

Marcelo de Almeida Maia
Orientador: Roberto da Silva Bigonha

Implementação Eficiente de uma Linguagem para Definição de Semântica

Dissertação apresentada ao Departamento de Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

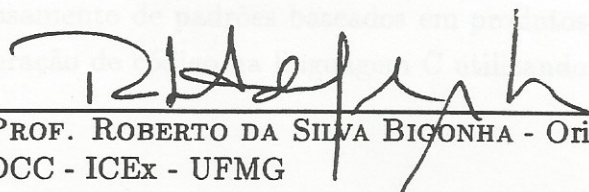
Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Belo Horizonte
Dezembro de 1994

FOLHA DE APROVAÇÃO

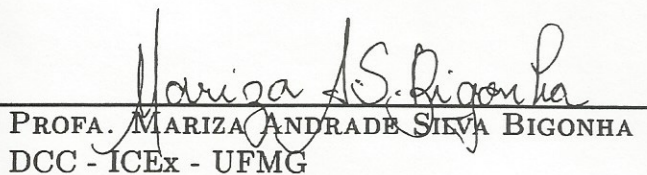
Implementação Eficiente de uma Linguagem para Definição de Semântica

MARCELO DE ALMEIDA MAIA

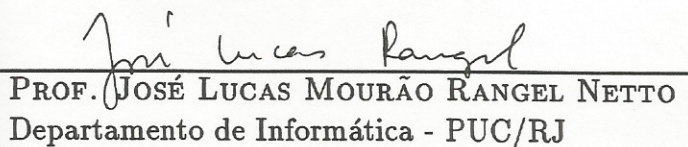
Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:



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



PROFA. MARIZA ANDRADE SILVA BIGONHA
DCC - ICEx - UFMG



PROF. JOSÉ LUCAS MOURÃO RANGEL NETTO
Departamento de Informática - PUC/RJ

Belo Horizonte, 05 de dezembro de 1994.

Resumo

Este trabalho trata da implementação eficiente de uma linguagem para definição de semântica denotacional. Os problemas envolvidos nesse trabalho estão relacionados com a implementação eficiente de linguagens puramente funcionais com semântica não-estrita. Existe também o aspecto de compilação eficiente das funções primitivas da linguagem. Os componentes principais do trabalho são a especificação da linguagem, a compilação de casamento de padrões baseados em produtos cartesianos, o processo de *lambda-lifting* e a geração de código na linguagem C utilizando uma máquina-G estendida.

Prefácio

Dedico esta dissertação aos meus pais, Alípio e Nina, que durante toda minha vida sempre apoiaram meus estudos.

Agradeço a todos que fizeram parte da minha vida.

Agradeço a Deus pela alegria de viver.

Abstract

This work focuses on the efficient implementation of a denotational semantics definition language. The problems covered on this work are related to the efficient implementation of purely functional languages with non-strict semantics. There is also the aspect of efficient compilation of the built-in functions of the language. The main components of the work are the language specification, the pattern-matching compilation based on cartesian products, the lambda-lifting process and the code generation into the C language using a extended G-machine.

Prefácio

Dedico essa dissertação aos meus pais Altair e Niza, que durante toda minha vida sempre apoiaram meus estudos.

Agradeço a todos me fizeram sorrir durante esses anos.

Agradeço a Cibele pela alegria que me proporcionou.

Agradeço aos meus amigos Léo, Juninho, Luiz, Humberto pela convivência quase sempre tranqüila.

Agradeço ao Prof. Roberto Bigonha pela participação intensa no trabalho.

Agradeço ao DEMAT/DECOM pelas liberações.

Agradeço aos amigos do DCC.

Agradeço ao CNPq pelo apoio financeiro.

2.3 Conclusão	12
3 Arquitetura do Compilador LAMB	13
3.1 Técnicas Existentes para Compilação de Linguagens Funcionais	13
3.2 Nossa Proposta	16
4 Representação Interna de Programas LAMB	18
4.1 Árvores de Sintaxe Abstrata	18
4.2 O Grafo	19
4.3 Gerenciamento de Memória e Coleta de Lixo	20
5 Compilação de Casamento de Padrões em LAMB	21
5.1 Especificação do Compilador de Casamento de Padrões	22
5.2 As Funções Seletoras	23
5.3 O Algoritmo para Compilação do Casamento de Padrões	23
6 Avaliação de uma Expressão LAMB	25
6.1 Avaliação Lazy	25
6.2 Construtores de Dados, Entrada e Saída	26
6.3 Forma Normal	27
6.3.1 Forma Normal com Cabeça Fraca	28
6.4 Avaliando Argumentos de Funções Built-in	29
6.5 Como Encontrar o Próximo Redex Top-level	29
6.5.1 A Pilha de Espinha	30
6.5.2 Inversão de Apontadores	31
6.5.3 Avaliação de Argumentos Usando Inversão de Apontadores	32

