




1. Publicação nº <i>INPE-3709-PRE/846</i>	2. Versão	3. Data <i>Novembro 1985</i>	5. Distribuição <input type="checkbox"/> Interna <input checked="" type="checkbox"/> Externa <input type="checkbox"/> Restrita
4. Origem <i>DIN/DCS</i>	Programa <i>INTAL</i>		
6. Palavras chaves - selecionadas pelo(s) autor(es) <i>LISP</i> <i>ALGOL</i> <i>COMPILAÇÃO</i>			
7. C.D.U.: <i>681.3.068</i>			
8. Título  <i>UM SISTEMA PARA A COMPILAÇÃO DE FUNÇÕES LISP</i>		10. Páginas: <i>06</i>	
		11. Última página: <i>04</i>	
9. Autoria <i>Edson Luiz França Senne</i> <i>Guilherme Bittencourt</i>		12. Revisada por  <i>Sandra Aparecida Sandri</i>	
Assinatura responsável 		13. Autorizada por  <i>Marco Antonio Raupp</i> <i>Diretor Geral</i>	
14. Resumo/Notas  <p><i>A linguagem LISP, apesar de ser largamente utilizada em sistemas de manipulação simbólica e inteligência artificial, sempre foi considerada como uma linguagem lenta quando processada em computadores convencionais. Este trabalho descreve um sistema desenvolvido com o objetivo de alcançar eficiência no processamento de programas LISP, que permite a compilação de procedimentos escritos em LISP, sem perder as vantagens do sistema de interpretação desta linguagem.</i></p>			
15. Observações <i>Este trabalho foi apresentado no 8º Congresso Nacional de Matemática Aplicada e Computacional, realizado em Florianópolis, SC, de 16 a 20 de setembro de 1985 e aceito para publicação em seus anais.</i>			

#### ABSTRACT

*The computer language LISP, although largely used for Symbolic Manipulation and Artificial Intelligence, has always been considered a slow language when processed in conventional computers. This work describes a system developed with the objective of achieving efficiency in the processing of LISP programs, that allows the compilation of LISP written programs, without losing the advantages of the interpretation system of this language.*

## UM SISTEMA PARA A COMPILAÇÃO DE FUNÇÕES LISP

Edson Luiz França Senne  
Guilherme Bittencourt

Instituto de Pesquisas Espaciais - INPE  
Ministério da Ciência e Tecnologia - MCT  
Caixa Postal 515 - 12225 - São José dos Campos - SP - Brasil

A linguagem LISP (McCarthy, 1960) a despeito de ser considerada eficiente para qualquer tipo de processamento simbólico sempre teve a "fama" de ser uma linguagem lenta. Isto se deve principalmente ao fato de que em geral são disponíveis apenas interpretadores LISP, uma vez que pela própria estrutura da linguagem a implementação de compiladores LISP apresenta algumas complicações (Honschopp et alii, 1983).

Este trabalho descreve um sistema construído com o objetivo de permitir a compilação de procedimentos escritos em LISP, sem perder as vantagens inerentes ao sistema de interpretação característico desta linguagem. O sistema compõe-se de três módulos: (i) um interpretador da linguagem LISP escrito em ALGOL; (ii) um tradutor de rotinas LISP para o ALGOL; (iii) um montador para integrar as rotinas traduzidas ao interpretador, de modo a obter um programa ALGOL no qual as funções do usuário são tratadas como primitivas da linguagem LISP. Neste processo uma limitação importante foi introduzida: os parâmetros da função do usuário são consideradas como parâmetros passados por valor em um procedimento ALGOL. Desta forma, uma vez que a linguagem LISP faz uso de escopo dinâmico e os valores de variáveis são determinados pelo ambiente de avaliação (Winston and Horn, 1981), a utilização de parâmetros de funções como variáveis "flutuantes" fica proibida. Se isto for desejado, o usuário deverá atribuir os valores dos parâmetros a variáveis, já que o controle de variáveis foi implementado de acordo com a estrutura da linguagem LISP.

O módulo interpretador reconhece dois tipos de estruturas: ÁTOMOS e LISTAS. Os ÁTOMOS podem ser de três tipos: símbolos alfa numéricos, números ou "strings". Já as listas são entidades delimitadas por parênteses, formadas por um número qualquer de elementos que podem ser tanto ÁTOMOS como novas LISTAS. Por exemplo:

A, Z123, XY2, 12, "ISTO É UM ÁTOMO": são átomos,

(A B C), (1 (A B) ((D))), ("22" 12 Z): são listas.

