

LR(1) parsing - canonical LR

$$\text{eg. } S \rightarrow CC \quad 0$$

$$C \rightarrow cC \quad 2 \Rightarrow \text{augmented grammar}$$

$$C \rightarrow d \quad 3$$

$$S' \rightarrow S \quad 0$$

$$S \rightarrow CC \quad 1$$

$$C \rightarrow cC \quad 2$$

$$C \rightarrow d \quad 3$$

2. găsim LR(1) items

$I_0 : S \rightarrow .S, \$ \rightarrow \$$ după prima prod (se numește lookahead)

$$S \rightarrow .CC, \$$$

$$C \rightarrow .cC, cld$$

$$C \rightarrow .d, cld$$

goto (I_0, S)

$$I_1 : S' \rightarrow S., \$$$

goto (I_0, C)

$$I_2 : S \rightarrow C.C, \$$$

$$C \rightarrow .CC, \$$$

$$C \rightarrow .d, \$$$

goto (I_0, C) ←

$$I_3 : C \rightarrow c.C, cld$$

$$(C \rightarrow .cC, cld) \rightarrow I_3$$

$$(C \rightarrow .d, cld) \rightarrow I_4$$

goto (I_0, d)

$$I_4 : C \rightarrow d., cld$$

goto (I_2, C)

$$I_5 : S \rightarrow CC., \$$$

goto (I_2, C)

$$I_6 : C \rightarrow c., C, \$$$

$$(C \rightarrow .cC, \$) \rightarrow I_6$$

$$(C \rightarrow .d, \$) \rightarrow I_7$$

goto (I_2, d)

$$I_7 : C \rightarrow d., \$$$

goto (I_3, C)

$$I_8 : C \rightarrow CC., cld$$

goto (I_6, C)

$$I_9 : C \rightarrow CC., \$$$

pentru lookahead: Be secol cumva dim

A → a, B(B, a) causa că este. în
B → .Y, FIRST(B a) păc lui într

- dacă Y este B luăm FIRST de la el
- dacă nu, copiem de la prod. din sus
- când facem goto și trecem . peste, păstrăm lookahead la fel

2. LR(1) parsing table

STATE	ACTION				GOTO	
	c	d	\$	S	C	
0	S3	S4		1	2	
1						shift
2	S6	S7				5
3	S3	S4				8
4	r3	r3				
5				r1		
6	S6	S7				9
7				r3		
8	r2	r2				
9				r2		

→ pt. shift: verificăm terminal

ex: în I_0 avem $C \rightarrow .cC, cld \rightarrow$ punem 13 în linia 0 coloana c
rezolvat în I_3

→ pt. reduce: verificăm la final producție

excepție: în $I_1 : S' \rightarrow S. \Rightarrow (1, \$) = \text{accept}$

ex: în I_4 avem $C \rightarrow d., cld$ lookahead
→ scriem în linia 4 și coloana c, d reduc 3
 \rightarrow scriem în linia 4 și coloana c, d reduc 3 \rightarrow $C \rightarrow d$ e a 3-a regulă

→ pt. goto: verificăm non-terminal

ex: în I_0 avem $S' \rightarrow S. \Rightarrow$ pe linia 0, coloana S scriem 1

rezolvat în I_1

3. how to parse the input string using the table

→ înainte să ajungem la asta vedem de ex că I_3 și I_6 sunt st. asemănătoare, dar au luka read diferit ⇒ facem LALR parsing table

I_{36} : $C \rightarrow c \cdot C, c \mid d \mid \$ \rightarrow$ doar combinații de la cele două

$C \rightarrow \cdot c C, c \mid d \mid \$$

$C \rightarrow \cdot d, c \mid d \mid \$$

I_{47} : $C \rightarrow d \cdot, c \mid d \mid \$$

I_{89} : $C \rightarrow \cdot c C, c \mid d \mid \$$

→ acum, unde aveam de ex. $S_3 \Rightarrow S_36$

→ ! la reduce nu, că se referă la nr. prod!

