# Uma proposta de linguagem de programação funcional com características adaptativas

***Ricardo Luis de Azevedo da ROCHA*[1,2]**
**e-mail: luis.rocha@poli.usp.br**

***João José NETO*[1]**
**e-mail: joao.jose@poli.usp.br**

*[1] Escola Politécnica da Universidade de São Paulo*
*Av. Luciano Gualberto, trav. 3, n. 158, Cidade Universitária*
*05508-900 – São Paulo, SP – Brasil*

*[2] Fundação Santo André, Faculdade de Engenharia, Curso de Eng. da Computação*
*Av. Príncipe de Gales, 821, Santo André, SP, Brasil, 09060-650*

## Resumo

O principal objetivo nesta pesquisa é investigar a viabilidade prática do uso da tecnologia adaptativa em um ambiente de programação funcional. Utilizou-se como base a linguagem de programação LISP, à qual se incorporou uma extensão adaptativa interpretada.

## Abstract

The main goal of this research is to investigate practical use of adaptive technology in a functional programming environment. We used the programming language LISP as a basis, and added to it an interpreted adaptive extension.

**Palavras-chave**: Linguagem de Programação, Linguagem de Programação Funcional, Tecnologia Adaptativa, Implementação de Modelo.

**Indicação**: Ao Congresso em Geral

# 1. Introdução

Um dispositivo é considerado adaptativo quando o seu comportamento muda dinamicamente, em resposta a estímulos de entrada, sem a interferência de agentes externos.

Para obter esta característica um dispositivo adaptativo deve ser automodificável, isto é, capaz de alterar sua própria estrutura quando em funcionamento.

A definição de computação adaptativa foi formalizada em [2], que a caracteriza para qualquer dispositivo baseado em regras. Este artigo tem por objetivo propor a inclusão de características adaptativas em uma linguagem funcional, como ponto de partida para caracterização de uma linguagem de comportamento adaptativo.

Uma característica dos dispositivos adaptativos que os torna particularmente atraentes para uma gama muito vasta de aplicações é sem dúvida a sua generalidade, que os habilita a atuarem em aplicações de extrema complexidade.

Isoladamente considerada, a característica da generalidade tende a exigir, tanto na teoria como na prática, que sejam criados recursos muito robustos e capazes de levar em conta uma ampla gama de situações, muitas de alta dificuldade.

Isso se reflete no aumento do tamanho, na perda de velocidade e na resultante queda de eficiência das implementações de dispositivos de alta generalidade, e leva, muitas vezes, para muitos problemas do dia-a-dia, à busca de soluções particulares e mais eficientes, porém menos flexíveis.

Como conseqüência direta, com freqüência têm sido descartadas pelos implementadores, inúmeras vezes a priori, as soluções abrangentes, as quais, no entanto, em situações de utilização prolongada e diversificada, definitivamente se mostram mais adequadas, apesar de apresentarem um custo inicial mais alto.

Todavia, é possível tirar vantagem tanto da abrangência como da eficiência, lançando-se mão dos dispositivos adaptativos: dotados de comportamento dinâmico, dão a seus projetistas a possibilidade de alterarem o conjunto de recursos por eles utilizados, em função das particulares necessidades da particular situação enfrentada em cada momento.

Assim, se adequadamente projetados, os dispositivos adaptativos podem operar de forma tal que: (a) nenhum recurso disponível seja incorporado, a não ser que estritamente necessário; (b) todo recurso utilizado e não mais necessário pode ser descartado, evitando-se assim onerar o sistema com custos adicionais evitáveis; (c) partes do dispositivo que estejam operando de forma ineficiente podem ser substituídas por outras mais adequadas, evitando-se assim o desperdício de tempo de processamento; (d) o dispositivo pode ser enriquecido pelo projetista com instrumentos de coleta de informações sobre sua operação, para efeito de auto-avaliação, os quais podem fornecer ao seu projetista informações muito preciosas para a sintonização de sua operação.

Assim procedendo, é possível manter disponível um banco de recursos (que representará o custo inicial da utilização dessa classe de dispositivos), e, através de um projeto criterioso, em que sejam considerados os pontos acima levantados, torna-se possível oferecer ao usuário toda a flexibilidade permitida pela generalidade do modelo e toda a economia proporcionada pelas características dinâmicas inerentes à adaptatividade.

