Московский авиационный институт

(национальный исследовательский институт)

Институт «Компьютерные науки и прикладная математика»

**Курсовая работа**

**«Разработать и реализовать МП-автомат с альтернативами»**

**по курсу**

**«Системы программирования»**

**IV семестр**

*Студент:* Белоносов К.А.*Группа: М8О-208Б-21*

*Руководитель:* Семёнов А. С.

*Оценка:*

*Дата:*

**Москва, 2023**

Разработка МП-автомата с альтернативами связана с некоторыми трудностями, возникающими при разборе. Мы можем пойти по такому пути разбора, который не даст нам разобрать нашу входную последовательность. Поэтому нужно придумать алгоритм с откатами.

Также необходимо разработать свой вариант грамматики с более удобным вариантом задания альтернативных правил.

Для начала разработаем алгоритм преобразования КС грамматики с альтернативами в МП-автомат

**Алгоритм построение МП-автомата из КС-грамматики с альтернативами:**

**Вход:**

**Выход:**

**Q = {q, }**

T

**Г =**

**F =**

**Z0 =** S

**foreach**()

**foreach**()

Каждую конфигурацию мы получаем из существующих правил и терминальных символом, также добавляем конфигурацию переходящую в заключительное состояние

Теперь разработаем алгоритм разбора МП-автомата с альтернативами, чтобы обойти все возможные разборы и найти нужный путь воспользуемся алгоритмом поиска в ширину (BFS).

**Алгоритм разбора МП-автомата с альтернативами**

**Вход: ,**  – Входная цепочка, где

**Выход: Bool** – принадлежность цепочки языку

q = q0

D =

D =

**while**()

**if**()

D =

**else**

**if**()

**else**

**continue**

Данный алгоритм хорошо решает задачу разбора, также может находить несколько вариантов разбора, при не однозначном разборе.

Рассмотрим пример разбора скобочной последовательности:

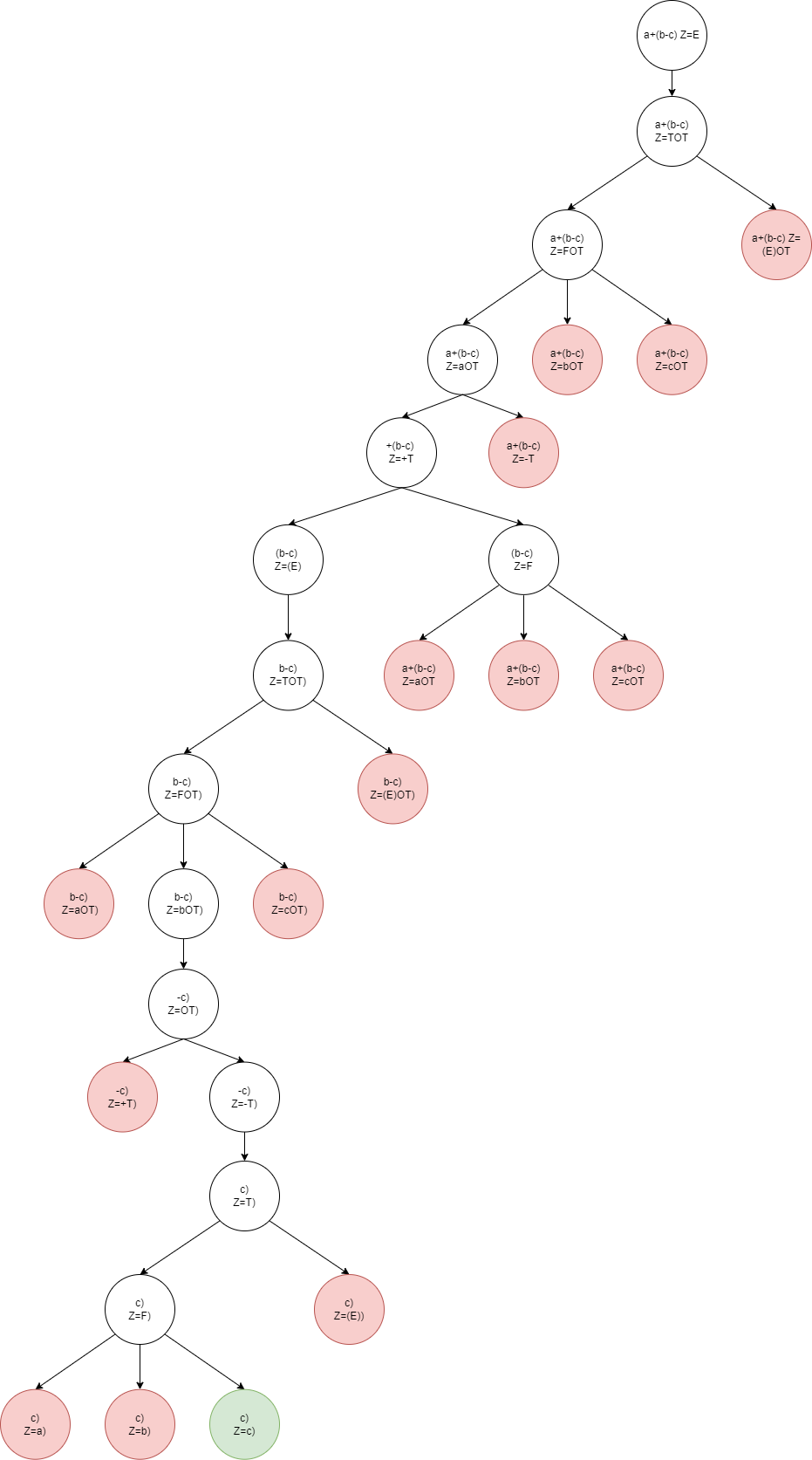
**G** = (**T**, **V**, **P**, **S0**), где