STATE	ACTION			GOTO	
	c	d	\$	S	C
0	S_36	S_{47}		1	2
1				accept	
2	S_36	S_{47}			5
36	S_36	S_{47}			89
47	π_3	π_3	π_3		
5				π_1	
89	π_2	π_2	π_2		

am combinat ce era la 3 cu
ce era la 6

ANALIZA DE SECVENTĂ EFIX LA FEL CA LA RESTUL

LR(0) parsing

e.g. $S \rightarrow AA \quad ①$
 $A \rightarrow aA \quad ② \Rightarrow$ augmented grammar:
 $A \rightarrow b \quad ③$
 \downarrow context free, fără ambiguitate

$$S' \rightarrow S$$

$$S \rightarrow AA$$

$$A \rightarrow aA$$

$$A \rightarrow b$$

1. canonical collection

$$\begin{aligned} I_0 : & S' \rightarrow S \\ & S \rightarrow AA \\ & A \rightarrow aA \\ & A \rightarrow b \end{aligned}$$

$$gto_0(I_0, S)$$

$$I_1 : S' \rightarrow S.$$

$$gto_1(I_0, A)$$

$$I_2 : S \rightarrow A.A$$

$$\begin{array}{l} A \rightarrow aA \rightarrow I_3 \\ A \rightarrow b \rightarrow I_4 \end{array}$$

$$gto_2(I_0, a)$$

$$\begin{array}{l} I_3 : A \rightarrow a.A \\ A \rightarrow aA \rightarrow I_3 \\ A \rightarrow b \rightarrow I_4 \end{array}$$

$$gto_3(I_0, b)$$

$$I_4 : A \rightarrow b.$$

$$gto_4(I_2, A)$$

$$I_5 : S \rightarrow AA.$$

$$gto_5(I_3, A)$$

$$I_6 : A \rightarrow aA.$$

2. LR(0) parsing table

state	action			goto	
	a	b	\$	A	S
0	S_3	S_4		2	1
1			accept		
2	S_3	S_4		5	
3	S_3	S_4		6	
4	r_3	r_3	r_3		
5	r_1	r_1	r_1		
6	r_2	r_2	r_2		

colectiv ($I_0 - I_6$)

→ pt shift: verificăm urmat de terminal

ex: în I_0 avem $A \rightarrow .aA \Rightarrow$ în linia 0 coloana a scriem S_3 (în I_3 am tratat cazul)

→ pt. goto: verificăm urmat de metterminal

ex: în I_0 avem $S \rightarrow .AA \Rightarrow$ pe linia 0 coloana A scriem 2

↓
tratăm I_2

pt. punct la final (simbol start): $S' \rightarrow S.$ → scriem pe linia 0 coloana $\$$ accept
↓
în I_1

→ pt. reduc: verificăm la final

avem nevoie de numărarea producțiilor pt. că A $\rightarrow b$ este a 3-a prod din lista

ex: în I_4 avem $A \rightarrow b. \Rightarrow$ la 4 scriem r_3 la toate coloanele

3. how to parse the input string using the table: aabb

STACK	INPUT	ACTION	STACK	INPUT	ACTION
\$0	aabb\$	$(0, a)$	$\$0A2b4$	\$	$r_3 \quad A \rightarrow b$
\$0a3	abb\$	S_3	$\$0A2A5$	\$	$r_1 \quad S \rightarrow AA$
\$0a3a3	bb\$	S_4 reduce by rule 3	$\$0S1$	\$	accept
\$0a3a3b4	b\$	$r_3 \quad A \rightarrow b$			
\$0a3a3A6	b\$	$r_9 \quad A \rightarrow aA$			
\$0a3A6	b\$	$r_2 \quad A \rightarrow aa$			
\$0A2	b\$	$r_2 \quad A \rightarrow aa$			
	b\$	S_4			

dacă avem nevoie de simbol de producție: 13223

luăm cele de la reduc învers

SLR (simple LR parsing)

1. avem gramatica \Rightarrow augmented grammar

$E \rightarrow E + T$	①	$E' \rightarrow E$ \rightarrow deoarece pt. simbolul de start
$E \rightarrow T$	②	$E \rightarrow E + T$
$T \rightarrow T * F$	③	$E \rightarrow T$
$T \rightarrow F$	④	$T \rightarrow T * F$
$F \rightarrow (E)$	⑤	$T \rightarrow F$
$F \rightarrow id$	⑥	$F \rightarrow (E)$
		$F \rightarrow id$

2. gasim canonical LR(0) collection

- avem nevoie de două funcții: closură și goto

$I_0 : E' \rightarrow .E$ closură (scriem prima prod., după vedem dacă avem neterm. după ., dacă da,
îi scriem și lui toate producțiile și tot așa)

