LR(1) - analyzátory

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

Syntaktický analyzátor jazyka LR(1)

- Najsilnejším zo základných syntaktických analyzátorov je analyzátor $LR(1)$.
- Platí, že $LR(0) \subset SLR(1) \subset LR(1)$.
- Jeho nevýhodou je jeho zložitejšia konštrukcia, než pri $LR(0)$, resp. $SLR(1)$.
- Ako z názvu vyplýva, jedná sa o LR-analyzátor, avšak tentokrát sa pri procese rozpoznávania pravých strán pravidiel uvažuje aj 1 vstupný symbol (na rozdiel od $LR(0)$-analyzátorov).
- Pri jeho konštrukcii sa používa tzv. $LR(1)$**-automat.**

LR(1)-automat

- *LR*(1)-automat sa konštruuje podobne, ako *LR*(0)-automat, s tým rozdielom, že sa pri procese rozpoznávania pravých strán pravidiel gramatiky **uvažujú vstupné symboly**, ktoré sa **očakávajú** na vstupe v momente **redukcie** podľa daného pravidla.
- Jeho stavy sú tvorené tzv. *LR*(1)-položkami.

$LR(1)$-položka

Definícia

Nech $G = (N, T, P, S)$ je bezkontextová gramatika. Potom LR(1) ***položka*** *je položka tvaru*

$$A \to \alpha_1 \bullet \alpha_2, l \tag{1}$$

pričom $A \to \alpha_1\alpha_2$ musí byť nejaké pravidlo z množiny pravidiel P gramatiky G, symbol $\bullet \notin N \cup T$ na pravej strane LR(1) položky označuje, aká časť pravej strany príslušného pravidla bola doteraz identifikovaná (a nachádza sa na vrchu zásobníka) a symbol $l \in T_\varepsilon$ predstavuje ***očakávaný symbol na vstupe*** *v momente, keď bude rozpoznaná* ***celá pravá strana*** *pravidla $A \to \alpha_1\alpha_2$ (resp. ε, ak bol už celý vstup spracovaný).*

$LR(1)$-automat

- Základný rozdiel medzi $LR(0)$-automatom a $LR(1)$-automatom je teda v tom, že $LR(1)$-položky obsahujú aj očakávaný symbol na vstupe v momente, keď bude rozpoznaná celá pravá strana.
- Tento symbol (resp. symboly) bude signalizovať **redukciu**, podobne ako symboly z $FOLLOW(A)$ pri $SLR(1)$-analyzátore.
- Keďže po redukcii podľa príslušného pravidla sa bude symbol l vo vetnej forme nachádzať za neterminálom A, nutne $l \in FOLLOW(A)$.
- Na základe tohto očakávaného symbolu potom možno riešiť konflikty presun-redukcia, resp. redukcia-redukcia $SLR(1)$-analyzátora.

LR(1)-automat

- Môže sa stať, že jedna *LR*(1)-položka obsahuje viacero očakávaných symbolov.
- V takom prípade ju zapíšeme skrátene ako $A \rightarrow \alpha_1 \bullet \alpha_2, \{l_1, ..., l_m\}$
- T.j. bola rozpoznaná časť α_1 pravidla $A \rightarrow \alpha_1\alpha_2$, pričom po rozpoznaní celej pravej strany sa na vstupe očakáva niektorý symbol z množiny $\{l_1, ..., l_m\}$.
- Pri konštrukcii *LR*(1)-automatu postupujeme obdobne, ako pri *LR*(0), s tým rozdielom, že je potrebné upraviť uzáverovú operáciu *CLOSURE*0 na *CLOSURE*1 kvôli očakávaným symbolom.

