# MEMÓRIAS DO WTA 2011

# QUINTO WORKSHOP DE TECNOLOGIA ADAPTATIVA

WTA 2011

Laboratório de Linguagens e Técnicas Adaptativas
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

São Paulo
2011

**WTA 2011 – Quinto Workshop de Tecnologia Adaptativa**

# MEMÓRIAS DO WTA 2011

# QUINTO WORKSHOP DE TECNOLOGIA ADAPTATIVA


Laboratório de Linguagens e Técnicas Adaptativas
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

São Paulo
2011

## FICHA CATALOGRÁFICA

**Workshop de Tecnologia Adaptativa (5 : 2011 : São Paulo)**
**Memórias do WTA´2011. – São Paulo ; EPUSP, 2011.**
**111 p.**

**ISBN 978-85-86686-61-0**


**1.Engenharia de computação(Congressos) 2.Teoria da computação (Congressos) 3.Teoria dos autômatos (Congressos) 4.Semântica de programação (Congressos) 5.Linguagens formais (Congressos) I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.**

**CDD 621.39**

# SUMÁRIO

# APRESENTAÇÃO

# WTA'2011

A quinta edição do WTA realizou-se em São Paulo, BRASIL, nos dias 31 de janeiro e 01 de fevereiro de 2011, na Escola Politécnica da Universidade de São Paulo, por meio do Departamento de Engenharia de Computação e Sistemas Digitais.

As contribuições encaminhadas na forma de artigos relacionados à Tecnologia Adaptativa, nas seguintes áreas, abrangeram, de forma não exclusiva, os tópicos abaixo:

TÓPICOS

- Teoria da Computação
- Autômatos
- Computação inspirada na biologia
- Compiladores
- Gramáticas
- Processamento de Linguagem Natural
- Jogos eletrônicos
- Tomada de decisões
- Inferência Gramatical

COMISSÃO DE PROGRAMA

Almir Rogério Camolesi (Assis, SP, Brasil)
Amaury Antônio de Castro Junior (Coxim, MS, Brasil)
Aparecido Valdemir de Freitas (São Caetano do Sul, SP, Brasil)
César Alberto Bravo Pariente(São Paulo, SP, Brasil)
Hemerson Pistori (Campo Grande, MS, Brasil)
Marcus Vinicius Midena Ramos (São Paulo, SP, Brasil)

COMISSÃO ORGANIZADORA

André Riyuiti Hirakawa (São Paulo, SP, Brasil)
Cinthia Itiki (São Paulo, SP, Brasil)
Italo Santiago Vega (São Paulo, SP, Brasil)
João José Neto, Chair (São Paulo, SP, Brasil)
Ricardo Luis de Azevedo da Rocha (São Paulo, SP, Brasil)

# Memórias do WTA 2011: Quinto Workshop de Tecnologia Adaptativa

R. L. A. Rocha

*Resumo* — **Esta publicação do Laboratório de Linguagens e Técnicas Adaptativas do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP é uma coleção de textos produzidos para o WTA 2011, o Quinto Workshop de Tecnologia Adaptativa, realizado em São Paulo nos dias 31 de Janeiro e 01 de Fevereiro de 2011. A exemplo da edição de 2010, este evento contou com uma forte presença da comunidade de pesquisadores que se dedicam ao estudo e ao desenvolvimento de trabalhos ligados a esse tema em diversas instituições brasileiras e estrangeiras, sendo o material aqui compilado representativo dos avanços alcançados nas mais recentes pesquisas e desenvolvimentos realizados.**

***Palavras-chave* — Adaptatividade, Autômatos adaptativos, Comportamento Auto-modificável, Sistemas Adaptativos, Tecnologia Adaptativa.**

## I. INTRODUÇÃO

Com muita satisfação apresentamos neste documento estas memórias com os artigos apresentados no WTA 2011 – Quinto Workshop de Tecnologia Adaptativa, realizado na EPUSP nos dias 31 de janeiro e 01 de fevereiro de 2011.

Constatou-se o terceiro sucesso desse evento pela efetiva participação de uma centena de pesquisadores, pelo recebimento de mais de 40 artigos, dos quais 9 foram selecionados completos, 11 selecionados como artigos que configuram pesquisa em andamento e 3 como comunicações. Os artigos mais bem avaliados foram aprimorados por seus autores para publicação na revista IEEE Latin American Transactions.

Na presente edição do WTA observaram-se algumas evoluções em relação à do ano anterior:

- **Chamada aberta de trabalhos**: facilitou-se assim o envolvimento e a aproximação de interessados externos;
- **Diversificação dos participantes e dos trabalhos:** foram recebidas e aceitas excelentes contribuições técnicas, oriundas de fora de S.Paulo e do Brasil;
- **Prestigiosos apoios:** da SBC (Sociedade Brasileira de Computação), SPC (Sociedad Peruana de Computación), IEEE (The Institute of Electrical and Electronics Engineering), EPUSP (Escola Politécnica da USP) e PCS (Dep. de Eng. de Computação e Sistemas Digitais).
- 

R. L. A. Rocha é pesquisador do Laboratório de Linguagens e Técnicas Adaptativas, Escola Politécnica da USP, São Paulo/SP, Brasil; fone: 55 11-3091-5583; e-mail: luis.rocha@poli.usp.br.

## II. ESTA PUBLICAÇÃO

Estas memórias espelham o conteúdo apresentado no WTA 2010.

As quatro seguintes áreas dominaram o WTA 2011:

- **Algoritmos**
- **Ambiente de execução adaptativo**
- **Aplicações on-line**
- **Processamento de linguagem natural**

A exemplo do que foi feito no ano anterior, todo o material referente aos trabalhos apresentados no evento estarão acessíveis no portal do WTA 2011, incluindo softwares e os slides das apresentações das palestras. Adicionalmente, o evento foi gravado em vídeo, em sua íntegra, e os filmes serão também disponibilizados aos interessados.

Esperamos que, pela qualidade e diversidade de seu conteúdo, esta publicação se mostre útil a todos aqueles que desejam adquirir ou aprofundar ainda mais os seus conhecimentos nos fascinantes domínios da Tecnologia Adaptativa.

## III. CONCLUSÃO

A repetição do sucesso das edições anteriores do evento, e o nível de qualidade dos trabalhos apresentados atestam a seriedade do trabalho que vem sendo realizado, e seu impacto junto à comunidade.

Somos gratos aos que contribuíram de alguma forma para o brilho do evento, e aproveitamos para estender a todos o convite para participarem da próxima edição, em 2012.

