Ferramenta de apoio às fases iniciais do ensino de linguagens formais e compiladores

Gustavo Prado Alkmim[1]; Bráulio Adriano de Mello[2]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)

[2] Ciência da Computação – Universidade Federal da Fronteira Sul (UFFS)
Chapecó, Santa Catarina, 89812-130

galkmim@gmail.com, bmello@dcc.ufla.br

Abstract. *This article presents the LexSint, a tool to support the teaching of formal languages and automata that allows you to generate lexical and syntactic recognizers from examples of code from the language is being construction. The structure of the examples is of simpler construction compared with the construction of the GLC, which facilitates the initial stages of learning. It simplifies the process of creating analyzers simple, the process of lexical and syntactic analysis is shown step by step tool.*

Resumo. *Este artigo apresenta o LexSint, uma ferramenta de apoio ao ensino de linguagens formais e autômatos que permite gerar reconhecedores (léxicos e sintáticos) a partir de exemplos de código da linguagem em construção. A estrutura dos exemplos é mais simples e intuitiva para alunos iniciantes nesta matéria facilitando as etapas iniciais do aprendizado. Além de simplificar o processo de criação de analisadores simples, o processo de análise léxica e sintática é mostrado passo a passo pela ferramenta.*

1. Introdução

O ensino da disciplina de compiladores é de grande importância para a formação superior em computação. A aplicação do formalismo de linguagens no projeto e construção de compiladores integra o conteúdo sugerido para o núcleo de tecnologias da computação. No entanto, a suficiente compreensão, pelos alunos, dos conceitos necessários para a construção de compiladores consome significativo tempo e esforço [Vegdahl 2001] [Alfred V. Aho 2007]. Uma primeira dificuldade é a demora para que o aluno interprete a diferença entre manipulação de símbolos, característica das máquinas reconhecedoras, e a manipulação de valores onde o desenvolvimento de programas é a atividade fim. Uma segunda dificuldade é o alcance de indicadores que apontam erros em etapas intermediárias da construção dos analisadores. Por exemplo, na construção de uma GLC o aluno identifica erros ou problemas essencialmente após o desenvolvimento de todo o processo de construção da GLC, da implementação dos reconhecedores e do seu uso para validação de um código de teste.

Esta realidade justifica o uso de ferramentas de apoio no ensino de linguagens formais e de compiladores [Juliano Henrique Foleiss 2009]. Este trabalho apresenta o projeto do LexSint, uma ferramenta educacional para a estudo de analisadores léxico e sintático que permite ao aluno gerar a GLC da linguagem através de exemplos das estruturas de código desejáveis. O LexSint tem como dados de entrada um arquivo texto

contendo exemplos destas estruturas. A partir deste arquivo, o LexSint extrai o automato finito e a gramática livre de contexto e gera os reconhecedores. O principal benefício do LexSint é exemplificar ou ilustrar quais os impactos de uma determinada estrutura de programa sobre a GLC que especifica formalmente a sintaxe linguagem de programação. Deste modo, o aluno reduz o tempo e esforço necessários para relacionar as produções da GLC com as estruturas sintáticas aceitas pela linguagem. Por exemplo, no LexSint o aluno pode criar um exemplo de código para iteração aninhada, gerar a GLC automaticamente e estudar as características da GLC. Esta característica contribui para diminuir o tempo e esforço do aluno iniciante para entender como estruturas sintáticas podem ser formalmente representadas através de regras gramaticais e produções.

A seção 2 comenta algumas ferramentas já existentes que podem ser utilizadas no ensino da disciplina de compiladores. Em seguida, na seção 3, o artigo apresenta o LexSint, suas características, estrutura e a fase atual de desenvolvimento.

2. Trabalhos Relacionados

Embora fundamental na formação superior em computação, formalismo de linguagens e compiladores são matérias cuja compreensão dos formalismos envolvidos e das fases de compilação não é simples. Esta característica justifica a proposição de ferramentas de apoio com fins didáticos. Esta seção aborda brevemente algumas ferramentas já existentes e algumas das suas principais características.

O C-gem [Jerônimo Backes 2006] auxilia na criação de analisadores léxico, sintático e semântico. Para a geração do analisador léxico, o usuário deve informar a definição de um autômato finito. O analisador sintático gerado pela ferramenta é do tipo SLR e o usuário deve criar a gramática livre de contexto utilizando a interface do C-gen.

