O ensino de Linguagens Formais vinculado ao ensino de Compiladores

Conference Paper · July 2003

CITATIONS
3

READS
338

1 author:

Olinto J V Furtado
Federal University of Santa Catarina
55 PUBLICATIONS **74** CITATIONS

SEE PROFILE

O ensino de Linguagens Formais vinculado ao ensino de Compiladores

Olinto José Varela Furtado

Departamento de Informática e de Estatística
Curso de Ciências da Computação
Centro Tecnológico – Universidade Federal de Santa Catarina
Caixa Postal 476 – 88.040 - 970 – Florianópolis SC – Brasil

olinto@inf.ufsc.br

Abstract. *Historically, teaching/learning the diverse aspects of the Computing Theory, in general, and of the Formal Languages Theory, in particular, have encompassed some problems due to this subject dryness and also to the lackof appropriate methodologies to approach it. In order to contribute to the reduction of these problems, this paper describes a more than 10-years-experience in teaching Formal Languages and Automata linked to the Compiler Construction teaching. This experience shows that the employed teaching methodology has entailed benefits to the learning of the computing theoretical aspects as well as to the applications which are based on these aspects.*

Resumo. *Historicamente o processo ensino/aprendizagem dos diversos aspectos da Teoria da Computação em geral e da Teoria das Linguagens Formais em particular tem sido problemático tanto pela aridez do assunto quanto pela falta de metodologias adequadas para abordá-lo. Visando contribuir na redução de tais problemas, este artigo descreve uma experiência de mais de 10 anos no ensino de linguagens formais e autômatos vinculado explicitamente ao ensino de Compiladores, mostrando que a metodologia usada tem trazido ganhos tanto no aprendizado dos aspectos teóricos da computação quanto das aplicações que fundamentam-se diretamente nesses aspectos.*

1. Introdução

O Ensino da Teoria da Computação como um todo e da Teoria de Linguagens Formais e Autômatos em particular, quando desvinculado da prática, tende a ser enfadonho, cansativo e pouco produtivo. Este quadro agrava-se quando levamos em conta a complexidade do assunto, a resistência natural dos alunos ao aprendizado de aspectos teóricos, a carência de professores atuando na área e a falta de metodologias adequadas a nossa realidade.

Esta era a realidade do Curso de Ciências da Computação da UFSC até o final da década de 80, quando a Teoria das Linguagens Formais e Autômatos era abordada na disciplina de Teoria da Computação, de uma maneira purista, pouco profunda e sem

vinculação direta com a prática de compiladores (ou seja, especificação/implementação dos aspectos léxicos, sintáticos e semânticos das linguagens de Programação).

Percebendo-se a dificuldade dos alunos frente à abstração do conteúdo e a pouca contribuição direta desse conteúdo na obtenção de bons resultados nas disciplinas que deveriam fundamentar-se nos mesmos – a disciplina de Construção de Compiladores em particular - e tendo em vista o fato de que a disciplina de Compiladores é a disciplina onde a integração entre teoria e prática mais se evidencia [WOO 1987], propôs-se uma metodologia para o ensino integrado de linguagens formais e compiladores.

Tal metodologia tem como objetivo tornar o processo de ensino/aprendizagem de linguagens formais e autômatos menos abstrato, mais agradável e mais efetivo, e ao mesmo tempo fundamentar melhor o ensino de compiladores; para tanto, usa-se a aplicação direta da teoria das linguagens formais na especificação e implementação de linguagens de programação, como fator de motivação natural para o ensino da própria teoria. Com isso visa-se alcançar um duplo benefício: aumentar o interesse pelos aspectos teóricos da computação bem como aplicar com mais propriedade essa teoria nas disciplinas que a utilizam direta ou indiretamente. Mais especificamente, visa-se melhorar o processo de ensino/aprendizagem dos conteúdos abrangidos pelas disciplinas de linguagens formais e de compiladores.