$E \rightarrow .E + T$ avem încercări să aducem la finalul producției (goto)
 $E \rightarrow .T$

$T \rightarrow .T * F$

$T \rightarrow .F$

$F \rightarrow .(E)$

$F \rightarrow .id$

punctul a trebat peste E

goto (I_0, E) \rightarrow la final \Rightarrow strict pt producția asta nu mai avem goto

$I_1 : E' \rightarrow E.$ în continuare verificăm dacă avem closură (aici nu avem)

$E \rightarrow E . + T$

goto ($I_0, ()$)

$I_4 : T \rightarrow (.E)$

$E \rightarrow .E + T$

$E \rightarrow .T$

$T \rightarrow .T * F$

$T \rightarrow .F$

$F \rightarrow .(E)$

$F \rightarrow .id$

goto ($I_1, +$)

$I_6 : E \rightarrow E . + T$

$T \rightarrow .T * F$

$T \rightarrow .F$

$F \rightarrow .(E)$

$F \rightarrow .id$

goto (I_6, T)

$I_9 : E \rightarrow E . T$

$T \rightarrow .T * F$

goto (I_{*}, F)

$I_{10} : T \rightarrow T . * F$

goto ($I_8,)$)

$I_{11} : F \rightarrow (E).$

goto (I_0, T)

$I_2 : E \rightarrow T.$

$T \rightarrow T . * F$

$E \rightarrow .E + T$

$E \rightarrow .T$

$T \rightarrow .T * F$

$T \rightarrow .F$

$F \rightarrow .(E)$

$F \rightarrow .id$

goto (I_0, id)

$I_5 : F \rightarrow id.$

goto (I_4, E)

$I_8 : F \rightarrow (E.)$

$E \rightarrow E . + T \rightarrow I_6$

3. SLR parsing table

state	action								goto
	+	*	()	id	\$	E	T	
0			S_4		S_5		1	2	3
1	S_6				acc				
2	r_2	S_*		r_2		r_2			
3	r_4	r_4		r_4		r_4			
4			S_4		S_5		8	2	3
5	r_6	r_6		r_6		r_6			
6			S_4		S_5		9	3	
*			S_4		S_5				10
8	S_6			S_{11}					
9	r_1	S_*		r_1		r_1			
10	r_3	r_3		r_3		r_3			
11	r_5	r_5		r_5		r_5			

↑
colectivile pe care le avem

terminal + \$ non-terminal

→ pt. shift verificăm în colecții când este. urm.
dă terminal

ex: în I_0 avem $F \rightarrow .(E)$
vedem unde a trecut o parte (\Rightarrow în I_4)
 \Rightarrow pe linia 0 și coloana (scriem S_4)

→ pt. completarea goto verificăm. urmat de metterm
ex: în I_0 avem $E^1 \rightarrow .E$ și $E \rightarrow .E + T$
caz tratat în I_1
 \Rightarrow pe linia 0 și coloana E scriem 1

→ pt. reduce verificăm. la final producție
ex: în I_2 avem $E \rightarrow T$. $E \rightarrow T$ este a 2-a reg. dim.
 \Rightarrow la 2 scriem regular r_2 în colecțile
follow (E) $\Rightarrow (2, \$), (2, +), (2,)$
excepție $E^1 \rightarrow E$.

găsit în $I_1 \Rightarrow$ linial și coloana $\$$ accept!

pentru reduce me trebuie FOLLOW de meterminală

$$\text{FOLLOW}(E) = \{ \$, +,) \}$$

$$\text{FOLLOW}(T) = \{ \$, +,), * \}$$

$$\text{FOLLOW}(F) = \{ \$, +,), * \}$$

4. how to parse the input string using the table : id + id + id , regulile : 1 $E \rightarrow E + T$ 5 $F \rightarrow (E)$
2 $E \rightarrow T$ 6 $F \rightarrow id$
3 $T \rightarrow T * F$
4 $T \rightarrow F$

