Construção de aplicação para geração dos conjuntos FIRST e FOLLOW a partir de uma GLC em linguagem C++

Acácia dos Campos da Terra[1], **Gabriel Batista Galli**[1],
João Pedro Winckler Bernardi[1], **Vladimir Belinski**[1]

[1]Ciência da Computação – Universidade Federal da Fronteira Sul (UFFS)
Caixa Postal 181 – 89.802-112 – Chapecó – SC – Brasil

{terra.acacia, g7.galli96, winckler.joao, vlbelinski}@gmail.com

Abstract. *This paper describes the implementation of an application in C++ programming language for the generation of sets FIRST and FOLLOW from a Context-Free Grammar (CFG). The particularities and operation of the application, such as its planning and code will be commented and analyzed in the work. At the end it will be checked the operation of the application by conducting tests.*

Resumo. *O presente trabalho descreve a implementação de uma aplicação em linguagem de programação C++ para a geração dos conjuntos FIRST e FOLLOW a partir de uma Gramática Livre de Contexto. As particularidades e funcionamento da aplicação, tal como seu planejamento e código serão comentados e analisados no trabalho. No final será verificado o funcionamento da aplicação através da realização de testes.*

1. Introdução

O presente trabalho objetiva descrever a construção de uma aplicação em linguagem de programação C++ para geração dos conjuntos FIRST e FOLLOW a partir de uma Gramática Livre de Contexto lida de um arquivo de entrada.

Na Seção 2 será realizada uma breve definição do que consistem os conjuntos FIRST e FOLLOW. Por sua vez, na Seção 3 será realizada uma descrição da aplicação, de seu planejamento e implementação. Em sequência, os testes que verificam seu funcionamento serão comentados na Seção 4. Por fim, na Seção 5 poderão ser encontradas as conclusões acerca do trabalho.

2. Conjuntos FIRST e FOLLOW

Conforme [Furtado] os conjuntos FIRST e FOLLOW podem ser definidos como:

Definição 2.1. Seja α uma sequência qualquer gerada por G. Definimos como sendo $FIRST(\alpha)$ o conjunto de símbolos terminais que iniciam α ou sequências derivadas (direta ou indiretamente) de α.

Observações: Se $\alpha = \varepsilon$ ou $\alpha \Rightarrow^* \varepsilon$, então $\varepsilon \in FIRST(\alpha)$.

Definição 2.2. Definimos $FOLLOW(A)$, para todo $A \in N$ como sendo o conjunto de símbolos terminais que podem aparecer imediatamente após A em alguma forma sentencial de G.

Observações: considere N como sendo o conjunto finito de símbolos não-terminais.

A descrição do cálculo dos conjuntos FIRST e FOLLOW pode ser verificado na descrição das funções `first` e `follow` apresentadas na Seção 3.

3. Descrição, Planejamento e Implementação da Aplicação

A aplicação desenvolvida possui como objetivo a geração dos conjuntos FIRST e FOLLOW a partir de uma Gramática Livre de Contexto, em notação BNF, lida de um arquivo de entrada (texto). Após a geração dos conjuntos ambos são salvos em arquivos `.csv` separadamente. Cabe destacar que para a representação de ε (épsilon) foi utilizado o símbolo $\&$.

A implementação da aplicação pode ser encontrada em três arquivos: `ndfa.cpp`, `automata.h` e `automata.cpp`.

No arquivo `ndfa.cpp` se encontra a função `main`, onde é realizada a leitura do arquivo de entrada, a impressão de mensagens no prompt a fim de permitir ao usuário o acompanhamento da aplicação e realizadas chamadas às funções responsáveis pelas ações executadas pela aplicação.

Por sua vez, no arquivo `automata.h` podem ser encontradas as definições das constantes utilizadas no trabalho, das estruturas de dados e os protótipos das funções implementadas em `automata.cpp`. Cabe destacar que a `struct symbol` representa um símbolo de uma produção, tanto um terminal quanto um não terminal, sendo que existe uma flag que é setada para 1 quando for terminal e 2 caso contrário. A `struct transition` representa uma transição e é constituída em um vetor de símbolos.

Em `automata.cpp` são encontradas 7 funções, as quais serão comentadas uma a uma a seguir.

Inicialmente, em relação à função `readgrammar` tem-se que sua funcionalidade consiste na leitura de uma Gramática Livre de Contexto na notação BNF e construção de seu respectivo Autômato Finito, o qual poderá ser não determinístico.

