

Bottom-up analýza, LR(0)-analyzátor

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

`viliam.hromada@stuba.sk`



Syntaktická analýza zdola-nahor (bottom-up)

- Druhý v praxi používaný prístup syntaktickej analýzy je analýza zdola-nahor.
- Syntaktický analyzátor hľadá odvodenie slova v danej gramatike, pričom strom odvodenia sa konštruuje **od listov ku koreňom**, t.j. začína sa od terminálov a končí sa s počiatočným neterminálom, pričom vstupný reťazec sa spracúva **zľava-doprava**. Taktiež poradie vytváraných listov v strome je **zľava-doprava**.
- Pri vytváraní derivačného stromu sa teda k potomkom vytvára rodič, t.j. pravidlo sa aplikuje opačne, kde sa pravá strana pravidla nahrádza ľavou stranou.
- Postupnosť použitých pravidiel gramatiky, generovaná syntaktickým analyzátorom, bude predstavovať **pravé odvodenie** slova v danej gramatike, avšak v opačnom poradí generovaných pravidiel.







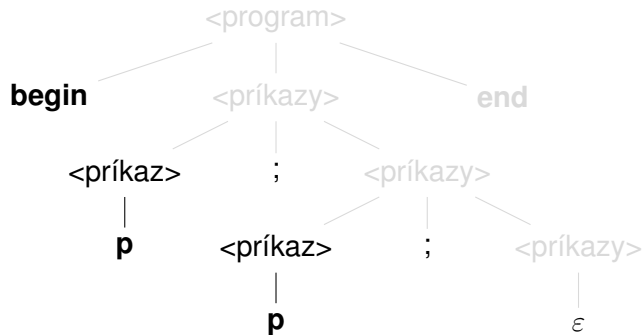






















Príklad (pokr.)

Ak by sme si vypísali čísla použitých pravidiel, dostali by sme postupnosť 4 4 3 2 2 1. Táto postupnosť **odzadu** predstavuje **pravú deriváciu**, t.j. nasledovné odvodenie:

$$\begin{aligned} \langle \text{program} \rangle &\Rightarrow_r \text{begin } \langle \text{príkazy} \rangle \text{ end} \Rightarrow_r \\ &\Rightarrow_r \text{begin } \langle \text{príkaz} \rangle ; \langle \text{príkazy} \rangle \text{ end} \Rightarrow_r \\ &\Rightarrow_r \text{begin } \langle \text{príkaz} \rangle ; \langle \text{príkaz} \rangle ; \langle \text{príkazy} \rangle \text{ end} \Rightarrow_r \\ &\Rightarrow_r \text{begin } \langle \text{príkaz} \rangle ; \langle \text{príkaz} \rangle ; \text{end} \Rightarrow_r \\ &\Rightarrow_r \text{begin } \langle \text{príkaz} \rangle ; \mathbf{p} ; \text{end} \Rightarrow_r \\ &\Rightarrow_r \text{begin } \mathbf{p} ; \mathbf{p} ; \text{end} \end{aligned}$$


Teoretický model bottom-up analyzátoru

- Podobne, ako pri top-down analyzátoch, aj tu sa postup konštrukcie derivačného stromu, dá modelovať pomocou zásobníkového automatu.
- Zásobník sa znovu použije na simuláciu odvodenia v danej gramatike, avšak tým, že v tomto prístupe sa pravidlá používajú tak, že sa pravá strana pravidla nahrádza ľavou, tak aj v zásobníku bude dochádzať k tzv. **redukcii**, t.j. aplikácie pravidla, kde sa celá jeho pravá strana nahrádza príslušným neterminálom na ľavej strane.
- V zásobníku teda neterminály vznikajú **redukciou**; terminály sú do zásobníka pridávané priamo z čítaného vstupu tzv. **presunom**.



Teoretický model bottom-up analyzátoru

- V zásobníku sa v každom momente nachádzajú **tie symboly gramatiky**, ktoré korešpondujú s **uzlami derivačného stromu**, ktoré nemajú rodiča.
- Preto sa na začiatku výpočtu v zásobníku nič nenachádza (v derivačnom strome nie je na začiatku žiaden symbol gramatiky) a na konci výpočtu sa snažíme dospieť do situácie, že sa v zásobníku nachádza len počiatočný neterminál S (pretože koreň derivačného stromu nemá rodiča).