STACK	INPUT	ACTION
0	$id + id + id \$$	shift 5 $\rightarrow (0, id)$
$0 id 5$	$* id + id \$$	reduce 6 \rightarrow reduce by rule 6: $F \rightarrow id$ ($1 \times 2 = 2$) \Rightarrow avem un simbol în dreapta
$0 F 3$	$* id + id \$$	reduce 4 $\rightarrow T \rightarrow F$
$0 T 2$	$* id + id \$$	shift 7
$0 T 2 * T$	$id + id \$$	shift 5
$0 T 2 * T id 5$	$+ id \$$	reduce 6 $\rightarrow F \rightarrow id$
$0 T 2 * T id 5$	$+ id \$$	reduce 3 $\rightarrow T \rightarrow T * F$ ($3 \times 2 = 6$ simbol.pop)
$0 T 2 * T F 10$	$+ id \$$	reduce 2 $\rightarrow F \rightarrow T$
$0 T 2$	$+ id \$$	shift 6
$0 E 1$	$+ id \$$	shift 5
$0 E 1 + 6$	$+ id \$$	reduce 6 $\rightarrow F \rightarrow id$
$0 E 1 + 6 id 5$	$+ id \$$	reduce 4 $\rightarrow T \rightarrow F$
$0 E 1 + 6 F 3$	$+ id \$$	reduce 1 $\rightarrow E \rightarrow E + T$ (6 simbol.pop)
$0 E 1 + 6 T 9$	$+ id \$$	accept
$0 E 1$	$+ id \$$	da că refacem un camp gel era errony

→ dacă refacem un camp gel era errony
→ dacă în tabel aveam mai mult de 2 val. per
celulă, nu era SLR(1) grammar

LL(1)

$$\begin{array}{l} \text{g. } E \rightarrow E + T \mid T \\ \quad T \rightarrow T * F \mid F \\ \quad F \rightarrow (E) \mid \text{id} \end{array}$$

1. eliminate left recursion

$$\begin{array}{l} E \rightarrow T E' \quad \text{(1)} \\ E' \rightarrow + T E' \mid E \quad \text{(2)} \end{array} \quad \left. \begin{array}{l} \text{scriem cele mărcurisire la st. următoare de un nou neterm. apoi în celelalte punem} \\ \text{partea mărcurisirii următoare de nou neterm } E \end{array} \right\}$$

$$T \rightarrow F T' \quad (4)$$

$$T' \rightarrow * F T' \mid E \quad (5) \quad (6)$$

$$F \rightarrow (E) \mid \text{id}$$

\Rightarrow în acest caz nu avem de făcut factorizări la dângă ceea ce avem prod. să înceapă fix la fel
Dacă vedeți un ex: $\boxed{E \rightarrow T + F}$ $\Rightarrow E \rightarrow T P$ luăm partea comună și facem un nou neterm.
 $E \rightarrow T$ $P \rightarrow + F$ cu ea

2. facem first și follow

AM MAI SCRIS REGULILE ÎN SEMINAR II

	FIRST	FOLLOW
E	(, id	\$,)
E'	+ , E	\$,)
T'	* , E	+ , \$,)
T	(, id	+ , \$,)
F	(, id	* , + , \$,)

3. predicting parsing table

	+	*	()	id	\$
E			(E \rightarrow TE', 1)		(E \rightarrow TE', 1)	
E'	(E' \rightarrow + TE', 2)			(E' \rightarrow E, 3)		(E' \rightarrow E, 3)
T			(T \rightarrow FT', 4)		(T \rightarrow FT', 4)	
T'	(T' \rightarrow E, 6)	(T' \rightarrow * FT', 5)		(T' \rightarrow E, 6)		(T' \rightarrow E, 6)
F			(F \rightarrow (E), 7)		(F \rightarrow id, 8)	
+	pop					
*		pop				
(pop			
)				pop		
id					pop	
\$						ACCEPT

term. și \$

1. L \rightarrow S pe linia lui L și coloana lui FIRST(S) unde S este primul număr din producție (first(neterm) = neterm)

2. L \rightarrow E pe linia lui L și coloana lui FOLLOW(L)

3. pop la ~~perechi~~ perechi astăzi term.

4. accept la (\$, \$)

5. erori oriunde altundeva

4. parsing the input string using this table : $\text{id} \# \text{id} + \text{id}$

$(\text{id} * \text{id} + \text{id} \$, E \$, \epsilon) \xrightarrow{\text{push 1}} (\text{id} * \text{id} + \text{id} \$, T E' \$, 1) \xrightarrow{\text{push 4}} (\text{id} * \text{id} + \text{id} \$, F T' E' \$, 14)$

$$\ddot{\sigma} \cdot ca(E, id) = (E \rightarrow TE', 1)$$

$\xrightarrow{\text{push 8}} (\text{id} * \text{id} + \text{id} \$, id T' E' \$, 148) \xrightarrow{\text{pop}} (*id + id \$, T' E' \$, 148) \xrightarrow{\text{push 5}}$

$$\text{num}(id, id)$$

$(*id + id \$, *FT' E' \$, 1485) \xrightarrow{\text{pop}} (id + id \$, FT' E' \$, 1485) \xrightarrow{\text{push 8}} (id + id \$, id T' E' \$, 14858)$