### *1.1 Motivação*

A motivação para este trabalho é fundamentada no argumento acima exposto, de que programas com comportamento adaptativo têm um potencial de desempenho capaz de superar o de programas convencionais equivalentes que resolvem o mesmo problema. Pesquisas recentes sobre tecnologia adaptativa como as conduzidas em [3, 4, 5] têm mostrado que, na prática, o acréscimo do custo inicial, associado à inclusão de características adaptativas, não é exagerado, podendo, com freqüência, mostrar-se extremamente baixo.

Isso sinaliza que, embora a formulação geral de um dispositivo adaptativo encerre em si uma complexidade computacional considerável, devido às inúmeras situações nada triviais com as quais se propõe a lidar no caso geral, em circunstâncias normais, que representam a maior parte dos casos práticos de interesse, a própria característica adaptativa desses dispositivos leva-os a assumirem configurações nas quais são totalmente dispensadas as formas mais sofisticadas de operação, recaindo-se dessa forma, com altíssima freqüência, em modelos muito eficientes e estruturalmente simples, e, portanto, com baixa complexidade computacional.

Outro aspecto motivador é o interesse de dispor de uma linguagem de programação adaptativa para uso normal.

Para isso, pretende-se, através da inserção, em uma linguagem de programação disponível, da possibilidade, inerente aos formalismos adaptativos, de exprimir com maior naturalidade atividades de auto-modificação espontânea.

Esperamos, como resultado, disponibilizar em breve um protótipo operante de uma linguagem funcional, com características de automodificação dinâmica, através de cuja utilização seja facilitada a tarefa de encontrar soluções adaptativas para problemas, soluções essas que se mostrem melhores ou, ao menos, mais expressivas que as usualmente obtidas com a ajuda de da implementação de modelos clássicos usando linguagens de programação tradicionais.

### *1.2 Objetivos*

O nosso principal objetivo nesta pesquisa é investigar a viabilidade prática do uso da tecnologia adaptativa em um ambiente de programação funcional.

Primeiramente, deseja-se projetar detalhadamente o funcionamento de uma extensão da linguagem LISP que incorpore à linguagem LISP original, que lhe servirá de substrato, a funcionalidade de uma linguagem adaptativa, e também propor a estrutura de um ambiente de programação apoiado nessa extensão.

Como resultado deverá surgir um primeiro protótipo executável dessa extensão, totalmente baseado na definição das extensões através de implementações fundamentadas em formalizações desenvolvidas usando a semântica operacional, e utilizando o próprio LISP como metalinguagem.

Pretende-se utilizar esse protótipo para sintonizar e ensaiar a linguagem estendida resultante desse experimento, modificando-a até que seja obtida uma versão aceitável, expressiva e de uso confortável.

Em suas etapas seguintes, esse projeto prevê a completa formalização da proposta, usando como metalinguagem o cálculo lambda, bem como a implementação de um ambiente de programação centrado nessa linguagem funcional estendida com características da computação adaptativa.

## 2. Fundamentos

### *2.1 Adaptatividade*

Dispositivos formais adaptativos são definidos como sendo aqueles cujo comportamento varia dinamicamente ao longo de sua operação, como reação do dispositivo, em determinadas configurações, à recepção de certos estímulos em sua entrada.

Em dispositivos adaptativos, essa reação faz parte das características intrínsecas do dispositivo, e deve ocorrer espontaneamente, sem a interferência do operador ou de outros agentes estranhos ao dispositivo.

Para incorporar tais características, um dispositivo adaptativo cujo comportamento seja completamente descrito por um conjunto de regras, nada mais deve fazer, para alterar esse comportamento, que modificar correspondentemente o conjunto de regras que o definem.

Para tornar isso viável, tais dispositivos automodificáveis devem antecipar todas as possíveis alterações que seu comportamento possa vir a sofrer sempre que ocorrer alguma situação que exija modificação do conjunto de regras de operação (note-se que não há necessidade de que todas essas alterações estejam integralmente definidas a priori, embora se exija que o dispositivo esteja apto a determiná-las em qualquer situação em que as modificações possam fazer-se necessárias).

Assim, os dispositivos adaptativos devem ser criados de tal modo que sejam capazes de detectar situações nas quais não lhes seja possível dar continuidade a sua operação sem antes alterar seu comportamento, e, em resposta a tal necessidade, promover todas as alterações necessárias antes de prosseguir.

