C-gen – Ambiente Educacional para Geração de Compiladores

Jerônimo Backes[1], Alessandra Dahmer[1]

[1]Universidade de Santa Cruz do Sul (UNISC) – Santa Cruz do Sul – RS – Brasil.
germanobax@yahoo.com.br, adahmer@unisc.br

Abstract. *The compilers discipline is considered complex due to the various techniques involved. Bounded with this factor, practically no tools with an exclusively educational approach on this subject are available, that enable students to explore and visualize the entire compilation process. To supply this need, the C-gen tool was developed, offering an usable graphical user interface, allowing the user to define the main phases of a compilation, exhibiting the recognition process step-to-step. To support free access to knowledge, this tool is available to use as free software, which allows the entire community to enhance it.*

Resumo. *A disciplina de compiladores é considerada complexa devido às inúmeras técnicas envolvidas. Aliado a este fator, praticamente inexistem ferramentas exclusivamente educacionais focadas neste tema, que permitam aos alunos explorarem e visualizarem todo o processo de compilação. Para suprir esta necessidade, a ferramenta C-gen foi desenvolvida, fornecendo uma interface gráfica usável, que possibilita a definição das principais fases da compilação por parte do usuário, exibindo o funcionamento do processo de reconhecimento passo a passo. Com o objetivo de incentivar o livre acesso ao conhecimento, esta ferramenta está disponível para uso como software livre, tornando-a passível de melhorias por parte de toda a comunidade.*

1 Introdução

O processo de comunicação entre o ser humano e o computador se dá através da tradução de uma determinada linguagem – definida e compreendida pelo ser humano – para a do computador. Este processo requer um tradutor, capaz de reconhecer a linguagem de entrada e produzir um texto equivalente em outra linguagem. Dá-se o nome de compilador ao tradutor capaz de converter textos de uma linguagem de entrada em código de máquina. [Aho, 1986]

Na disciplina de compiladores, são estudadas inúmeras técnicas utilizadas neste processo, comumente dividido em fases, ou passos. Cada um é responsável por tarefas específicas que podem ser implementadas com o auxílio de diversos métodos.[Kakde, 2002] A quantidade destes, aliado à sua complexidade inerente, dificultam o processo de aprendizado. Não obstante, a falta de meios que permitam a aplicação dos conceitos rápida e facilmente, obscurecem a compreensão do funcionamento de um determinado passo, bem como de todo o processo de compilação.

A implementação de um pequeno compilador, ou de partes deste, é o método comum utilizado para a aplicação do conteúdo visto em sala de aula. Mas tal procedimento somente pode ser solicitado quando o aluno conhecer claramente a

metodologia a ser empregada para tal fim.

Muitas ferramentas podem ser utilizadas para a confecção de compiladores [Mak, 1996], mas a maioria é destinada ao uso profissional, ou são incapazes de ser utilizadas em mais de um passo da compilação. Nenhuma apresenta uma interface gráfica apropriada, que oriente o aluno e que tenha capacidade de exibir o funcionamento de todo o processo.

Para sanar esta carência, a ferramenta C-gen foi desenvolvida, permitindo que o aluno possa utilizá-la enquanto aprende, possibilitando a rápida experimentação da teoria e a satisfatória visualização de seu funcionamento, facilitando a compreensão do conteúdo. Ademais, a ferramenta permite a aplicação de diferentes métodos de reconhecimento para cada passo da compilação. São considerados passos de análise léxica, sintática e semântica; sendo que este último pode ser utilizado para geração de código.

As seções a seguir compreendem a análise das ferramentas já existentes e utilizadas em ambientes educacionais (seção 2), as funcionalidades projetadas para o sistema (seção 3), sua implementação (seção 4) e o sistema principal (seção 5). Finalmente, a conclusão do presente trabalho é apresentada (seção 6).

