SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta Informatiky a Informačných Technológií

SEMESTRÁLNY PROJEKT

Tabuľkou riadený syntaktický analyzátor zhora nadol

Bc. Pavol Michálek

Bc. Michal Cihák

Softérové jazyky ZS 2014/2015
Cvičiaci: Ing. Peter Lacko, PhD.

Obsah

1 Zadanie

Úlohou tohto projektu je vytvoriť tabuľkou riadený syntaktický analyzátor (SA) zhora nadol pre definovanú gramatiku – littleXML.

LittleXML je jazyk pre podmnožinu XML súborov (napr. bez možnosti atribútov vo vnútri značiek). [Príloha A]

Syntax tohto jazyka je špecifikovaná notáciou BNF Backus-Naur Form. [Príloha B]

1.1 Percentuálny podiel autorov na projekte

Michal Cihák	50%
Pavol Michálek	50%

2 Analýza a Návrh

V tejto kapitole sa nachádzajú príklady viet daného jazyka, prepis jeho gramatiky z BNF do gramatických pravidiel, transformovanie gramatiky tak, aby bola LL(1), vypísanie množín FIRST a FOLLOW pre každý neterminál a tabuľka prechodov tejto gramatiky.

2.1 Príklady viet daného jazyka

V nasledujúcej tabuľke sa nachádzajú rôzne príklady jazyka littleXML.

<e/>		<_/>		<:/>	
<eA/>	<e:_-/>	<_AbC/>	<_.-_:/>	<:-._:/>	<:_flkblk-/>
<element>WORD</element>					
<?xml version=143.3000?><element>WORD ABCDEF @</element>					
<?xml version=143.3000?><element><element1><element2>WORDS</element2></element1></element>					

2.2 Prepis gramatiky z BNF do gramatických pravidiel

V nasledujúcej tabuľke sa nachádza prepísaná gramatika z BNF do gramatických pravidiel. Tabuľka vyjadruje skrátený zápis v tvare: α (ľavá strana) → β1|β2| ... |βn (pravá strana). Terminály sú podfarbené.

č.	ĽAVÁ STRANA	PRAVÁ STRANA		
1	xmldocument	element	xmldecl element	
2	xmldecl	<?xml version= vernumb ?>		
3	vernumb	number . number		
4	number	digit	digit number	
5	element	emptyelemtag	starttag words endtag	starttag elements endtag
6	emptyelemtag	< name />		
7	elements	ε	element	
8	starttag	< name >		
9	endtag	</ name >		
10	words	ε	word words	
11	word	char	char word	
12	char	letter	digit	@ % ! * §
13	name	letter name1	_ name1	: name1
14	name1	ε	namechar name1	
15	namechar	letter	digit	. - _ :
16	letter	A-Z	a-z	
17	digit	0-9		

Gramatika nespĺňa podmienky pre deterministickú analýzu v riadkoch 4,5,11, kde je potrebné vykonať ľavú faktorizáciu.

2.3 Transformácia gramatiky do LL(1)

V tejto kapitole sa nachádza transformácia gramatiky tak, aby spĺňala podmienky LL(1).

2.3.1 Odstránenie ľavej rekurzie

Odstrániť ľavú rekurziu nebolo potrebné, keďže sa v danej gramatike nenachádza.

2.3.2 Ľavá faktorizácia

V tomto bode je vykonaná ľavá faktorizácia a zároveň substitúcia na niektorých pravidlách a následne odstránenie nepotrebných pravidiel.

č.	ĽAVÁ STRANA	PRAVÁ STRANA		
1	xmldocument	element	<?xml version= number . number ?> element	
~~2~~	~~xmldecl~~	~~<?xml version= vernumb ?>~~		
~~3~~	~~vernumb~~	~~number . number~~		
2	**number**	**digit number1**		
3	**number1**	**ε**	**number ε**	
4	**element**	**< name tag**		
5	**element1**	**element2 endtag**	**endtag**	
6	**element2**	**word**	**element**	
7	**tag**	**/>**	**> element1**	
~~6~~	~~emptyelemtag~~	~~< name />~~		
~~7~~	~~elements~~	~~ε~~	~~element~~	
~~8~~	~~starttag~~	~~< name >~~		
8	endtag	</ name >		
~~10~~	~~words~~	~~ε~~	~~word words~~	
9	**word**	**char word1**		
10	**word1**	**ε**	**word**	
11	char	letter	digit	@ % ! * § ‖
12	name	letter name1	_ name1	: name1
13	name1	ε	namechar name1	
14	namechar	letter	digit	. - _ :
15	letter	A-Z	a-z	
16	digit	0-9		