O artigo [2] apresenta todos os detalhes do formalismo que define os dispositivos adaptativos baseados em regras, e o presente trabalho segue o formalismo aí proposto, assim como a terminologia e a notação adotada.

Em resumo, um formalismo adaptativo dirigido por regras pode ser visto como a composição de um formalismo subjacente não-adaptativo, definido por um conjunto de regras, ao qual tenha sido acoplado um mecanismo adaptativo, responsável por associar a cada regra do formalismo subjacente alguma ação adaptativa, encarregada de modificar o conjunto de regras subjacente e seu relacionamento com o mecanismo adaptativo.

Assim, cada um dos passos de operação de um dispositivo adaptativo consiste na seleção de uma regra a ser executada (com base na configuração corrente e no particular estímulo de entrada recebido), e de alguma ação adaptativa que eventualmente esteja com ela relacionada; a seguir, a regra selecionada determina a nova configuração do dispositivo, e a ação adaptativa associada, as eventuais alterações a serem sofridas pelo conjunto de regras.

Uma vez aplicadas a regra e as ações adaptativas, o dispositivo volta a selecionar a próxima regra, reiniciando o ciclo, que se repete até que o dispositivo atinja alguma configuração de aceitação ou de rejeição para a cadeia de entrada que estiver sendo processada.

Geral e independente do particular formalismo escolhido, essa formulação tem ampla aplicação, podendo ser adotada também em ambientes de programação, considerada a linguagem de programação como formalismo subjacente.

### *2.2 Programação funcional*

O modelo de computação proposto por Church em cálculo lambda, a chamada definibilidade lambda, vem a ser a base do paradigma de programação funcional [1]. Isto ocorre porque os programas funcionais são mais próximos de especificações de problemas computacionais do que programas imperativos.

As maiores dificuldades em relação à implementação de ambientes de programação (editores, compiladores) funcionais residiam nas questões de eficiência de espaço ocupado, e de desempenho. Entretanto já há algum tempo soluções para os problemas mencionados foram encontradas, e ambientes de programação funcional comerciais podem ser encontrados [6].

Uma forma simples e usual de implementação de linguagem funcional consiste na "aplicação", por exemplo, uma expressão da forma '*F A*' é avaliada da seguinte forma, primeiramente as expressões *F* e *A*, em qualquer ordem, e depois se avalia a ligação (*contrato*) entre *F* e *A*. Uma linguagem implementada desta forma é LISP.

## 3. Proposta

Com base na implementação baseada no conceito de aplicação, e procurando explorar o conceito da adaptatividade, anteriormente discutido, chega-se à proposta que passamos a apresentar.

Nossa meta, na presente proposta, é a especificação de uma linguagem com características adaptativas, obtida por extensão da linguagem funcional LISP mediante a utilização de mecanismos adaptativos implementados com o auxílio do conceito de aplicação.

Nessa linguagem adaptativa que desejamos obter, o formalismo não-adaptativo que lhe servirá de substrato corresponde a alguma implementação disponível da linguagem de programação LISP.

Do ponto de vista conceitual, vamos adotar, como formalismo subjacente do formalismo que estamos propondo, um núcleo básico puramente funcional da linguagem LISP.

Na prática, vamos trabalhar com esse núcleo como um subconjunto da implementação disponível, e, para tanto, simplesmente deixaremos de utilizar as construções implementadas que não pertençam ao núcleo escolhido.

A despeito das diferenças que se notam entre as diversas linguagens funcionais existentes, e de algumas características não funcionais que elas possam apresentar, o processo de programação, nestas linguagens, é baseado na construção de funções e em sua avaliação pela aplicação das mesmas a um particular conjunto de dados.

Entretanto os dados, em uma linguagem de programação funcional, também podem ser interpretados como funções lambda, ou como abreviações das mesmas.

Com base nessas considerações, escolheu-se, como núcleo da linguagem, a ser usado como formalismo subjacente da linguagem adaptativa aqui proposta, um subconjunto equivalente ao cálculo lambda, já que este é o modelo de computação mais aderente às linguagens funcionais, e que dá margem, através do mecanismo de aplicação, à criação dos mecanismos adaptativos da linguagem proposta, na forma de uma extensão do núcleo.

Dessa maneira, a forma escolhida para conectar os conceitos de adaptatividade e de programação funcional foi a de usar o mecanismo original de avaliação da linguagem disponível para criar, por extensão funcional, um avaliador adaptativo.