## AGRADECIMENTO

Às instituições que apoiaram o WTA 2011, à comissão organizadora, ao pessoal de apoio e a tantos colaboradores voluntários, cujo auxílio propiciou o êxito que tivemos a satisfação de observar.

São Paulo, fevereiro de 2011.
Prof. Dr. Ricardo Luis de Azevedo da Rocha
LTA – PCS - EPUSP

# Créditos

São muitos os que deram ao nosso evento um auxílio decisivo, que viabilizou sua realização e permitiu que tivesse todo o êxito que apresentou.

Nosso agradecimento inicial a todos os que, de modo formal, institucionalmente apoiaram o WTA 2011:

- SBC – Sociedade Brasileira de Computação;
- SPC – Sociedade Peruana de Computação;
- IEEE – The Institute of Electrical and Electronics Engineering;
- EPUSP – Escola Politécnica da USP;
- PCS – Dep. de Engenharia de Computação e Sistemas Digitais.

Gostaríamos de agradecer ao IEEE, através do Profa. Dra. Mirela Sechi Moretti Annoni Notare, pela parceria com o nosso evento, e pela publicação dos melhores trabalhos na revista IEEE Latin American Transactions.

Agradecemos também, à Microsoft, que, através da pronta ação do Prof. Dr. Jorge Luís Risco Becerra, prestou ao evento uma importante colaboração mediante a cessão dos equipamentos utilizados na mostra de software do evento.

Registramos a seguir nosso reconhecimento público nominal àqueles que, direta ou indiretamente, contribuíram, com suas idéias e seu trabalho dedicado, para a organização e para a viabilização final desta edição do WTA (ordem alfabética):

- Prof. Amaury Antônio de Castro Júnior, a quem devemos: a gestão junto à SBC para apoio ao evento; a atualização do conteúdo do portal do LTA, especialmente das publicações; a criação do novo portal do LTA, com recursos mais automáticos que permitirão uma atualização mais rápida e segura e também pela participação no tutorial deste ano;
- Prof. Dr. João José Neto, pela presteza com que aceitou nosso convite para conduzir os trabalhos com a costumeira competência nos dois dias de duração do evento
- Aos mestrandos Luís Emílio Cavechiolli Dalla Valle, a quem devemos: a divulgação do evento; a coleta das inscrições para o evento; a organização dos horários das apresentações; a criação e manutenção do novo portal do WTA; a criação do pôster do WTA

Agradecemos finalmente a boa vontade da qual resultou o importante auxílio prestado ao WTA 2011 pelos seguintes profissionais (ordem alfabética):

- Leia Sicília, que secretariou o evento, pela permanente alegria e boa vontade que demonstrou, antes e durante seus trabalhos na recepção do WTA 2011;
- Nilton Araújo do Carmo, pelos trabalhos de apoio técnico prestados durante o evento, inclusive pela gravação de todas as imagens dos trabalhos do WTA 2011;

# TUTORIAL

O tutorial apresentado pelo prof. Dr. Carlos Eduardo Cugnasca apresentou um conjunto de contribuições teóricas e metodológicas para a área de automação agrícola e biodiversidade usando tecnologia adaptativa.

# ARTIGOS

# Oportunidades de Aplicação das Tecnologias Adaptativas para a Modelagem em Agricultura de Precisão

Fabiana S. Santana, Antonio M. Saraiva e Renata L. Stange

***Resumo* — A agricultura de precisão tem o objetivo de identificar áreas com diferentes níveis de produtividade, a fim de permitir a aplicação otimizada de insumos, o gerenciamento dos recursos agrícolas, a manutenção da qualidade da produção e também promover a sustentabilidade em campo. A agricultura de precisão depende da aplicação de tecnologias sofisticadas e do uso de sistemas de informação. As unidades de gerenciamento diferenciado permitem agrupar áreas de uma fazenda ou propriedade de forma contínua, auxiliando o seu gerenciamento. O resultado da aplicação do algoritmo *k-means* para identificar unidades de gerenciamento diferenciado é apresentado, discutindo os testes realizados e a sua versão adaptativa. Também são apresentados outros problemas de modelagem em agricultura de precisão onde a aplicação de tecnologias adaptativas parece ser um importante caminho a ser explorado.**

***Palavras-chave* — Tecnologias adaptativas (*adaptive technology*), algoritmos adaptativos (*adaptive algorithms*), agricultura de precisão (*precision agriculture*), modelagem em agricultura de precisão (*precision agriculture modelling*).**

## IV. Introdução

AGRICULTURA de precisão busca identificar áreas com diferentes níveis de produtividade no campo, a fim de oferecer um tratamento individual e diferenciado para cada uma delas [10] e, com isto, uniformizar os resultados obtidos [3]. Desta forma, é possível realizar a aplicação de insumos de maneira otimizada, a fim de uniformizar as condições do solo e procurar uniformizar a produtividade no campo.

Como potenciais resultados, destacam-se o melhor gerenciamento dos recursos agrícolas, o aumento da produtividade, o uso sustentável dos recursos disponíveis, a manutenção da qualidade da colheita, a segurança alimentar, a prosperidade agrícola e o desenvolvimento econômico [6].

Fabiana S. Santana é professora do Centro de Matemática, Computação e Cognição da Universidade Federal do ABC (fabiana.santana@gmail.com).
Antonio M. Saraiva é professor do Departamento de Engenharia de Computação e Sistemas Digitais da Universidade de São Paulo (saraiva@usp.br).
Renata L. Stange é aluna de mestrado do Departamento de Engenharia de Computação e Sistemas Digitais da Universidade de São Paulo (rlstange@usp.br).

A variabilidade sempre deve ser estudada considerando variáveis pré-definidas. A definição depende de diversos fatores, incluindo dados das culturas que estão sendo analisadas, aspectos químicos ou físicos do solo, dados climatológicos, aspectos específicos do ponto de vista agronômico e aspectos econômicos, ou de mercado.

A agricultura de precisão [4] [16] também deve oferecer suporte para a tomada de decisão aos produtores agrícolas, considerando aspectos gerenciais e técnicos. São muitos os aspectos a serem analisados em agricultura de precisão, bem como as variáveis envolvidas, o que resulta na necessidade de aplicação de tecnologias sofisticadas para tratá-las adequadamente [9] [19]. Por exemplo, pode ser necessário interpretar grandes quantidades de dados coletados em campo por sensores instalados em tratores e outros equipamentos, comparados a dados climáticos obtidos de provedores de dados na Internet, como o portal WorldClim [http://www.worldclim.org/].

A Tecnologia da Informação, TI, pode ser aplicada em agricultura de precisão para suportar as decisões associadas à produção agrícola [15]. A TI deve ser utilizada para coletar, armazenar, processar e analisar as variáveis e os dados envolvidos em seus processos. Para que a tomada de decisão seja ágil, recomenda-se o desenvolvimento de sistemas de informação capazes de tratar as grandes quantidades de dados e as variáveis necessárias [10].

