LR(0) and SLR detailed explanation EN RO

LR(0) detailed explanation

ENGLISH

LR(0) analysis items have the form ($[A \to \alpha . \beta]$).
Such an item is obtained from a grammar production by inserting a dot somewhere among the symbols on the right-hand side, at the beginning, between symbols, or at the end.

Steps of LR(0) syntactic analysis:

1. Computation of the canonical collection of states
2. Construction of the LR(0) parsing table
3. Parsing the input sequence

---

1. Canonical collection of states

Each state in the canonical collection represents a set of LR(0) analysis items. The construction of the canonical collection uses the **closure** and **goto** functions.

The **closure** function is applied to a set of analysis items and produces another set of analysis items. If I = a set of analysis items is the argument of the closure function, then (closure(I)) includes the set I. In addition, if (closure(I)) contains an analysis item of the form ($[A \to \alpha . B\beta]$) (with the dot in front of a nonterminal, here (B)), then it also contains all analysis items derived from the productions of that nonterminal (B), with the dot placed at the beginning of the right-hand side of those productions (i.e., all items of the form ($[B \to . \gamma]$), where ($B \to \gamma$) is any production of (B)). This procedure is repeated until no **new** items are added to (closure(I)).

The **goto** function is applied to a pair ((s, X)), where (s) is a set of analysis items (a state), and (X) is a grammar symbol (terminal or nonterminal). The result of applying the function is a set of analysis items. The computation procedure is based on identifying all analysis items in state (s) that have the dot immediately in front of symbol X. In all such items, the dot is moved after symbol X, and then

the closure function is applied to the resulting set. If there is no analysis item in (s) with the dot in front of (X), then the result of the goto function is the empty set.

The states of the canonical collection are denoted ($s_0, s_1, \ldots, s_n$).

State s_0 is computed using the formula
s_0 = closure({$[S' \rightarrow . S]$})
where S' is the new start symbol added by augmenting the grammar.

For each newly obtained state s_k, the result of the goto function is computed for every pair consisting of state s_k and each terminal and nonterminal of the grammar. If the result of the goto function is not the empty set and does not already belong to the canonical collection, it is added to the canonical collection (and assigned an index). The procedure continues until no new states are added to the collection.

---

2. LR(0) parsing table

The LR(0) parsing table contains one row for each state in the canonical collection. Column-wise, the table is divided into two parts: the **action** part and the **goto** part. The action part has a single column, while the goto part contains one column for each terminal and nonterminal of the grammar.

State	Action	Goto					
		t1	...	tn	N1	...	Nk
0	shift						

The action part for row (i) (corresponding to state s_i) is filled as follows: the form of the items in state s_i is analyzed. If, in the analysis items of the state, the dot is **not** at the end, the action is **shift** (the symbol “s”, "sh" is entered in the corresponding cell or write "shift" if you have enough space). If the state contains the analysis item ($[S' \rightarrow S.]$), the action is **accept** (“acc” is entered). If state s_i contains an analysis item of the form $[A \rightarrow \alpha.]$ (with the dot at the end), then the corresponding action is **reduce**, and the cell is filled with the string “rj”, where j is the number of the production $A \rightarrow \alpha$ from the initial grammar.

If conflicts appear in the table (e.g., a state contains one analysis item indicating reduction and another indicating shift, or two items indicating reduction but with different productions), then the grammar is not of type LR(0).

The goto part of the table is filled according to the results of the goto function obtained during the computation of the canonical collection.

All table cells that remain unfilled indicate an error.

---

3. LR(0) parser operation

The LR(0) parser is a **shift/reduce** parser. It operates using two stacks (the working stack and the input stack), with the result placed on the output tape. Each of the two stacks has its top oriented toward the other, and both are bounded by the symbol `$`. In the initial configuration, the working stack contains the symbol `$` and the initial state (0), while the input stack contains the sequence to be parsed. To decide the action (shift or reduce), the state at the top of the working stack and the corresponding action in the parsing table are consulted.

- **If the state indicates a shift**, a symbol is moved from the top of the input stack to the top of the working stack, after which the result from the parsing table of the goto function (computed for the previous state and the shifted symbol) is pushed onto the working stack. (There must always be a state at the top of the working stack.)
- **If the state indicates a reduction with production (j)**, the right-hand side of production (j) (the handle) is identified at the top of the working stack and replaced with the nonterminal on the left-hand side of that production, after which the goto function is applied again. The number of the production used for the reduction is added to the beginning of the output tape.
- **If the state indicates acceptance**, the analyzed sequence is syntactically correct, and the output tape will contain the sequence of production numbers used to derive that sequence (via rightmost derivations), starting from the grammar's start symbol.

If, during the analysis, it is necessary to consult an empty cell in the parsing table, then the input sequence does not belong to the language generated by

the grammar (it is syntactically incorrect).

SLR

S → simple
L → left-to-right scan of the input
R → rightmost derivation in reverse

SLR is very similar to **LR(0)**, having one difference:
for reductions, the prediction of length **1** is considered (**FOLLOW**),
instead of prediction of length **0** used in **LR(0)**.