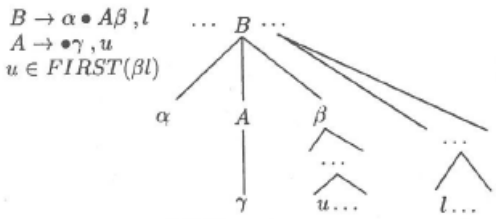
Uzáver množiny $LR(1)$ položiek

Vstup: Bezkontextová gramatika $G = (N, T, P, S)$, množina $LR(1)$ položiek S
Výstup: Doplnenie S o $LR(1)$ položky, ktoré implicitne obsahuje

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

Vysvetlenie uzáverovej operácie *CLOSURE*1

- Uvažujme položku $B \to \alpha \bullet A\beta, l$



Obsah zásobníka	Zvyšok vstupu
$\ldots\alpha$	...
...	
$\ldots\alpha\gamma$	$u\ldots$
$\ldots\alpha A$	$u\ldots$
...	
$\ldots\alpha A\beta$	$l\ldots$
$\ldots B$	$l\ldots$

Figure: Prevzaté z Dedera, Ľ.: Počítačové jazyky a ich spracovanie

Vysvetlenie uzáveru položky $B \to \alpha \bullet A\beta, l$

- Podobne ako pri *LR*(0), ak sa má ako ďalší symbol na vrchu zásobníka rozpoznať A, potom je najprv potrebné rozpoznať niektoré z A-pravidiel (t.j. jeho pravú stranu), aby sa po jeho redukcii na vrch zásobníka dostalo práve A.
- V okamihu redukcie tohto A-pravidla musí byť na vstupe taký terminál, ktorým sa môžu začínať reťazce odvodené z časti β pravidla $B \to \alpha A\beta$, tzn. symbol z množiny $FIRST(\beta l)$.
- $FIRST(\beta l)$ preto, lebo ak β môže generovať/odvodiť ε, tak sa na vstupe môže očakávať aj symbol l.
- Preto musíme k položke $B \to \alpha \bullet A\beta, l$ pridať **všetky** položky odvodené z A-pravidiel, t.j. $A \to \bullet\gamma, u$, kde γ sú pravé strany všetkých A-pravidiel a u sú symboly z množiny $FIRST(\beta l)$.

Príklad

Uvažujme gramatiku:

$S \to E;$

$E \to E + T$

$E \to T$

$T \to \text{id}$

$T \to (E)$

Počiatočný stav bude tvorený uzáverom položky $S' \to \bullet S, \{\varepsilon\}$, t.j. $CLOSURE1(\{S \to \bullet S, \{\varepsilon\}\})$

Príklad

$S' \to \bullet S$, $\{\varepsilon\}$
$S \to \bullet E;$, $\{\varepsilon\}$
$E \to \bullet E + T$, $\{;\}$
$E \to \bullet T$, $\{;\}$
$E \to \bullet E + T$, $\{+\}$
$E \to \bullet T$, $\{+\}$
$T \to \bullet \mathrm{id}$, $\{;\}$
$T \to \bullet (E)$, $\{;\}$
$T \to \bullet \mathrm{id}$, $\{+\}$
$T \to \bullet (E)$, $\{+\}$

Príklad

$S' \to \bullet S$, $\{\varepsilon\}$
$S \to \bullet E;$, $\{\varepsilon\}$
$E \to \bullet E + T$, $\{;\}$
$E \to \bullet T$, $\{;\}$
$E \to \bullet E + T$, $\{+\}$
$E \to \bullet T$, $\{+\}$
$T \to \bullet \mathrm{id}$, $\{;\}$
$T \to \bullet (E)$, $\{;\}$
$T \to \bullet \mathrm{id}$, $\{+\}$
$T \to \bullet (E)$, $\{+\}$

Príklad

$S' \to \bullet S$, $\{\varepsilon\}$
$S \to \bullet E;$, $\{\varepsilon\}$
$E \to \bullet E + T$, $\{;\}$
$E \to \bullet T$, $\{;\}$
$E \to \bullet E + T$, $\{+\}$
$E \to \bullet T$, $\{+\}$
$T \to \bullet \mathrm{id}$, $\{;\}$
$T \to \bullet (E)$, $\{;\}$
$T \to \bullet \mathrm{id}$, $\{+\}$
$T \to \bullet (E)$, $\{+\}$

