Trabalho de Linguagens Formais, Autômatos e Computabilidade

Implementação de Gramáticas Regulares

Rafael Francisco Ferreira, Luís Fernando S. Gaspar

Ciência da Computação – Universidade Estadual do Paraná (UNESPAR)
Apucarana – PR – Brasil
2017

rafaelfrancisco_97@hotmail.com, gasparfluis@gmail.com

Resumo

Este documento apresenta um relatório técnico sobre a implementação de uma ferramenta que simula uma Gramática Regular, iformada pelo usuário, e retorna ao mesmo o resultado das derivações feitas para chegar à palavra testada. É apresentada também uma introdução a Gramáticas Regulares, seguida das decisões de projeto e conclusões sobre a implementação.

1. Introdução

Assim como nas linguagens naturais, as linguagens formais também podem ser especificadas através de “gramáticas” a ela associadas.

As gramáticas, também conhecidas como dispositivos generativos, dispositivos de síntese ou ainda dispositivos de geração de cadeias, constituem sistemas formais baseados em regras de substituição, através das quais é possível sintetizar o conjunto das cadeias que compõem uma determinada linguagem.

As regras assim definidas especificam combinações válidas dos símbolos que compõem o alfabeto e o conjunto dessas regras que formam uma “gramática” deve ser suficiente para permitir a elaboração de qualquer palavra, frase ou discurso corretamente construída em qualquer linguagem, e não deve permitir a construção de qualquer cadeia que não pertença a linguagem.

Noam Chomsky, considerado “o pai da linguística moderna”, formalizou as gramáticas generativas pela primeira vez em 1956, ele as classificou em tipos, conhecida hoje como Hierarquia de Chomsky. De forma sucinta, a diferença entre esses tipos, é que eles têm cada vez mais rigorosas regras

de produção. Dois importantes tipos são as gramáticas livres de contexto e as gramáticas regulares, sendo esta, objeto principal desse estudo.

Baseado nos estudos de Chomsky sobre as gramáticas e o conhecimento sobre ele adquiridos em sala, propõe-se o desenvolvimento de uma ferramenta que simule o funcionamento de uma Gramática Regular, com ênfase em seus dois tipos, são eles GLUD (Gramática Linear Unitária a Direita) e GLUE (Gramática Linear Unitária a Esquerda).

2. Formalizando uma Gramática

Uma gramática é uma quádrupla ordenada G = (V, T, P, S) onde:

V são variáveis utilizadas para a construção da gramática;

T são todos os terminais, ou seja, o alfabeto da linguagem;

P conjunto finito de pares, denominado regras de produção tal que a primeira componente é palavra de $(V \quad T)^{¿}$ e a segunda componente é a palavra de $(V \quad T)^{¿}$;

S elemento de partida para as derivações $S \quad V$.

Uma regra de produção (a, β) é representada por a →β. As regras definem as condições de geração das palavras da linguagem. Em outras palavras, a é uma cadeia qualquer constituídas por V.

A aplicação de uma regra de produção é denominada derivação de uma palavra. A aplicação sucessiva de regras de produção permite derivar as palavras da linguagem representada pela gramática.

A derivação é a substituição de uma subpalavra de acordo com uma regra de produção.

3. Gramática Regular

Uma gramática regular é o mecanismo gerador para as linguagens regulares, ou seja, as linguagens que podem ser reconhecidas por um Autômato Finito Determinístico ou um Autômato Finito Não Determinístico, podendo ser dividido em quatro tipo, sendo eles:

- **Gramática Linear a Direita**
  Todas as regras de produção seguem o formato $A \rightarrow wB$ ou $A \rightarrow w$.
- **Gramática Linear a Esquerda**
  Todas as regras de produção seguem o formato $A \rightarrow Bw$ ou $A \rightarrow w$.
- **Gramática Linear Unitária a Direita**
  Se todas as regras de produção são como a Gramática Linear a Direita, adicionalmente $|w| \leq 1$.
- **Gramática Linear Unitária a Esquerda**
  Se todas as regras de produção são como a Gramática Linear a Esquerda, adicionalmente $|w| \leq 1$.

