
Linguagem de Definição e Geração de Analisadores Sintáticos em Semântica Denotacional Legível

José Leite da Silva Júnior

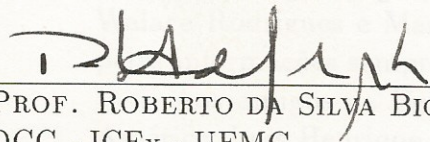
Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Março de 1993

FOLHA DE APROVAÇÃO

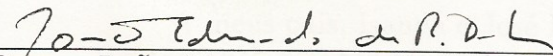
Linguagem de Definição e Geração de Analisadores Sintáticos em Semântica Denotacional Legível

JOSÉ LEITE DA SILVA JÚNIOR

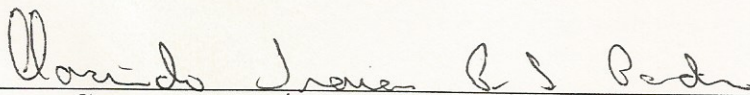
Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:



PROF. ROBERTO DA SILVA BIGONHA - Orientador
DCC - ICEx - UFMG



PROF. JOÃO EDUARDO DE REZENDE DANTAS
DCC - ICEx - UFMG



PROF. CLARINDO ISAIÁS PEREIRA DA SILVA E PÁDUA
DCC - ICEx - UFMG

Belo Horizonte, 29 de março de 1993.

Agradecimentos

Deus não nos propõe senão enigmas
Feodor Dostoiévski

À CAPES e à UFMG pelo apoio financeiro.

Ao meu orientador, prof. Roberto Bigonha, pelo constante incentivo, ajuda na solução de diversos problemas e orientação precisa.

Aos colegas do grupo de linguagens de programação — Lucas M. Amaral, Wallace Rodrigues e Mariza Bigonha — pela ajuda em diversos problemas e pelas informações sempre muito valiosas.

Aos amigos de Belo Horizonte, Daniel, Eveline, Geraldo, Higino, José Américo, Luis Henrique, Márcio e Vivien, pela companhia e apoio.

Aos *hermanos* de Fortaleza, Alfredo, Antônio Leite, Beatriz, Catarina, Edite, Ênio, Erivaldo, Galileu, Herbert, Mário, Paulo, Sobral, Tibério, pelos conselhos, amizade, cartas, telefonemas (mesmo às 2:00 h), cerveja e biscoitos.

A meus pais, Isauda e José Leite, minhas irmãs, Márcia e Nadja, e meus irmãos, Cláudio, Augusto e Marcos, pelo patrocínio, carinho, presença, amor e saudade.

Este trabalho apresenta uma nova abordagem à definição formal de linguagens de programação. Partindo do fato que os principais problemas encontrados em definições formais são sua baixa legibilidade e dificuldade de compreensão, foi examinada a adequação da aplicação das técnicas de *literate programming* a semântica denotacional.

O resultado da análise de *literate programming* no contexto de semântica formal foi a definição da linguagem de especificação LDS. Essa linguagem une prosa informal e equações semânticas numa mesma estrutura, promovendo um estilo de composição de definições onde a ênfase se encontra na estruturação e apresentação das definições.

A existência de dois modos de exposição (prosa e equações) torna mais evidente o duplo caráter (as duas audiências) das descrições em LDS: processamento pelo computador para obtenção automática de um compilador e a leitura por pessoas para a comunicação de idéias sobre uma linguagem de programação. Seguindo o modelo de *literate programming*, dois programas foram desenvolvidos para dirigir as definições à audiência desejada.

A segunda parte da dissertação trata da primeira fase do processo de geração automática de compiladores em LDS, i.e., a produção dos analisadores léxico e sintático a partir da definição sintática da linguagem. O algoritmo de geração de analisadores e sua implementação são descritos, juntamente com o esquema de compactação de tabelas de análise usado nos compiladores produzidos. Comentários sobre a linguagem de especificação sintática e sua interação com o algoritmo de geração também são apresentados.

Abstract

This work presents a new approach to the formal definition of programming languages. One assumption underlies the proposed approach: the main problems of formal definitions are their low legibility and difficult of comprehension. To solve these problems, it was verified the suitability of applying the techniques of literate programming to denotational semantics.

