GAS
Gerador de Analisadores Sintáticos

Olinto José Varela Furtado
José Eduardo De Lucca
Edison Linhares de Oliveira

Laboratório de Integração de Software e Hardware (LISHA)
Departamento de Ciências Estatísticas e da Computação (CEC)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis - Santa Catarina - Brasil

RESUMO

Este artigo descreve um gerador de analisadores sintáticos chamado GAS e fornece uma visão geral do ambiente no qual está inserido, detalhando algumas ferramentas de apoio. GAS gera analisadores sintáticos para várias linguagens através de diversas técnicas de análise sintática. Adicionalmente fornece documentação dos passos intermediários, possibilidade de solução interativa de conflitos e um simulador de análise sintática.

ABSTRACT

This paper describes a syntatic analyzer generator called GAS and provides a general vision of the environment which it is in and also gives some details about supporting tools. GAS generates syntatic analyzer for several languages through many syntatic analysis techniques. Furthermore, GAS provides intermediate steps documentation, possibility of conflicts interactive solution and a syntatic analyzer simulator.

KEYWORDS: Linguagens Formais, Compiladores, Compiler-Compilers

INTRODUÇÃO

O processamento de linguagens constitui uma atividade básica no desenvolvimento de software. Esta realidade motivou e continua motivando a pesquisa de técnicas que permitam a especificação formal de linguagens e a construção de seus processadores, o que tem resultado no surgimento de diversos formalismos para a especificação dos aspectos léxicos, sintáticos e semânticos de linguagens. Baseados nestes formalismos, surgiram muitas ferramentas capazes de automatizar desde a geração de algumas etapas do processo de compilação até a geração completa de um compilador.

Lex/Yacc [01] e LADE [02] são exemplos de ferramentas baseadas nestes formalismos e nas técnicas de compilação elaboradas a partir dos mesmos (ver também [09] [10] [11] [12]

[13] e [14]). Contudo, estas ferramentas não objetivam a transparência do processo de geração, omitindo os passos intermediários envolvidos e isolando o usuário. Além disso, as mesmas não oferecem flexibilidade nem na escolha do formalismo/notação de especificação (algumas até criando seus próprios formalismos) nem na escolha da técnica a ser utilizada na geração. Tais características impedem seu uso no ensino de linguagens formais e compiladores e dificulta sobremaneira o seu uso profissional, uma vez que obriga o usuário a adaptar-se à ferramenta e não ao contrário, como estabelecem os princípios da Engenharia de Software.

Objetivando o ensino adequado de Linguagens Formais e Compiladores e o desenvolvimento de processadores de linguagens foi concebido o Ambiente para Ensino e Desenvolvimento de Compiladores, o qual prima pela transparência na realização das tarefas e pela flexibilidade das ferramentas que o compõem. Dentre as ferramentas serão destacados neste artigo o GAS - Gerador de Analisadores Sintáticos, o Edigralc - Editor de Gramáticas Livres de Contexto e o Edigraf - Editor Gráfico de Autômatos Finitos.

O GAS será o objeto principal deste artigo, por ser uma ferramenta de muita importância prática e por sua abrangência teórica, envolvendo direta ou indiretamente uma gama considerável de fundamentos básicos da teoria das linguagens formais e das técnicas de compilação.

Nas seções seguintes será apresentada a descrição geral do Ambiente, destacando as ferramentas Edigralc e Edigraf e detalhando as principais características e potencialidades do GAS.

AMBIENTE PARA ENSINO E DESENVOLVIMENTO DE COMPILADORES E LINGUAGENS

O Ambiente para Ensino e Desenvolvimento de Compiladores e Linguagens vem sendo projetado e implementado no Departamento de Ciências Estatísticas e da Computação da Universidade Federal de Santa Catarina, com o objetivo de apoiar o ensino de técnicas formais de especificação de linguagens e fornecer meios para a criação de processadores de linguagens (tradutores, compiladores, editores dirigidos pela sintaxe etc.) de forma automática, prática e eficiente.