Teoretický model bottom-up analyzátor

Bottom-up syntaktický analyzátor pre bezkontextovú gramatiku $G = (N, T, P, S)$ bude (nedeterministický) zásobníkový automat, ktorý pracuje na základe nasledovných princípov:

- Automat začína výpočet s **prázdny** zásobníkom.
- Ďalej automat opakuje (nedeterministicky) nasledovné akcie:
 - **Presun** - aktuálny vstupný symbol sa presunie na vrch zásobníka,
 - **Redukcia** - viaže sa na konkrétne pravidlo $A \rightarrow \alpha$, v zásobníku pravú stranu pravidla α nahradí ľavou stranou A .
- Automat slovo akceptuje (vyhodnotí ako syntakticky správne, t.j. z jazyka), ak na konci v zásobníku ostane len počiatočný neterminál S a vstup bol celý spracovaný.
- Ak automat pri každej redukcii pošle na výstup číslo pravidla, získa sa postupnosť pravidiel, ktorá v obrátenom poradí dáva pravú deriváciu vstupného slova.



Syntaktická analýza bottom-up I

Vstup: Bezkontextová gramatika $G = (N, T, P, S)$, reťazec w

Výstup: Zistenie, či $w \in L(G)$ a postupnosť pravidiel, ktorá v opačnom poradí generuje pravú deriváciu w

- 1: **pokiaľ** možno vykonať niektorú akciu **rob**
- 2: **vyber a vykonaj** niektorú z nasledovných akcií:
- 3: **presun**
- 4: vlož do zásobníka aktuálny vstupný symbol $t, t \neq \varepsilon$
- 5: prejdi na ďalší vstupný symbol t
- 6: **koniec presun**
- 7: **redukcia**
- 8: **ak** $A \rightarrow \alpha \in P$ a zároveň α sa nachádza na vrchu
- 9: zásobníka **potom**
- 10: odstráň α zo zásobníka
- 11: vlož A do zásobníka
- 12: pošli na výstup pravidlo $A \rightarrow \alpha$



Syntaktická analýza bottom-up II

```

13:      koniec ak
14:      koniec redukcia
15:      koniec vyber a vykonaj
16:      koniec pokiaľ
17:      ak zásobník obsahuje len  $S$  potom
18:          akceptuj vstupné slovo
19:      inak
20:          neakceptuj vstupné slovo
21:      koniec ak

```



Príklad

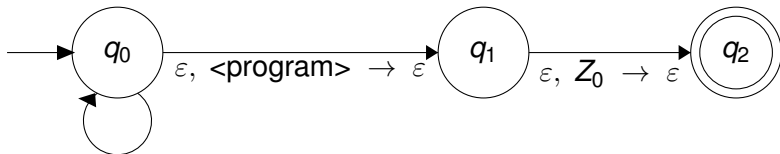
Uvažujme gramatiku zo slajdu č. 3 a vetu **begin p ; p ; end**.

r.	Zásobník $\square$	Zvyšok vstupu:	Akcia
1	ϵ	begin p; p; end	P
2	begin	p; p; end	P
3	begin p	; p; end	R4
4	begin<príkaz>	; p; end	P
5	begin<príkaz>;	p; end	P
6	begin<príkaz>; p	; end	R4
7	begin<príkaz>;<príkaz>	; end	P
8	begin<príkaz>;<príkaz>;	end	R3
9	begin<príkaz>;<príkaz>;<príkazy>	end	R2
10	begin<príkaz>;<príkazy>	end	R2
11	begin<príkazy>	end	P
12	begin<príkazy>end	ϵ	R1
13	<program>	ϵ	A



- Zásobník sme písali *sklopený doprava*, t.j. v tomto prípade je prvok na vrchu zásobníka napísaný najviac vpravo (viď ikona zásobníka v záhlaví tabuľky).
- Je to preto, aby bolo vidieť, že zreťazením obsahu zásobníka (od dna po vrch) a neprečítanej časti vstupu dostávame **pravé vetné formy**.
- Automat je nedeterministický! **Presun** sa môže vykonať v podstate hocikedy, taktiež pravidlo č. 3 sa môže použiť pri **Redukcii** v podstate hocikedy - ich iné použitie, než je uvedené v príklade - by malo za následok neakceptáciu slova (štandardný príklad činnosti nedeterministického ZA).
- Použité pravidlá pri redukcii boli: 4 4 3 2 2 1. Ich použitím v obrátenom poradí, t.j. 1 2 2 3 4 4, by sme vedeli zostrojiť pravú deriváciu daného slova.