O VAST (*Visualization of Abstract Syntax Tree*) [Francisco J. Almeida-Martínez 2008] é uma ferramenta gráfica para a geração e a visualização de árvores de sintaxes abstratas independentemente do gerador de *parser*. Uma de suas limitações é que o gerador de parser deve ser implementado apenas na linguagem Java.

O Verto [Carlos Sérgio Schneider 2005] é um compilador educativo escrito na linguagem Java e elaborado na forma de um software livre sob a licença GPL (GNU Public License). O usuário informa um código-fonte escrito em uma linguagem similar ao português com uma sintaxe próxima a da linguagem C e o compilador Verto gera o código macro-assembler intermediario e o código-objeto na linguagem César [Weber 2001]. No entanto, o aluno não visualiza etapas intermediárias do processo.

Outras ferramentas bastante exploradas que auxiliam na criação de compiladores são Lex (A Lexical Analyzer Generator) [Doug Brown 1995], Yacc [Doug Brown 1995], flex [flex 2010] e bison [Bison 2010]. O Lex e o Flex são geradores de analisadores léxicos. O Yacc e o Bison são geradores de analisadores sintáticos. O uso destas ferramentas em ambientes didáticos tem como ponto fraco comum a complexidade para alunos iniciantes. As ferramentas voltadas para uso em produção não contemplam questões didáticas com finalidade de ensino.

4. O LexSint

O LexSint é uma ferramenta de apoio ao ensino de linguagens formais e compiladores. Ela tem o aluno iniciante nesta matéria como principal beneficiado. A principal característica é a geração automática do autômato finito e da gramática livre de contexto a partir de exemplos da estrutura do código da linguagem alvo. LexSint interpreta os

exemplos de código escritos (Figura 1) pelo aluno e apresenta como resultado uma GLC válida. Afinada com os propósitos de ensino, LexSint permite ao aluno acompanhar, em passos intermediários, a montagem das produções e regras gramaticais facilitando a compreensão. O LexSint oferece suporte para a análise léxica e para a sintática.

```
** GRUPOS **
COMANDOS for if while
SINAIS + = - * /
SINAIS_COMPARACAO < > ==

** PADROES **
for (i=1;i!!SINAIS_COMPARACAO!!0;i++) do { !!COMANDOS!! }
```

Figura 1 – Exemplo de sintaxe do arquivo 'exemplos'

O módulo gerador de gramáticas, inicialmente, gera um arquivo com estrutura pré formatada para a construção das regras gramaticais, local onde o aluno insere os tokens e os padrões das estuturas sintáticas da linguagem. A partir de cada linha do arquivo de exemplos o LexSint cria regras automaticamente e as insere no arquivo estruturado já no formato previsto para a GLC. O processo de geração da GLC não verifica questões de validade da linguagem, apenas reproduz o conjunto de regras a partir dos exemplos. Eventuais erros cometidos pelo usuário na construção dos exemplos também serão transferidos para a GLC. Ao identificar a ocorrência dos erros, o usuário pode alterar o conjunto de exemplos e gerar novamente as regras da GLC.

A primeira seção do arquivo 'exemplos', iniciada pelo rótulo **GRUPOS** (Figura 1), identifica os tokens. A primeira palavra de cada linha é o identificador do grupo e as outras palavras são os tokens pertencentes a este grupo. Na geração da GLC, cada grupo de tokens (descritos em uma linha) se torna uma regra. O nome do grupo é o símbolo ‘não terminal’ da regra e cada palavra é uma produção da regra.

A segunda seção, iniciada no rótulo **PADROES** (Figura 1) possui os exemplos das estruturas sintáticas que devem ser aceitas. Cada linha desta seção representa uma estrutura a ser analisada pelo reconhecedor. Cada linha representa uma regra na GLC para facilitar o entendimento do aluno. O símbolo não terminal da regra é a primeira palavra e toda a linha é a produção da regra. Na primeira seção cada palavra é uma produção diferente, ou seja, o 'não terminal' pode gerar várias tokens diferentes enquanto na segunda seção cada linha gera uma única produção. A Figura 2 mostra a GLC gerada pelo arquivo da Figura 1. A seção de grupos facilita a escrita do arquivo de exemplos pelo usuário na criação de regras semelhantes ao mesmo tempo em que permite a criação de estruturas mais complexas tais como iterações aninhadas.

```
COMANDOS => for | if | while
SINAIS => + | = | - | * | /
SINAIS_COMPARACAO => < | > | ==
FOR => for (i=1;iSINAIS_COMPARACAO0;i++) do {!COMANDOS}
INICIO => FOR
```