4. Decisões de Projeto para a Implementação

4.1 Linguagem

A linguagem escolhida para a implementação foi Java, por sua portabilidade e pelo nível de conhecimento sobre a mesma dos membros da equipe de desenvolvimento. A linguagem Java é multiplataforma e também de fácil entendimento à qualquer programador, trazendo benefícios de portabilidade ao usuário final e de fácil entendimento a todos que participam do desenvolvimento em grupo.

4.2 Interface com o Usuário

A interface com o usuário foi feita utilizando formulários, baseados em Jframes java, oferecidos pela linguagem, que tornam fácil e rápida a captura dos dados e também oferecem uma melhor apresentação, comparados às ferramentas que rodam em consoles e terminais, bem como facilitam o tratamento dos dados e erros do usuário, controles de acesso às informações e praticidade em testes.

4.3 Estruturas de Dados

A implementação toda utiliza apenas dois tipos de estruturas de dados para armazenar os diversos tipos de informações capturadas do usuário. Uma delas é o vetor. Vetores foram utilizados para armazenamento temporário de informações, como Strings ou números inteiros que posteriormente seriam utilizados para checagem de erros, ou seriam armazenados em Listas. As listas são o segundo tipo utilizado. Nelas, foram armazenados todos os conjuntos necessários para o funcionamento do programa, como: conjunto de Variáveis, conjunto de Terminais, Regras de uma variável, passos de derivação, armazenamento temporário de regras para reordenação de listas, entre outros.

4.4 Tratamentos de Erro e Decisões Relevantes

Algumas decisões relevantes foram tomadas. Dentre elas, a decisão de tratar todos os erros possíveis por parte do usuário antes de qualquer teste ou adição de regras foi tomada. Este tratamento de erros consiste em uma série de checagens e condições, onde, se uma delas não é satisfeita, o programa para e informa o erro ao usuário, somente continuando a execução após o erro ser corrigido.

Uma particularidade da implementação, era a necessidade de uma prioridade nas regras de produção. Para não incomodar o usuário e não

comprometer a praticidade, foi implementada uma função de reordenação das regras, que reordena a lista de regras da variável toda vez que a lista for modificada, mantendo sempre a ordem de prioridade necessária para o bom funcionamento da aplicação.

4.5 Dificuldades Encontradas

Foram encontradas algumas dificuldades durante a implementação. Dentre elas, à de implementar gramáticas ambíguas do tipo wB|wB, com Bs diferentes, onde a ferramenta vai optar sempre pela primeira das duas regras que for reconhecida por ela. As ambiguidades do tipo w|wB, são reconhecidas normalmente.

Devido ao grande número de casos e opções de caminhos, o grupo de desenvolvimento não conseguiu implementar e tratar todos os casos, portanto, o funcionamento poderá variar, dependendo da gramática de entrada. Mas no geral, as gramáticas GLUD e GLUE tem bom funcionamento em boa parte dos casos, desde que não envolvam ambiguidade.

5. Conclusão

O desenvolvimento passou por várias etapas e várias modificações, só tendo seu fim ao atingir o prazo de entrega. Ao analisar o que foi implementado, notamos que os casos tratados funcionam bem para as GLUD e GLUE que não são ambíguas. Para as ambíguas, funciona, porém a ferramenta pode entrar em *loop* ou errar o resultado.

A ferramenta foi desenvolvida com base em aprendizagens adquiridas em sala de aula, tendo seu funcionamento limitado ao conhecimento dos desenvolvedores.

Podemos concluir, com base na ferramenta final entregue ao fim do prazo, que seu funcionamento ainda contém falhas, variando conforme a gramática inserida e com o seu tipo, e nota-se também, a falta da decisão por parte da ferramenta em casos de ambiguidades na gramática. Porém, boa parte das gramáticas conseguem ser processadas. Todos as falhas ainda poderão ser corrigidas em versões futuras da ferramenta.

6. Referências

Menezes, P. B. ; Linguagens Formais e Autômatos. 3ª edição. Ed. Sagra Luzzato. 2000.

Vieira, Newton José. Introdução aos Fundamentos da Computação. São Paulo. Pioneira Thomson Learning. 2006;

Ramos, Marcus Vinícius. Linguagens Formais e Autômatos. Universidade Federal do Vale do São Francisco. 2008;

Material e exemplos fornecidos em aula pelo Professor Maurílio Campano Jr., na matéria de “Linguagens Formais, Autômatos e Computabilidade” - Ciência da Computação, Universidade Estadual do Paraná, Campus Apucarana, 2017.