Nas seções subseqüentes apresenta-se o plano de ensino da disciplina INE5317 – Linguagens Formais e Compiladores, criada para operacionalizar a metodologia proposta, e descreve-se de forma comentada a metodologia utilizada no ensino de cada item do programa criado, destacando-se os objetivos almejados, os métodos empregados e os resultados alcançados.

2. A disciplina de Linguagens Formais e Compiladores

A disciplina de Linguagens Formais e Compiladores, criada para operacionalizar a metodologia proposta, está alocada na quinta fase do curso de Ciências da Computação da UFSC, possui uma carga horária de 72 horas aula, tem como pré-requisito a disciplina de Estrutura de Dados e é pré-requisito das disciplinas de Teoria da Computação e de Construção de Compiladores.

Originalmente seu conteúdo era um tópico da disciplina de Teoria da Computação, o qual foi extraído visando abordá-lo com maior abrangência e profundidade (conforme previsto nas Diretrizes Curriculares de Cursos da área de Computação e Informática [MEC 2002]) e também vinculá-lo diretamente com a especificação e implementação dos aspectos léxicos, sintáticos e semânticos de linguagens de programação.

Embora em uma primeira leitura o sequenciamento das disciplinas (Linguagens Formais precedendo Teoria da Computação) possa parecer equivocado, o mesmo vem de encontro com a metodologia proposta, a qual visa motivar o aluno para a importância da teoria da computação, a partir da constatação prática de sua aplicabilidade na ciência da computação em geral e na área de Compiladores em particular.

A contextualização da teoria das linguagens formais frente à teoria da computação dá-se no primeiro capítulo da disciplina onde, a partir de uma breve introdução à teoria da

computação, infere-se que todo problema computacional pode ser tratado como um problema de Linguagem [WOO 1987], não havendo, portanto, problema algum de descontinuidade de conteúdo.

O plano de ensino da disciplina é definido em função de seus objetivos (gerais e específicos), seu programa, critério de avaliação e bibliografia. Na seqüência são apresentados os objetivos, o programa e a bibliografia; os instrumentos de avaliação são abordados implicitamente na descrição da metodologia apresentada na próxima seção.

2.1. Objetivos

A disciplina proposta tem como **objetivo geral** o estudo da teoria das linguagens formais visando sua aplicação na especificação de linguagens de programação e na construção de compiladores.

Dentre os **objetivos específicos** da disciplina, destacamos os seguintes: 1. dar uma visão geral do processo de compilação sob o ponto de vista de implementação. 2. Dar uma visão geral da Teoria da Computação sob a ótica da Teoria das Linguagens Formais. 3. Dar ao aluno noções de linguagens formais e suas representações. 4. Estudar autômatos e gramáticas, enfatizando seus aspectos teóricos e suas aplicações. 5. Abordar as técnicas formais de análise léxica e sintática.

2.2. Programa

Para atingir os objetivos descritos na seção anterior, desenvolveu-se o seguinte programa:

Capítulo I - (8 horas/aula) – Introdução a Compiladores, Teoria da computação e Teoria das linguagens formais.

Capítulo II - (18 horas/aula) – Gramáticas: Motivação. Definição formal. Derivação e redução. Sentença, forma sentencial e linguagens. Tipos de gramáticas e linguagens (hierarquia de Chomsky). Sentença vazia. Recursividade das Gramáticas Sensíveis ao Contexto.

Capítulo III - (16 horas/aula) – Autômatos Finitos e Conjuntos Regulares: Autômatos Finitos Determinísticos (AFD) e Não-Determinísticos (AFND). Transformação de AFND para AFD. Relação entre Autômatos Finitos e Gramáticas Regulares. Minimização de AFD. Conjuntos regulares e Expressões Regulares. Implementação de Autômatos Finitos. Propriedades e problemas de decisão das Linguagens Regulares. Aplicações de Autômatos Finitos e Expressões Regulares.