Príklad

$S' \to \bullet S$, $\{\varepsilon\}$
$S \to \bullet E;$, $\{\varepsilon\}$
$E \to \bullet E + T$, $\{;\}$
$E \to \bullet T$, $\{;\}$
$E \to \bullet E + T$, $\{+\}$
$E \to \bullet T$, $\{+\}$
$T \to \bullet \mathrm{id}$, $\{;\}$
$T \to \bullet (E)$, $\{;\}$
$T \to \bullet \mathrm{id}$, $\{+\}$
$T \to \bullet (E)$, $\{+\}$

Príklad

$S' \to \bullet S$, $\{\varepsilon\}$
$S \to \bullet E;$, $\{\varepsilon\}$
$E \to \bullet E + T$, $\{;\}$
$E \to \bullet T$, $\{;\}$
$E \to \bullet E + T$, $\{+\}$
$E \to \bullet T$, $\{+\}$
$T \to \bullet \mathrm{id}$, $\{;\}$
$T \to \bullet (E)$, $\{;\}$
$T \to \bullet \mathrm{id}$, $\{+\}$
$T \to \bullet (E)$, $\{+\}$

Príklad

$S' \to \bullet S \quad , \{\varepsilon\}$
$S \to \bullet E; \quad , \{\varepsilon\}$
$E \to \bullet E + T \quad , \{;\}$
$E \to \bullet T \quad , \{;\}$
$E \to \bullet E + T \quad , \{+\}$
$E \to \bullet T \quad , \{+\}$
$T \to \bullet \mathrm{id} \quad , \{;\}$
$T \to \bullet (E) \quad , \{;\}$
$T \to \bullet \mathrm{id} \quad , \{+\}$
$T \to \bullet (E) \quad , \{+\}$

Príklad

T.j. v stručnom zápise:

$$
\begin{array}{ll}
S' \to \bullet S & ,\{\varepsilon\} \\
S \to \bullet E; & ,\{\varepsilon\} \\
E \to \bullet E + T & ,\{;,+\} \\
E \to \bullet T & ,\{;,+\} \\
T \to \bullet \mathrm{id} & ,\{;,+\} \\
T \to \bullet (E) & ,\{;,+\}
\end{array}
$$

Vytvorenie $LR(1)$-automatu

- Do gramatiky doplníme pravidlo $S' \to S$, kde S' je nový počiatočný neterminál.
- Počiatočný stav $LR(1)$-automatu obsahuje položky $CLOSURE1(\{S' \to \bullet S, \{\varepsilon\}\})$.
- T.j. počiatočný stav vystihuje situáciu, že na akceptáciu vstupného slova je potrebné na záver robiť redukciu podľa pravidla $S' \to S$. Pozícia $\bullet$ vystihuje situáciu, že na začiatku je zásobník prázdny, a teda nebola rozpoznaná žiadna časť pravej strany.
- ε ako očakávaný symbol predstavuje fakt, že po redukcii podľa pravidla už nemôže na vstupe ostať žiadny symbol, t.j. tam očakávame prázdny reťazec ε.

Vytvorenie *LR*(1)-automatu

- Ak stav *s* obsahuje *LR*(1)-položky, v ktorých stojí • pred symbolom *X* (či už terminálom alebo neterminálom), t.j. tvaru $A \rightarrow \alpha \bullet X\beta, \{l_1, ..., l_m\}$, potom je potrebné zo stavu *s* viesť prechod na *X* do stavu $CLOSURE1(\{A \rightarrow \alpha X \bullet \beta, \{l_1, ..., l_m\}\})$. Rozpoznanie symbolu *X* totižto **nezmení** očakávané symboly pre dané pravidlo.
- Ak dostávame taký stav, ktorý v automate existuje, použijeme existujúci.
- **Pozor!!!** *LR*(1)-automat môže obsahovať 2 rôzne stavy, ktoré síce obsahujú rovnaké rozpoznané časti pravidiel, ale líšia sa v očakávaných symboloch.