2.3.3 Finálna LL(1) gramatika

ĽAVÁ STRANA	PRAVÁ STRANA											
xmldocument	1	element	2	<?xml version= number . number ?> element								
number	3	digit number1										
number1	4	ε	5	number								
element	6	< name tag										
element1	7	element2 endtag	8	endtag								
element2	9	word	10	element								
tag	11	/>	12	> element1								
endtag	13	</ name >										
word	14	char word1										
word1	15	ε	16	word								
char	17	letter	18	digit	19	@	%	!	*	§		
name	20	letter name1	21	_ name1	22	: name1						
name1	23	ε	24	namechar name1								
namechar	25	letter	26	digit	27	:	_	31	-	28	.	
letter	29	A-Z | a-z										
digit	30	0-9										

2.4 Množiny FIRST a FOLLOW

V nasledujúcich tabuľkách sú vypísané všetky množiny FIRST a FOLLOW pre každý neterminál v transformovanej gramatike.

2.4.1 FIRST pre každý neterminál

ĽAVÁ STRANA	množina FIRST
xmldocument	{ <, <?xml }
number	{0..9}
number1	{ε, 0..9}
element	{<}
element1	FI(element2) + FI(endtag) = FI(word) + FI(element) = { </, <, a-z, A-Z, 0-9, *, &, !, @, #, ... }
element2	{ <, a-z, A-Z, 0-9, *, &, !, @, #, ...}

tag	{ />, > }
endtag	{ </ }
word	FI(char) = { a-z, A-Z, 0-9, *, &, !, @, #, ...}
word1	{ ε, a-z, A-Z, 0-9, *, &, !, @, #, ...}
char	{ a-z, A-Z, 0-9, *, &, !, @, #, ...}
name	{ a-z, A-Z, : , _ }
name1	{ ε, a-z, A-Z, 0-9, ,, -, _, ; }
namechar	{ A..Z, a..z, 0..9, ., -, _, ; }
letter	{ A..Z, a..z }
digit	{ 0..9 }

2.4.2 FOLLOW pre každý neterminál

ĽAVÁ STRANA	FOLLOW
xmldocument	{ $ }
number	{ . , ?> }
number1	FO(number) = { . , ?> }
element	FO(element2) + FO(xmldocument) = FI(endtag) + { $ } = { </ , $ }
element1	FO(tag) = FO (element) = { </ , $ }
element2	FI(endtag) = { </ }
tag	FO(element) = { </ , $ }
endtag	FO(element1) = { </ , $ }
word	FO(element2) + FO(word1) = { </ }
word1	FO(word) = { </ }
char	FI(word1) = { </, a-z, A-Z, 0-9, *, &, !, @, #, ...}
name	FI(tag) + { > } = { >, /> }
name1	FO(name) = { >, /> }
namechar	FI(name1) + FO(name1) = { >, />, a-z, A-Z, 0-9, ,, -, _, ;}
letter	FO(namechar) + FI(name1) + FO(name1) + FO(char) = { >, />, </, a-z, A-Z, 0-9, ., -, _, ; ,*, &, !, @, #, ...}
digit	FO(namechar) + FO(char) + FI(number1) + FO(number) = { . , ?>, >, />, </, a-z, A-Z, 0-9, -, _, ;, *, &, !, @, #, ...}

2.4.3 Splnenie podmienok LL(1)

Pre každé pravidlo platí, že jeho množiny FIRST jeho pravých strán nezačínajú rovnakým terminálom.

Pre každé pravidlo, kde sa na pravej strane nachádza epsilon platí, že žiadne množiny FOLLOW ľavej strany týchto pravidiel nezačínajú rovnakým terminálom.

Z toho vyplýva, že transformovaná gramatika je LL(1).

2.5 Tabuľka prechodov transformovanej gramatiky

Na nasledujúcej strane sa nachádza tabuľka prechodov pre transformovanú gramatiku LL(1) .