Capítulo IV - (14 horas/aula) – Gramáticas Livres de Contexto (GLC) e Autômatos de Pilha (PDA): Introdução. Árvore de derivação. Formas de derivação em GLC. Ambiguidade. Transformações e simplificações de GLC. Tipos especiais de GLC. First e Follow. PDA. Equivalência entre PDA e GLC. Propriedades e problemas de decisão das Linguagens Livres de Contexto. Aplicações.

Capítulo V - (16 horas/aula) – Análise Sintática: Introdução. Classes de analisadores sintáticos. Técnicas de análise sintática ascendentes e descendentes.

2.3. Bibliografia

A bibliografia complementar utilizada nesta disciplina, como não poderia deixar de ser, inclui livros clássicos das áreas de linguagens formais ([HOP 1969] [HOP 1979]) e compiladores ([AHO 1972] [AHO1986]), além de publicações mais recentes e acessíveis como [MEN 1998], [LEW 1998], [DIV 1999] e [SUD 1997]. Um destaque especial da bibliografia adotada é [WOO 1987], usado na contextualização da teoria das linguagens formais como um dos propósitos da teoria da computação.

Apesar da diversidade e da qualidade das obras acima referidas, a bibliografia básica (texto) usada nesta disciplina é uma apostila [FUR 2002] criada especificamente para a metodologia proposta. Esta apostila é uma evolução das anotações de aula usadas no início da década de 90 (sua primeira versão foi escrita em 1992), e baseia-se em obras clássicas das áreas envolvidas. A principal vantagem da adoção desta apostila reside no fato de que ela reúne, de forma ordenada, na abrangência e na profundidade desejada, todo o conteúdo a ser abordado.

3. A metodologia proposta

Buscando nas aplicações imediatas a motivação para o aprendizado do conteúdo proposto, a disciplina inicia-se com uma Introdução à Compiladores, na qual enfatiza-se a estrutura geral de um compilador baseado no modelo análise/síntese [AHO 1986] e aborda-se as funções básicas de cada uma das fases que compõem o processo de compilação, ilustrando-as a partir de um programa simples completo. Durante essa exposição, ao abordar-se a etapa de análise, dividida nas fases de análise léxica, sintática e semântica, chama-se a atenção para a necessidade de uma especificação precisa dos aspectos léxicos, sintáticos e semânticos de uma linguagem de programação, visando a implementação sistemática dos analisadores destes aspectos. Neste momento os modelos formais de especificação (gramáticas e autômatos) são preliminarmente introduzidos e é enfatizado que o estudo de tais modelos é o objeto da Teoria das Linguagens Formais e Autômatos.

Colocada à necessidade prática e vislumbrado na teoria das linguagens formais os fundamentos (os modelos) para tais necessidades, apresenta-se uma visão teórica/formal de tais modelos, vinculando-os à teoria da computação. Parte-se então para uma introdução à teoria da computação, focada no estudo dos problemas computacionais (computabilidade) e na questão da decidibilidade, inferindo-se que tais problemas podem ser tratados como problemas de linguagem e que, portanto, a teoria da computação pode ser estudada sob a ótica da teoria das linguagens formais [WOO 1987]. Desta forma, fica estabelecido um elo direto entre as duas disciplinas.

Isto posto, apresenta-se as definições básicas necessárias para uma primeira definição matemática de linguagem : $\mathbf{L} \subseteq \mathbf{V}^*$. Na sequência, o estudo de linguagens é

associado ao estudo das formas de representação de tais linguagens [HOP 1979], ou seja, os sistemas geradores (gramáticas) e os sistemas reconhecedores (autômatos) de linguagens.

3.1. Capítulo II – Gramáticas

Neste capítulo as gramáticas são introduzidas de forma tradicional e intuitiva, através de uma analogia com a especificação sintática das linguagens naturais (da língua portuguesa mais especificamente). Em seguida, formaliza-se a definição deste modelo de representação de linguagens e introduz-se os tipos de gramáticas (Hierarquia de Chomsky), voltando a vincular os diferentes tipos de gramáticas com os diferentes aspectos que compõem a especificação de uma linguagem de programação; mais especificamente, correlaciona-se Gramáticas Regulares com os aspectos léxicos e Gramáticas Livres de Contexto com os aspectos sintáticos de linguagens de programação.