V tomto ZA dovoľíme vybrať zo zásobníka viacero symbolov pri 1 prechode:



end, $\varepsilon \rightarrow$ end

$$; , \varepsilon \rightarrow ;$$
$$\mathbf{p}, \varepsilon \rightarrow \mathbf{p}$$
$$\varepsilon, \mathbf{end}\langle\text{príkazy}\rangle\mathbf{begin} \rightarrow \langle\text{program}\rangle$$
$$\varepsilon, \langle \text{príkazy} \rangle; \langle \text{príkaz} \rangle \rightarrow \langle \text{príkazy} \rangle$$
$$\varepsilon, \varepsilon \rightarrow \langle \text{príkazy} \rangle$$
$$\varepsilon, \mathbf{p} \rightarrow \langle \text{príkaz} \rangle$$

Syntaktické analyzátory *LR* jazykov

Pri svojej činnosti budú *LR*-analyzátory využívať 2 tabuľky:

- *ACTION*[s, t] - udáva akciu, ktorú analyzátor vykoná, ak je na vrchu zásobníka stav s a na vstupe symbol $t \in T_\epsilon$ (ϵ znamená, že bol prečítaný celý vstup), akcie sú: **P**(presun), **R** _{i} (redukcia podľa i -teho pravidla), **A**(akceptácia), **E**(chyba).
- *GOTO*[s, X] - udáva aký nový stav sa vloží na vrch zásobníka, ak na vrchu je stav s a do zásobníka teoretického modelu by sa vkladal symbol X ($X \in N \cup T$).
- *ACTION* má teda podobnú úlohu ako *RT* pri *LL*-analyzátoroch; *GOTO* je „prevod“ medzi symbolmi gramatiky a stavmi, resp. medzi teoretickým modelom a *LR* analyzátorom.



Syntaktický analyzátor jazykov LR I

Vstup: Bezkontextová gramatika $G = (N, T, P, S)$, reťazec w , tabuľky *ACTION*, *GOTO*

Výstup: Zistenie, či $w \in L(G)$ a postupnosť pravidiel, ktorá v opačnom poradí generuje pravú deriváciu w

- 1: vlož do zásobníka začiatkový stav s_0
- 2: $t \leftarrow$ prvá lexéma zo vstupu
- 3: **pokiaľ** true **rob**
- 4: $s \leftarrow$ symbol na vrchu zásobníka
- 5: **ak** *ACTION*[s, t] = Presun **potom**
- 6: vlož do zásobníka stav *GOTO*[s, t]
- 7: $t \leftarrow$ ďalšia lexéma zo vstupu
- 8: **inak**
- 9: **ak** *ACTION*[s, t] = Redukcia podľa $A \rightarrow \alpha$ **potom**
- 10: vyber zo zásobníka vrchných $|\alpha|$ stavov
- 11: $\acute{s} \leftarrow$ nový symbol na vrchu zásobníka
- 12: vlož do zásobníka stav *GOTO*[$\acute{s}, A$]



Syntaktický analyzátor jazykov LR II

13: **inak**
 14: **ak** *ACTION*[*s*, *t*] = Akceptácia **potom**
 15: akceptuj vstupné slovo
 16: **inak**
 17: neakceptuj vstupné slovo
 18: **koniec ak**
 19: **koniec ak**
 20: **koniec ak**
 21: **koniec pokiaľ**

Príklad

Uvažujme gramatiku zo slajdu č. 3 a vetu **begin p ; p ; end**. Uvažujme, že už máme zostrojené tabuľky *ACTION* a *GOTO*.

<i>ACTION</i>	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
begin	P								
end		R3	P				R3	R2	
;					P	R4			
p		P					P		
ε				R1					A

<i>GOTO</i>	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
begin	s_1								
end			s_3						
;					s_6				
p		s_5					s_5		
<program>	s_8								
<príkazy>		s_2					s_7		
<príkaz>		s_4					s_4		

Príklad

r.	Zásobník □	Zvyšok vstupu:	Akcia
1	s_0	begin p; p; end	P
2	$s_0 s_1$	p; p; end	P
3	$s_0 s_1 s_5$	; p; end	R4
4	$s_0 s_1 s_4$	; p; end	P
5	$s_0 s_1 s_4 s_6$	p; end	P
6	$s_0 s_1 s_4 s_6 s_5$	; end	R4
7	$s_0 s_1 s_4 s_6 s_4$	; end	P
8	$s_0 s_1 s_4 s_6 s_4 s_6$	end	R3
9	$s_0 s_1 s_4 s_6 s_4 s_6 s_7$	end	R2
10	$s_0 s_1 s_4 s_6 s_7$	end	R2
11	$s_0 s_1 s_2$	end	P
11	$s_0 s_1 s_2 s_3$	ε	R1
12	$s_0 s_8$	ε	A