Vytvorenie $LR(1)$ automatu

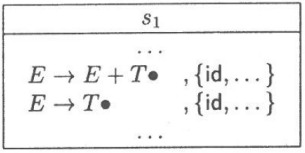
Vstup: Bezkontextová gramatika $G = (N, T, P, S)$
Výstup: Vytvorenie $LR(1)$ automatu

1: pridaj do gramatiky pravidlo $S' \rightarrow S$
2: vytvor stav $s_0 = CLOSURE1(\{S' \rightarrow \bullet S, \{\varepsilon\}\})$ a označ ho ako nespracovaný
3: **pokiaľ** existujú nespracované stavy **rob**
4: vyber ľubovoľný nespracovaný stav s a označ ho ako spracovaný
5: **pre všetky** $X \in N \cup T$ také, že s obsahuje položky $A \rightarrow \alpha \bullet X\beta, \{l_1, ..., l_m\}$ **rob**
6: $\acute{s} = CLOSURE1(\{A \rightarrow \alpha X \bullet \beta, \{l_1, ..., l_m\} \mid$
$A \rightarrow \alpha \bullet X\beta, \{l_1, ..., l_m\} \in s\})$
7: **ak** $\acute{s}$ sa nenachádza medzi stavmi $LR(1)$ automatu **potom**
8: zaraď $\acute{s}$ medzi stavy a označ ho ako nespracovaný
9: **koniec ak**
10: zaznamenaj prechod zo stavu s do stavu $\acute{s}$ na symbol X
11: **koniec pre**
12: **koniec pokiaľ**

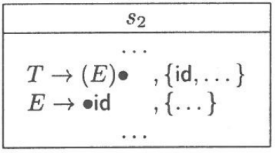
ACTION, *GOTO* pre *LR*(1)-analyzátor

- Tabuľka *GOTO* sa zostrojí analogicky ako pri *LR*(0), resp. *SLR*(1) analyzátore.
- Tabuľka *ACTION* sa zostrojí analogicky ako pri *SLR*(1), teda pri redukcii sa koná redukcia vtedy, ak je na vstupe jeden z očakávaných symbolov danej položky.
- Redukciu signalizujú položky $B \rightarrow \rho\bullet, \{l_1, ..., l_m\}$, avšak:
  1. Pri *SLR*(1) analyzátore $\{l_1, ..., l_m\} = FOLLOW(B)$,
  2. Pri *LR*(1) analyzátore $\{l_1, ..., l_m\} \subseteq FOLLOW(B)$.
- Preto pri *LR*(1) analyzátoroch vzniká menšie riziko konfliktu, a teda trieda jazykov, pre ktoré sa dá zostrojiť *LR*(1) analyzátor, je **väčšia**, než trieda *SLR*(1) jazykov. Preto $SLR(1) \subset LR(1)$.
- Ak tabuľka *ACTION LR*(1) analyzátora neobsahuje konflikty, príslušnú gramatiku nazývame *LR*(1)-gramatikou.

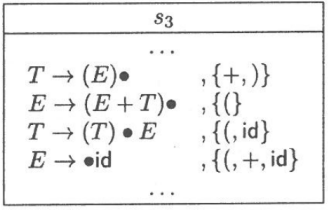
Konflikty *LR*(1)-analyzátora



Redukcia-redukcia



Presun-redukcia



Bez konfliktov

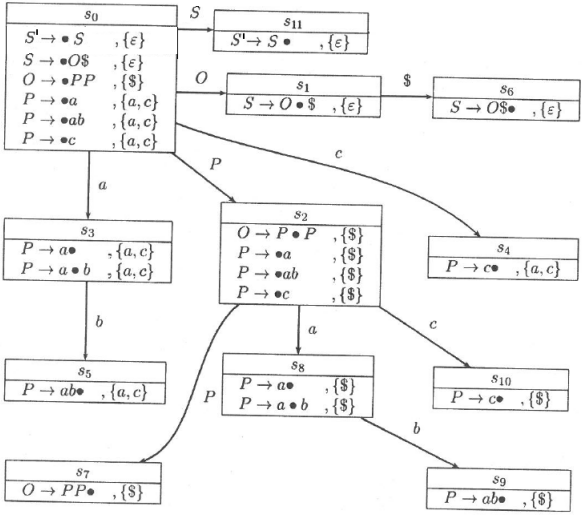
Figure: Prevzaté z Dedera, Ľ.: Počítačové jazyky a ich spracovanie

