SLR gramatiky

Formální jazyky a překladače - cvičení 12

Analyzujte řetězec "abaaab" metodou zdola nahoru

```
S --> aAS     (1)
S --> b       (2)
A --> bA      (3)
A --> a       (4)

(q, abaaab, #) :-        start
(q, baaab, #a) :-        přijetí terminálního symbolu, dále jen přijetí
(q, aaab, #ab) :-        přijetí
(q, aab, #aba) :-        přijetí
(q, aab, #abA) :-        redukce podle pravidla (4), dále jen (4); redukují se nejpravjší symboly
(q, aab, #aA) :-         (3)
(q, ab, #aAa) :-         přijetí
(q, b, #aAaa) :-         přijetí
(q, b, #aAaA) :-         (4)
(q, e, #aAaAb) :-        přijetí
(q, e, #aAaAS) :-        (2)
(q, e, #aAS) :-          (1)
(q, e, #S) :-            (1)
(r, e, e)                akceptováno
```

Zatímco v analýze shora dolů bylo zřejmé, podle kterých pravidel symboly rozkládat, v tomto případě již není situace tak jednoduchá. Dokud se nenaučíte LR gramatiky, můžete brát uvedený příklad jako kouzlo, kdy se symboly přijímají a redukují tak, aby to vyšlo. Pokud se vám takové vysvětlení nezdá, ukážeme si alespoň náznak deterministického postupu.

Množinu G vytvoříme složitější, G = {#, a_1, a_2, b_1, b_2, A_1, A_2, S}. Tzv. rozkladová tabulka vypadá takto:

M	a	b	A	S
#	a_1	b_1		
a_1	a_2	b_2	A_1	
a_2	redukce podle (4)			
b_1	redukce podle (2)			
b_2	a_2	b_2	A_2	
A_1	a_1	b_1		S
A_2	redukce podle (3)			
S	redukce podle (1)			

V horní řadě je vstupující symbol ze vstupního (rozpoznávaného) řetězce (symboly A a S si vysvětlíme za chvíli), v levém sloupci symbol na vrcholu zásobníku (tj. v našem zápisu konfigurace automatu znak nejvíc vpravo). Pokud je na vrcholu zásobníku např. "#" a na vstupu "a", vložím do zásobníku symbol "a_1". Je-li na vrcholu zásobníku např. "A_2", sezobnu z vrcholu zásobníku dva symboly, které musí být "$b_?A_2$" (viz přepisovací pravidlo 3) a do zásobníku vložím symbol "A" (opět viz pravidlo 3). Co konkrétně vložím, záleží na momentálním vrcholu zásobníku; proto jsou v horní řadě tabulky symboly "A" a "S", protože i ty mohu do automatu vkládat. Lépe se princip pochopí na příkladu:

```
(q, abaaab, #) :-            start
(q, baaab, # a1) :-          na vrcholu byl symbol "#", chci přijmout "a", proto vložím (dle tabulky) "a1"
(q, aaab, # a1 b2) :-        na vrcholu byl symbol "a1", chci přijmout "b", proto vložím (dle tabulky) "b2"
(q, aab, # a1 b2 a2) :-      na vrcholu byl symbol "b2", chci přijmout "a", proto vložím (dle tabulky) "a2"
(q, aab, # a1 b2 A2) :-      na vrcholu byl symbol "a2", budu redukovat podle pravidla "A --> a";
                             ze zásobníku sezobnu "a2" a vložím "A"; protože na vrcholu zásobníku zbyl symbol
                             "b2", vložím (dle tabulky) "A2")
(q, aab, # a1 A1) :-         na vrcholu byl symbol "A2", budu redukovat podle pravidla "A --> bA";
                             ze zásobníku sezobnu "b2 A2" a vložím "A"; protože na vrcholu zásobníku zbyl symbol
                             "a1", vložím (dle tabulky) "A1")
(q, ab, # a1 A1 a1) :-       přijetí
(q, b, # a1 A1 a1 a2) :-     přijetí
(q, b, # a1 A1 a1 A1) :-     redukce (4)
(q, e, # a1 A1 a1 A1 b1) :-  přijetí
(q, e, # a1 A1 a1 A1 S) :-   redukce (2)
(q, e, # a1 A1 S) :-         redukce (1)
(q, e, # S) :-               redukce (1)
(r, e, e)                    akceptace
```