O avaliador adaptativo assim obtido mantém o controle sobre as construções utilizadas pelo programador, lançando mão do avaliador existente para efetuar a avaliação das construções do núcleo, contidas no programa, e tomando para si a tarefa de identificar, manipular e interpretar adequadamente as construções adaptativas utilizadas no programa através da execução de funções que compõem os mecanismos de extensão implementados.

Um esquema desse processo está ilustrado na figura 1.

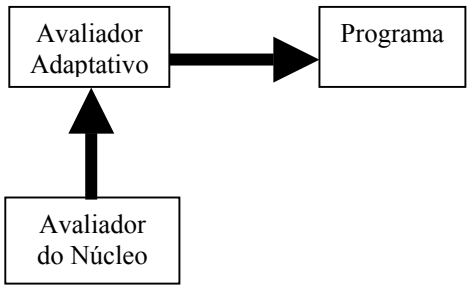


Figura 1: Esquema de funcionamento proposto

O avaliador adaptativo determina quais ações adaptativas devem ser executadas em cada momento, e deve controlar a execução das mesmas.

Para tanto, necessita manter um registro de todas as operações associadas à execução de cada uma das ações adaptativas utilizadas pelo usuário em seu programa, de modo que, toda vez que uma delas for ativada pela execução de alguma construção adaptativa em seu programa, a correspondente ação adaptativa é então identificada, passando o avaliador adaptativo a executá-la.

Assim sendo, ativada uma ação adaptativa, o avaliador adaptativo aplicará adequadamente as funções que implementam o mecanismo adaptativo da linguagem estendida, de forma que seja executada, sobre a instância corrente do programa adaptativo do usuário, cada uma das atividades que compõem tal ação.

Dispõe-se de três atividades básicas, denominadas ações adaptativas elementares, com as quais as ações adaptativas podem ser especificadas: as ações de inserção, as de remoção e as de consulta.

Através de ações elementares, podem ser definidas aquelas ações adaptativas associadas a cada instrução do programa, especificando-se, dessa forma, quais são, durante a execução do programa, as operações de edição que se deseja aplicar ao programa adaptativo a cada vez que for executada a instrução à qual essa ação adaptativa estiver associada.

Para ser capaz de manipular todas essas operações, o avaliador adaptativo deverá, portanto, dispor de funções que se encarreguem da aplicação das três operações primitivas mencionadas sobre a instância corrente do programa adaptativo em execução.

O arranjo da figura 1 permite, pois, explorar ambas as características desejadas, facilitando a construção de programas dinamicamente mutáveis, bem como sua especificação através de uma linguagem funcional adaptativa, cuja base de programação seja, portanto, a mesma utilizada pela linguagem LISP empregada em sua implementação.

Na prática, o avaliador adaptativo funciona da seguinte forma: cada construção sintática do programa adaptativo recebe inicialmente, do seu ambiente de execução, uma identificação unívoca (optamos pela simplicidade, atribuindo a cada uma um número natural seqüencial).

Sempre que necessário localizar uma dessas construções, para efeito de aplicação de uma das ações adaptativas elementares de eliminação, inserção ou consulta, essa identificação é utilizada pelo avaliador.

Naturalmente, é necessário que o usuário tenha algum tipo de acesso a uma ou outra construção do seu programa, cuja alteração pretende promover através de ações adaptativas, e para isso rótulos simbólicos são associados pelo usuário, à sua escolha, com a finalidade de permitir que tais construções sejam devidamente referenciadas pelo programa.

Como algumas construções sintáticas, inicialmente inexistentes, podem ser incluídas no programa, por obra das ações adaptativas, durante a sua execução, sempre que necessário for efetuar futuras referências a tais construções, o usuário deverá tomar o cuidado de associar nomes simbólicos a elas no ato de sua criação, ou tais construções deixarão de ser acessíveis para efeito de referência.

Naturalmente, por sua vez, algumas construções inicialmente existentes podem ser eliminadas, e caso isso ocorra com uma construção identificada simbolicamente pelo usuário, essa identificação perderá sua função, e não mais poderá ser utilizada, sob pena de perda de consistência das referências simbólicas.