2 Ferramentas Analisadas

Visando identificar quais recursos deveriam ser implementados em uma ferramenta educacional, foram avaliadas algumas ferramentas cujo propósito é auxiliar a confecção de compiladores, descobrindo os pontos fracos e fortes de cada uma, considerando seu uso em um ambiente educacional.

Foram analisados aspectos de interface (design, interação e usabilidade) e aprendizagem do conteúdo (facilidade de aplicação da fundamentação teórica, capacidade de aplicação das diversas ramificações teóricas e relação entre uso e aprendizado) das ferramentas COCO/R [Mössenböck, 2005], Flex[Paxson, 2005] e Bison[Donelly, 2005]. Tais ferramentas foram escolhidas para análise por serem muito utilizadas no meio acadêmico na implementação de compiladores. O resultado é exibido na tabela 1.

Tabela 1. Análise das Ferramentas COCO/R, Flex e Bison

Ferramenta analisada	***Característica***					
	Design da interface com o usuário	Facilidade de uso da ferramenta	Interação com o usuário	Facilidade de aplicação da fundamentação teórica	Capacidade de aplicação das ramificações teóricas	Relação entre uso e aprendizado
Coco/R						
Flex						
Bison						
Legenda	 Bom		 Regular/Limitado		 Ruim/Inexistente	

Os geradores de compiladores avaliados são amplamente conhecidos por sua eficiência, mas conforme a análise realizada, não fornecem um ambiente propício para sua utilização educacional, especialmente se o objetivo for acompanhar o aluno no

processo de aprendizado. Conclui-se, portanto, que uma ferramenta que se proponha a facilitar a compreensão da teoria e orientar a sua aplicação certamente seria bem-vinda no meio acadêmico.

3 Funcionalidades do Sistema

Este projeto objetivou o atendimento dos seguintes requisitos:

1. Fornecer um ambiente que possibilite a definição de analisadores léxicos, sintáticos e semânticos;
2. Disponibilizar uma interface gráfica adequada para a definição dos passos listados no item 1;
3. Permitir o acompanhamento do processo de reconhecimento das fases da compilação, passo a passo;
4. Gerar, para cada passo criado, um objeto salvo em disco capaz de realizar o processo de reconhecimento, para integração em quaisquer aplicativos;
5. Permitir a expansão das funcionalidades do programa através da instalação de *plugins*, que implementem algum passo da compilação com técnicas diferentes;
6. Fornecer interfaces de comunicação padrão entre os passos, permitindo a criação de *plugins* compatíveis com os módulos já criados [Birsan, 2005] e responsáveis por outras etapas da compilação;

Assim, a arquitetura deste sistema é organizada de tal forma que cada passo do processo possa ser executado independentemente do posterior, comunicando-se com este através de uma representação intermediária, conforme ilustra a figura 1.