LR-analyzátory

Vidíme, že konštrukcia *LR*-analyzátor (t.j. bottom-up / zdola-nahor parsera) je v podstate o konštrukcii tabuliek *ACTION* a *GOTO*. K tejto konštrukcii existuje viacero prístupov, podľa ktorých rozoznávame rôzne *LR* analyzátory:

- *LR*(0) - konštrukcia *ACTION* nevyužíva vstupné symboly, len stavy v zásobníku; v praxi sa príliš nevyužíva, pretože množina jazykov *LR*(0) je malá - slúži ako základ pre zložitejšie a v praxi používané konštrukcie, ako napr.
- *LR*(1) - použiteľný pre širokú triedu gramatík, z hľadiska konštrukcie najzložitejší;
- *SLR*(1) a *LALR*(1) - najčastejšie praktické voľby, z hľadiska konštrukcie sa nachádzajú „medzi“ *LR*(0) a *LR*(1).



LR(0) syntaktický analyzátor

- $LR(0)$ analyzátory pri rozhodovaní, akú akciu majú vykonať (*ACTION* tabuľka), nevyužívajú vstupné symboly (resp. využívajú 0 vstupných symbolov - preto $LR(0)$).
- Pri rozhodovaní používajú len **aktuálny stav na vrchu zásobníka**.
- V podstate vždy, ak sa rozpozná pravá strana nejakého pravidla, tak sa robí príslušná **redukcia**, inak sa robí **presun**.
- V praxi nemá uplatnenie ako samostatný analyzátor; využíva sa však ako základ komplikovanejších analyzátorov $SLR(1)$, $LALR(1)$, $LR(1)$.



LR(0)-automat LR(0) syntaktického analyzátor

- Konštrukcia tabuliek *ACTION* a *GOTO* analyzátor *LR(0)* začína konštrukciou tzv. *LR(0)-automatu* pre danú gramatiku $G = (N, T, P, S)$.
- Tento *LR(0)-automat* je v podstate deterministický konečný automat, ktorý pomocou svojich **stavov** rozpoznáva časti pravých strán pravidiel gramatiky, ktoré sa môžu nachádzať v zásobníku teoretického modelu bottom-up analyzátor.
- Využíva sa teda na určovanie toho, aké časti (predpony) pravých strán ktorých pravidiel sa môžu nachádzať na vrchu zásobníka.
- **Stavy** tohoto *LR(0)-automatu* sa potom používajú ako symboly v zásobníku *LR*-analyzátor.



Konštrukcia $LR(0)$ -automatu

Konštrukcia $LR(0)$ -automatu je založená na $LR(0)$ položkách.

Definícia

*Nech $G = (N, T, P, S)$ je bezkontextová gramatika. Potom $LR(0)$ **položka** je položka tvaru*

$$A \rightarrow \alpha_1 \bullet \alpha_2 \tag{1}$$

pričom $A \rightarrow \alpha_1 \alpha_2$ musí byť nejaké pravidlo z množiny pravidiel P gramatiky G a symbol $\bullet \notin N \cup T$ na pravej strane $LR(0)$ položky označuje, aká časť pravej strany príslušného pravidla bola doteraz identifikovaná (a nachádza sa na vrchu zásobníka).



LR(0)-položka

LR(0)-položka - špeciálne prípady výskytu symbolu $\bullet$:

- Na začiatku pravej strany indikuje, že sa na vrchu zásobníka nenachádza ani jeden symbol z pravej strany príslušného pravidla (t.j. napr $E \rightarrow \bullet T$)
- Na konci pravej strany indikuje, že sa na vrchu zásobníka nachádza celá pravá strana príslušného pravidla a teoreticky môže teda prísť ku redukcii podľa daného pravidla (t.j. napr $E \rightarrow E + T\bullet$)



Konštrukcia $LR(0)$ -automatu

- Stavy $LR(0)$ -automatu budú tvoriť množiny $LR(0)$ položiek, popisujúcich pravé strany všetkých pravidiel, ktoré boli doteraz identifikované a nachádzajú sa na vrchu zásobníka.
- Napríklad, ak by sa v nejakom stave nachádzala trojica:
 <príkaz> $\rightarrow$ **id** • := <výraz>
 <príkaz> $\rightarrow$ **id** • : <príkaz>
 <príkaz> $\rightarrow$ **id** •

znamená to, že bol rozpoznaný identifikátor **id** ako prvý symbol pravej strany troch rôznych pravidiel. V každom stave (množinách položiek) je vždy nutné uvažovať všetky možnosti / všetkých kandidátov.