Tabuľka prechodov

	<?xml version=	?>	.	<	</	>	/>	a-z, A-Z	0..9	_	:	@|%|!|*|§|' '|	-	$
xmldocument	<?xml version= number . number ?> element, 2			element, 1										
number									digit number1, 3					
number1		ε, 4	ε, 4						number, 5					
element				< name tag, 6										
element1				element2 endtag, 7	endtag, 8			element2 endtag, 7	element2 endtag, 7			element2 endtag, 7		
element2				element, 10				word, 9	word, 9			word, 9		
tag						> element1, 12	/>, 11							
endtag					</ name >, 13									
word								char word1, 14	char word1, 14			char word1, 14		
word1					ε, 15			word,16	word,16			word,16		
char								letter, 17	digit, 18			@|%|!|*|§|' ', 19		
name								letter name1, 20		_ name1, 21	: name1, 22			
name1			namechar name1, 24			ε, 23	ε, 23	namechar name1, 24	namechar name1, 24	namechar name1, 24	namechar name1, 24		namechar name1, 24	
namechar			. , 28					letter, 25	digit, 26	_ , 27	: , 27		- , 31	
letter								a-z, A-Z ,29						
digit									0-9, 30					
<?xml version=	V													
?>		V												
.			V											
<				V										
</					V									
>						V								
/>							V							
a-z, A-Z								V						
0..9									V					
_										V				
:											V			
@|%|!|*|§												V		
-|_|:													V	
#														A

3 Implementácia

Riešenie bolo implementované ako webová aplikácia. Algoritmus bol implementovaný v jazyku JavaScript s použitím knižnice jQuery. Lexikálny analyzátor bol implementovaný ako súčasť tabuľkou riadeného syntaktického analyzátora (SA). SA analyzuje postupnosť lexikálnych jednotiek na vstupe. Ak postupnosť zodpovedá vete jazyka, skončí SA prijatím, inak oznámi chybu. SA vypíše protokol o svojej činnosti, ktorý obsahuje všetky uskutočnené akcie pri parsovaní vstupu a prípadné chyby.

Implementácia umožňuje výber vstupu z preddefinovaných testovacích viet alebo zadanie vlastnej. SA bol otestovaný na všetkých vetách vytvorených v 2.1 a úspešne tieto vety prijal.

Prílohy

A littleXML

```
xmldocument ::= [xmldecl] element .
xmldecl ::= '<?xml'  'version=' vernumb '?>' .
vernumb ::= number '.' number .
element    ::= emptyelemtag | starttag (words |elements) endtag .
starttag ::= '<' name '>' .
endtag     ::= '</' name '>' .
words ::= {word} .
elements ::= [element] .
emptyelemtag ::= '<' name '/>' .
name ::= (letter | '_' | ':') {namechar} .
namechar ::=  letter | digit | '.' | '-' | '_' | ':' .
letter     ::= 'A' | .. | 'Z' | 'a' | .. | 'z' .
number     ::= digit {digit} .
digit ::= '0' | .. | '9' .
word ::= char {char} .
char ::= letter | digit | .. a ďalšie znaky okrem znakov < > (stačí
vymenovať niekoľko) .
```

B BNF Backus-Naur Form (BNF)

terminály

- ohraničené úvodzovkami, napr. "a" "+" "begin"
- niekedy sa namiesto úvodzoviek používajú apostrofy ‘ ’

neterminály

- uzavreté v lomených zátvorkách, napr. <neterminál>
- často sa lomené zátvorky pre neterminál vynechávajú

pravidlá

- končia bodkou (niekedy bodkočiarkou); bodka na konci pravidla sa niekedy vynecháva
- majú tvar <neterminál> := postupnosť_terminálov_a_neterminálov . napr.: <prog> := "begin" <stmt_list> "end" .
- niekedy sa namiesto := používa znak =, alebo znak →, alebo skupina znakov ::=

– ak sa nepoužívajú lomené zátvorky pre neterminál, pravidlo je napr. v tvare prog ::= "begin" stmt_list "end"

–

dodatočné symboly

| alternatíva (or)

napr. <number> := <integer> | <real>

( ) zoskupenie

napr. <exp> := <var> | (<var> "+" <var>)

[ ] nula alebo jeden výskyt

napr. <if> := "if "<test> "then" <stmt> ["else" <stmt>]

{ } opakovanie nula až veľa krát

napr. <ident_list> := <ident> { "," <ident>}

Transformačné pravidlá pre prepis z BNF do pravidiel

[E] sa transformuje na E | e

(E | F) G sa transformuje na E G | F G

E := F{aF} sa transformuje na

E ::= F | FaE

E := F{aG} sa transformuje na

E::=FX

X ::= aGX | e