



Universidade Estadual de Santa Cruz – UESC

Relatórios de Implementações de Autômato de Pilha para GLC

Docente César Alberto Bravo Pariente

Discente Matheus Miranda Brandão

Matrícula 201820065

Disciplina Compiladores.

Curso Ciência da Computação

Semestre 2022.2

Ilhéus – BA

2022

Índice

GLC LL	3
Produções:	3
Compilando e Executando	4
Testes	5
$m() \{ r(1); \}$	5
$m() \{ h=(x+y); r(0); \}$	6
$m() \{ (1-1); r(1); \}$	7
$m() \{ w(1) \{ (1/x); \}; r(1); \}$	8
$n() \{ (0/y); r(y); \} \quad g() \{ i=y; r(x); \} \quad m() \{ (1-x); r(0); \}$	10
$m() \{ w(x) \{ f(y) \{ k=(1+(1*0)); \}; \}; r(0); \}$	13
Link para download	15
Referências	16

GLC LL

O projeto consiste na implementação em C de um algoritmo que busca simular um autômato pilha que reconhece a linguagem gerada por uma gramática livre de contexto. A execução do código recebe como entrada a palavra e retorna a Tabela de Parsing, suas Produções e a struct referente a Árvore de Análise.

O autômato implementado ignora o token ' '.

Produções:

$p_1: S \rightarrow M \mid GM \mid NGM$

$p_4: N \rightarrow n() \{ C; r(E); \}$

$p_5: G \rightarrow g() \{ C; r(E); \}$

$p_6: M \rightarrow m() \{ C; r(E); \}$

$p_7: E \rightarrow 0 \mid 1 \mid x \mid y \mid (EXE)$

$p_{12}: X \rightarrow + \mid - \mid * \mid /$

$p_{16}: C \rightarrow h=E \mid i=E \mid j=E \mid k=E \mid z=E \mid (EXE) \mid w(E) \{ C; \} \mid f(E) \{ C; \} \mid o(E;E;E) \{ C; \}$

Compilando e Executando

Para a execução não é necessário o uso de nenhuma dependência, basta compilá-lo normalmente.

```
$ gcc proj2_a.c -o proj2_a
```

Ao executar é necessário digitar o nome do arquivo destino contendo as palavras, caso contrário resultará em erro.

Exemplo:

```
$ ./proj2_a examples/inputs.txt
```

Neste projeto pode-se adicionar num .txt todas as palavras separadas por uma quebra de linha.

Exemplo:

```
m(){ r(1); }
m(){ h=(x+y); r(0); }
m(){ (1-1); r(1); }
m(){ w(1) { (1/x); }; r(1); }
n() { (0/y); r(y); } g() { i=y; r(x); } m() { (1-x); r(0); }
m(){w(x){f(y){k=(1+(1*0));};};r(0);}
```

Testes

Caso a entrada dada seja incorreta o programa irá imprimir uma mensagem de erro, sua tabela, produções e a árvore sintática até o momento de erro, então pulará a linha para ler a próxima palavra. Como outputs temos a tabela de parsing, as produções a árvore sintática (no formato '[index | mapeamento | token]'), foi considerado o pior caso, onde temos uma árvore n-ária de 12. A árvore foi feita seguindo o modelo de árvore compacta.

Para criação de palavras compatíveis com a linguagem gerada pela GLC foi utilizado o website "CFG Developer".

m(){ r(1); }

Palavra 1:

i	Qi	Token	Stack	Pi
0	Q0	m		-
1	Q1	m	M	P1
2	Q1	((P6
2	Q1	()	-
2	Q1)	{	-
2	Q1	{	C	-

Erro.

Producoes: P1 P6

Arvore: [0|0| S], [1|1| M], [2|13| m], [3|14| (], [4|15|)], [5|16| {], [6|17| C], [7|18| ;], [8|19| r], [9|20| (], [10|21| E], [11|22|)], [12|23| ;], [13|24| }]

m(){ h=(x+y); r(0); }

Palavra 2:

i	Qi	Token	Stack	Pi
0	Q0	m		-
1	Q1	m	M	P1
2	Q1	((P6
2	Q1	()	-
2	Q1)	{	-
2	Q1	{	C	-
3	Q1	=	=	P16
3	Q1	=	E	-
4	Q1	x	E	P11
5	Q1	+	X	P9
6	Q1	y	E	P12
7	Q1))	P10
7	Q1)	;	-
7	Q1	;	r	-
7	Q1	r	(-
7	Q1	(E	-
8	Q1))	P7
8	Q1)	;	-
8	Q1	;	}	-
8	Q1	}		-

Palavra aceita.