Como os dados podem vir de fontes distintas e obedecer a diferentes padrões ou formatos, é necessário um esforço adicional para prover a interoperabilidade e a integração entre as fontes de dados e os sistemas relacionados à tomada de decisão.

Considerando que a tendência atual da computação é utilizar sistemas baseados em Internet [8] e orientados a serviços para viabilizar a troca de dados, a integração e a interoperabilidade entre máquinas, equipamentos e sistemas [17] e que as funcionalidades da agricultura de precisão devem evoluir com a tecnologia, uma solução que permita a fácil integração de novas funcionalidades é extremamente desejável [6].

Em [12], foi proposta uma infraestrutura para a solução de um problema em biodiversidade, denominado modelagem de nicho ecológico. A partir dos estudos para agricultura de precisão feitos por [14], [6] e [5] e, a partir da grande similaridade encontrada entre estes problemas [13], foi

possível estender a infraestrutura, com pequenos ajustes, para o domínio da agricultura de precisão. A Figura 1 apresenta a arquitetura da solução orientada a serviços para agricultura de precisão.

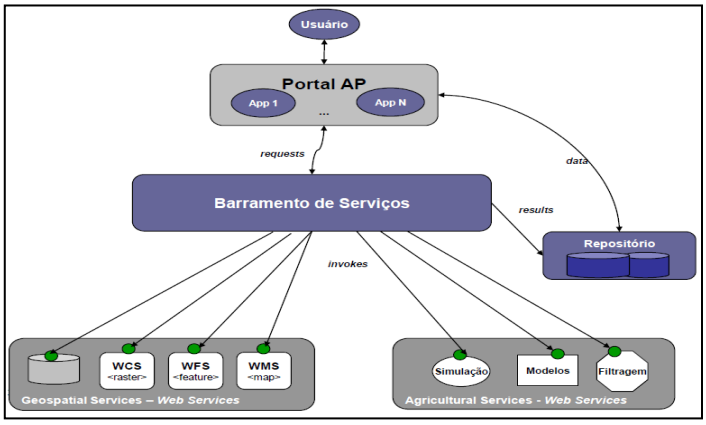



Figura 1 – Arquitetura da solução orientada a serviços para agricultura de precisão.

Uma vez definida a infraestrutura, o próximo passo foi desenvolver o portal para permitir a integração de serviços de interesse para agricultura de precisão. A Figura 2 apresenta o portal com a implementação de um serviço específico, baseado em um algoritmo de filtragem de dados de produtividade de máquina.

Agora, o grande desafio, além de desenvolver métodos e serviços gerenciais para auxiliar a administração da propriedade agrícola, é desenvolver métodos e algoritmos capazes de identificar a variabilidade no campo, identificar suas causas através da análise das variáveis de interesse, e oferecer suporte para a tomada de decisão em agricultura de precisão. Ao conjunto de métodos e algoritmos aplicados ao domínio da agricultura de precisão, denominamos modelagem em agricultura de precisão.

Este trabalho discute aplicações da tecnologia adaptativa para a modelagem em agricultura de precisão e apresenta uma solução preliminar para o problema de identificação de unidades de gerenciamento diferenciado, conhecidas como "zonas de manejo" na terminologia antiga, e que são de extremo interesse para o proprietário rural, quando da definição de culturas a serem produzidas em cada área de sua propriedade.

## V. Material e Métodos

### A. O Problema da Modelagem em Agricultura de precisão

A agricultura de precisão, desde o seu surgimento, sempre teve a sua natureza dirigida ao tratamento dos dados e ao uso intensivo de tecnologia em diversas frentes da engenharia, incluindo equipamentos eletrônicos, como tecnologia embarcada em máquinas e equipamentos, e sistemas de informação (ZHANG et al., 2002).

As etapas básicas relacionadas à construção de sistemas de informação para agricultura de precisão estão sendo endereçadas, na medida em que sensores foram ou estão sendo instalados em equipamentos e no campo para a obtenção dos dados necessários, com destaque para os dados de produtividade, de solo e climáticos [6]. Também estão sendo feitos esforços no sentido de se criar padrões e aplicá-los para a coleta dos dados existentes, bem como para o desenvolvimento de sistemas de informação mais preparados para tratá-los [12].

Porém, embora seja possível obter os dados e, com a tecnologia atual, integrá-los na maioria dos casos, ainda não existem métodos satisfatórios para construir modelos confiáveis para agricultura de precisão. Desta forma, os sistemas disponíveis apenas coletam os dados e deixam a critério dos agrônomos a definição, através de métodos não automatizados, como as análises serão feitas. Em geral, aplicam-se fórmulas definidas pelo agrônomo com base em


Figura 2 – Portal para agricultura de precisão, exibindo o resultado do serviço para filtragem de dados de produtividade de máquina.

alguns estudos ou critérios genéricos encontrados na literatura.

### *B. Unidades de Gerenciamento Diferenciado*

Uma fazenda ou unidade produtiva é normalmente dividida em regiões menores, denominadas talhões, a partir dos quais as unidades de gerenciamento diferenciado são identificadas.

Uma unidade de gerenciamento diferenciado é definida em função da similaridade das variáveis estudadas [3]. Por exemplo, se o objetivo for medir a produtividade e se a variável considerada for a quantidade de grãos colhidos, serão consideradas similares, a partir de intervalos pré-estabelecidos, as áreas onde a quantidade de grãos colhidos assumirem valores próximos, considerando o campo de forma contínua.

A partir da definição das unidades de gerenciamento diferenciado, outros problemas interessantes podem ser resolvidos. Entre eles, destaca-se o problema de definir e analisar as variáveis que têm influência sobre estas unidades, para interferir no processo. Por exemplo, no caso da definição de unidades de gerenciamento diferenciado em função da produtividade, como no exemplo anterior, fatores como a quantidade e o tipo de insumos aplicados ao solo, irrigação, quantidade de chuvas e outros fatores climáticos devem ser considerados, pois eles podem interferir diretamente na produtividade.

Estes problemas representam grandes oportunidades para a aplicação de algoritmos adaptativos.

### *C. Algoritmos de Agrupamento (Clustering)*

O agrupamento de dados, também chamado de *clustering*, é um problema fundamental em mineração de dados, e tem o objetivo de determinar um conjunto finito de categorias para descrever um conjunto de dados, de acordo com padrões de objetos similares [7]. Portanto, as técnicas de agrupamento procuram explorar a semelhança entre os objetos, agrupando-os de acordo com padrões semelhantes em grupos ou categorias.