The result of the analysis of literate programming in a denotational semantics context was the design of the specification language LDS. This language associates in the same structure informal prose and semantic equations, promoting a style of definition composition where the emphasis is in the definition structuring and presentation.

The two modes of exposition (prose and equations) make more evident the two objectives (two audiences) of LDS descriptions: computer processing, to the automatic generation of compilers, and human reading, to communicate ideas about a programming language. Following the literate programming model, two programs were implemented to direct the definitions to the desired audience.

The second part of this dissertation deals with the first phase of automatic generation of compilers, i.e., the production of lexical analyzers and parsers from the syntatic definition. The algorithm to construct the analyzers and its implementation are described, along with the method for compactation of parsing tables used in the analyzers produced. Comments about the syntatic specification language and its interaction with the algorithm are also presented.

Sumário

Capítulo 1 Introdução	1
1.1 Descrições de Linguagens de Programação	1
1.2 O Estilo Proposto	4
1.3 Perspectiva da dissertação	5
Capítulo 2 Semântica Denotacional	6
2.1 Introdução	6
2.2 Aplicações	8
2.3 Conceitos e Notação	11
Capítulo 3 Programação Legível	14
3.1 Motivações e Premissas	14
3.2 O Estilo	19
3.3 O Sistema WEB	22
3.4 Críticas	24
3.4.1 Problemas	24
3.4.2 Vantagens	26
3.5 Simulando Programação Legível	28
3.6 Conclusões	29
Capítulo 4 A Linguagem LDS	30
4.1 Introdução	30
4.2 O Ambiente de Definição e Implementação de Linguagens de Programação	31
4.3 A Estrutura de LDS	33

4.3.1	Formatação de Definições	35
4.3.2	A Linguagem de Especificação Sintática	37
4.3.3	A Linguagem de Especificação Semântica	43
A Linguagem SDL		44
4.3.4	Os Comandos LDS	51
4.4	Definições em LDS	54
Capítulo 5 A Implementação de LDS		59
5.1	Introdução	59
5.2	<i>Tangle</i>	60
5.3	<i>Weave</i>	63
Capítulo 6 Uma Aplicação		69
6.1	A Linguagem Descrita	69
6.2	A Sintaxe de ASPLE	71
6.2.1	Cláusulas Sintáticas	71
6.2.1.1	Sintaxe de Programas	71
6.2.1.2	Sintaxe de Declarações	72
6.2.1.3	Sintaxe de Comandos	72
6.2.1.4	Sintaxe de Expressões	73
6.2.2	Domínios Sintáticos	74
6.2.3	Estrutura Léxica	74
6.3	A Semântica de ASPLE	75
6.4	Semântica Estática	75
6.4.1	Contexto	75
6.4.2	O Ambiente de Tipos	76
6.4.3	Verificação de Tipos em Programas	76
6.4.4	Verificação de Tipos em Declarações	77
6.4.5	Verificação de Tipos de Identificadores	78
6.4.6	Verificação de Tipos em Comandos	79
6.4.6.1	Comando de Atribuição	80
6.4.6.2	Comandos Estruturados	81
6.4.7	Verificação de Tipos em Expressões	82
6.4.7.1	Operações Binárias	83
6.4.7.2	Comparação de Valores	84
6.4.7.3	Sub-expressões	84
6.4.7.4	deref	84

6.4.7.5	base-level	85
6.5	Semântica Dinâmica	85
6.5.1	Contexto	85
6.5.2	O Ambiente de Execução	86
6.5.3	Domínios de Continuação	86
6.5.4	Continuação Inicial	87
6.5.5	Máquina Abstrata	87
6.5.5.1	Estado Inicial	88
6.5.6	Funções Básicas	88
6.5.6.1	Manipulação de Memória	89
6.5.6.2	Entrada e Saída	90
6.5.7	Programas	91
6.5.8	Declarações	92
6.5.9	Alocação de Memória	92
6.5.10	Comandos	93
6.5.11	Expressões	95
6.5.11.1	Sub-expressões	95
Capítulo 7 O Compilador SSL		104
7.1	Análise Sintática	104
7.2	Geração de Analisadores Sintáticos	107
7.3	Geração de Analisadores Léxicos	112
Capítulo 8 Conclusões		117
8.1	A Linguagem LDS e o Compilador SSL	117