Sumário

Lista de Figuras	6
1 Introdução	1
2 A linguagem LAMB	3
2.1 Descrição Informal de LAMB	3
2.1.1 Expressões LAMB	4
2.1.2 Padrões	5
2.1.3 Operadores Pré-definidos	7
2.2 Descrição Formal de LAMB	10
2.2.1 Sintaxe da Linguagem LAMB	11
2.2.2 Semântica da Linguagem LAMB	11
2.3 Conclusão	12
3 Arquitetura do Compilador LAMB	13
3.1 Técnicas Existentes para Compilação de Linguagens Funcionais	13
3.2 Nossa Proposta	16
4 Representação Interna de Programas LAMB	18
4.1 Árvores de Sintaxe Abstrata	18
4.2 O Grafo	19
4.3 Gerenciamento de Memória e Coleta de Lixo	20
5 Compilação de Casamento de Padrões em LAMB	21
5.1 Especificação do Compilador de Casamento de Padrões	22
5.2 As Funções Seletoras	23
5.3 O Algoritmo para Compilação do Casamento de Padrões	23
6 Avaliação de uma Expressão LAMB	25
6.1 Avaliação <i>Lazy</i>	25
6.2 Construtores de Dados, Entrada e Saída	26
6.3 Forma Normal	27
6.3.1 Forma Normal com Cabeça Fraca	28
6.4 Avaliando Argumentos de Funções <i>Built-in</i>	29
6.5 Como Encontrar o Próximo <i>Redex Top-level</i>	29
6.5.1 A Pilha da Espinha	30
6.5.2 Inversão de Apontadores	31
6.5.3 Avaliação de Argumentos Usando Inversão de Apontadores	32

6.5.4	Pilhas $\times$ Inversão de Apontadores	33
6.6	Reduzindo uma Aplicação Lambda	34
6.6.1	Substituição com Apontadores para os Argumentos	34
6.6.2	Reescrevendo a Raiz do <i>Redex</i>	34
6.6.3	Construindo uma Nova Instância do Corpo Lambda	35
6.7	Reduzindo a Aplicação de uma Função <i>Built-in</i>	37
6.8	O Algoritmo de Redução Completo	38
6.9	Nodos de Indireção	38
6.10	Implementando o Combinador Paradoxal <i>Y</i>	42
6.11	Conclusão	42
7	Compilador de Expressões LAMB para Supercombinadores	43
7.1	Supercombinadores e <i>Lambda-lifting</i>	43
7.1.1	A Idéia de Compilação	43
7.1.2	Solucionando o Problema das Variáveis Livres	44
7.1.3	Transformando Abstrações Lambda em Supercombinadores	47
7.2	Supercombinadores Recursivos	50
7.2.1	Notação	50
7.2.2	<i>LET's</i> e <i>DEF's</i> nos Corpos de Supercombinadores	50
7.2.3	<i>Lambda-lifting</i> na Presença de <i>DEF's</i>	51
7.2.4	Gerando Supercombinadores com Corpos do Tipo Grafo	52
7.2.5	Um Exemplo	52
7.2.6	Abordagens Alternativas	53
7.2.7	Simplificações em Tempo de Compilação	54
7.3	<i>Lambda-lifting</i> Totalmente <i>Lazy</i>	55
7.3.1	<i>Full Laziness</i>	55
7.3.2	Expressões Livres Maximais	56
7.3.3	<i>Lambda-lifting</i> Usando Expressões Livres Maximais	57
7.3.4	Um Exemplo	59
7.3.5	Implementando o <i>Lambda-lifting</i> Totalmente <i>Lazy</i>	59
7.3.6	Eliminando <i>Full Laziness</i> Redundante	62
7.4	Conclusão	63
8	O Compilador de Supercombinadores	65
8.1	A Linguagem Fonte para o Compilador de Supercombinadores	66
8.2	A Linguagem Objeto para o Compilador de Supercombinadores	66
8.2.1	Notação	68
8.2.2	Transições de Estado para a Máquina-G	69
8.3	A Compilação	79
8.3.1	Pilhas e Contextos	80
8.4	Os Esquemas de Compilação para os Supercombinadores	82
8.4.1	Notação	82
8.4.2	O Esquema F de Compilação	83
8.4.3	O Esquema S de Compilação	83
8.4.4	O Esquema S2 de Compilação	85
8.4.5	O Esquema R de Compilação	86
8.4.6	O esquema RS de Compilação	91
8.4.7	O Esquema E de Compilação	93

8.4.8	O Esquema ES de Compilação	97
8.4.9	O esquema B de Compilação	98
8.4.10	O Esquema C de Compilação	100
8.4.11	O Esquema CS de Compilação	103
8.4.12	Funções <i>built-in</i>	105
8.5	Conclusão	106
●	A Pragmática de Implementação de Semântica Denotacional	108
9.1	Estilos de Semântica Denotacional	109
9.2	Semântica de Continuação × Semântica Direta	110
9.3	Conclusão	110
10	Conclusão e Trabalhos Futuros	114
	Bibliografia	116
A	Formato da Árvore Sintática de um Programa LAMB	121
B	Sintaxe da Linguagem LAMB	127
C	Semântica da Linguagem LAMB	129
D	Semântica Direta da Linguagem LAMB	139
E	Especificação do Compilador de Casamento de Padrões	141
F	Estrutura do Compilador de Supercombinadores para C	148

Lista de Figuras

3.1	Resumo de possibilidades para compilação de linguagens funcionais	14
3.2	Estrutura do Compilador LAMB	17
4.1	Representação <i>unboxed</i> para $(*\ 4\ 2)$	20
6.1	Um exemplo de pilha da espinha	31
6.2	Caminhamento na espinha usando inversão de apontadores	32
6.3	$(LAM\ x.\ NOT\ x)\ TT \rightarrow NOT\ TT$	34
6.4	Compartilhamento de argumentos	35
6.5	Reescrevendo a raiz do <i>redex</i>	35
6.6	Redução de uma abstração lambda	35
8.1	Contexto do compilador de supercombinadores	66
8.2	Notação BNF para a linguagem fonte de supercombinadores	67
9.1	Exemplo de avaliação com semântica de continuação	111
9.2	Exemplo de avaliação com semântica direta	112
F.1	Estrutura do Compilador de Supercombinadores	148