Por sua vez, a função `first` calcula os conjuntos FIRST da GLC passada como entrada, iterando sobre todas as transições de todos os estados. Se o primeiro símbolo da transição atual for terminal ou ε o adiciono no conjunto FIRST do estado que está sendo analisado. Caso seja não-terminal $(*)$ analiso todos os símbolos do conjunto FIRST desse não-terminal, adicionando o símbolo ao conjunto FIRST do estado inicialmente analisado caso seja um terminal e realizando o seguinte procedimento caso seja um ε: olhando para o próximo símbolo que segue o não terminal que levou a análise de seu conjunto FIRST, é adicionado o símbolo ao conjunto FIRST do estado inicialmente analisado caso esse seja um terminal, adicionado ε se não houver mais nenhum símbolo e realizado todo o procedimento de tratamento de símbolo não-terminal a partir de $(*)$ caso assim o seja. Tal processo é repetido até que nenhuma alteração nos conjuntos FIRST ocorra na iteração.

No que lhe diz respeito, a função `follow` calcula os conjuntos FOLLOW da GLC passada como entrada, adicionando \$ (marca de final de sentença) no conjunto FOLLOW de 'S' (símbolo inicial da gramática em questão) e também iterando sobre todas as transações de todos os estados. Consistindo em duas etapas, na primeira são ignorados os

símbolos que dão nome a regra (os que precedem ::=) e analisados todos os não-terminais. Verificando o primeiro símbolo após um não-terminal: se esse for terminal o adiciono no conjunto FOLLOW do não-terminal que o antecede; se for não-terminal analiso o conjunto FIRST desse não-terminal e adiciono todos os terminais (ε não é adicionado) no conjunto FOLLOW do não-terminal que o antecede; se não houver mais símbolos nada é feito. Essa etapa é realizada uma única vez.

Na segunda etapa, a qual é repetida até que nenhuma alteração ocorra nos conjuntos FOLLOW na iteração, são analisados todos os não-terminais que correspondem ao último símbolo de suas produções. Nessa etapa, o FOLLOW desses símbolos passa a ser seu FOLLOW antigo somado ao FOLLOW do símbolo que dá nome a sua regra. Além disso, se ε fizer parte do conjunto FIRST desse não-terminal e caso o símbolo que o antecede também seja um não-terminal então o FOLLOW desse não-terminal antecendente também será seu FOLLOW antigo somado ao FOLLOW do símbolo que dá nome a sua regra.

A respeito da construção dos conjuntos também cabe destacar que é a estrutura global `map<string, set<char> > frst, fllw;` que mapeia um estado (a string) para um conjunto de caracteres (os símbolos) que compõem seus conjuntos FIRST e FOLLOW.

Em relação à `printfirst` tem-se que sua funcionalidade consiste na impressão do conjunto FIRST gerado, enquanto que a função `printfllw` realiza a impressão do conjunto FOLLOW construído. A respeito da função `printfa` tem-se que ela realiza a impressão do autômato recebido como entrada em notação BNF.

Por fim, a função `csv` salva o conjunto FIRST gerado em um arquivo denominado `first.csv` e o conjunto FOLLOW gerado em um arquivo denominado `follow.csv`.

4. Testes

Para a realização dos testes foram criados quatro arquivos, nomeados `test.in`, `frst.in`, `frst2.in` e `frst3.in`. Cada um desses arquivos armazena uma Gramática Livre de Contexto a partir da qual foram gerados os conjuntos FIRST e FOLLOW relacionados a essa GLC.

A partir da realização dos testes pôde ser verificado que os resultados obtidos condizem com as saídas esperadas, essas que podem ser verificadas em dois arquivos `.csv` gerados, um para o conjunto FIRST e outro para o conjunto FOLLOW. Isso demonstra que a aplicação construída atende aos objetivos a qual se propõe.

5. Conclusão

Conclui-se que a aplicação desenvolvida atende ao objetivo ao qual se propõe, consistindo em uma aplicação em linguagem de programação C++ para geração dos conjuntos FIRST e FOLLOW a partir de uma Gramática Livre de Contexto (GLC).

No trabalho foi apresentada uma definição do que consistem os conjuntos FIRST e FOLLOW, assim como realizada a exposição do planejamento, funcionamento, particularidades e implementação de uma aplicação para sua geração.

Por fim, foram citados e comentados os arquivos utilizados nos testes para a verificação da funcionalidade da aplicação desenvolvida.

Referências

Furtado, O. J. V. *Linguagens formais e compiladores*. Universidade Federal de Santa Catarina, Centro Tecnológico, Departamento de Informática e de Estatística.