Considerando o problema de identificação das unidades de gerenciamento diferenciado, destaca-se o algoritmo *k-means* [2] [18] dentre os demais algoritmos de agrupamento, pois este método, em sua versão não-adaptativa, já foi utilizado para resolver este tipo de problema e os resultados foram bastante positivos para regiões com predomínio de culturas de café [11].

Basicamente, o *k-means* é um algoritmo de aprendizagem não-supervisionada que divide um conjunto de dados X em k clusters disjuntos, de modo que os dados de um determinado grupo possuam algum grau de similaridade entre eles [18], que é basicamente o que se deseja identificar com as unidades de gerenciamento diferenciado.

## VI. RESULTADOS

Para encontrar as unidades de gerenciamento diferenciado e realizar esta etapa da modelagem em agricultura de precisão, foram executados os seguintes passos: 1) Definição do problema: definir a área a ser analisada e os tipos de análise a serem realizados; 2) Identificar as variáveis que podem interferir na identificação das unidades de gerenciamento diferenciado, sabendo que a escolha destas variáveis depende do objetivo da modelagem; 3) Obter os dados necessários para a modelagem; 4) Executar o algoritmo *k-means* para encontrar as unidades de gerenciamento diferenciado; 5) Projetar os resultados na forma de mapas georreferenciados e tabelas de aplicação, permitindo a análise dos dados obtidos.

A partir de 12016 amostras de dados de produtividade, obtidos a partir de sensores instalados em máquinas utilizadas para a colheita de grãos, e do uso da ferramenta *Weka 3: Data Mining Software in Java* [http://www.cs.waikato.ac.nz/ml/weka/], foi realizada a simulação para a obtenção das unidades de gerenciamento diferenciado através da aplicação do algoritmo *k-means*. Os resultados da simulação estão nas Figuras 3 e 4.

```
=== Run information ===

Scheme:     weka.clusterers.XMeans -I
1 -M 1000  -J 1000  -L 2 -H 4 -B 1.0  -C
0.5 -D "weka.core.EuclideanDistance  -
R first-last"  -S 10
Relation:    produtividade
Instances:   12016
Attributes:  4
             Moisture
             Wet_lg_yld
             Latitude
             Longitude
Test mode:   evaluate on training data

=== Model and evaluation on training
set ===
XMeans
======
Requested iterations         : 1
Iterations performed         : 1
Splits prepared              : 2
Splits performed             : 2
Cutoff factor                : 0.5
Percentage of splits accepted
by cutoff factor             : 0 %
------
Cutoff factor                : 0.5
------

Cluster centers              : 4 centers

Cluster 0
        0.0367156208277770364
9.1295060080106 67E-4  -
22.69948079706273  -
49.984846164218936

Cluster 1
        17.432303040483788
0.009003074080295605  -
22.6984198360490 34  -
49.985845654963946

Cluster 2
        13.389391056137013
0.006460133206470033  -
22.693911574215036  -
49.99064685823021

Cluster 3
        16.99206102117061
0.008456475716064754  -
22.69581996450814  -
49.9887359019303 5

Distortion: 2898.153613
BIC-Value : 12336.563107
Clustered Instances

0      883 (  7%)
1     5417 ( 45%)
2     1969 ( 16%)
3     3747 ( 31%)
```

Figura 3 – Resultado da simulação do algoritmo k-means através da ferramenta WEKA 3 – Data Mining Software in Java.

A Figura 3 apresenta os dados numéricos, que resultaram na identificação de quatro clusters, ou seja, quatro regiões de gerenciamento diferenciado. A Figura 4 apresenta os mesmos resultados de forma gráfica.

Uma versão adaptativa do *k-means* foi proposta em [1]. A diferença entre este algoritmo e o *k-means* tradicional está na inclusão de uma taxa de aprendizagem adaptativa. Porém, os testes preliminares com esta versão adaptativa não apresentaram diferenças significativas em relação aos testes realizados com o *k-means* tradicional, o que ocorreu provavelmente em função do uso de dados com pequenas variações nos valores da amostra utilizada para os testes.

Mesmo que ainda sejam bastante preliminares, o uso de algoritmos de agrupamento para identificação de unidades de gerenciamento diferenciado mostrou-se satisfatório, e novos testes devem ser desenvolvidos com outros conjuntos de dados, mais complexos e mais adequados, buscando avaliar o quanto se pode evoluir com esta técnica na solução deste problema. Da mesma forma, outros algoritmos dee agrupamento devem ser avaliados.


Figura 4 – Resultado gráfico da simulação do algoritmo k-means através da ferramenta WEKA 3 – Data Mining

## VII. CONCLUSÃO

Embora a aplicação de tecnologias sofisticadas e o uso de sistemas de informação para a agricultura de precisão já sejam aplicados para o gerenciamento da maioria das atividades em campo, o esforço da pesquisa na área nos últimos anos esteve concentrado no desenvolvimento de sistemas integráveis e na interoperabilidade, com o desenvolvimento de padrões para a obtenção e tratamento de dados no campo.

A evolução natural desta tecnologia é a modelagem em agricultura de precisão, que tem o objetivo de construir modelos para identificar as unidades de gerenciamento diferenciado e analisar as causas das diferenças dentro de um mesmo talhão, a fim de suportar a tomada de decisão com base nos dados obtidos.

Este trabalho apresenta uma pequena evolução na solução do problema de identificação de regiões de gerenciamento diferenciado, mas esta ainda é uma área de pesquisa incipiente, onde existem poucos métodos de análise disponíveis e uma série de aspectos do problema ainda não foram abordados.

Portanto, o grande desafio ainda permanece como um problema em aberto, que é utilizar a adaptatividade para obter modelos que permitam a tomada de decisão em agricultura de precisão de forma a interferir na produtividade da colheita. Desta forma, as oportunidades para a aplicação de tecnologias adaptativas para a modelagem em agricultura de precisão são muitas, e devem ser exploradas.

## AGRADECIMENTOS

Os autores agradecem à Financiadora de Estudos e Projetos do Ministério da Ciência e Tecnologia, FINEP/MCT, pelo suporte ao projeto PROSENSAP, sob o convênio n.º 01.08.0566.00, do qual faz parte a pesquisa realizada neste trabalho.

## REFERÊNCIAS

[1] CHINRUNGRUENG, C.; SEQUIN, C.H. Optimal adaptive k-means algorithm with dynamic adjustment of learning rate. 2002. IEEE Transactions on neural networks. Vol 6:1, pp 157-169.

[2] MACQUEEN, J.B. Some methods for classification and analysis of multivariate observations, in: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, volume 1: Statistics, 1967, pp. 281–297.