Uzáver množiny $LR(0)$ -položiek

- Keďže je nutné uvažovať všetkých kandidátov, je potrebné do množiny $LR(0)$ -položiek zahrnúť tie položky, ktoré sú tam síce implicitne zahrnuté, ale nie explicitne uvedené.
- Napríklad, ak je v množine položka, kde $\bullet$ stojí pred neterminálom A , to znamená, že ak sa má ako ďalší symbol rozpoznať neterminál A , musí sa najprv rozpoznať pravá strana niektorého z pravidiel, ktoré má neterminál A ako ľavú stranu; pričom po následnej redukcii sa rozpozná práve neterminál A .
- Keďže vopred nie je známe, o ktoré pravidlo s neterminálom A na ľavej strane môže ísť, treba do množiny položiek pridať **všetky** takéto pravidlá ako položky: $A \rightarrow \bullet\gamma$, kde $A \rightarrow \gamma \in P$.
- Takto pridávame do množiny položky, kým v množine existujú nespracované pravidlá tvaru $B \rightarrow \alpha \bullet A\beta$. Výsledok nazývame **uzáver množiny $LR(0)$ položiek**, resp. *CLOSURE*0.



Uzáver množiny $LR(0)$ položiek

Vstup: Bezkontextová gramatika $G = (N, T, P, S)$, množina $LR(0)$ položiek S

Výstup: Doplnenie S o $LR(0)$ položky, ktoré implicitne obsahuje

- 1: **funkcia** $CLOSURE_0$ (množina $LR(0)$ položiek S)
- 2: **pokiaľ** v S existujú nespracované položky tvaru $B \rightarrow \alpha \bullet A\beta$
- 3: vyber z S ľubovoľnú položku tvaru $B \rightarrow \alpha \bullet A\beta$
- 4: označ ju ako spracovanú
- 5: pridaj do S všetky položky tvaru $A \rightarrow \bullet\gamma$, kde $A \rightarrow \gamma \in P$
- 6: **koniec pokiaľ**
- 7: **vrát** S
- 8: **koniec funkcia**



Príklad

Gramatika:

$G = (N, T, P, S)$, $N = \{S, E, T\}$, $T = \{;, +, (,), \text{id}\}$ s pravidlami:

1. $S \rightarrow E$;
2. $E \rightarrow E + T$
3. $E \rightarrow T$
4. $T \rightarrow \text{id}$
5. $T \rightarrow (E)$

Ako sa zostrojí množina $CLOSURE_0(\{T \rightarrow (\bullet E)\})$.



Príklad

1. V prvom kroku obsahuje množina len položku $T \rightarrow (\bullet E)$, t.j. pridajú sa E -pravidlá v tvare položiek:

$$E \rightarrow \bullet E + T$$

$$E \rightarrow \bullet T$$

2. Položka $T \rightarrow (\bullet E)$ je teda spracovaná. Ďalej si vyberieme jednu z 2 novopridaných, povedzme, $E \rightarrow \bullet E + T$. V tomto prípade by sme pridávali položky vytvorené z E -pravidiel, t.j. také, ktoré tam už sú. Čiže aj položka $E \rightarrow \bullet E + T$ je spracovaná.
3. Vyberieme zatiaľ poslednú nespracovanú položku, $E \rightarrow \bullet T$. Pre ňu musíme do množiny pridať položky:

$$T \rightarrow \bullet \text{id}$$

$$T \rightarrow \bullet (E)$$

4. Keďže obe posledne pridané pravidlá nemajú za symbolom $\bullet$ neterminál, algoritmus končí (neexistuje nespracovaná položka tvaru $B \rightarrow \alpha \bullet A\beta$).