### *3.1 Associação dos conceitos funcional e adaptativo*

Na presente proposta, o papel do avaliador adaptativo é central nesse tipo de implementação usando como base uma linguagem funcional, já que permite estender linguagem, sem alterar sua sintaxe ou semântica original. Esquematicamente tem-se a seguinte estrutura:

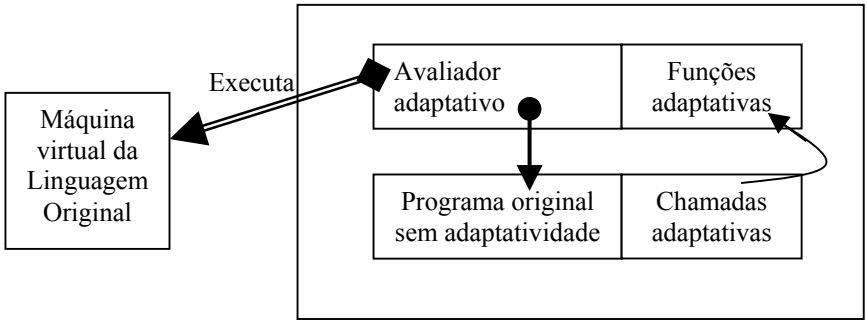


Figura 2: Esquema da estrutura de funcionamento da linguagem funcional adaptativa proposta.

Assim um programa funcional adaptativo pode ser interpretado como sendo composto de duas partes, a primeira, puramente funcional, e a segunda, referente à adpatatividade encerrada no programa em questão.

Essa porção adaptativa do programa compõe-se de duas partes: na primeira, encontra-se o conjunto de todas as definições das funções adaptativas a serem utilizadas pelo programa, as quais são agregadas ao avaliador adaptativo para que possam ser acionadas sempre que necessário; na segunda, tem-se o conjunto de todas as ações adaptativas empregadas no programa, ou seja, das chamadas que são feitas, pelas construções sintáticas da linguagem subjacente, às funções adaptativas a elas atreladas, que são explicitamente referenciadas no corpo do programa adaptativo.

Sempre que uma parte do programa à qual esteja associada uma ação adaptativa é ativada, a correspondente função adaptativa é executada, e nessa ocasião o programa em execução e as chamadas de funções adaptativas a ele atreladas podem ser modificados por obra do mecanismo adaptativo da linguagem, através das operações de edição implementadas pelas ações adaptativas elementares que constituem a ação adaptativa ativada.

Na implementação proposta, o avaliador adaptativo encapsula todas essas características adaptativas, incumbindo-se de, sempre que necessário, efetuar todas as alterações necessárias no programa em execução, de forma que todas as operações referentes à adaptatividade do programa sejam reduzidas a operações da linguagem funcional subjacente, o que permite, dessa maneira, que o programa adaptativo seja executado, em última instância, pela máquina virtual da linguagem original.

### *3.2 Formalização da associação proposta*

Pode-se formalizar a associação acima descrita admitindo-se a existência de um avaliador funcional $\lambda_{EVAL}$ para alguma linguagem funcional, o qual efetua a avaliação de qualquer expressão lambda fornecida retorna o resultado dessa avaliação.

Em um ensaio experimental inicial, construímos uma função lambda, com o papel de um avaliador adaptativo, e essa função, incumbida de executar ações adaptativas sobre um programa funcional passado como argumento.

Seja $\lambda_{ADAP}$ tal avaliador adaptativo, e sejam P o programa funcional, A o conjunto de ações adaptativas associadas às construções sintáticas de P, em correspondência a um conjunto de funções adaptativas definidas em F.

Desta forma, a avaliação do programa P, cujas ações adaptativas estão em A, referentes às funções adaptativas em F, é dada pela seguinte expressão lambda:

$$(\lambda_{EVAL}\ (\lambda_{ADAP}\ (P\ (A)))\ F)$$

Dessa maneira, através das ações adaptativas, o avaliador adaptativo pode operar sobre o programa em execução. Apesar de fazerem parte do programa final, as funções adaptativas são descritas à parte, compondo com o avaliador adaptativo um conjunto escrito na linguagem funcional original.

## 4. Exemplo Ilustrativo

Para ilustrar esta proposta empregamos, como exemplo, uma implementação de tabela de decisão adaptativa em paradigma funcional, isto é, implementada em LISP puro.

A estrutura adotada para a implementação de tabelas em LISP assemelha-se à sugerida em [6], ou seja, os modelos são estruturados segundo um conjunto de listas, a cada um de cujos membros se associa uma propriedade.

De forma semelhante, uma linha ou uma coluna dessas tabelas é representada na forma de um conjunto de estados em um autômato.