[3] MOLIN, J. P. Tendências da agricultura de precisão no Brasil. In: Congresso Brasileiro de Agricultura de Precisão – CONBAP, Piracicaba, 2008.

[4] MOLIN, J. P.; MILAN, M.; NESRALLAH, M. G. T.; CASTRO, C. N.; GIMENEZ, L. M. Utilização de dados georreferenciados na determinação de parâmetros de desempenho em colheita mecanizada. Engenharia Agrícola, Jaboticabal, v.26-3, p.759-767, set./dez. 2006.

[5] MURAKAMI, E.; SARAIVA, A. M.; RIBEIRO JR., L. C. M.; CUGNASCA, C. E.; HIRAKAWA, A. R., CORREA, P. L. P. An infraestructure for the development of distributed service-oriented information systems for precision agriculture. Computer Electronics Agriculture, 2007. Doi:10.1016/j.compag.2006.12.010.

[6] MURAKAMI, E. Uma infra-estrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão. Tese de Doutorado. Escola Politécnica da Universidade de São Paulo. São Paulo. Brasil. 2006.

[7] NALDI, M. C., CAMPELLO, R. J. G. B., HRUSCHKA, E. R., CARVALHO, A. C. P. L. F. (2010) "Efficiency issues of evolutionary k-means" Applied Soft Computing 11 (2011), p. 1938-1952.

[8] PASLEY, J. How BPEL and SOA are changing Web Services development. IEEE Internet Computing, v.9-3, p.60-67. 2005.

[9] PLANT, R.E. Site-specific management: the application of information technology to crop production. Computers and Electronics in Agriculture, v.30, p.9–29, 2001

[10] ROBERT, P.C. Precision Agriculture: research needs and status in the USA. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 2., 1999, Danemark. Proceedings. SCI/Sheffield Academic Press, 1999. p.19-33

[11] RODRIGUES JR., F. M. Geração de zonas de manejo para cafeicultura usando sensor spad e análise foliar. Dissertação de Mestrado, 2008, Universidade Federal de Viçosa, Minas Gerais.

[12] SANTANA, F.S. Uma infraestrutura orientada a serviços para a modelagem de nicho ecológico. 141p. Tese (Doutorado), Escola Politécnica, Universidade de São Paulo, São Paulo, 2009.

[13] SANTANA, F.S.; MURAKAMI, E.; SARAIVA, A.M.; CORREA, P.L.P. A Comparative Study between Precision Agriculture and Biodiversity Modelling Information Systems. Proceedings of the 6th Biennal Conference of the European Federation of IT in Agriculture, EFITA. Glasgow: C. Parker, S. Skerratt, C. Park, J. Shields, v.1, p.1-6, 2007.

[14] SARAIVA, A. M. Tecnologia da Informação na agricultura de precisão e biodiversidade: estudos e proposta de utilização de web services para desenvolvimento e integração de sistemas. Tese de livre docência. Escola Politécnica da Universidade de São Paulo. São Paulo, Brasil. 2003.

[15] SARAIVA, A.M. Um Modelo de Objetos para Sistemas Abertos de Informações de Campo para Agricultura de Precisão – MOSAICo. 1998. 235p. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 1998.

[16] SORENSEN, C.G., et al. Information sources and decision making on precision farming. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, 6., 2002, Wisconsin. Proceedings. Madison, ASA/CSSA/SSSA, 2002. p.1683-1695.

[17] STAL, M. Service-Oriented Architecture Principles and Technologies. Has been published as: Using Architectural Patterns and Blueprints for Service-Oriented Architecture. IEEE Software. 2006. V. 23-2, March/April.

[18] STEINLEY, D. K-means clustering: A half-century synthesis, British Journal of Mathematical and Statistical Psychology 59 (34) (May 2006) 1–34.

[19] ZHANG, N.; WANG, M.; WANG, N. Precision agriculture - a worldwide overview. Computers and Electronics in Agriculture, v.36, p. 113-132, 2002.

# Convolução Adaptativa

R. Camargo, Luís Raunheitte

***Resumo*. Este artigo tem como objetivo apresentar um método de utilização de técnicas convolucionais associadas a um núcleo subjacente para o desenvolvimento de um dispositivo adaptativo aplicado a uma base de dados objetivando funções de identificação de padrões. Essa aplicação representa a utilização de procedimentos adaptativos contendo funções de mineração de dados.**

*Palavras-chave*: **Adaptativo, núcleo subjacente, padrões, base de dados.**

## I. Introdução

Convolução é um processo aplicado normalmente em processamento de imagem como uma técnica para analisar e modificar quadros. A convolução aplicada a informação (dados), organizada de forma matricial, pode ser entendida como uma operação entre duas matrizes, geralmente bidimensionais, uma das quais é a matriz original e a outra é uma matriz de menor dimensão chamada de matriz de convolução ou núcleo de convolução. O núcleo de convolução representa uma função matemática qualquer e é aplicada sobre cada elemento da matriz original p(x,y) sua vizinhança imediata, resultando em uma nova matriz pc(x,y), que reflete a relação da matriz original com a função matemática dada pelo núcleo de convolução.

Pode-se fazer uma analogia da aplicação da técnica da convolução em banco de dados, com sua aplicação no processamento digital de sinais, onde sinal deve ser entendido como uma grandeza que varia no tempo e no espaço. Neste contexto o sinal é separado em componentes mais simples, de forma que cada componente seja processado separadamente e depois de processadas as componentes o resultado é reunido novamente.

## II. Tecnologia Adaptativa

Na área da Tecnologia Adaptativa existem inúmeros estudos de técnicas com aplicações em diversas áreas [3] [5], cujas contribuições têm estimulado o desenvolvimento de novos recursos computacionais.

Nos dispositivos adaptativos desenvolvidos, encontram-se formalismos conhecidos e tradicionais, tais como autômatos de pilha estruturados, *statecharts*, redes de Markov, gramáticas, árvores de decisão, tabelas de decisão, entre outros [5]. Isso mostra que há certa facilidade de uso das técnicas adaptativas, uma vez que [3] define um dispositivo adaptativo como um dispositivo formado por uma camada subjacente (núcleo do sistema) representada por um formalismo conhecido não-adaptativo e uma camada adaptativa, cujas funções agem sobre o núcleo, o que lhe confere a capacidade de automodificação, sem interferência externa, alterando suas estruturas topológicas, adaptando-se às necessidades requeridas de problemas específicos. Essa característica pode, desta maneira, conferir aos métodos adaptativos a classificação de sistemas inteligentes.