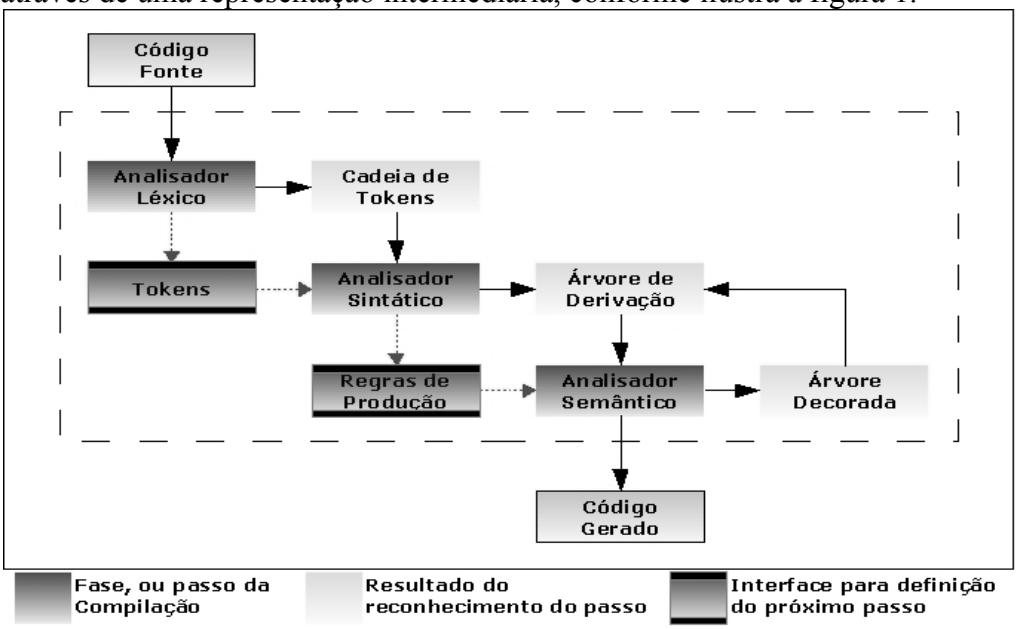

Figura 1. Arquitetura do protótipo

Onde os passos da compilação são implementados como *plugins* que devem reconhecer as respectivas entradas e produzir saídas correspondentes à sua função. Além disso, um *plugin* para um dado passo deve fornecer sua definição, informando o que é capaz de reconhecer, para que o subseqüente seja construído com base em suas capacidades.

Assim, um analisador léxico deve reconhecer um código fonte e produzir uma seqüência de *tokens*, bem como informar quais *tokens* é capaz de reconhecer. Um analisador sintático deve reconhecer a seqüência de *tokens* e produzir uma árvore de derivação, bem como informar suas regras de produção. Finalmente, um analisador semântico deve reconhecer uma árvore de derivação e produzir uma árvore decorada, além de mensagens de erro, ou código fonte se desejado.

4 Implementação

Seguindo os moldes definidos pelo projeto do sistema, as funcionalidades, implementadas na linguagem de programação Java 1.5, são: gerenciador de classes, geradores de analisadores léxicos, sintáticos e semânticos e sistema principal.

A opção da implementação utilizando uma linguagem de programação gratuita, bem como a distribuição deste software como livre, permitem que quaisquer pessoas possam utilizá-lo e redistribuí-lo, bem como permite que interessados no projeto possam expandir suas funcionalidades, corrigir erros, produzir traduções e *plugins,* sem custo monetário algum, beneficiando todos os envolvidos.

A interface dos itens a seguir não será discutida detalhadamente devido à grande quantidade de funcionalidades. Ao leitor interessado, a ajuda do programa fornece , em seus pormenores, todas as informações sobre o uso e as capacidades do *software.*

4.1 O Analisador Léxico

O *plugin* responsável pela edição de analisadores léxicos foi criado para produzir um analisador através da definição de autômatos. Esta é informada pelo usuário, responsável pela criação de estados e transições do autômato, bem como os caracteres reconhecidos em cada transição.

Finalizada a edição do autômato, ao iniciar o processo de reconhecimento, este será convertido em um autômato finito determinístico (AFD) e sua tabela de transições será exibida na tela de reconhecimento, conforma a figura 2.