Estes objetivos estão presentes em todas as ferramentas que compõem o Ambiente, desde os editores até os geradores. O Ambiente pode ser subdividido, quanto a classe das ferramentas, em: Módulo de Linguagens Regulares (LR) e Módulo de Linguagens Livres de Contexto (LLC); e quanto à funcionalidade em Ferramentas de Edição, de Geração, de Simulação e de Manipulação/Transformação. A figura 1 apresenta todas as ferramentas e seus interrelacionamentos.

O conjunto de editores usado na especificação das linguagens regulares permite a edição nos seguintes formalismos: Autômatos Finitos-Tabela de Transições (EdiTab), Expres-

sões Regulares (ER-Edit), Gramáticas Regulares (Edigreg) e Autômatos Finitos-diagramas de transição (Edigraf). A migração entre as notações é automática, sendo que o Editab executa ainda transformações sobre autômatos finitos (determinização e minimização).

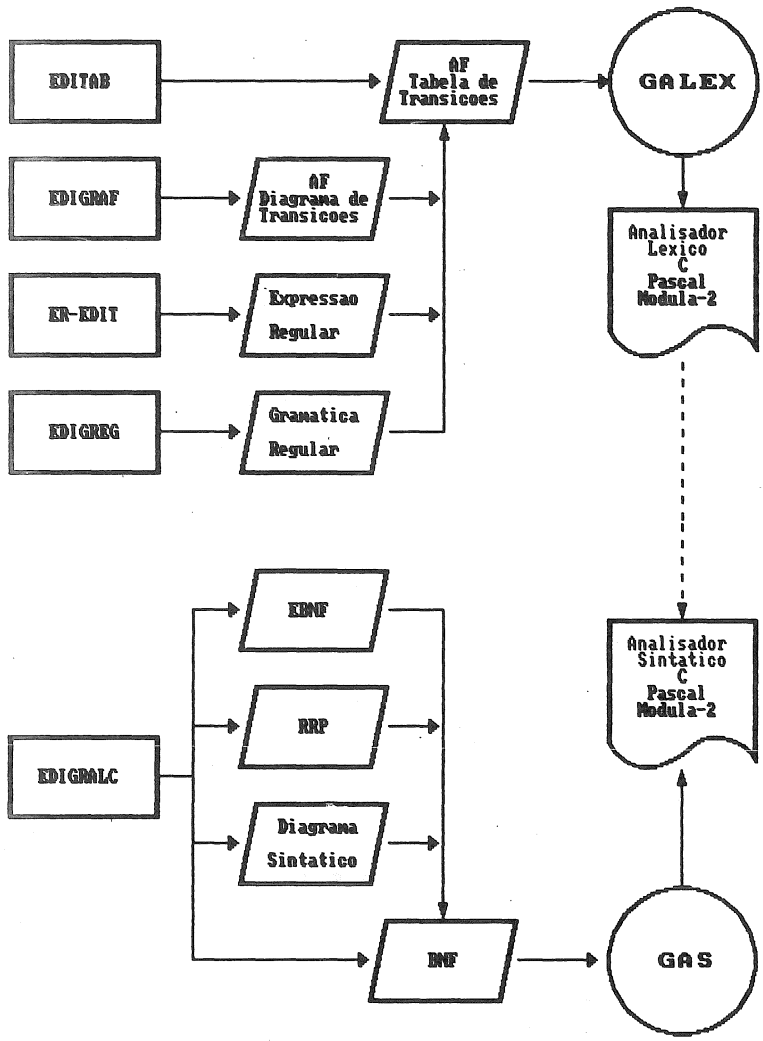

FIGURA 1 - Estrutura Geral do Ambiente

A simulação dinâmica do processo de reconhecimento léxico de uma sentença é realizado tanto pelo Editab quanto pelo Edigraf (ver adiante) e visa a depuração da especificação

e/ou compreensão do processo. O GALEX - Gerador de Analisadores Léxicos é responsável pela geração automática de analisadores léxicos, que podem ser empregados na construção de compiladores, scanners e outros programas que trabalham com aspectos regulares de linguagens. O GALEX é perfeitamente integrável ao GAS.