Príklad

Zostrojte $LR(1)$ analyzátor pre gramatiku:

1. $S \rightarrow O\$$
2. $O \rightarrow PP$
3. $P \rightarrow a$
4. $P \rightarrow ab$
5. $P \rightarrow c$

Príklad - *LR*(1) automat



Príklad - tabuľka *ACTION*

ACTION	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}
$		P						R2	R3	R4	R5	
a	P		P	R3	R5	R4						
b				P					P			
c	P		P	R3	R5	R4						
ε							R1					A

Príklad - tabuľka *GOTO*

GOTO	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}
$		s_6										
a	s_3		s_8									
b				s_5					s_9			
c	s_4		s_{10}									
S	s_{11}											
O	s_1											
P	s_2		s_7									

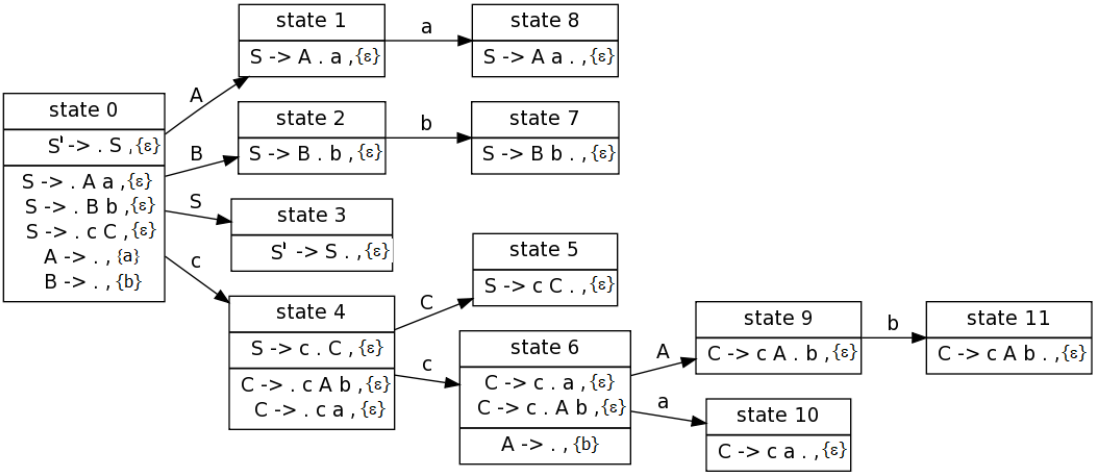
Príklad č. 2

Zostrojte $LR(1)$-analyzátor pre gramatiku $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$.
Pravidlá P:

1. $S \to Aa$
2. $S \to Bb$
3. $S \to cC$
4. $A \to \varepsilon$
5. $B \to \varepsilon$
6. $C \to cAb$
7. $C \to ca$

Pre túto gramatiku obsahoval $SLR(1)$ analyzátor konflikty. Skúsme teda, či pomôže zostrojenie $LR(1)$.

LR(1)-automat


state 0
S' -> . S , {ε}
S -> . A a , {ε}
S -> . B b , {ε}
S -> . c C , {ε}
A -> . , {a}
B -> . , {b}
state 1
S -> A . a , {ε}
state 2
S -> B . b , {ε}
state 3
S' -> S . , {ε}
state 4
S -> c . C , {ε}
C -> . c A b , {ε}
C -> . c a , {ε}
state 5
S -> c C . , {ε}
state 6
C -> c . a , {ε}
C -> c . A b , {ε}
A -> . , {b}
state 7
S -> B b . , {ε}
state 8
S -> A a . , {ε}
state 9
C -> c A . b , {ε}
state 10
C -> c a . , {ε}
state 11
C -> c A b . , {ε}
A
B
S
c
a
b
C
c
A
a
b

ACTION&GOTO

ACTION	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
a	R4	P					P					
b	R5		P				R4			P		
c	P				P							
ε				A		R3		R2	R1		R7	R6