Sob o ponto de vista mais teórico, usa-se o teorema da recursividade das Gramáticas Sensíveis ao Contexto, para estabelecer-se um vínculo com a questão da decidibilidade de problemas e enfatiza-se as conseqüências do uso de uma gramática irrestrita (tipo 0) para, por exemplo, representar a sintaxe de uma linguagem de programação. Com exemplos como este, consegue-se de forma simples, prática e natural mostrar a importância da questão da decidibilidade em particular e da teoria da computação em geral.

A avaliação do conteúdo relativo a este capítulo dá-se na forma de exercícios que visam formalizar (através de gramáticas dos diferentes tipos) linguagens enunciadas informalmente. Nestes exercícios explora-se também a questão da expressividade dos diferentes tipos de gramáticas e naturalmente a relação existente entre eles.

3.2. Capítulo III – Autômatos Finitos e Conjuntos Regulares

Todo o conteúdo deste capítulo é apresentado tendo como motivação a fase de análise léxica do processo de compilação. Neste contexto é mostrada a equivalência entre Gramáticas Regulares, Expressões Regulares e Autômatos Finitos, e é enfatizado o uso de Expressões Regulares como mecanismo adequado para a especificação e de Autômatos Finitos como mecanismo natural para implementação dos aspectos léxicos de Linguagens de Programação. A questão da determinização (obtenção de um AFD a partir de um AFND) e minimização de Autômatos Finitos também é apresentada neste contexto.

Para consolidar o aprendizado do conteúdo teórico deste capítulo, são implementados algoritmos clássicos de determinização, minimização e de equivalência entre mecanismos. A complexidade computacional do processo de reconhecimento também é explorada através da implementação e/ou utilização de reconhecedores genéricos determinísticos e não-determinísticos.

A vinculação deste conteúdo teórico com a prática de compiladores ocorre naturalmente ao definir-se uma linguagem regular formada pelos símbolos válidos (itens léxicos) de uma linguagem de programação. O reconhecedor implementado é um protótipo do analisador léxico de tal linguagem! Com base nisso, infere-se a viabilidade da geração automática de analisadores léxicos a partir da especificação formal dos aspectos léxicos de uma linguagem.

Com isso, na disciplina de Construção de Compiladores após uma breve revisão dos aspectos teóricos necessários, parte-se imediatamente para a especificação léxica da linguagem a ser utilizada na disciplina e a implementação do analisador léxico correspondente; note-se que, como esses aspectos já foram abordados teoricamente e experimentados na prática, mesmo em se tratando de um curso que tem a computação como fim, o uso de geradores automáticos é adequado, pois não interfere na profundidade com que o conteúdo é ministrado, agiliza a implementação e produz um analisador léxico correto com relação à especificação léxica da linguagem em questão.

3.3. Capítulos IV e V – GLC, PDA e Análise Sintática

Estes capítulos visam apresentar a teoria de parsing a partir do estudo dos modelos formais que lhe dão sustentação: GLC (Gramáticas Livres de Contexto) e PDA (Push-Down Automata – Autômatos de Pilha).

Neste sentido é enfatizado o uso de GLC como mecanismo de especificação sintática e de PDA como um modelo natural de um analisador sintático. Também é enfatizado o uso de GLC como base para definição formal dos aspectos semânticos de Linguagens (via Gramática de Atributos, por ex.) e para a implementação de Esquemas de tradução dirigidos pela sintaxe.