Para a especificação de linguagens livres de contexto a ferramenta Edigralc - Editor de Gramáticas Livres de Contexto permite utilizar as notações BNF, EBNF, RRP e Diagramas Sintáticos. A migração entre as notações também é automática. O GAS - Gerador de Analisadores Sintáticos é a ferramenta central deste módulo e usa como ferramentas de apoio os editores dos Módulos LR e LLC.

Pode-se utilizar as ferramentas de modo independente (stand-alone) ou em conjunto, o que implica na flexibilidade do Ambiente. Outro aspecto que também evidencia a flexibilidade é a possibilidade de escolha das ferramentas/notações de edição para as diversas etapas, o que liberta o usuário das limitações de outros sistemas que possuem o mesmo propósito. Além disso, utilizam-se notações exatamente como definidas na bibliografia, unindo definitivamente a teoria com a prática.

A característica de simulação dinâmica presente em algumas ferramentas - as principais - é também de fundamental importância nas duas óticas do Ambiente: para ensino pois permite a visualização e acompanhamento do processo de reconhecimento/análise facilitando a compreensão do mesmo; e para desenvolvimento como suporte à depuração da especificação e testes de desempenho.

FERRAMENTAS DE APOIO AO GAS

Tomando o GAS como peça central do Ambiente, pois centraliza e interliga os outros diversos componentes, são apresentadas a seguir algumas das principais ferramentas que servem de apoio ao GAS na tarefa de construção de analisadores sintáticos.

i) EDIGRALC - Editor de Gramáticas Livres de Contexto

O EDIGRALC permite a edição de especificações de Linguagens Livres de Contexto através de Gramáticas por quatro notações distintas: BNF - Backus-Naur Form, EBNF - Extended BNF, RRP- Regular Right-hand Production e Diagramas Sintáticos [04],[05].

Os quatro editores que compõem o Edigralc são dirigidos pela sintaxe da própria notação, ou seja, verifica-se em tempo de edição se a estrutura da notação escolhida está sendo respeitada, permitindo e apoiando as correções caso sejam necessárias. Também é realizada uma análise de consistência quanto à símbolos inalcançáveis e/ou inférteis. Estas facilidades trazem vantagens de produtividade para a edição e a ga-

rantia da correção do resultado, isto é, que a gramática editada estará no formato escolhido.

A notação Diagrama Sintático permite a especificação através da notação proposta por Conway [07], baseada em primitivas gráficas. A utilização de recursos gráficos favorece a compreensão da especificação efetuada bem como sua documentação.

O Edigralc está diretamente relacionado com o GAS, pois as gramáticas editadas são utilizados para a geração dos analisadores sintáticos.

ii) EDIGRAF - Editor Gráfico de Autômatos Finitos

O Edigraf se propõe a editar representações gráficas de autômatos finitos (Diagramas de Transição) e permitir a especificação de Linguagens Regulares, muito especialmente os aspectos léxicos de linguagens de programação. Tanto autômatos finitos determinísticos quanto não-determinísticos podem ser editados através desta ferramenta.

Possui três características fundamentais: consistência visual, simplicidade e objetividade; resultando uma ferramenta amigável e prática. Algumas características dos autômatos podem ser configurados dinamicamente: rotulação de estados automática, geração de tokens e um dicionário de estados (que contém informações sobre o estado - e que visa a documentação automática).

A simulação do processo de reconhecimento de uma sentença submetida pelo usuário é realizada passo-a-passo, de modo animado, permitindo ao usuário acompanhar o reconhecimento e depurar o autômato. Quando da simulação de um autômato finito não-determinístico, as situações de não-determinismo são tratadas com backtrack, que é indicado ao usuário e realizado automaticamente.

Uma Biblioteca Expansível de Conjuntos está disponível para agrupar símbolos do alfabeto, visando facilitar a manipulação de rótulos nas arestas e a própria visualização do autômato que está sendo editado.