Príklad

Množina $CLOSURE_0(\{T \rightarrow (\bullet E)\})$ je teda:

$$T \rightarrow (\bullet E)$$

$$E \rightarrow \bullet E + T$$

$$E \rightarrow \bullet T$$

$$T \rightarrow \bullet id$$

$$T \rightarrow \bullet (E)$$

Konštrukcia $LR(0)$ -automatu

- Pomocou množín položiek $LR(0)$ zostrojíme $LR(0)$ -automat, ktorého stavy budú môcť predstavovať zásobníkové symboly LR -analyzátoru.
- **Na začiatku** gramatiku upravíme pridaním **nového počiatočného symbolu** $\hat{S}$ a pridáme pravidlo $\hat{S} \rightarrow S$.

Konštrukcia $LR(0)$ -automatu

- Počiatočný stav $LR(0)$ -automatu bude stav s_0 , ktorý obsahuje položky:
 $CLOSURE_0(\{\hat{S} \rightarrow \bullet S\})$

podľa umelo pridaného pravidla $\hat{S} \rightarrow S$.

- Tento počiatočný stav reprezentuje situáciu na začiatku syntaktickej analýzy, keď na akceptáciu vstupného slova je potrebné rozpoznať pôvodný počiatočný neterminál gramatiky S .
- Pozícia $\bullet$ indikuje, že v zásobníku ešte nemáme neterminál S - čo je samozrejmé, keďže sme na začiatku.



Konštrukcia $LR(0)$ -automatu

Pre stav s určíme nasledovníkov a prechody do nich nasledovne:

1. Ak stav s obsahuje položky, v ktorých $\bullet$ stojí pred (terminálnym alebo neterminálnym) symbolom X (t.j. položka tvaru $A \rightarrow \alpha \bullet X\beta$),
2. potom je potrebné viesť prechod zo stavu s na symbol X do stavu:

$$CLOSURE_0(\{A \rightarrow \alpha X \bullet \beta\})$$

kde $A \rightarrow \alpha \bullet X\beta \in s$.

Cieľový stav (nasledovník s pre nejaký symbol X) teda zohľadňuje rozpoznanie symbolu X na správnom mieste v rámci pravej strany pravidla. Ak dostaneme stav, ktorý už existuje, tak použijeme existujúci.



Vytvorenie $LR(0)$ automatu

Vstup: Bezkontextová gramatika $G = (N, T, P, S)$, S sa nenachádza na pravej strane žiadneho pravidla

Výstup: Vytvorenie $LR(0)$ automatu

- 1: vytvor stav $s_0 = CLOSURE_0(\{\dot{S} \rightarrow \bullet S\})$ a označ ho ako nespracovaný
- 2: **pokiaľ** existujú nespracované stavy **rob**
- 3: vyber ľubovoľný nespracovaný stav s a označ ho ako spracovaný
- 4: **pre všetky** $X \in N \cup T$ také, že s obsahuje položku $A \rightarrow \alpha \bullet X\beta$ **rob**
- 5: $\dot{s} = CLOSURE_0(\{A \rightarrow \alpha X \bullet \beta \mid A \rightarrow \alpha \bullet X\beta \in s\})$
- 6: **ak** $\dot{s}$ sa nenachádza medzi stavmi $LR(0)$ automatu **potom**
- 7: zaraď $\dot{s}$ medzi stavy a označ ho ako nespracovaný
- 8: **koniec ak**
- 9: zaznamenaj prechod zo stavu s do stavu $\dot{s}$ na symbol X
- 10: **koniec pre**
- 11: **koniec pokiaľ**



Príklad

Zostrojte $LR(0)$ -automat ku gramatike zo slajdu č. 29.

Najprv pridáme nový počiatkový neterminál $\acute{S}$ a pravidlo $\acute{S} \rightarrow S$. Potom počiatkový stav $s_0 = CLOSURE_0(\{\acute{S} \rightarrow \bullet S\})$. Po uzávere sa doň dostanú ďalšie položky a výsledný tvar:

$$\acute{S} \rightarrow \bullet S$$

$$S \rightarrow \bullet E;$$

$$E \rightarrow \bullet E + T$$

$$E \rightarrow \bullet T$$

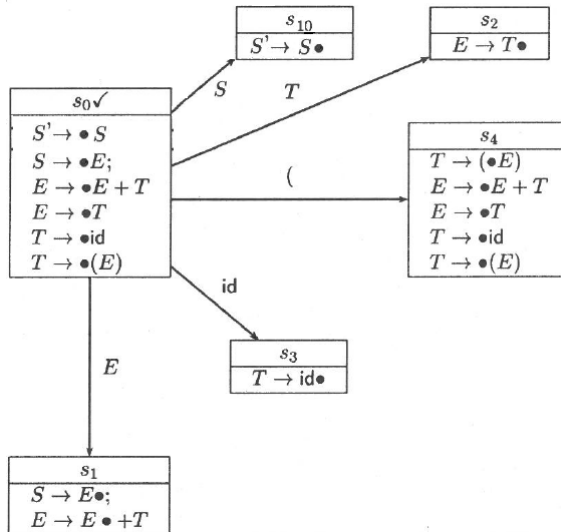
$$T \rightarrow \bullet id$$

$$T \rightarrow \bullet (E)$$

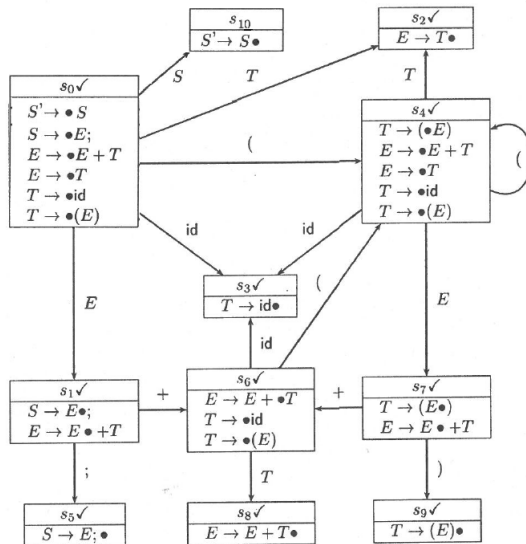
s_0 označíme ako spracovaný a určíme nasledovníkov pre tie symboly, pred ktorými je $\bullet$ v položkách stavu s_0 , t.j. symboly: $\{S, E, T, id, (\}$.



Príklad (pokr.)



Príklad (pokr.)



LR(0)-analyzátor

- Na základe $LR(0)$ -automatu je možné posúdiť, či pôvodná gramatika generuje $LR(0)$ -jazyk.
- Taktiež je z neho možné zostrojiť *ACTION* a *GOTO* tabuľky.
- Symbol ● v položkách označuje, aká časť pravidla bola rozpoznaná.
- Ak sa automat dostane do stavu s položkou, kde ● je na konci pravidla (v príklade stavy s_2, s_3, s_5, s_8, s_9 , potom prichádza do úvahy redukcia podľa daného pravidla.
- Ak je v danom stave **len** takáto položka, potom to jednoznačne signalizuje danú redukciu - $LR(0)$ -automat teda „rozpoznáva“ pravé strany pravidiel v zásobníku a pravá strana je rozpoznaná vtedy, ak sa nachádza celá v zásobníku, t.j. ● je na konci pravej strany.



LR(0)-analyzátor - ACTION

- LR(0) analyzátor pre rozhodovanie o ďalšej akcii **nevyužíva** vstupný symbol, t.j. tabuľka *ACTION* má v tomto prípade len 1 riadok, pretože akcia bude závisieť **len od aktuálneho stavu na vrchole zásobníka**.
- Položka tvaru $\hat{S} \rightarrow S\bullet$ v stave s , kde $\hat{S}$ je počiatočný neterminál, signalizuje akceptáciu. Preto $A \in ACTION[s]$.
- Položky tvaru $A \rightarrow \alpha\bullet$ v stave s , okrem položky signalizujúcej akceptáciu, signalizujú redukciu podľa pravidla $A \rightarrow \alpha$. Preto $R_{A \rightarrow \alpha} \in ACTION[s]$.
- Položky tvaru $A \rightarrow \alpha \bullet t\beta$, $t \in T$ v stave s signalizujú presun. Preto $P \in ACTION[s]$.



Príklad

Pre gramatiku zo slajdu č. 29 bude *ACTION* analyzátoru *LR(0)* vyzerat':

<i>ACTION</i>	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
	P	P	R4	R5	P	R2	P	P	R4	R6	A

LR(0)-analyzátor - *GOTO*

- Tabuľku *GOTO* zostrojíme na základe *LR(0)*-automatu nasledovne:
- Ak v *LR(0)* automate existuje prechod zo stavu s na symbol X do stavu $\acute{s}$, potom $GOTO[s, X] = \acute{s}$.