Para implementar uma tabela de decisão adaptativa, utilizamos a estrutura descrita em [2]. A tabela de decisão adaptativa é composta das seguintes linhas e colunas:

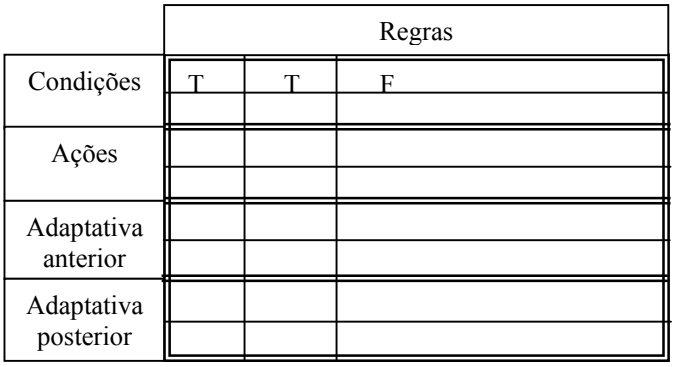


Figura 3: Componentes da representação de uma tabela de decisão adaptativa.

A representação desta tabela de decisão em linguagem funcional, no presente caso a linguagem LISP, pode ser feita através do seguinte esquema:

- Na figura 3, cada regra (coluna) representa uma possível transição do autômato. Assim, a configuração corrente do autômato é representada, de fato, pelo conjunto de colunas que se referem às possíveis transições a partir da corrente configuração. Cada regra codifica a situação à qual se aplica nas células correspondentes às linhas rotuladas como condições, e codifica a nova configuração através das células correspondentes às linhas rotuladas como ações.
- Em LISP, como tal estrutura é representada através de listas, adota-se o seguinte:
  - A lista mais abrangente representa uma regra;
  - Cada regra do formalismo subjacente é representada por uma lista formada de duas listas, representando, respectivamente, condições e ações;
  - As associações do formalismo subjacente ao mecanismo adaptativo são supridas por outra lista, composta por sua vez de duas listas, que indicam a particular forma de ativação das funções adaptativas, anteriores e as posteriores, associadas à regra correspondente;
  - Cada função adaptativa é formalizada por meio de uma lista de ações adaptativas elementares, cada qual com o formato similar ao de uma regra;
  - Cada coluna da tabela (e, portanto, cada uma das listas que as representam) tem um marcador, representado em seu primeiro elemento, que indica o papel desempenhado pela mesma – regra, ação adaptativa elementar de inclusão, idem de eliminação, idem de consulta, etc;
- A tabela de decisão tem duas regras identificadas na lista mais abrangente, a regra inicial, e a regra final.

Assim a operação do formalismo pode ser controlada através da escolha da seqüência de ações executadas, a partir da aplicação da regra inicial, com base nos símbolos de entrada encontrados.

É natural que, se permitidas situações de não-determinismo, será necessário levar em consideração o conjunto de todas as possíveis configurações seguintes a partir de cada configuração de partida.

Uma possível representação para a tabela de decisão a que nos referimos é representada por uma lista da seguinte forma:

```
(Tabela
   ( Regra 1
        (lista de condições) (lista de ações) (ação adaptativa anterior) (ação adaptativa posterior)  )
   (Regra 2     ...      ...      ...     )
        .....
   regra-inicial
```

regra-final )

```
(defun busca-estado (est lis)
"*** Busca na lista mais abrangente as regras cujo estado de partida é est ***"
  (let ((res nil))
    (dolist (transicao lis res)
      (if (equal est (first transicao))
          (setf res (append res (list transicao))))))
  )
```

Assim cada regra presente na lista resultante deve ser executada, ou seja, a seqüência de regras deve ser seguida até que seja executada a regra final da tabela, ou então até que, por qualquer razão, a inexistência de previsão de uma configuração seguinte não permita o prosseguimento do processo.

```
(defun mudar-est (modelo n-est)
    "*** Troca o valor do estado do modelo para n-est ***"
  (let ((res nil))
    (dotimes (est *num-automa* res)
      (let ((val (nth est *lista-exec*)))
        (if (eq est modelo)
            (setf res (append res (list n-est)))
          (setf res (append res (list val)))))))
  )

(defun transicionar (modelo estado simb)
    "*** Efetuar uma transição em um modelo a partir de estado e simb ***"
  (let ((lista-est (nth modelo *lista-ok*))
        (tran nil))
    (if (not (null (setf tran (first (busca-estado estado (busca-simb simb
lista-est))))))
        (setf *lista-exec* (mudar-est modelo (third tran)))
      (setf *lista-exec* (mudar-est modelo 0))))
  )
```

Como se pode notar, a aplicação de uma ação adaptativa pode modificar a dimensão e o conteúdo da tabela que estiver sendo utilizada, permitindo que o modelo em execução seja, a cada passo, diferente do que era no passo anterior.