Como resultado de uma edição, obtem-se um automato finito em um formato compatível com o exigido pelo GALEX (Gerador de Analisadores Léxicos), que se responsabiliza pela interação com o GAS.

GAS - Gerador de Analisadores Sintáticos

GAS é um gerador de analisadores sintáticos baseados em tabelas de parsing que aceita como entrada a especificação sintática da linguagem através de uma GLC na notação BNF quando usado stand-alone ou nas notações BNF, EBNF, RRP e de Diagramas Sintáticos quando usado de forma integrada ao Ambiente.

A GLC de entrada passa inicialmente por um processo de validação, envolvendo aspectos léxicos (verifica se os símbolos terminais e não-terminais utilizados estão dentro do padrão especificado), sintáticos (verifica se a entrada realmente é uma GLC na notação correta) e semânticos (verifica a existência de símbolos e produções duplamente definidos, existência de símbolos inférteis/inalcançáveis etc.).

Após validada a entrada, o GAS interage com o usuário permitindo a escolha da técnica de análise sintática a ser utilizada (preditiva LL(1), SLR(1), LALR(1) ou LR(1) - ver [04] [05] [08]) e a linguagem na qual o Parser deverá ser gerado (C, Pascal ou Modula-2). O GAS verifica ainda a adequabilidade da GLC à técnica escolhida, gerando documentação completa dos testes realizados. Caso as condições iniciais (relativas a cada técnica) estejam satisfeitas, passa-se à geração da Tabela de Parsing. Os conflitos porventura existentes (indicando a não adequação da GLC à técnica escolhida) podem então ser resolvidos de maneira interativa com o usuário e então a Tabela de Parsing pode ser gerada. Adicionalmente, o GAS gera arquivo com os conflitos existentes e seu histórico, permitindo que a GLC seja revista à luz dos problemas detectados.

Uma vez gerada a Tabela de Parsing, o GAS permite a simulação do processo de análise sintática dispondo de um editor de sentenças/formas sentenciais e permitindo o acompanhamento dinâmico do processo, através da visualização da pilha sintática e da ação a ser efetuada em cada passo.

As tabelas geradas são responsáveis por todo o processo de análise sintática, sendo a base para gerar o analisador e efetuar a simulação. Assim sendo, procurou-se projetar eficientemente as estruturas de armazenamento das mesmas, levando-se em consideração tanto a técnica de análise sintática envolvida como a linguagem na qual o Parser será gerado. Particularmente para as técnicas da família LR implementou-se técnicas de compactação das tabelas, resultando em redução consideravel no tamanho das mesmas (acima da média conseguida por outras ferramentas).

O Parser gerado é uma implementação exata da técnica escolhida (com a vantagem de a Tabela de Parsing estar armazenada de forma a otimizar o desempenho do mesmo), não necessitando nenhuma alteração para a inclusão dos analisadores léxicos e semânticos. O analisador léxico pode ser desenvolvido de forma independente, bastando que haja sincronização em relação aos tokens a serem utilizados; ou pode ser obtido pelo Gerador de Analisadores Léxicos (GALEX) do Ambiente, podendo usufruir das facilidades de edição e validação da especificação léxica, providas pelo Módulo LR. Verificações Semânticas e ações de Geração de Código podem ser anexados ao Parser gerado e serão ativadas automaticamente por este através de referências explícitas incluídas livremente no lado direito das produções. O momento exato da ativação destas ações pode também ser visualizado no processo de simulação, permitindo ao usuário a análise de seus efeitos no processo geral de tradução.

A flexibilidade do GAS na escolha da técnica de análise sintática propicia o ensino adequado destas técnicas, pois permite a comparação entre elas, envolvendo desde as características estruturais da GLC de entrada até a forma final do Parser gerado.

A documentação gerada automaticamente a cada passo do processo, serve como material complementar para estudo/fixação dos conceitos envolvidos, como base para uma análise mais profunda da GLC visando adequação à técnica escolhida e como documentação final da etapa de Análise Sintática dentro do processo de contrução de um compilador.