Neste sistema os diferentes tipos de átomos são representados por vetores, e as listas são representadas por uma matriz bidimensional. Sempre que durante uma operação faltar espaço nestas estruturas, estas serão aumentadas através do comando RESIZE do ALGOL (Segre, 1981), com a particularidade de que, uma vez terminada a operação, o espaço disponível é remanejado de modo que possa ser novamente utilizado numa operação de "garbage collection" (Cohen, 1981).

Cada função da linguagem LISP é realizada através de um procedimento ALGOL adequado, existindo além destes os procedimentos específicos para a manipulação das estruturas internas e para a entrada e saída, isto é, para a transformação da representação interna (matrizes e vetores) nas listas e nos átomos que serão impressos, e das listas e dos átomos lidos na representação interna.

A rotina "EVAL" é responsável pela avaliação de qualquer entidade (LISTAS ou ÁTOMOS) submetida ao interpretador. Através dela é feito o encaminhamento do controle para uma das rotinas internas que implementam a semântica das primitivas LISP, ou para alguma rotina traduzida.

O módulo tradutor recebe como entrada um arquivo de funções LISP do tipo EXPR (Marti et alii, 1979), pois, uma vez que a linguagem ALGOL não permite procedimentos com um número variável de parâmetros, as funções do tipo FEXPR não podem ser traduzidas, devendo ser

definidas através do interpretador. Como saída este módulo produz os arquivos: TRADUTOR/SAÍDA, que contém os procedimentos ALGOL equivalentes às funções LISP definidas no arquivo de entrada, e TRADUTOR/FUNÇÕES, que armazena os nomes e o número de parâmetros das funções traduzidas e também os nomes das variáveis globais que deverão ser declaradas. Devido ao arquivo TRADUTOR/FUNÇÕES, é possível utilizar o módulo tradutor repetidas vezes para vários arquivos de programas LISP.

No processo de tradução, as referências às funções primitivas são transformadas em chamadas às rotinas internas do interpretador, e as referências às funções definidas pelo usuário são transformadas em chamadas aos procedimentos ALGOL traduzidos (ou declarados como FORWARD, caso não tenham sido traduzidos ainda). Um tratamento especial é dado às seguintes primitivas LISP: DE, SET, SETQ, COND, PROG, PROGN, QUOTE, RETURN, GO, além das funções de número variável de parâmetros: LIST, PLUS, TIMES, AND, OR, MAX e MIN.

O módulo montador recebe como entrada os arquivos TRADUTOR/SAÍDA, TRADUTOR/FUNÇÕES e LISP/INTERPRETADOR (que contém o módulo interpretador) e produz como saída um novo interpretador incorporando os procedimentos do arquivo TRADUTOR/SAÍDA como primitivas.

Em testes preliminares realizados obtiveram-se tempos de processamento da ordem de sete vezes menores que os tempos gastos pela mesma rotina em um sistema LISP interpretado (Marti et alii, 1979), sem considerar o tempo que seria necessário para definir as rotinas, pois no presente sistema as rotinas do usuário são compiladas juntamente com o interpretador, não utilizando nenhum tempo de processamento para que sejam definidas.

Utilizando a experiência obtida com este sistema pretende-se substituir a linguagem ALGOL pela linguagem PASCAL, visando tornar o sistema transportável para microcomputadores.

## REFERÊNCIAS BIBLIOGRÁFICAS

- COHEN, J. Garbage collection of linked data structures. *Computing Surveys*, 13(3):341-367, 1981.
- HONSCHOPP, U.; LIPPE, W.M.; SIMON, F. Compiling functional languages for von Neumann machines. *ACM Sigplan Notices*, 18(6):22-27, 1983. Proceedings of the Sigplan'83 Symposium on Programming Languages Issues in Software Systems.
- MARTI, J.B.; HEARN, A.C.; GRISS, C. *Standart LISP report*. Salt Lake City, UT., University of Utah, 1979.
- McCARTHY, J. Recursive functions of symbolic expressions and their computation by machine, Part. I. *Communications of the ACM*, 12(3): 184-195, 1960
- SEGRE, L.M. *Linguagem de programação ALGOL*. Campus, Rio de Janeiro, RJ, 1981.
- WINSTON, P.H.; HORN, B.K.P. *LISP*. Addison-Wesley, Reading, MA, 1981.