GOTO	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
S	s3											
A	s1						s9					
B	s2											
C					s5							
a		s8					s10					
b			s7							s11		
c	s4				s6							

Výpočet *cca*

r.	Zásobník ⊏ Zvyšok vstupu	Akcia
1	s_0 *cca*	P
2	$s_0 s_4$ *ca*	P
3	$s_0 s_4 s_6$ *a*	P
4	$s_0 s_4 s_6 s_{10}$ ε	R7
5	$s_0 s_4 s_5$ ε	R3
6	$s_0 s_3$ ε	A

T.j. pravá derivácia *cca* (postupne pravidlá č. 3 a 7): $S \Rightarrow cC \Rightarrow cca$

Výpočet *ba*

r.	Zásobník ⊏ Zvyšok vstupu	Akcia
1	s_0 *ba*	R5
2	$s_0 s_2$ *ba*	P
3	$s_0 s_2 s_7$ *a*	SYNTAX ERROR

T.j. pravá derivácia *ba* neexistuje (syntaktický analyzátor vrátil chybu) a teda reťazec *ba* nepatrí do jazyka generovaného príslušnou gramatikou.

Rozdiel *SLR*(1) a *LR*(1)

- Na príklade sme videli ukážku gramatiky, ktorá sa nedala spracovať *SLR*(1)-analyzátorom, ale dala sa spracovať *LR*(1)-analyzátorom.
- Pri *SLR*(1)-analyzátore závisia očakávané symboly na vstupe pri redukcii od samotnej gramatiky (množina *FOLLOW*).
- Pri *LR*(1) gramatikách závisia očakávané symboly na vstupe pri redukcii od toho, v akom *stave* sa nachádza rozpoznávanie pravých strán gramatiky (očakávané terminály na vstupe v momente redukcie).
- *LR*(1) teda zohľadňujú aj *kontext / vzťah* medzi jednotlivými pravidlami počas derivácie - zatiaľ čo *SLR*(1) nie.

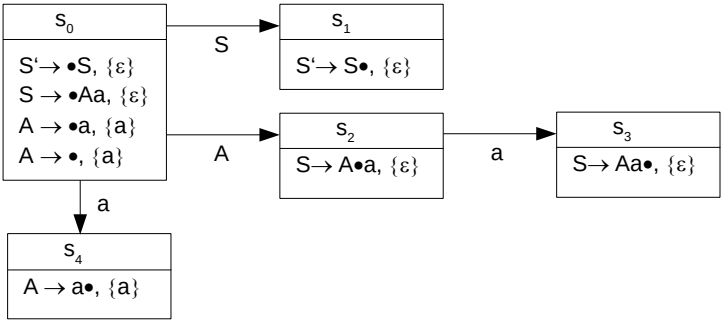
Niektoré gramatiky nie sú ani $LR(1)$

Uvažujme jednoduchú gramatiku $G = (\{S, A\}, \{a\}, P, S)$, kde pravidlá P:

1. $S \to Aa$
2. $A \to a$
3. $A \to \varepsilon$.

Táto gramatika generuje **konečný jazyk** $L = \{a, aa\}$. Navyše je **jednoznačná** a jazyk **deterministický**. Avšak gramatika nie je $LR(1)$, pretože:

LR(1)-automat



ACTION&GOTO

ACTION	s0	s1	s2	s3	s4
a	P/R3		P		R2
ε		A		R1	

GOTO	s0	s1	s2	s3	s4
S	s1				
A	s2				
a	s4		s3		

Poznámka ku konfliktu

- Vidíme, že v tabuľke *ACTION* je pre stav s_0 konflikt Presun-Redukcia pre symbol *a* na vstupe.
- Samozrejme, že jazyk, ktorý daná gramatika generuje, sa dá popísať aj inou - ekvivalentnou - gramatikou, ktorá je spracovateľná $LR(1)$ analyzátorom (alebo aj $LR(0)$ či $LL(1)$).

Použitá literatúra

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

Dedera, Ľ: *Počítačové jazyky a ich spracovanie*.

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