**T** = {a, b, c, -, +, (, )};

**V** = {E, T, F, P, S, O};

**S0** = E.

Построим дерево разбора для входной цепочки: a+(b-c)



При обходе этого дерева мы могли пойти по неправильному пути разбора на высоте 6 и не разобрать нашу входную цепочку. Но благодаря обходу в ширину мы смогли посмотреть все пути найти

**Листинг:**

***PDAAlt.cs***

1. using RaGlib.Automata;
2. using RaGlib.Core;
3. using System;
4. using System.Collections.Generic;
5. using static System.Formats.Asn1.AsnWriter;
6. namespace RaGlib
7. {
8. // Класс для создания списка альтернативных правил
9. public class DeltaConfig
10. {
11. int countOfAlt;
12. public List<DeltaQSigmaGamma> listOfDeltaRule;
13. public DeltaConfig()
14. {
15. listOfDeltaRule = new List<DeltaQSigmaGamma>();
16. countOfAlt = 0;
17. }
18. public void addRule(string LHSQ, string LHSS, string LHSZ, List<Symbol> RHSQ, List<Symbol> RHSZ)
19. {
20. listOfDeltaRule.Add(new DeltaQSigmaGamma(LHSQ, LHSS, LHSZ, RHSQ, RHSZ));
21. countOfAlt++;
22. }
23. }
24. // МП-автомат с альтернативами
25. public class PDAAlt : PDA
26. {
27. public PDAAlt(
28. List<Symbol> Q,
29. List<Symbol> Sigma,
30. List<Symbol> Gamma,
31. Symbol Q0,
32. Symbol z0,
33. List<Symbol> F) :
34. base(Q, Sigma, Gamma, Q0, z0, F)
35. {
36. }
37. // Метод для создания дельта правил из дельта конфигурации
38. public void addDeltaRule(DeltaConfig configuration)
39. {
40. foreach (var rule in configuration.listOfDeltaRule)
41. {
42. this.Delta.Add(rule);
43. }
44. }
45. // Создать делата правило
46. public override void addDeltaRule(string LHSQ, string LHSS, string LHSZ, List<Symbol> RHSQ, List<Symbol> RHSZ)
47. {
48. this.Delta.Add(new DeltaQSigmaGamma(LHSQ, LHSS, LHSZ, RHSQ, RHSZ));
49. }
50. // Найти все дельта правила с указанной левой частью
51. public List<DeltaQSigmaGamma> findAllDelta(Symbol z)
52. {
53. var deltaList = new List<DeltaQSigmaGamma>();
54. foreach (var delta in this.Delta)
55. if (delta.LHSZ.symbol.Equals(z.symbol)) deltaList.Add(delta);
56. return deltaList;
57. }
58. // Запуск разбора (В основе лежит поиск в ширину BFS)
59. public override bool Execute(string str)
60. {
61. currState = this.Q0;
62. int i = 0;
63. str = str + "ε";
64. DeltaQSigmaGamma delta = null;
65. Queue<Tuple<DeltaQSigmaGamma, Stack<Symbol>, int>> q = new Queue<Tuple<DeltaQSigmaGamma, Stack<Symbol>, int>>();
66. // Задаем начальное состояние
67. List<DeltaQSigmaGamma> start = findAllDelta(Z.Peek());
68. foreach (var v in start)
69. {
70. q.Enqueue(new Tuple<DeltaQSigmaGamma, Stack<Symbol>, int>(v, new Stack<Symbol>(Z), i));
71. }
72. // Запускаем разбор
73. while (q.Count != 0)
74. {
75. Tuple<DeltaQSigmaGamma, Stack<Symbol>, int> state = q.Dequeue();
76. delta = state.Item1;
77. Stack<Symbol> store = new Stack<Symbol>(state.Item2);
78. int j = state.Item3;
79. Console.WriteLine("Current string: " + str.Substring(j));
80. Console.WriteLine(" step 1");
81. delta.Debug();
82. bool next = true;
83. if (!delta.LHSS.symbol.Equals("ε"))
84. {
85. for (; j < str.Length;)
86. {
87. if (str[j].Equals("ε")) return true;
88. Console.WriteLine(" step 2 " + str[j] + " " + delta.LHSS);
89. if (str[j].ToString().Equals(delta.LHSS.symbol))
90. {
91. Console.WriteLine(" step 3 " + str[j]);
92. if (delta.RHSZ[0].symbol.Equals("ε"))
93. store.Pop();
94. else
95. store.Push(delta.RHSZ[0]);
96. } else
97. {
98. next = false;
99. }
100. j++;
101. if (str[j] == 'ε') return true;
102. break;
103. }
104. }
105. else
106. {
107. store.Pop();
108. delta.RHSZ.Reverse();
109. foreach (var symbol in delta.RHSZ)
110. store.Push(symbol);
111. delta.RHSZ.Reverse();
112. }
113. if(next)
114. {
115. List<DeltaQSigmaGamma> vertex = findAllDelta(store.Peek());
116. foreach (var v in vertex)
117. {
118. q.Enqueue(new Tuple<DeltaQSigmaGamma, Stack<Symbol>, int>(v, new Stack<Symbol>(store), j));
119. }
120. } else
121. {
122. Console.WriteLine("[X] Fail branch");
123. }
124. }
125. return false;
126. }
127. }
128. }