A execução do formalismo adaptativo após a aplicação de uma ação adaptativa deve corresponder portanto à tentativa de, a partir da configuração corrente, aplicar alguma regra da nova tabela, mudando de configuração e executando as ações previstas. O processo segue até que se atinja a regra final ou um impasse em que não seja possível evoluir para uma nova configuração.

### *4.1 Avaliação*

Embora simples, a solução adotada neste ensaio, na forma de um interpretador, foi capaz de materializar e mostrar a viabilidade de implementação de um formalismo dinâmico (no caso, uma tabela de decisões adaptativa) através de uma extensão adaptativa da linguagem LISP. O desempenho do primeiro protótipo não foi adequado, porque cada modelo gerado, potencialmente não-determinístico, a cada passo aumenta, combinatoriamente, o número de caminhos possíveis. Procurando uma solução melhor, estamos simplificando o tratamento pela linearização da execução a cada passo (efetuando em cada modelo esse processo em ordem lexicográfica, conforme [6]).

## 5. Conclusão

Como esperado, restringindo apenas a modelos determinísticos, melhora-se muito o desempenho. Entretanto, acreditamos que o trabalho mostrou viável nossa proposta: devemos buscar a formalização adequada de uma linguagem funcional adaptativa, e definir suas sintaxe e semântica.

O trabalho desenvolvido é parte de inicial de uma pesquisa na linha de linguagens funcionais, conduzida no Laboratório de Linguagens e Tecnologias Adaptativas (LTA)[*]. Como resultado, apresenta a formulação em linguagem funcional com características adaptativas, e um exemplo de aplicação das mesmas na implementação de tabelas de decisão adaptativas.

A proposta deste artigo é de ilustrar a utilização de características adaptativas em uma linguagem funcional através de um avaliador escrito na própria linguagem, agindo, portanto, como um interpretador. O próximo passo dessa pesquisa será definir completamente a extensão sintática da linguagem funcional adotada, e produzir, usando-a como linguagem de implementação, um ambiente adequado de programação para a nova linguagem funcional adaptativa proposta.

## 6. Referências


[1] Barendregt, H. P. **The Lambda Calculus: its Syntax and Semantics.** North-Holland. Amsterdam, 1984.

[2] Neto, J. J. **Adaptive Rule-Driven Devices - General Formulation and Case Study.** Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol.2494, Pretoria, South Africa, July 23-25, 2001, pp. 234-250.

[3] Neto, Joao Jose and Moraes, Miryam de. **Using Adaptive Formalisms to Describe Context-Dependencies in Natural Language.** Lecture Notes in Artificial Intelligence. N.J. Mamede, J. Baptista, I. Trancoso, M. das Graças, V. Nunes (Eds.): Computational Processing of the Portuguese Language 6th International Workshop, PROPOR 2003, Volume 2721, Faro, Portugal, June 26-27, 2003, pp 94-97.

[4] Menezes, Carlos Eduardo Dantas de e Neto, João José. **Um método híbrido para a construção de etiquetadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos.** Anais da Conferencia Iberoamericana en Sistemas, Cibernética e Informática, 19-21 de Julio, 2002, Orlando, Florida.

[5] Pistori, Hemerson e Neto, João José. **AdapTree - Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas.** Anais Conferência Latino Americana de Informática - CLEI 2002. Montevideo, Uruguai, Novembro, 2002.

[6] Rocha, R. L. A. e Neto, J. J. **Construção e Simulação de Modelos Baseados em Autômatos Adaptativos em Linguagem Funcional** Proceedings of ICIE 2001 - International Congress on Informatics Engineering, Buenos Aires: Computer Science Department - University of Buenos Aires, v. CD-ROM, p. 509-521, 2001

[7] Abelson, H.; Sussman, G. J. **Structure and Interpretation of Computer Programs.** MIT Press. Cambridge, MA, 1996.


[*] www.pcs.usp.br/~lta