Este estudo inicia com a demonstração da capacidade e adequação (expressividade e performance) das GLG e dos PDA para a especificação sintática de linguagens. Na seqüência é discutida a questão da ambigüidade a partir de construções típicas de linguagens de programação, passando-se então para o estudo das simplificações e transformações clássicas de GLC – vinculando-se cada simplificação ou transformação às necessidades das diferentes técnicas de análise sintática abordadas. A equivalência entre GLC e PDA é introduzida de forma paralela à demonstração da adequação destes mecanismos para especificação dos aspectos sintáticos e implementação de seus analisadores.

O estudo integrado dos mecanismos de especificação e das técnicas de implementação torna evidente a possibilidade de geração automática de analisadores sintáticos a partir da especificação formal da sintaxe de uma linguagem, estabelecendo naturalmente a importância do uso de mecanismos formais e conseqüentemente a aplicabilidade dos modelos que compõem a teoria das linguagens formais e, por extensão, da Teoria Computação.

Para consolidação dos conhecimentos obtidos neste capítulo, são especificadas pequenas linguagens para as quais são construídos manualmente analisadores sintáticos – usando-se as principais técnicas disponíveis. Posteriormente são utilizados geradores de analisadores sintáticos didáticos disponíveis (GAS [LIN 1991] e GALS [GUE 2002], por exemplo) para geração de analisadores para as mesmas gramáticas; isto permite a validação dos analisadores construídos manualmente e a experimentação dos conceitos (ambigüidade, fatoração e recursão à esquerda, por exemplo) e técnicas (análise comparativa via simulação) estudados. Com isso, na disciplina de Construção de Compiladores o estudo da análise sintática reduz-se a uma revisão conceitual rápida do modelo de especificação e da

técnica de análise a serem utilizados, permitindo que os alunos dediquem-se mais efetivamente à especificação sintática da linguagem a ser usada na disciplina e à implementação do analisador correspondente.

4. Considerações Finais

Este artigo descreveu a metodologia utilizada na disciplina de Linguagens Formais e Compiladores do curso de graduação em Ciências da Computação da UFSC, a qual foi criada visando a redução dos problemas normalmente encontrados no ensino dos aspectos teóricos da computação e a utilização mais efetiva de tais aspectos nas disciplinas tecnológicas do curso.

Os resultados alcançados com a prática dessa metodologia, não só confirmam as suposições iniciais, como também apontam para a melhoria de vários indicadores relativos ao desempenho dos acadêmicos. Dentre os benefícios pedagógicos obtidos podemos destacar os seguintes:

- Aprendizado mais abrangente, mais profundo e mais efetivo tanto do conteúdo relativo à disciplina de Linguagens Formais e Compiladores quanto da disciplina de Construção de Compiladores.
- Menor resistência e maior motivação para estudo dos aspectos teóricos da computação.

A constatação de tais benefícios pode ser inferida a partir de diferentes perspectivas, com base nos seguintes fatos:

- Redução significativa no percentual de abandono da disciplina (em média de 30 para 10% entre a primeira e a segunda avaliação da disciplina);
- Redução de 50% no índice de reprovação dos alunos freqüentes (de 30 para 15%);
- Aumento da média dos alunos aprovados, caracterizando um melhor desempenho.
- Maior motivação/dedicação nas disciplinas de Teoria da Computação e Construção de Compiladores, resultando em um melhor desempenho.
- Possibilidade de uma abordagem com maior profundidade do conteúdo previsto na disciplina de Construção de Compiladores – decorrente de um conhecimento mais efetivo dos modelos formais usados na especificação/implementação de linguagens.
- Maior incidência de alunos desenvolvendo seus trabalhos de conclusão de Curso nas áreas de Linguagens formais e compiladores ([CAV 2002], [GUE 2002] e [KUE 2001], por exemplo);
- Aumento significativo da aplicação das técnicas de compilação em outras áreas ([CAS 2001], [MEI 2002] e [MEL 2002], por exemplo).
- Maior incidência de egressos do curso buscando orientação para o emprego dos modelos e técnicas estudados nas disciplinas, no desenvolvimento de sistemas aplicativos em geral.