As ações adaptativas são implementadas na camada adaptativa e são responsáveis pelas alterações no conjunto de regras, gerando uma nova configuração do dispositivo [3]. De maneira geral, as ações adaptativas permitem que regras sejam consultadas, eliminadas ou incluídas no sistema.

## III. Núcleo de Convolução

Uma maneira de se entender a convolução é como uma operação que copia uma matriz a partir de cada localização de elemento para outra, considerando o valor de todos os elementos na área onde a cópia acontece, para produzir a alteração no valor original do elemento da matriz, [4].

Pode-se descrever a convolução como um processo de somas ponderadas. Cada elemento da matriz na vizinhança é multiplicado pelo seu similar no núcleo de convolução; a soma de todos os produtos resulta no novo valor do elemento central de interesse. Cada elemento do núcleo de convolução é um fator de ponderação (também chamado de coeficiente de convolução). O arranjo dos fatores de ponderação no núcleo, bem como o tamanho do núcleo, determina o tipo de transformação que será aplicada ao dado da representação matricial. Mudando um fator de ponderação no núcleo de convolução muda-se a magnitude e até o sinal de toda a soma afetando o valor atribuído ao elemento de interesse.

A convolução por soma ponderada apresenta um problema na sua aplicação nas fronteiras da matriz. Com o movimento do núcleo de convolução através da matriz, ao chegar a fronteira da mesma quando o elemento de interesse estiver na fronteira, uma parte dos coeficientes do núcleo não estarão sobre elementos da matriz. Uma maneira de contornar este problema é ignorar as fronteiras da matriz no cálculo, outra é duplicar os dados da fronteira, de forma a adicionar uma fronteira à matriz original, permitindo assim o cálculo sobre a fronteira original.

A operação de convolução substitui o valor do elemento da matriz pela soma de seu valor com a valor dos elementos das vizinhanças, tudo multiplicado por um fator chamado de "núcleo de convolução", supondo que se use uma vizinhança de 3X3 elementos, chamando de p(x,y) os pontos de matriz e

R. Camargo – Universidade Presbiteriana Mackenzie (correspondência: R. Oscar Freire, 235 – ap.31- São Paulo, SP, Brasil – cep: 01426-001; e-mail: rucamargo@usp.br).
Luís Tadeu Mendes Raunheitte – Universidade Presbiteriana Mackenzie; e-mail: raunheitte@mackenzie.br

os pontos do núcleo de N(x,y) onde x = 0, 1 ou 2, então o elemento central, p(1,1) será substituído pela soma dos pontos, vezes o valor do núcleo.

p(1,1) = p(0,0) * N(0,0) + p(1,0) * N(1,0) + p(2,0) * N(2,0) + p(0,1) * N(0,1) + p(1,1) * N(1,1) + p(2,1) * N(2,1) + p(2,2) * N(2,2).
(1)

ou

$$p(1,1) = \sum_{m,n}^{2} N(m,n) * p(m,n) \tag{2}$$

Esta é uma operação de correlação. Para convolução pode-se inverter a ordem dos valores do núcleo. A correlação é mais fácil de se entender e muitas convoluções de núcleo são simétricas, sendo equivalentes à correlação. Para convoluir uma área da matriz, deve-se repetir esta operação para cada posição de elemento na matriz de dados. Em cada ponto, deve-se multiplicar os valores do núcleo com os valores da matriz sobre ela, somar o resultado, e substituir o elemento do centro do núcleo com o valor. A equação então se torna:

$$p(x,y) = \sum_{m,n=0}^{2} N(m,n) * p(x+m, y+n) \tag{3}$$

Convoluir uma área de tamanho X por Y com um núcleo de tamanho m por n requer X∗Y∗m∗n multiplicações e somas. Então uma matriz de 256 por 256 com um núcleo de 3 por 3 requer 589.824 multiplicações e somas. A **Erro! Fonte de referência não encontrada.** 4 representa o processo de convolução.

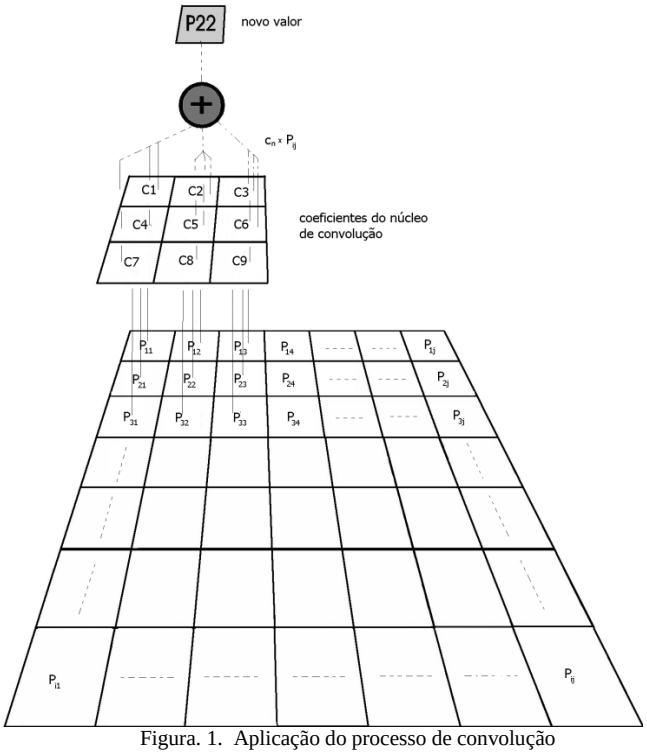


Figura. 1. Aplicação do processo de convolução

Quando se aplica a convolução para um problema de processamento de dados pensa-se na convolução como um filtro espacial. Em um filtro espacial, a convolução de núcleo é essencialmente para análise de uma pequena parte da matriz de dados que se quer amplificar ou detectar. Nesse contexto a escolha eficiente do núcleo pode detectar características na matriz de dados que permitam reconhecer padrões a partir da diferença de frequência dos resultados obtidos. Através da identificação de um determinado padrão o método proposto permitirá alterações no núcleo de convolução o que torna adaptativa a técnica, essa identificação será feita a partir de um índice obtido a partir da própria operação. O índice servirá como endereço que aponta para uma tabela de consulta onde será obtido um novo núcleo de convolução, núcleo de convolução adaptativo.

Considerando-se um sinal de entrada que passa através de um dado sistema resultando em um sinal de saída x[n], (como mostrado na Figura.2), este sinal pode ser decomposto nas componentes de entrada ou seja componentes harmônicos, são freqüências que compõe o sinal x0[n], x1[n], x2[n], etc. Cada componente de entrada é individualmente aplicada ao sistema, resultando em um conjunto de componentes de saída y0[n], y1[n], y2[n], etc. Estas componentes são então sintetizadas na forma do sinal de saída y[n]. Dessa forma o sinal de saída obtido é idêntico ao produzido pela passagem direta do sinal de entrada através do sistema. Os sinais de entrada e saída podem ser vistos como a soma de sinais mais simples.