Figura 2. A GLC gerada pelo arquivo de exemplos da Figura 1

```
Expressão: for (i=1;i<0;i++) do { while }
1) INICIO
2) FOR
3) for (i=1;iSINAIS_COMPARACAO0;i++) do { COMANDOS }
4) for (i=1;i<0;i++) do {COMANDOS}
5) for (i=1;i<0;i++) do {while}
```

Figura 3. Derivação da expressão utilizando GLC da Figura 2

A Figura 3 ilustra a derivação de uma expressão utilizando a gramática livre de contexto da Figura 2.

A versão atual do LexSint não possui interface gráfica. O aluno edita o arquivo de exemplos e executa o LexSint passando este arquivo como argumento. Todo o processo de geração dos analisadores léxico e sintático é mostrado passo a passo para o aluno, sendo mostrados, por exemplo, a geração dos conjuntos *first* e *follow* e a tabela de itens válidos, entre outros. A ferramenta está preparada para realizar a análise de um código fonte qualquer fornecido pelo usuário. Nesta opção o usuário visualiza passo a passo todo o processo de reconhecimento com objetivos de aprendizagem. Ao final do processo a ferramenta apresenta o resultado do trabalho de reconhecimento léxico e sintático.

5 Conclusão

Neste artigo foi apresentado o LexSint, uma ferramenta voltada para o ensino de linguagens formais e compiladores. O LexSint oferece suporte para as etapas de reconhecimento léxico e sintático tendo como objetivo principal apoiar a aprendizagem.

A principal característica que o difere de outras ferramentas já desenvolvidas é a geração automática do automato finito e a gramatica livre de contexto usando um arquivo de exemplos fornecido pelo aluno. A construção de exemplos, inicialmente, é mais intuitiva para o aluno se comparada com a construção de regras sintáticas da GLC. Esta estratégia permite a criação de analisadores para linguagens mais complexas com menos esforço para estudante, fator importante para o ensino dos fundamentos de linguagens formais e compiladores.

Todas as etapas da análise, incluindo a geração dos analisadores, são mostradas ao usuário em detalhes, o que também interfere de forma positiva no aprendizado. A versão atual da ferramenta oferece suporte para a análise léxica e sintática usando os exemplos como fonte das regras e o gerador da GLC está em fase de implementação.

Referências

Aho, A.V., Sethi, R. and Ullman, J.D. (2007). Compilers: Principles, Techniques and Tools. Pearson Education, 2nd edition.

Bison (2010). GNU parser generator. Disponível em http://www.gnu.org/software/bison/, acessado em 03/2010.

Schneider, C.S., Passerino, L.M. e Oliveira, R. F. (2005). Compilador educativo verto: ambiente para aprendizagem de compiladores. RENOTE.

Brown, D. and Levine, J. (1995). lex e yacc. O'Reilly Media, Inc., 2nd edition. flex (2010). The Fast Lexical Analyzer. Disponível em http://flex.sourceforge.net/, acessado em 03/10.

Almeida-Martínez, F.J. and Urquiza-Fuentes, J. N. V.-I. (2008). Vast - visualization of abstract syntax trees within language processors courses. SoftVis 2008 - Proceedings of the 4th ACM Symposium on Software visuallization, pages 209–210.

Jerônimo Backes, A. D. (2006). C-gen – ferramenta de apoio ao estudo de compiladores. WEI - XIV Workshop sobre Educação em Computação, pages 71–78.

Foleiss, J. H., Guilherme Puglia Assunção, Cruz, E. H. M., Gonçalves, R.A.L. e Feltrin, V.D (2009). Scc: Um compilador C como ferramenta de ensino de compiladores. WEAC2009 - Workshop Educação em Arquitetura de Computadores, pages 15–22.

Weber, R. F. (2001). Fundamentos de arquitetura de computadores. Sagra Luzzatto, 3rd edition.