Lab 2 documentation

Grammar:

-List<string> N;

-List<string> Epsilon;

-List<string> P;

Finite Automata:

-List<string> lambda;

-List<string> Q;

-List<string> Epsilon;

-List<string> F;

Convert grammar to finite automata:

public bool TryConvertToFinite()

{

if (!this.grammar.CheckIfRegular())

return false;

this.finite = new FiniteAutomata();

finite.SetInitialState("S");

finite.SetAlphabet(Copy(this.grammar.GetSetOfTerminals()));

finite.SetFiniteSetOfStates(Copy(this.grammar.GetSetOfNonTerminals()));

int ok = 1;

foreach (string item in this.grammar.GetSetOfProductions())

{

string[] itemSplit = item.Split(new string[] { "->" }, StringSplitOptions.None);

string[] RHSSplit = itemSplit[1].Split('|');

for (int i = 0; i < RHSSplit.Length; i++)

{

char[] chars = RHSSplit[i].ToCharArray();

string str = this.grammar.ComposeString(chars);

if (itemSplit[0].Equals('S') && str.Equals("$"))

finite.SetFinalState("S");

if (str.Equals("") && ok.Equals(1))

{

finite.AddK();

ok = 0;

}

if (str.Equals(""))

{

finite.AddToLambda(itemSplit[0] + "," + chars[0].ToString() + "=" + "&");

}

else

{

finite.AddToLambda(itemSplit[0] + "," + chars[0].ToString() + "=" + str);

}

}

}

return true;

}

Check if grammar is regular:

public bool CheckIfRegular()

{

Regex regex = new Regex(this.Lower);

foreach(string item in this.P)

{

string[] itemSplit = item.Split(new string[] { "->" }, StringSplitOptions.None);

string[] RHSSplit = itemSplit[1].Split('|');

for (int i = 0; i < RHSSplit.Length; i++)

{

char[] chars = RHSSplit[i].ToCharArray();

if (!regex.IsMatch(chars[0].ToString()))

return false;

if (!itemSplit[0].Equals("S") && ComposeString(chars).Equals("$"))

return false;

if (itemSplit[0].Equals("S") && ComposeString(chars).Equals("$"))

this.flag = 0;

if (ComposeString(chars).Equals("S") && flag == 0)

return false;

}

}

return true;

}

Convert finta automata to grammar:

public void ConvertToGrammar()

{

this.grammar = new Grammar();

this.grammar.SetNonTerminals(Copy(this.finite.GetFiniteSetOfStates()));

this.grammar.SetTerminals(Copy(this.finite.GetAlphabet()));

List<string> products = new List<string>();

foreach(string item in this.finite.GetLambda())

{

string[] itemSplit = item.Split('=');

string[] LHSSplit = itemSplit[0].Split(',');

if (itemSplit[1].Equals("&"))

products.Add(LHSSplit[0] + "->" + LHSSplit[1]);

products.Add(LHSSplit[0] + "->" + LHSSplit[1] + itemSplit[1]);

}

if (this.finite.CheckInitState())

products.Add(this.finite.GetInitialState() + "->$");

this.grammar.SetProductions(products);

}