$\xrightarrow{\text{pop}} (+id \$, T' E' \$, 14858) \xrightarrow{\text{push 6}} (+id \$, E' \$, 148586) \xrightarrow{\text{push 2}} (+id \$, +TE' \$, 1485862)$

$\xrightarrow{\text{pop}} (id \$, TE' \$, 1485862) \xrightarrow{\text{push 4}} (id \$, FT' E' \$, 1485862) \xrightarrow{\text{push 8}} (id \$, id T' E' \$, 14858628)$

$\xrightarrow{\text{pop}} (\$, T' E' \$, 14858628) \xrightarrow{\text{push 6}} (\$, E' \$, 148586286) \xrightarrow{\text{push 3}} (\$, \$, 1485862863) \xrightarrow{\text{ACCEPT}}$

șirul producător este 1485862863

Amalizator descendente cu revenire

- eg. $S \rightarrow aSbS$ ①
 $S \rightarrow aS$ ②
 $S \rightarrow c$ ③

1. $cb \in L(G)$?

$$(q_1, 1, \epsilon, S) \xrightarrow{\text{expandare}}$$

$$(q_1, 1, S_1, aSbS) \xrightarrow{i.m.}$$

$$(q_1, 1, S_1, aSbS) \xrightarrow{\text{altă încercare I}}$$

$$(q_1, 1, S_2, aS) \xrightarrow{i.m.}$$

$$(q_1, 1, S_2, aS) \xrightarrow{\text{altă încercare I}}$$

$$(q_1, 1, S_3, c) \xrightarrow{\text{avans}}$$

$$(q_1, 2, S_3, c, \epsilon) \xrightarrow{i.m.}$$

$(q_1, 2, S_3, c, \epsilon) \xrightarrow{\text{revenire}} \rightarrow \text{avem terminal} \text{ si nu putem sa expandam dupa el}$

$$(q_1, 1, S_3, c) \xrightarrow{\text{altă încercare II}}$$

$$(q_1, 1, \epsilon, S) \Rightarrow cb \notin L(G)$$

2. $acb \in L(G)$

$$(q_1, 1, \epsilon, S) \xrightarrow{\text{expandare}}$$

$$(q_1, 1, S_1, aSbS) \xrightarrow{\text{avans}}$$

$$(q_1, 2, S_1, aS, SbS) \xrightarrow{\text{expandare}}$$

$$(q_1, 2, S_1, aS_1, aSbSbS) \xrightarrow{i.m.}$$

$$(q_1, 2, S_1, aS_1, aSbSbS) \xrightarrow{\text{altă încercare I}}$$

$$(q_1, 2, S_1, aS_2, aSbS) \xrightarrow{i.m.}$$

$$(q_1, 2, S_1, aS_2, aSbS) \xrightarrow{\text{altă încercare I}}$$

$$(q_1, 2, S_1, aS_3, cbS) \xrightarrow{\text{avans}}$$

$$(q_1, 3, S_1, aS_3, c, bS) \xrightarrow{\text{avans}}$$

$$(q_1, 4, S_1, aS_3, cb, S) \xrightarrow{\text{expandare}}$$

$$(q_1, 4, S_1, aS_3, cbS_1, aSbS) \xrightarrow{i.m.}$$

$$(q_1, 4, S_1, aS_3, cbS_1, aSbS) \xrightarrow{\text{altă încercare}}$$

$$(q_1, 4, S_1, aS_3, cbS_2, aS) \xrightarrow{i.m.}$$

$$(q_1, 4, S_1, aS_3, cbS_2, aS) \xrightarrow{\text{altă încercare I}}$$

$$(q_1, 4, S_1, aS_3, cbS_3, c) \xrightarrow{\text{avans}}$$

$$(q_1, 5, S_1, aS_3, cbS_3, c, \epsilon) \xrightarrow{\text{succes}}$$

$$(t_5, S_1, aS_3, cbS_3, c, \epsilon)$$

$$\Rightarrow acb \in L(G)$$