***GrammarAlt.cs***

1. using RaGlib.Automata;
2. using RaGlib.Core;
3. using RaGlib.Grammars;
4. using System;
5. using System.Collections.Generic;
6. namespace RaGlib
7. {
8. public class GrammarAlt : Grammar
9. {
10. public GrammarAlt(List<Symbol> T, List<Symbol> V, string S0) : base(T, V, S0)
11. {
12. }
13. public GrammarAlt(List<Symbol> T, List<Symbol> V, List<Production> production, string S0) : base(T, V, S0)
14. {
15. }
16. public GrammarAlt() : base()
17. {
18. }
19. public new PDAAlt Transform()
20. {
21. var Q = new List<Symbol>() { "q" };
22. var Sigma = new List<Symbol>(this.T);
23. var F = new List<Symbol>();
24. var Gamma = new List<Symbol>(this.V);
25. foreach (var t in this.T)
26. Gamma.Add(t);
27. var Q0 = "q0";
28. var Z = new Stack<Symbol>();
29. Z.Push(new Symbol("z0"));
30. Z.Push(this.S0);
31. var Delta = new List<DeltaQSigmaGamma>();
32. DeltaQSigmaGamma delta = null;
33. var rhsq = new List<Symbol>() { "q" };
34. var rhsz = new List<Symbol>() { "ε" };
35. foreach (var p in this.P)
36. {
37. if (!this.V.Contains(p.LHS.symbol))
38. {
39. Console.Write($"This grammar cant have rule: {p.LHS.symbol} -> ");
40. foreach (var s in p.RHS)
41. {
42. Console.Write(s.symbol);
43. }
44. Console.Write("\n");
45. return null;
46. }
47. foreach (var t in p.RHS)
48. {
49. if (!this.T.Contains(t) && !this.V.Contains(t))
50. {
51. Console.Write($"This grammar cant have rule: {p.LHS.symbol} -> ");
52. foreach (var s in p.RHS)
53. {
54. Console.Write(s.symbol);
55. }
56. Console.Write("\n");
57. return null;
58. }
59. }
60. }
61. foreach (var p in this.P)
62. {
63. delta = new DeltaQSigmaGamma("q", "ε", p.LHS.symbol, rhsq, p.RHS);
64. Delta.Add(delta);
65. }
66. foreach (var t in this.T)
67. {
68. delta = new DeltaQSigmaGamma("q", t.symbol, t.symbol, rhsq, rhsz);
69. Delta.Add(delta);
70. }
71. delta = new DeltaQSigmaGamma("q0", "ε", "z0", new List<Symbol> { "qf" }, new List<Symbol> { "ε" });
72. Delta.Add(delta);
73. PDAAlt KA = new PDAAlt(Q, Sigma, Gamma, Q0, "z0", F);
74. KA.Delta = Delta;
75. KA.Z.Push(this.S0);
76. return KA;
77. }
78. public new void AddRule(string LeftNoTerm, List<Symbol> RHS)
79. {
80. var TmpRHS = new List<Symbol>();
81. foreach (var s in RHS)
82. {
83. if (s.symbol == "|")
84. {
85. this.P.Add(new Production(LeftNoTerm, TmpRHS));
86. TmpRHS = new List<Symbol>();
87. }
88. else
89. {
90. TmpRHS.Add(s);
91. }
92. }
93. if (TmpRHS.Count > 0)
94. {
95. this.P.Add(new Production(LeftNoTerm, TmpRHS));
96. }
97. }
98. }
99. }

***Program.cs***

1. case "7.13": // МП-автомат с альтернативами
2. {
3. var altGramm = new GrammarAlt(new List<Symbol>() { "a","b","c", "-", "+", "(", ")"},
4. new List<Symbol>() { "E", "T", "F", "P", "S", "O"},
5. "E");
6. altGramm.AddRule("E", new List<Symbol>() { "T", "O", "T"});
7. altGramm.AddRule("O", new List<Symbol>() { "+", "|", "-"});
8. altGramm.AddRule("T", new List<Symbol>() { "F", "|", "(", "E", ")"});
9. altGramm.AddRule("F", new List<Symbol>() { "a", "|", "b", "|", "c"});
11. var altPDA = altGramm.Transform();
12. altPDA.Debug();
13. Console.WriteLine("\nEnter the line :");
14. // Example: a+(b-c)
15. Console.WriteLine(altPDA.Execute(Console.ReadLine()).ToString());
16. }
17. break;