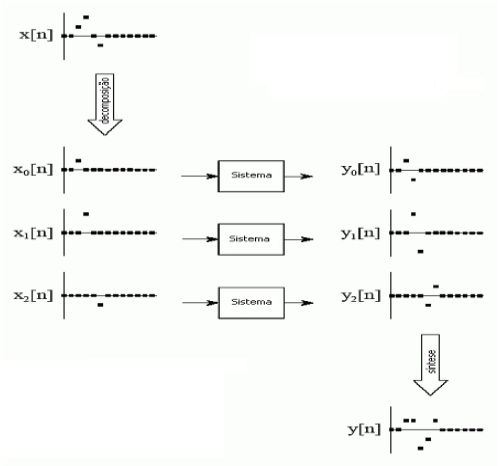



Figura.2. Decomposição do sinal
Fonte: Smith (1997)

Utilizando um sinal representado por suas funções de impulso aplicado a um dado sistema, este produz uma resposta que pode ser determinada pela soma ponderada das respostas individuais dos impulsos aplicados ao sistema.

$$\sum_k c_k \cdot \delta[n-k] \longrightarrow \boxed{\text{sistema}} \longrightarrow \sum_k c_k \cdot h[n-k] \quad (4)$$

Onde: ck é o número de impulsos aplicados ao sistema,
δ[n-k] impulsos de entrada deslocados no tempo e
h[n-k] resposta impulsiva do sistema para cada impulso de
entrada.

Assim a definição matemática de convolução é:

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k] \quad (5)$$

onde x[n] é o sinal de entrada, h[N] é a resposta impulsiva, e y[n] é a saída. O símbolo * indica a operação de convolução. Logo a convolução pode ser obtida pela soma dos resultados das operações de multiplicação dos termos x[k] pelos termos h[n] deslocados no tempo.

Neste artigo é apresentada a convolução aplicada à informação, organizada de forma matricial, e pode ser entendida como uma operação entre duas matrizes, geralmente bidimensionais, uma das quais é a matriz original e a outra é uma matriz de menor dimensão chamada de matriz de convolução ou núcleo de convolução.

O núcleo de convolução representa uma função matemática e é aplicada sobre cada elemento da matriz original p(x,y) na sua vizinhança imediata, resultando em uma nova matriz pc(x,y), que reflete a relação da matriz original com a função matemática dada pelo núcleo de convolução.

Pode-se considerar a convolução como a aplicação de uma máscara de resposta à matriz original de acordo com critérios bem definidos. Na convolução temos dois componentes:

- Um ou mais núcleos de convolução ci(x,y)
- A operação de convolução

Esse trabalho ainda está em fase de estudo e estão sendo testados vários filtros (matriz 3 x 3 e matriz 5 x 5). Os filtros correspondem a uma variedade de filtros direcionais simétricos onde as posições mais próximas do pixel central da matriz de convolução interferem no resultado com peso superior àqueles nas extremidades, ajustando as variações discretas do gradiente nas direções horizontal, vertical e diagonal [4].

A convolução sendo aplicada no seu modo mais simples tem como objetivo a identificação de pontos salientes que se destacam de seus vizinhos, não sendo necessariamente um valor alto do ponto de vista absoluto.

| Máscara de detecção de pontos | Imagem | Máscara de resposta |
|---|---|---|
| $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ | $\begin{bmatrix} 2 & 2 & 2 \\ 2 & 8 & 2 \\ 2 & 2 & 2 \end{bmatrix}$ | $\begin{bmatrix} -2 & -2 & -2 \\ -2 & 64 & -2 \\ -2 & -2 & -2 \end{bmatrix}$ |

(6)

## IV. CONCEITOS DE MINERAÇÃO DE DADOS

Abordando pesquisas que tem como objetivo básico a identificação de padrões encontram-se estudos sobre Mineração de Dados ou Data Mining que tiveram origem em análise estatística na década de 60, os quais evoluíram, nos anos 80, para novas técnicas de inteligência artificial, tais como lógica fuzzy, redes neurais, árvores de decisão [7].

Mineração de dados, segundo [8], consiste em um conjunto de técnicas utilizadas na exploração de conjuntos de dados, normalmente mantidos em tabelas, formando um banco de dados. A mineração de dados tem como objetivo o descobrimento de relacionamentos complexos envolvendo conceitos, tais como: padrões, regras, fatos em dados armazenados.

Conforme [9], o processo de descoberta de conhecimento e mineração de dados (KDD, *Knowledge Discovery and Data Mining*), pode ser tratado em quatro etapas:

1. Seleção de dados: etapa para determinar o agrupamento de dados e atributos de interesse;
2. Limpeza dos dados: consiste na remoção de ruídos, na transformação de alguns campos e na criação de campos combinados;
3. Mineração dos dados: aplicação de algoritmos específicos para extrair padrões de interesse;
4. Avaliação: etapa em que os padrões descobertos são disponibilizados para os usuários em forma inteligível, facilitando a visualização.

Deve-se ressaltar que o conceito de padrão, segundo [9], é uma unidade de informação ou atributo de um registro que se repete, ou então é uma sequência de informações/atributos

presentes em uma estrutura que se repete.

Existem cinco tipos de técnicas [8], usadas para a Mineração de Dados:

1. Associações: que identificam afinidades entre um conjunto de dados em um grupo de registros; por exemplo: 72% de todos os registros que contêm itens A, B e C, também contêm itens D e E; dessa maneira, regras associativas procuram estabelecer ligações entre um elemento e outro;
2. Padrões Sequênciais: que identificam sequências de registros que ocorrem em decorrência de outros. Por exemplo: na ocorrência de um evento A, 32% dos clientes com determinadas características realizarão o evento B, em um determinado espaço de tempo;
3. Classificação: que divide as classes predefinidas, permitindo que registros de uma classe permaneçam próximos. Exemplificando: poderia haver classes de registros quanto à frequência do comparecimento de clientes em uma agência bancária: infrequentes (nunca frequentam a agência), frequentes (comparecem de modo frequente) e ocasionais (ocasionalmente frequentam a agência);
4. Agrupamento: a partir da base de dados, descobre classes ocultas, enquanto que a classificação já inicia com classes predefinidas;
5. Previsão: que tem como objetivo o cálculo de previsão do valor futuro de uma variável, como por exemplo, prever uma determinada projeção de vendas, considerando registros devidamente classificados.