A partir dos resultados positivos obtidos, estamos trabalhando no desenvolvimento de um ambiente integrado (tutorias e ferramentas) para o ensino de Linguagens formais e Compiladores. Outro trabalho em desenvolvimento é a proposição de uma metodologia similar para a disciplina de Introdução à Compiladores, oferecida desde o segundo semestre de 2002 para o curso de Sistemas de Informação, atualmente em implantação na UFSC.

5. Referências Bibliográficas

[AHO 1972] - AHO, A. V., ULLMAN, J. D. The Theory of Parsing, Translation, and Compiling. Volume I: Parsing. Ed Prentice-Hall, Inc. 1972, 542p

[AHO 1986] - AHO, A. V., SETHI, R.,ULLMAN, J. D.. Compiler - Principles, Techniques and tools. Ed. Addison-Wesley, 1986.

[CAS 2001] – CASCAES, J. Uma Linguagem para Análise de Bilhetes de Tarifação de Centrais Telefônicas, TCC – Curso de Ciências da Computação, UFSC, Florianópolis, SC, 2001.

[CAV 2002] – CAVANUS, J. , HILDEBRAND, R. W. - PHP-BR : Um Ambiente de Desenvolvimento gratuito e open-source para a Linguagem PHP, TCC – Curso de Ciências da Computação, UFSC, Florianópolis, SC, 2001.

[DIV 1999] - DIVERIO, T. A., MENEZES, P. B., Teoria da Computação – Máquinas Universais e Computabilidade. Ed. Sagra Luzzatto, Porto Alegre, 1999

[FUR 2002] - FURTADO, O. J. V. Apostila de Linguagens Formais e Compiladores – Curso de Ciências da Computação, UFSC, 2002.

[GUE 2002] - GUESSER, C. E. GALS - Gerador de Analisadores Léxicos e Sintáticos, TCC – Curso de Ciências da Computação, UFSC, Florianópolis, SC, 2002

[HOP 1969] - HOPCROFT, J. E., ULLMAM, J. D. Formal Languagens and Their Relations to Automata. Addison-Wesley, 1969.

[HOP 1979] - HOPCROFT, J. F., ULLMAN, J. D.. Introduction to Automata Theory, Languagens and Computation. Ed. Addison-Wesley, 1979.

[KUE 2001] – KUELKAMP, K., Implementação da Linguagem Java/RTR, TCC – Curso de Ciências da Computação, UFSC, Florianópolis, SC, 2001.

[LEW 1998] - LEWIS, H. R. e PAPADIMITRIOU, C. H. , Elementos de Teoria da Computação, Ed. Bookmam, 2. edição, 1998.

[LIN 1991] – LINHARES, E., GAS - Gerador de Analisadores Sintáticos, TCC – Curso de Ciências da Computação, UFSC, Florianópolis, SC, 1991.

[MEC 2002] – Diretrizes Curriculares de Cursos da área de Computação e Informática, http://www.mec.gov.br/sesu/ftp/curdiretriz/computacao/co_diretriz.rtf

[MEI 2002] – MEIRE, S. de F. Concepção de Ambiente Computacional para Extração de Informações dos Registros de Chamadas Telefônicas Aplicando Técnicas de Compilação, Dissertação de Mestrado, CPGCC, UFSC, Florianópolis, SC, 2002.

[MEL 2002] – MELO, A . S., Reconhecimento de padrões sobre Sinais de ECG utilizando Técnicas Sintáticas, TCC – Curso de Ciências da Computação, UFSC, Florianópolis, SC, 2002.

[MEN 1998] - MENESES, P. B. Linguagens Formais e Autômatos, Ed. Sagra Luzzato, 2. edição, 1998.

[SUD 1997] - SUDKAMP, T. A. Languages and Machines – An Introduction to the Theory of Computer Science, 2. edição, Ed. Addison Wesley, 1997.

[WOO 1987] - WOOD, D. Theory of Computation. Ed. John Wiley & Sons, 1987.