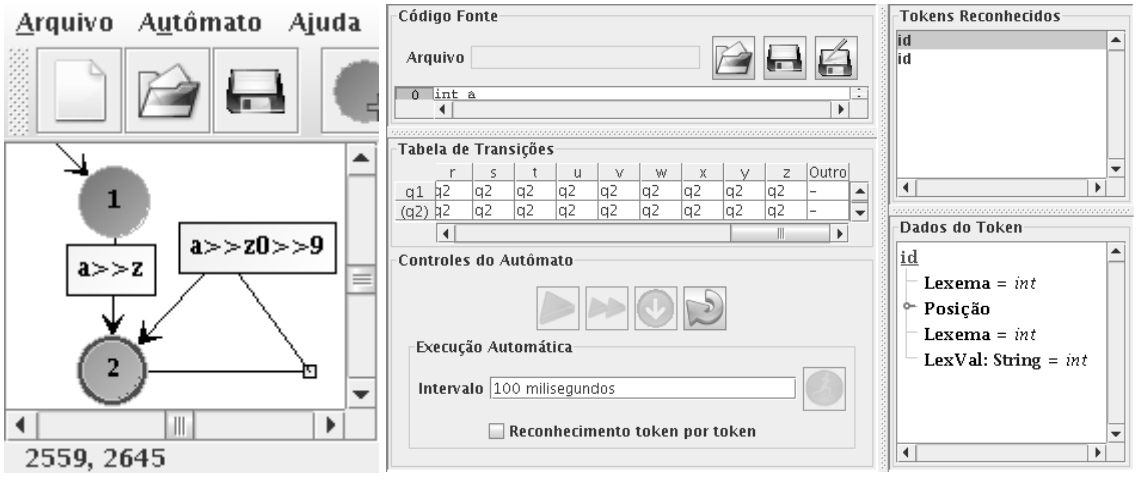

Figura 2. Tela de edição de um autômato e respectiva tela de reconhecimento

4.2 O Analisador Sintático

Através da especificação de uma gramática, um analisador sintático SLR é produzido pelo *plugin* responsável pela geração de analisadores sintáticos, considerando como terminais os *tokens* reconhecidos pelo léxico. A gramática é montada no editor através do *mouse,* tarefa esta realizada pelo usuário, que deve criar os símbolos não-terminais da gramática e posicioná-los dentro do editor. Isto elimina a necessidade de uma linguagem de definição, o que evita a necessidade do aprendizado de uma nova notação, que à primeira vista pode não ser clara para o usuário. A tela de edição e de reconhecimento é exibida na figura 3.

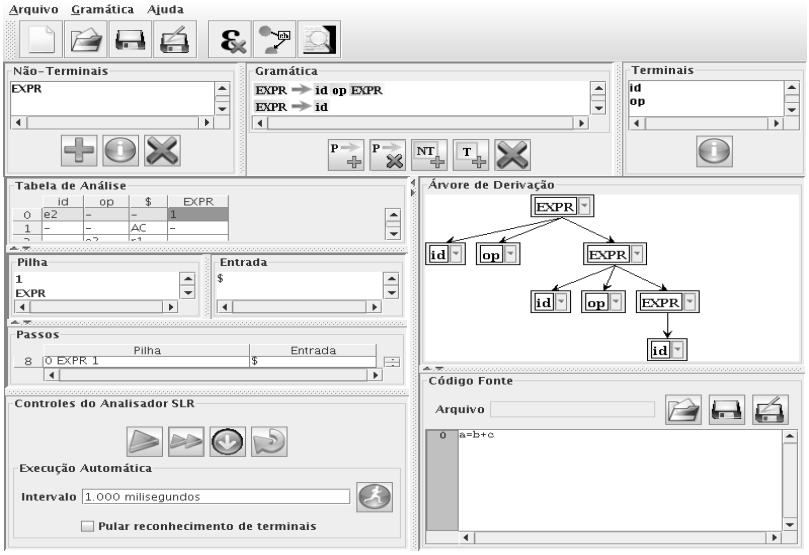

Figura 3. Tela de edição de gramáticas e respectiva tela de reconhecimento

4.3 O Analisador Semântico

O gerador de analisadores semânticos implementado estende a gramática de um analisador sintático, de forma a realizar o processo de tradução dirigida por sintaxe ao percorrer a árvore de derivação produzida pelo passo anterior. Assim, o usuário deve selecionar as produções da gramática e inserir ações semânticas entre os símbolos destas. Ademais, os atributos dos símbolos reconhecidos podem ser manipulados nestas ações, definidas em um editor que reconhece uma linguagem simples, de sintaxe semelhante a C, com recursos de auto-completar e correção de código. É possível gerar código em qualquer formato, através de uma função de impressão que pode ser utilizada dentro das ações semânticas. A figura 4 exibe o editor. A tela de reconhecimento é semelhante a do analisador sintático.

