DCC - UFMG

Proposta de Projeto Orientado I

**Uma biblioteca de analisadores
sintáticos**

Aluno: Thadeu Lima de Souza Cascardo

Orientadora: Mariza Andrade da Silva Bigonha

Data: 18 de Julho de 2005

1. Introdução

A produção de compiladores, interpretadores de linguagens de programação e outras tarefas como a leitura de arquivos de configuração ou de entrada para qualquer aplicação, ou mesmo a interpretação de linguagens naturais envolvem um processo denominado análise sintática.

Em geral, esse processo envolve a especificação da linguagem em uma gramática, na maioria das vezes, em uma forma denominada BNF (Backus-Naur Form). Ferramentas de geração de compiladores são utilizadas para traduzir uma linguagem fonte na forma de uma gramática em código objeto. Especificamente, em relação à análise sintática, tem-se como exemplos o bison [1] e styx [2] para C, Javacc [3], Cup [4] e ANTLR [5] para Java, entre muitos outros.

2. Motivação

O processo de análise sintática pode oferecer algumas limitações. Uma delas é que o mesmo é executado em tempo de compilação, o que impediria uma possível extensão ou adaptação da linguagem em tempo de execução.

Outra limitação é o algoritmo utilizado para a análise sintática ou a geração do analisador. Alguns algoritmos são mais eficientes em termos de espaço ou tempo e uma escolha poderia ser feita por parte do programa ou do usuário em tempo de execução dependendo dos recursos disponíveis no ambiente.

Por fim, algumas gramáticas não podem ser utilizadas com alguns analisadores, e, geralmente, o desenvolvedor deve reescrevê-la utilizando alguns algoritmos conhecidos.

3. Objetivos

Uma biblioteca de analisadores e geradores não teria as limitações descritas na Seção 2. A implementação de diferentes algoritmos permitiria a escolha entre otimizar tempo e espaço por parte do desenvolvedor em tempo de execução, utilizando o analisador mais apropriado.

Este trabalho se propõe a estudar a viabilidade de implementar alguns dos algoritmos aqui enumerados sob uma mesma interface, permitindo uma maior flexibilidade para o desenvolvedor de aplicações.

4. Trabalhos Relacionados

As linguagens podem ser classificadas segundo seu poder de expressão. Entre elas, temos as linguagens regulares e as linguagens livres de contexto, as mais utilizadas e referenciadas na literatura, pois podem ser analisadas de forma eficiente por programas de computador.

Estas linguagens podem ser representadas de diversas formas. Entre as representações mais populares, temos a BNF (Backus Naur Form) [6]. Para linguagens livres de contexto, há ainda a TDPL (top-down parsing language) [7] e, recentemente, a PEG (Parsing Expression

Grammar) [8]. Para as linguagens regulares, existem as expressões regulares.

As BNF's permitem a descrição de linguagens ambíguas, enquanto as PEG's não. As PEG's também podem ser utilizadas para descrever algumas linguagens que não são livres de contexto, mais poderosas.

Estas linguagens podem ser analisadas utilizando diferentes algoritmos. Para as linguagens livres de contexto, temos duas classes de algoritmos, os "top-down" e os "bottom-up". Geralmente, os "bottom-up" são mais eficientes, enquanto os "top-down" são mais restritivos.

Entre os algoritmos "top-down", temos o "recursive descent parser" [9] [10], que executa em tempo exponencial, e o "packrat parser" [11], que, apesar de executar em tempo linear, consome muito espaço em memória. Ambos algoritmos não reconhecem gramáticas recursivas à esquerda e ambíguas.

Os algoritmos "bottom-up" utilizam uma técnica denominada "shift-reduce". Esses algoritmos são muito eficientes, mas dependem de ferramentas para a geração das tabelas de análise sintática.

Há ainda o "Earley parser" [12] e o "CYK parser" [13] [14] [15], que podem analisar as linguagens descritas por qualquer gramática e executam em tempo cúbico.

Os geradores de analisadores são algoritmos que geram código capaz de analisar uma linguagem a partir de uma descrição da mesma, seja em BNF, TDPL ou PEG. A TDPL e a PEG são mais apropriadas para a geração de algoritmos "top-down".

5. Plano de Trabalho

O trabalho proposto está organizado em quatro etapas. São elas:

- Desenvolvimento de alguns algoritmos de análise sintática: os algoritmos "top-down", "bottom-up", "Earley parser" e "CYK parser" enumerados na Seção 4 são candidatos à implementação.
- Desenvolvimento de interface para representação de linguagens: uma API (Application Programming Interface) para a representação de linguagens como BNF's e PEG's será implementada.
- Desenvolvimento de analisadores de representações como BNF's e PEG's: estas representações serão analisadas utilizando a própria biblioteca desenvolvida, permitindo que aplicações obtenham descrições de linguagens a partir de uma entrada fornecida pelo usuário. Assim, geradores de código ou máquinas de busca podem ser implementadas facilmente.
- Testes de linguagens de programação e arquivos de configuração: testes serão desenvolvidos utilizando a biblioteca para a análise de linguagens de programação, arquivos de configuração e outras linguagens.

5.1. Cronograma

No trabalho proposto, as quatro etapas descritas nesta seção serão executadas da seguinte forma:

Como Projeto Orientado I, dois algoritmos serão estudados e implementados e a API será desenvolvida. Testes de linguagens de fácil análise, como Scheme, serão realizados.

Como trabalho futuro, a ser realizado em Projeto Orientado II, outros algoritmos serão implementados além dos analisadores de BNF's e PEG's e testes de linguagens mais elaboradas serão realizados.