Príklad

Pre gramatiku zo slajdu č. 29 bude *GOTO* analyzátor *LR(0)* vyzerat':

<i>GOTO</i>	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
;		s_5									
+		s_6						s_6			
(	s_4				s_4		s_4				
)								s_9			
id	s_3				s_3		s_3				
<i>E</i>	s_1				s_7						
<i>T</i>	s_2				s_2		s_8				
<i>S</i>	s_{10}										

Keďže *LR(0)* je deterministický automat, tabuľka má v každom políčku max. 1 stav.



Konflikty $LR(0)$ -analyzátor

- Daný stav však nemusí obsahovať len 1 položku s ● na konci pravej strany. Môže totiž obsahovať:
 1. 2 rôzne $LR(0)$ položky s ● na konci. Potom nevieme, podľa ktorého pravidla vykonať redukciu; ide o konflikt tzv. **redukcia-redukcia** syntaktického analyzátor $LR(0)$.
 2. $LR(0)$ položky s ● na konci aj vo vnútri (alebo na začiatku) pravidla. Potom nevieme, či vykonať redukciu podľa pravidiel s ● na konci, alebo presun; ide o konflikt **presun-redukcia** syntaktického analyzátor $LR(0)$.

Jeden stav môže súčasne obsahovať aj viacero konfliktov.

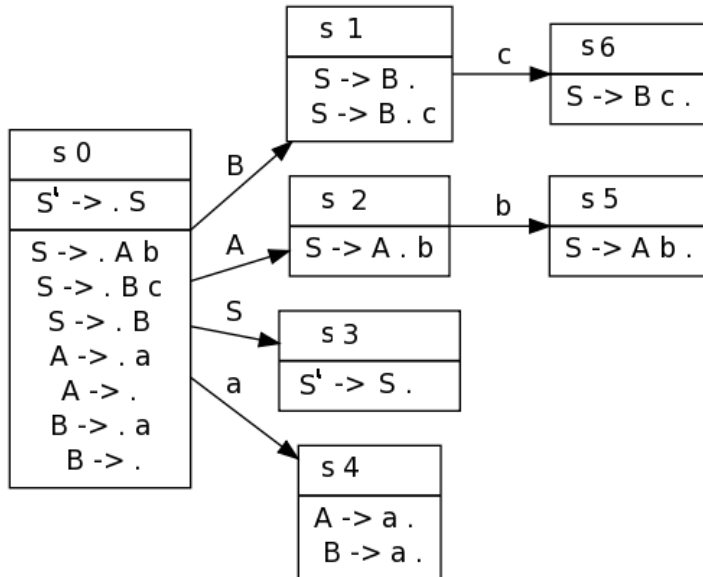


Príklad gramatiky s konfliktami $LR(0)$ analyzátor

Nech je daná gramatika $G = (\{S, A, B\}, \{a, b, c\}, P, S)$ s pravidlami:

1. $S \rightarrow Ab$
2. $S \rightarrow Bc$
3. $S \rightarrow B$
4. $A \rightarrow a$
5. $A \rightarrow \varepsilon$
6. $B \rightarrow a$
7. $B \rightarrow \varepsilon$

LR(0) automat



Tabuľka *ACTION*

ACTION	s_0	s_1	s_2	s_3	s_4	s_5	s_6
	P/R5/R7	P/R3	P	A	R4/R6	R1	R2

Vidíme, že v tabuľke sa nachádzajú konflikty:

- Presun/redukcia v stave s_1
- Redukcia/redukcia v stave s_4
- Presun/redukcia/redukcia v stave s_0

Gramatika sa teda nedá spracovať *LR(0)* analyzátorom.

LR(0)-analyzátor

- Ak $LR(0)$ -analyzátor (stavy $LR(0)$ -automatu) neobsahuje konflikty, potom vstupnú gramatiku nazývame $LR(0)$ **gramatikou** a možno k nej zostrojiť $LR(0)$ syntaktický analyzátor podľa spomenutých princípov.
- Napríklad tabuľka *ACTION* na slajde č. 41 neobsahovala konflikty, teda gramatika zo slajdu č. 29 **je** $LR(0)$ -**gramatika**.
- Tabuľka *ACTION* na slajde č. 47 obsahovala konflikty, teda gramatika zo slajdu č. 45 **nie je** $LR(0)$ gramatika.

Použitá literatúra

Aho, A., Lam, M., Sethi, R., Ullman, J.: *Compilers: Principles, techniques and tools.*

Dedera, L': *Počítačové jazyky a ich spracovanie.*

Linz, P.: *An Introduction to Formal Languages and Automata.*