Producoes: P1 P6 P16 P11 P9 P12 P10 P7

Arvore: [0|0| S], [1|1| M], [2|13| m], [3|14| (], [4|15|)], [5|16| {], [6|17| C], [7|18| ;], [8|19| r], [9|20| (], [10|21| E], [11|22|)], [12|23| ;], [13|24| }], [14|205| h], [15|206| =], [16|207| E], [17|2485| (], [18|2486| E], [19|2487| X], [20|2488| E], [21|2489|)], [22|29833| x], [23|29845| +], [24|29857| y], [25|253| 0]

m(){ (1-1); r(1); }

Palavra 3:

i	Qi	Token	Stack	Pi
0	Q0	m		-
1	Q1	m	M	P1
2	Q1	((P6
2	Q1	()	-
2	Q1)	{	-
2	Q1	{	C	-
3	Q1	1	E	P21
4	Q1	-	X	P8
5	Q1	1	E	P13
6	Q1))	P8
6	Q1)	;	-
6	Q1	;	r	-
6	Q1	r	(-
6	Q1	(E	-
7	Q1))	P8
7	Q1)	;	-
7	Q1	;	}	-
7	Q1	}		-

Palavra aceita.

Producoes: P1 P6 P21 P8 P13 P8 P8

Arvore: [0|0| S], [1|1| M], [2|13| m], [3|14| (], [4|15|)], [5|16| {], [6|17| C], [7|18| ;], [8|19| r], [9|20| (], [10|21| E], [11|22|)], [12|23| ;], [13|24| }], [14|205| (], [15|206| E], [16|207| X], [17|208| E], [18|209|)], [19|2473| 1], [20|2485| -], [21|2497| 1], [22|253| 1]

m(){ w(1) { (1/x); }; r(1); }

Palavra 4:

i	Qi	Token	Stack	Pi
0	Q0	m		-
1	Q1	m	M	P1
2	Q1	((P6
2	Q1	()	-
2	Q1)	{	-
2	Q1	{	C	-
3	Q1	((P22
3	Q1	(E	-
4	Q1))	P8
4	Q1)	{	-
4	Q1	{	C	-
5	Q1	1	E	P21
6	Q1	/	X	P8
7	Q1	x	E	P15
8	Q1))	P9
8	Q1)	;	-
8	Q1	;	}	-
8	Q1	}	;	-
8	Q1	;	r	-
8	Q1	r	(-
8	Q1	(E	-
9	Q1))	P8
9	Q1)	;	-
9	Q1	;	}	-

9 Q1 } -

Palavra aceita.

Producoes: P1 P6 P22 P8 P21 P8 P15 P9 P8

Arvore: [0|0| S], [1|1| M], [2|13| m], [3|14| (], [4|15|)], [5|16| {], [6|17| C], [7|18| ;],
[8|19| r], [9|20| (], [10|21| E], [11|22|)], [12|23| ;], [13|24| }], [14|205| w], [15|206| (],
[16|207| E], [17|208|)], [18|209| {], [19|210| C], [20|211| ;], [21|212| }], [22|2485| 1],
[23|2521| (], [24|2522| E], [25|2523| X], [26|2524| E], [27|2525|)], [28|30265| 1],
[29|30277| /], [30|30289| x], [31|253| 1]

n() { (0/y);r(y); } g() { i=y;r(x); } m() { (1-x);r(0); }

Palavra 5:

i	Qi	Token	Stack	Pi
0	Q0	n		-
1	Q1	n	N	P3
2	Q1	((P4
2	Q1	()	-
2	Q1)	{	-
2	Q1	{	C	-
3	Q1	0	E	P21
4	Q1	/	X	P7
5	Q1	y	E	P15
6	Q1))	P10
6	Q1)	;	-
6	Q1	;	r	-
6	Q1	r	(-
6	Q1	(E	-
7	Q1))	P10
7	Q1)	;	-
7	Q1	;	}	-
7	Q1	}	G	-
8	Q1	((P5
8	Q1	()	-
8	Q1)	{	-
8	Q1	{	C	-
9	Q1	=	=	P16
9	Q1	=	E	-

10	Q1	;	;	P10
10	Q1	;	r	-
10	Q1	r	(-
10	Q1	(E	-
11	Q1))	P9
11	Q1)	;	-
11	Q1	;	}	-
11	Q1	}	M	-
12	Q1	((P6
12	Q1	()	-
12	Q1)	{	-
12	Q1	{	C	-
13	Q1	1	E	P21
14	Q1	-	X	P8
15	Q1	x	E	P13
16	Q1))	P9
16	Q1)	;	-
16	Q1	;	r	-
16	Q1	r	(-
16	Q1	(E	-
17	Q1))	P7
17	Q1)	;	-
17	Q1	;	}	-
17	Q1	}		-

Palavra aceita.