Figura 4. Tela de edição de ações semânticas.

5 O Programa Principal

O programa principal habilita todos os *plugins* produzidos para a ferramenta, instalados no diretório "*plugins"*, permitindo sua utilização em um projeto de compilador, onde são sincronizados para que alterações em um passo reflitam nos subseqüentes. Em um projeto de compilador, todos os arquivos responsáveis por manter a definição de cada passo são gerenciados, e todos os analisadores produzidos pelos editores são automaticamente salvos em uma pasta de saída, junto ao objeto compilador, que pode ser carregado em quaisquer aplicativos Java que o usuário produzir, realizando o processo de reconhecimento ou alguma parte deste.

Finalizado o projeto do compilador, pode-se iniciar o processo de reconhecimento de código, que exibe todos os passos, da análise léxica até a semântica, de acordo com a implementação de cada *plugin.* Um projeto pode possuir vários analisadores semânticos, pois podem ser necessárias várias passagens na árvore de derivação.

6 Conclusão

O programa produzido elimina a necessidade do emprego de muito tempo e esforço para aprender a utilizar ferramentas cujo foco é voltado a aplicações profissionais, que exigem que as técnicas já estejam dominadas, ao invés de auxiliar o aluno a dominá-las.

A interface produzida permite a fácil aplicação e experimentação da teoria, com funcionalidades suficientes para servir de acompanhamento no decorrer da disciplina, auxiliando na eliminação de incertezas sobre o funcionamento de um compilador.

Porquanto a ferramenta permite expansão através de *plugins*, quaisquer novas funcionalidades podem ser desenvolvidas e utilizadas em conjunto com os *plugins* já implementados. Esta característica permite que todo o conteúdo concernente a disciplina possa ser utilizado na ferramenta. Funcionalidade esta que até o momento nenhum outro software de cunho educacional oferece.

Na qualidade de software livre (disponível em *http://sourceforge.net/projects/c-gen*), este projeto possui capacidade de rápida expansão, através da contribuição voluntária em seu desenvolvimento, beneficiando, destarte, toda a sociedade através de esforços altruísticos para a consecução do objetivo de facilitar o ensino da disciplina de compiladores.

Assim, espera-se que este trabalho sirva como auxílio para alunos e professores a elucidar e aplicar grande parte do conteúdo desta área, bem como apresentar a importância deste conhecimento, incentivar o interesse dos alunos e tornar mais palpável o uso deste na solução dos mais diversos problemas enfrentados durante a carreira de um cientista da computação.

Referências

Aho, A.; Sethi, R.; Ullman, J. (1986) “Compilers: Principles, Techniques, and Tools”. Editado por Addison Wesley, Estados Unidos.

Donelly, C.; Stallmann, R. (2005) “Bison – The Yacc-compatible parser generator”, http://www.gnu.org/software/bison/manual/, Maio/2005.

Mössenböck, Hanspeter. “The Compiler Generator Coco/R User Manual.”, (2005), http://www.ssw.uni-linz.ac.at/Research/Projects/Coco, Abril/2005.

Paxson, Vern. “Flex – A scanner Generator.” (1998), http://www.gnu.org/software/flex/manual/, Maio/2005.

Mak, Ronald. “Writing Compilers And Interpreters An Applied Approach Using C++”.(1996), Editado por Wiley Computer Publishing.

Kakde, O. G. “Algorithms For Compiler Design”. Editado por Charles River Media, 2002, Hingham – Massachusetts.

Birsan, Dorian. “On Plugins: And Extensible Architectures”, (2005), http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=286, Maio/2005