chi,\omega \in V^*, X \in N$

# Gramatica LL(3)

Fie $G_1 = (T, N, E, P)$

- $T = \{a, b, c\}$, $N = \{Z, X, Y\}$
- cu productiile P
  - (1) $Z \to X$
  - (2,3) $X \to Y|bYa$
  - (4,5) $Y \to c|ca$
- Care sunt stringurile generate impreuna cu derivarea lor?
- Care e derivarea pt *bcaa* considerand si lookahead?
  $Z \Rightarrow X \Rightarrow ?$
- Care e automatul pt analiza descendenta? E determinist?

# Rezumat

Recursive descent parsing

Predictive parsing

FIRST, FOLLOW

Intro LL(k) grammars

# kahoot

Ce e adevarat pentru automatul stiva $A$ construit pt analiza descendenta a G?
$G = (\{a, b\}, \{S\}, \{S \to ab|aSb\}, S)$,

1. stiva contine initial a
2. are o singura stare
3. stiva contine initial S
4. vocabularul stivei este $\{a, b, S\}$
5. $bqa \to q$ este o productie
6. $Sq \to bSaq$ este o productie

Automatul stiva:

- $T = \{0, 1\}$, $Q = \{q\}$, $q_0 = q$, $F = \{q\}, S = \{0, 1, S\}, s_0 = S$
- cu productiile R
  1. $Sq \to 1S0q$
  2. $1q1 \to q$,
  3. $0q0 \to q$,
  4. $Sq \to q$,