SLR steps

1. canonical collection (same as for LR(0))
2. SLR table (similar to LR(0)) with one mention:
   - for terminal symbols, **action** and **goto** columns merge together
3. use the table for parsing (same process as for LR(0))

---

1. Canonical collection of states

Exactly the same as for LR(0)

2. SLR table

The table is similar with the one for LR(0), being split in two parts: **action** and **goto**. In the action part we add a column for each terminal and a column for the $ symbol. In the goto part, we add a column for each non-terminal, excepting the augmented start symbol S'

State	Action					Goto			
	t1	t2	...	tn	$	N1	N2	...	Nk
s0									

where t1, t2, ..., tn are terminals and N1, N2 non-terminals

Rules for fill in the table cells

- If $[A \to \alpha.\beta] \in s_i$ then $\text{action}(i, u) = \text{shift}$ for all $u \in \Sigma \quad (\text{terminals})$ and $\text{goto}(i, u) \neq \varnothing$
- If $[A \to \beta.] \in s_i \quad \text{and} \quad A \neq S'$ then $\text{action}(i, u) = \text{reduce}_x$
  where:
  - x is the production number
  - $u \in \text{FOLLOW}(A)$
    In other words, what this means is that if in state s_k the dot is at the end of a production you have to compute FOLLOW of left side of production. For all the elements in the FOLLOW set you will fill in with reduce in the table
- If $[S' \to S.] \in s_i$ then $\text{action}(i, \$) = \text{accept}$
- If $\text{goto}(s_i, X) = s_j$ then $\text{goto}(i, X) = j$

3. Syntactic analysis / Parsing

Same as for LR(0)

ROMANIAN

Elementele de analiza LR(0) au forma $[A \to \alpha.\beta]$ Un astfel de element se obtine pe baza unei productii a gramaticii, prin adaugarea unui punct undeva intre simbolurile partii drepte, la inceputul sau la sfarsitul acesteia.

Pasii analizei sintactice LR(0):

1. Calculul colectiei canonice de stari
2. Constructia tabelului de analiza LR(0)
3. Analiza secventei
4. Fiecare stare din colectia canonica reprezinta o multime de elemente de analiza LR(0). In constructia colectiei canonice sunt folosite functiile “closure” si “goto”.

---

1. Colectia canonica de stari

Functia “closure” se aplica unei multimi de elemente de analiza si are ca si rezultat o multime de elemente de analiza. Daca I (= multime de elemente de analiza) este argumentul functiei “closure”, atunci closure(I) va include multimea I. In plus, daca closure(I) contine un element de analiza de forma $[A \to \alpha . B\beta]$ (cu punct in fata unui neterminal, aici B), atunci va contine si toate elementele de analiza obtinute pe baza productiilor acelui neterminal (B), cu punct in fata partilor drepte ale respectivelor productii (deci toate elementele de forma $[B \to . \gamma]$, unde $B \to \gamma$ e o productie oarecare a lui B). Procedeul se repeta pana cand in closure(I) nu se mai adauga elemente noi.

Functia “goto” se aplica unei perechi (s,X), unde s este multime de elemente de analiza (o stare), iar X este un simbol al gramaticii (terminal sau neterminal). Rezultatul aplicarii functiei e o multime de elemente de analiza. Procedeul de calcul se bazeaza pe identificarea tuturor elementelor de analiza din starea s care contin punct imediat in fata simbolului X. In toate aceste elemente gasite, se trece punctul dupa simbolul X, dupa care se aplica functia “closure” multimii rezultate. Daca in multimea s nu exista nici un element de analiza cu punct in fata lui X, atunci rezultatul functiei “goto” e multimea vida.

Starile colectiei canonice le notam $s_0, s_1, \ldots, s_n$

Starea s_0 se calculeaza dupa formula s_0=closure({$[S' \to . S]$}), unde S’ este noul simbol de start adaugat prin imbogatirea gramaticii.

Petru fiecare stare nou obtinuta s_k, se calculeaza rezultatul functiei “goto” pentru fiecare pereche formata din starea s_k si fiecare din terminalele si neterminalele gramaticii. Daca rezultatul functiei “goto” e diferit de multimea vida si nu apartine deja colectiei canonice, se agauga in colectia canonica (i se atribuie in continuare un index). Procedeul continua pana in momentul in care nu se mai adauga stari noi in colectie.

---

2. Tabelul de analiza

Tabelul de analiza LR(0) contine cate o linie pentru fiecare stare din colectia canonica. La nivelul coloanelor, tabelul este structurat pe doua parti: partea “action” si partea “goto”. Partea “action” are o singura coloana, in timp ce partea “goto” contine cate o coloana pentru fiecare terminal si neterminal din gramatica.