Producoes: P3 P4 P21 P7 P15 P10 P10 P5 P17 P10 P9 P6 P21 P8 P13
P9 P7

Arvore: [0|0| S], [1|1| N], [2|2| G], [3|3| M], [4|13| n], [5|14| (], [6|15|)], [7|16| {],
[8|17| C], [9|18| ;], [10|19| r], [11|20| (], [12|21| E], [13|22|)], [14|23| ;], [15|24| }],
[16|205| (], [17|206| E], [18|207| X], [19|208| E], [20|209|)], [21|2473| 0],
[22|2485| /], [23|2497| y], [24|253| y], [25|25| g], [26|26| (], [27|27|)], [28|28| {],
[29|29| C], [30|30| ;], [31|31| r], [32|32| (], [33|33| E], [34|34|)], [35|35| ;],
[36|36| }], [37|349| i], [38|350| =], [39|351| E], [40|4213| y], [41|397| x], [42|37| m],
[43|38| (], [44|39|)], [45|40| {], [46|41| C], [47|42| ;], [48|43| r], [49|44| (],
[50|45| E], [51|46|)], [52|47| ;], [53|48| }], [54|493| (], [55|494| E], [56|495| X],
[57|496| E], [58|497|)], [59|5929| 1], [60|5941| -], [61|5953| x], [62|541| 0]

$m()\{w(x)\{f(y)\{k=(1+(1*0));\};\};r(0);\}$

Palavra 6:

i	Qi	Token	Stack	Pi
0	Q0	m		-
1	Q1	m	M	P1
2	Q1	((P6
2	Q1	()	-
2	Q1)	{	-
2	Q1	{	C	-
3	Q1	((P22
3	Q1	(E	-
4	Q1))	P9
4	Q1)	{	-
4	Q1	{	C	-
5	Q1	((P23
5	Q1	(E	-
6	Q1))	P10
6	Q1)	{	-
6	Q1	{	C	-
7	Q1	=	=	P16
7	Q1	=	E	-
8	Q1	1	E	P11
9	Q1	+	X	P8
10	Q1	(E	P12
11	Q1	1	E	P11
12	Q1	*	X	P8
13	Q1	0	E	P14

14	Q1))	P7
14	Q1))	-
14	Q1)	;	-
14	Q1	;	}	-
14	Q1	}	;	-
14	Q1	;	}	-
14	Q1	}	;	-
14	Q1	;	r	-
14	Q1	r	(-
14	Q1	(E	-
15	Q1))	P7
15	Q1)	;	-
15	Q1	;	}	-
15	Q1	}		-

Palavra aceita.

Producoes: P1 P6 P22 P9 P23 P10 P19 P11 P8 P12 P11 P8 P14 P7 P7

Arvore: [0|0| S], [1|1| M], [2|13| m], [3|14| (], [4|15|)], [5|16| {], [6|17| C], [7|18| ;],
[8|19| r], [9|20| (], [10|21| E], [11|22|)], [12|23| ;], [13|24| }], [14|205| w],
[15|206| (], [16|207| E], [17|208|)], [18|209| {], [19|210| C], [20|211| ;], [21|212| }],
[22|2485| x], [23|2521| f], [24|2522| (], [25|2523| E], [26|2524|)], [27|2525| {],
[28|2526| C], [29|2527| ;], [30|2528| }], [31|30277| y], [32|30313| k], [33|30314| =],
[34|30315| E], [35|363781| (], [36|363782| E], [37|363783| X], [38|363784| E],
[39|363785|)], [40|4365385| 1], [41|4365397| +], [42|4365409| (], [43|4365410| E],
[44|4365411| X], [45|4365412| E], [46|4365413|)], [47|52384921| 1],
[48|52384933| *], [49|52384945| 0], [50|253| 0]

Link para download

Código fonte e exemplos encontram-se para download no seguinte link:

<https://github.com/MatBrands/Compiladores/tree/master/Proj2/Proj2a%20Compactado>

Referências

<https://web.stanford.edu/class/archive/cs/cs103/cs103.1156/tools/cfg/>