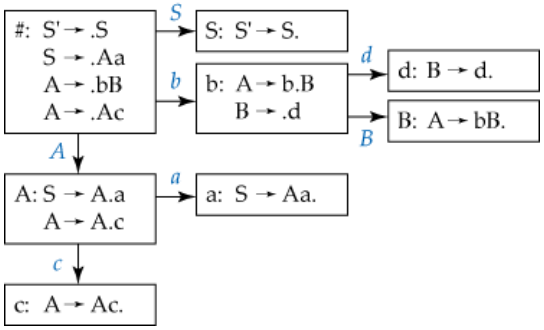
SLR gramatiky

1. Zjistěte, zda je daná gramatika SLR(1) a sestavte rozkladovou tabulku

   ```
   S --> Aa  (1)
   A --> bB  (2)
   A --> Ac  (3)
   B --> d   (4)
   ```

   Řešení

   Do gramatiky doplníme pravidlo 0: `S' --> S`, od kterého začneme budovat množinu LR(0) položek:



V množině LR(0) položek nejsou žádné konflikty, proto je gramatika LR(0). Nyní snadno sestavíme rozkladovou tabulku:

M	akce	S	A	B	a	b	c	d
#		S	A			b		
S	AKC							
A					a		c	
B	R2							
a	R1							
b				B				d
c	R3							
d	R4							

Pro kontrolu zkusíme akceptovat řetězec bdcca:

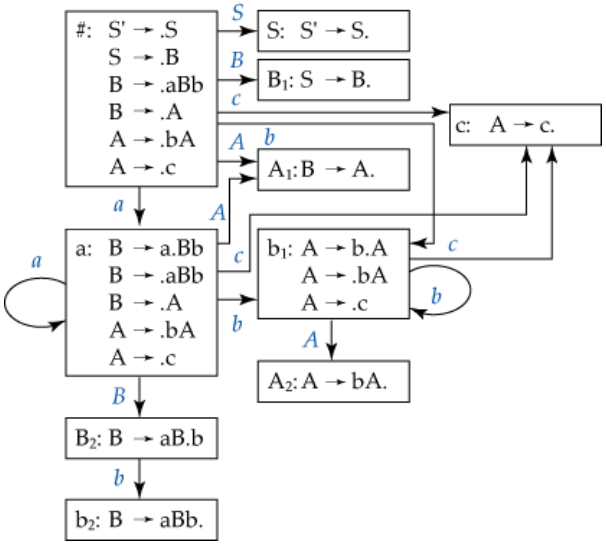
```
(#, bdcca, e) :- (#b, dcca, e) :- (#bd, cca, e) :- (#bB, cca, 4) :-
(#A, cca, 42) :- (#Ac, ca, 42) :- (#A, ca, 423) :- (#Ac, a, 423) :-
(#A, a, 4233) :- (#Aa, e, 4233) :- (#S, e, 42331) :- AKC
```

2. Zjistěte, zda je daná gramatika SLR(1) a sestavte rozkladovou tabulku

```
S --> B     (1)
B --> aBb   (2)
B --> A     (3)
A --> bA    (4)
A --> c     (5)
```

Řešení

Do gramatiky doplníme pravidlo 0: `S' --> S`, od kterého začneme budovat množinu LR(0) položek:



V množině LR(0) položek jsou některé označeny stejným názvem. K jejich odlišení použijeme indexy. Ty jsou na obrázku už vyznačeny, v průběhu kreslení se indexy nedělají. Jinak v položkách nejsou žádné jiné konflikty, proto je gramatika LR(0). Nyní snadno sestavíme rozkladovou tabulku:

M	akce	S	A	B	a	b	c
#		S	A_1	B_1	a	b_1	c
S	AKC						
A_1	R3						

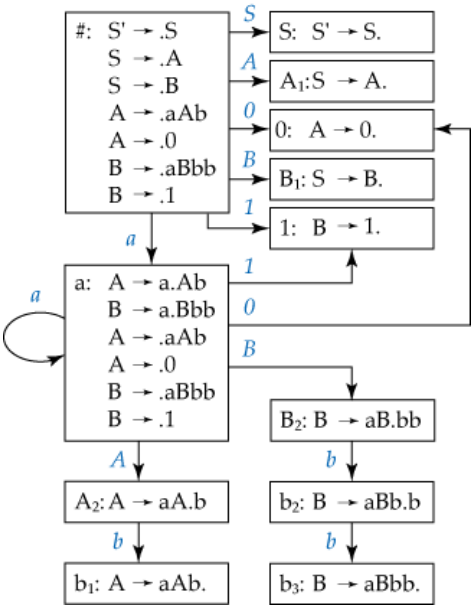
A_2	R4						
B_1	R1						
B_2						b_2	
a			A_1	B_2	a	b_1	c
b_1			A_2			b_1	c
b_2	R2						
c	R5						

3. Zjistěte, zda je daná gramatika SLR(1) a sestavte rozkladovou tabulku

```
S --> A     (1)
S --> B     (2)
A --> aAb   (3)
A --> 0     (4)
B --> aBbb  (5)
B --> 1     (6)
```

Řešení

Do gramatiky doplníme pravidlo 0: `S' --> S`, od kterého začneme budovat množinu LR(0) položek:



Tato gramatika je zajímavá tím, že není typu LL. Lze se o tom přesvědčit poměrně snadno. Generovaný jazyk jsou řetězce a0b, aa0bb, aaa0bbb, ... a a1bb, aa1bbbb, aaa1bbbbbb, ... Použijeme-li analýzu shora dolů a prohlížíme n symbolů dopředu, nemusí být zřejmé, zda časem přijde symbol "1" nebo symbol "0". To ovšem v analýze shora dolů potřebujeme hned na začátku analýzy - má se použít pravidlo 1 nebo 2? Proto neexistuje takové n, pro které je daná gramatika LL(n).

V množině LR(0) položek ale žádné konflikty nejsou, proto je gramatika LR(0). Snadno sestavíme rozkladovou tabulku:

M	akce	S	A	B	a	b	0	1
#		S	A_1	B_1	a		0	1
S	AKC							
A_1	R1							
A_2						b_1		
B_1	R2							
B_2						b_2		
a			A_2	B_2	a		0	1
b_1	R3							
b_2						b_3		
b_3	R5							
0	R4							
1	R6							

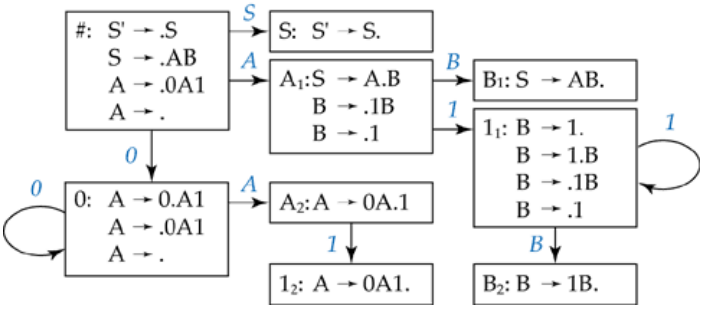
4. Zjistěte, zda je daná gramatika SLR(1) a sestavte rozkladovou tabulku

```
S --> AB    (1)
A --> 0A1   (2)
```

```
A --> e      (3)
B --> 1B     (4)
B --> 1      (5)
```

Řešení

Do gramatiky doplníme pravidlo 0: `S' --> S`, od kterého začneme budovat množinu LR(0) položek:



Oproti předchozím gramatikám vidíme rozdíl. Některé položky v sobě obsahují jak pravidla s tečkou na konci, tak pravidla s tečkou uprostřed. Není tedy na první pohled zřejmé, zda redukovat nebo přijmout další symbol. Můžeme-li se o dalším postupu rozhodnout na základě vstupu (zkoumáním dalších n symbolů), řekneme o gramatice, že je typu SLR(n). V našem případě se můžeme roznodnout podle jednoho vstupujícího symbolu, gramatika je tedy SLR(1).