State	Action	Goto					
		t1	...	tn	N1	...	Nk
0	shift						

Completarea partii “action” pentru linia i (corespunzatoare starii s_i) se face dupa cum urmeaza: Se analizeaza forma elementelor din starea s_i. In cazul in care in elementele de analiza ale starii punctul NU este la sfarsit, actiunea este de deplasare (se completeaza simbolul “s”, "sh" in celula respectiva sau scrieti diresct shift daca aveti loc). In cazul in care in starea respectiva gasim elementul de analiza $[S' \rightarrow S.]$, actiunea este de acceptare (se completeaza “acc”). In cazul in care starea s_i contine un element de analiza de forma $[A \rightarrow a.]$ (cu punctul la sfarsit), atunci actiunea corespunzatoare va fi de reducere si se completeaza celula cu stringul “rj”, unde j este numarul productiei $A \rightarrow a$

Daca apar conflicte la nivelul tabelului (de ex. o stare contine un element de analiza care indica reducere si un altul care indica deplasare, sau doua care indica reducere, dar cu productii diferite), atunci gramatica nu este de tip LR(0).

Partea “goto” a tabelului se completeaza conform rezultatelor functie “goto”, obtinute pe parcursul calculului colectiei canonice.

Toate celulele ramase necompletate in tabel indica eroare.

---

3. Analiza sintactica / Parsarea

Analizorului sintactic LR(0) e un analizor de tip deplasare(shift)/reducere. Lucreaza cu doua stive (stiva de lucru si stiva de intrare), rezultatul fiind depus in banda de iesire. Fiecare din cele doua stive are varful orientat spre cealalta si ambele sunt marginite de simbolul $. In configuratia initiala, stiva de lucru contine simbolul $ si starea initiala (0), iar stiva de intrare contine secventa de analizat. Pentru a decide actiunea (deplasare sau reducere), intotdeauna se consulta starea din varful stivei de lucru si actiunea aferenta din tabelul de analiza.

- **Daca starea indica deplasare(shift)**, atunci se muta un simbol din varful stivei de intrare in varful stivei de lucru, dupa care se aduga in varful stivei de lucru rezultatul din tabel al functiei goto, calculat pentru vechea stare si

simbolul deplasat. (Intotdeauna in varful stivei de lucru trebuie sa ramana o stare).

- **Daca starea indica reducere cu productia j**, atunci se identifica in varful stivei de lucru partea dreapta a productiei j (mansa) si se inlocuieste cu neterminalul din partea stanga a acelei productii, dupa care se aplica din nou “goto”. Numarul productiei cu care se face reducerea se va adauga la inceputul benzii de iesire.
- **Daca starea indica acceptare**, secventa analizata e corecta sintactic si in banda de iesire vom avea succesiunea numerelor de productii folosite pentru derivarea acelei secvente (prin derivari de dreapta) pornind de la simbolul de start al gramaticii.

Daca in timpul analizei e necesara consultarea unei celule vide din tabel, atunci secventa de analizat nu apartine limbajului generat de gramatica (e sintactic incorecta).

SLR

S → simplu
L → parcurgere de la stânga la dreapta a secvenței de intrare
R → derivari de dreapta, în ordine inversă

SLR este foarte similar cu **LR(0)**, având o singură diferență importanta: pentru reduceri, este considerată predicția de lungime **1** (**FOLLOW**), în locul predicției de lungime **0** folosita pentru **LR(0)**.

Pașii SLR

1. colecția canonică (aceeași ca pentru LR(0))
2. tabelul SLR (similar cu cel pentru LR(0)), cu o mențiune:
   - pentru simbolurile terminale, coloanele **action** și **goto** se unesc
3. folosirea tabelului pentru analiză sintactică (același proces ca pentru LR(0))

1. Colecția canonică de stări

Exact aceeași ca pentru LR(0)

2. Tabelul SLR

Tabelul este similar cu cel pentru LR(0), fiind împărțit în două părți: **action** și **goto**. În partea **action** se adaugă câte o coloană pentru fiecare terminal și o coloană pentru simbolul $. În partea **goto** se adaugă câte o coloană pentru fiecare non-terminal, cu excepția simbolului de start augmentat S'.

State	**Action**					**Goto**			
	t1	t2	...	tn	$	N1	N2	...	Nk
s0									

unde **t1, t2, ..., tn** sunt terminale, iar **N1, N2, ..., Nk** sunt non-terminale.

Reguli de completare a tabelului

- Daca $[A \to \alpha . \beta] \in s_i$ atunci $\text{action}(i, u) = \text{shift}$ (deplasare) pentru toate $u \in \Sigma \quad (\text{terminale})$ si $\text{goto}(i, u) \neq \varnothing$
- Daca $[A \to \beta .] \in s_i \quad \text{si} \quad A \neq S'$ atunci $\text{action}(i, u) = \text{reduce}_x$ (reducere) unde:
  - x este numarul productiei
  - $u \in \text{FOLLOW}(A)$
    Cu alte cuvinte, acest lucru înseamnă că dacă în starea s_k punctul se află la sfârșitul unei producții, trebuie calculată mulțimea **FOLLOW** a părții stângi a producției. Pentru toate elementele din mulțimea **FOLLOW**, se va completa în tabel acțiunea de **reducere** pe randul starii s_i

3. Analiza sintactica / Parsarea

La fel ca la LR(0)