A simulação do processo de Análise Sintática possibilita ao estudante não só a percepção dinâmica do funcionamento da técnica escolhida facilitando o aprendizado da mesma como também proporciona uma visão global do processo de tradução dirigido pela sintaxe, explicitando a ativação tanto do analisador léxico, como do analisador semântico/gerador de código. Por outro lado, a simulação permite a experimentação da técnica mais adequada bem como a validação/depuração da especificação sintática da linguagem por parte do profissional.

CONCLUSÃO

Neste artigo foi apresentada uma visão geral do Ambiente para Ensino e Desenvolvimento de Compiladores e Linguagens a partir da descrição do GAS e de algumas outras ferramentas correlacionadas.

Atualmente o Ambiente está sendo utilizado no ensino de linguagens formais e compiladores nos programas de graduação e pós-graduação em Ciência da Computação da UFSC, e algumas de suas ferramentas já estão sendo utilizadas - de forma integrada ou independente - em projetos de pesquisa que envolvem direta ou indiretamente o processamento de linguagens [03].

Particularmente o GAS tem sido bastante procurado e usado como alternativa à ferramentas correlatas disponíveis, tanto em função do Ambiente que o suporta como também devido às suas características de transparência, flexibilidade, fidelidade aos aspectos formais, documentação final, possibilidade de simulação/depuração e abrangência.

Como evolução do GAS, a curto prazo, está prevista a implementação de técnicas de recuperação de erros e uso de gramáticas de atributos para especificações semânticas e geração de código. A premissa para estas e futuras evoluções do Ambiente prevê a manutenção dos objetivos e características que nortearam sua criação.

BIBLIOGRAFIA

[01] JOHNSON, S.C. - YACC - Yet Another Compiler Compiler, Technical Report 32, AT&T Bell Laboratories, 1978.

[02] XORIAN Technologies Pte. Ltd. - LADE Reference Manual, Singapura, 1991

[03] UGGIONI, D.; DE LUCCA, J. E.; FURTADO, O. J. F. - GENESE, Um Editor Genérico Dirigido pela Sintaxe, Anais de las 1as Jornadas de Ingenieria de Sistemas Informaticos y de Computacion, Quito-Equador, novembro 1990.

[04] AHO, A. V. e ULLMAN, J. D. - Compiler: Principles, Techniques and Tools, Addison-Wesley, 1986.

[05] TREMBLAY,J. P. e SORENSON, P. G. - The Theory and Practice of Compiler Writing, McGraw-Hill, 1985

[06] GOUGH, K. J. - Syntax Analysis and Software Tools, Addison-Wesley, 1988.

[07] CONWAY, M. E. - Design of a Separable Transition-Diagram Compiler, Comm. ACM, 6,7, 396-408, 1963

[08] HOPCROFT, J. E. e ULLMAN, J. D. - Introduction to Automata Theory, Languages and Computation - Addison-Wesley, 1979

[09] HORSPOOL, R. N. e LEVY, M. R. - MKScan: An Interactive Scanner Generator,Software - Practice and Experience, vol 17 num 6, 369-378, junho 1987.

[10] QUADROS, F. S. e PRICE, A. M. A. - GAL: A Lexical Analyser Generator From Graphical Specification, Pós-Graduação em Ciência da Computação, UFRGS, 1988

[11] GRUNE, D. e JACOBS, C. J. H. - A Programmer-friendly LL(1) Parser Generator, Software - Practice and Experience, vol 18 num 1,29-38, janeiro 1988.

[12] SPECTOR, D. - Efficient Full LR(1) Parser Generation, SIGPLAN Notices, vol 23 num 12, dezembro 1988.

[13] MILTON, D. R.; KIRCHOFF, L. W. e ROWLAND, B. R. - An ALL(1) Compiler Generator, SIGPLAN Notices, vol 14 num 8, agosto 1979.

[14] HEURING, V. P. - The Automatic Generation of Fast Lexical Analyzers, Software - Practice and Experience, vol 16 num 9, 801-808, 1986.