Rozkladová tabulka bude nyní o něco složitější. Sloupec "akce" se musí rozpadnout na několik případů podle toho, jaký symbol je zrovna na vstupu. Důležité je to právě v těch položkách, kde je možná redukce i přijetí. Je-li v položce pravidlo `X --> alfa.` a na vstupu je symbol z follow(X), můžeme redukovat. Je-li v položce pravidlo `X --> alfa.q beta` a na vstupu symbol q, můžeme ho přijmout. Nevznikne-li aplikací těchto pravidel žádný konflikt, je gramatika SLR(1).

M	akce			S	A	B	0	1
	0	1	e					
#	P	3		S	A_1		0	
S			A					
A_1		P				B_1		1_1
A_2		P						1_2
B_1			1					
B_2			4					
0	P	3			A_2		0	
1_1		P	5			B_2		1_1
1_2		2						

Při akceptaci musíme s tabulkou zacházet mírně odlišně než v předchozích případech. Postupujeme takto: je-li na vstupu znak x, na vrcholu zásobníku symbol Y a v tabulce akcí "P", symbol x přijmeme. To uděláme tak, že do zásobníku vložíme symbol, který je v tabulce na řádku Y v sloupci x (v rozkladové části). Je-li na vstupu znak x, na vrcholu zásobníku symbol Y a v tabulce akcí číslo, musíme redukovat podle příslušného pravidla. K tomu musíme odebrat z vrcholu zásobníku tolik symolů, kolik předepisuje dané pravidlo (tj. pravou stranu tohoto pravidla, na případné indexy u symolů nehledíme), čímž se na vrcholu zásobníku octne symbol Z. Nyní musíme místo odebraných symbolů do zásobníku vrátit nový neterminální symbol, např. Q (dle příslušného přepisovacího pravidla). Do zásobníku tedy vložíme symbol, který je na řádku Z ve sloupci Q tabulky.

Zkusíme si akceptovat řetězce 0111 a 11:

```
(#, 0111, e) :- (#0, 111, e)          na vrcholu #, na vstupu 0 => přijetí, vložíme 0
             :- (#0A2, 111, 3)        na vrcholu byla 0, va vstupu 1 => redukce podle pravidla 3, tj.
                                      A --> e; z vrcholu zásobníku odebereme "e" (tj. nic), čímž se
                                      na vrcholu zásobníku ocitne symbol 0; a vkládáme symbol A,
                                      tj. dle tabulky A2
             :- (#0A212, 11, 3)        přijetí 1
             :- (#A1, 11, 32)         na vrcholu byl symbol 12, na vstupu 1 => redukujeme
                                      podle pravidla 2, tj. A --> 0A1; z vrcholu zásobníku odebereme
                                      symboly 0A1 (nehledě na indexy), čímž se na vrcholu objeví
                                      symbol #; pravidlo 2 požaduje vložení A, tj. podle tabulky
                                      vložíme A1
             :- (#A111, 1, 32)
             :- (#A11111, e, 32)
             :- (#A111B2, e, 325)
             :- (#A1B1, e, 3254)
             :- (#S, e, 32541)
             :- AKC

(#, 11, e)   :- (#A1, 11, 3)
             :- (#A111, 1, 3)
             :- (#A11111, e, 3)
             :- (#A111B2, e, 35)
             :- (#A1B1, e, 354)
             :- (#S, e, 3541)
             :- AKC
```

Poslední změna: 15.12.2014