## V. RECONHECIMENTO DE PADRÕES

Como colocado inicialmente esta pesquisa tem como objetivo utilizar técnicas adaptativas associadas ao processo de convolução aplicado a banco de dados para identificação de padrões.

Parte inicial desta seção deve ser expandido e utilizado como introdução do trabalho.

Conforme [5], o formalismo adaptativo se mostra uma opção a ser utilizada na aprendizagem computacional, destacando trabalhos realizados no reconhecimento de imagens e linguagens. Tratando o reconhecimento de padrões, [5] ainda destaca o reconhecimento ótico de caracteres (OCR), baseado em classes de técnicas de aprendizagem tais como: árvores de decisão, redes neurais artificiais, sentenças em lógica de predicados, conjunto de regras "se-então", autômatos, redes *bayesianas* e memorização (*instance-based learning*). Com relação às regras "se-então", pode-se entender como uma opção sem o domínio de conceitos complexos, permitindo uma maior transparência do modelo.

Uma proposta de trabalho com a utilização do conceito de adaptatividade no reconhecimento de padrões em uma base de dados é composta pelas seguintes etapas:

a) a utilização da técnica de associação usada na Mineração de dados;

definição de uma camada de convolução que aplicada gera o fator de ponderação, que determina o tipo de transformação aplicada na representação matricial.

Esta última camada será denominada Convolução adaptativa, pois apresentará seus fatores de ponderação de forma variável, havendo a possibilidade de inclusão ou exclusão de novos fatores de ponderação, alterando os coeficientes do núcleo de convolução face aos resultados obtidos.

## VI. CONVOLUÇÃO ADAPTATIVA

Pesquisas já realizada comprovam que a adaptatividade refere-se a um conceito considerando a "experiência anterior" adquirida por um dispositivo adaptativo, baseado em um histórico de operações onde um sistema pode tomar a decisão de modificar seu comportamento [6].

O formalismo convolução adaptativa, aplicado na identificação de padrões, é definido a seguir:

Elementos do formalismo ND = (C, NR, S, $c_0$, A, NA):

- ND – dispositivo descrito pelo conjunto de regras NR;
- C – Conjunto de configuração do Núcleo por tipo de aplicação (tipo de BD);
- NR – Núcleo de Convolução Inicial (Ni)
- Li – limiar de comparação para testar os resultados da primeira convolução entre a Matriz do Banco de Dados por Ni. Após a execução do procedimento Estratégico que converte a Base de dados - BD em Matriz do Banco de Dados;
- S – Tipo de Padrão a ser identificado com base no Limiar de Comparação - Li.
- $c_0 \in C$, como configuração inicial;
- A $\subseteq$ C
- NA, com $\varepsilon \in$ NA é um conjunto de todos símbolos possíveis de saída de AD.

O conjunto de regras são os índices (coeficientes) da expressão da convolução:

$$\sum_{k=-\infty}^{\infty} x[k].h[n-k]$$

A estrutura da convolução adaptativa é construída conforme os procedimentos especificados a seguir na Figura 3:

A partir de um banco de dados organizado, tendo seus eventos associados às suas dimensões, é aplicada a mineração de dados associando fatos, considerando o limiar inicial, fornecido pelo decisor. O limiar deve ser entendido como suporte mínimo descrito anteriormente.

O passo seguinte é montar uma matriz, baseada na classificação dos eventos.

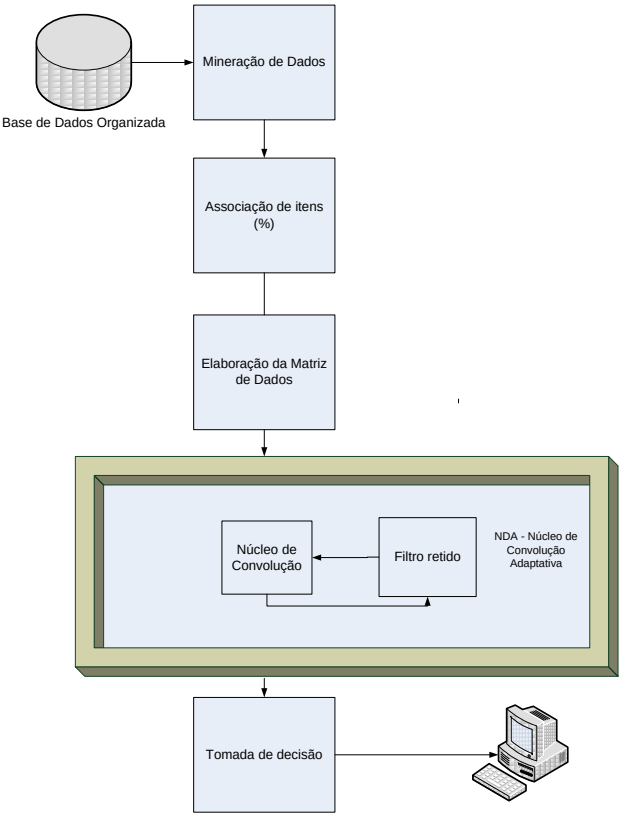



Figura.3. Modelo do Estudo

Sobre a matriz de dados é aplicado o núcleo inicial, definido pela transformada de Laplace, conforme [1].

Baseado no cálculo da convolução o resultado é comparado com o limiar inicial. Se resultado da convolução for menor que o limiar inicial, é alterado o valor central do núcleo. Caso o resultado da convolução seja maior ou igual, é verificada a existência de filtros já utilizados e válidos, no caso de não existir é criado um novo núcleo.

O resultado é apresentado de forma gráfica identificando regiões e padrões a serem interpretados por um decisor (especialista). Sendo o núcleo analisado e validado, é armazenado para uso futuro.

O estudo exploratório, para implementação do modelo de convolução adaptativa sobre a matriz de dados, estão em andamento, bem como análise do comportamento do algoritmo.

Na Figura. 4 é apresentada uma etapa dos estudos exploratório.

Como modelo hipotético de aplicação da técnica adaptativa proposta foi utilizada uma Matriz de Transição 100 x 100 com números aleatórios, simulando a obtenção dos dados originados em um banco de dados, contendo informações sobre o consumo em um supermercado.

Essa Matriz foi criada de forma idealizada contendo grandes grupos de consumo.

A matriz foi convertida para leitura no Excel com arquivo denominado PNA.xls, gerada a partir do banco de dados. Os valores dos elementos da matriz foram tratados como elementos de imagem (pixels). Dessa forma tornou-se possível visualizar o referido banco de dados como uma imagem, a Figura. 4 apresenta essa visualização.

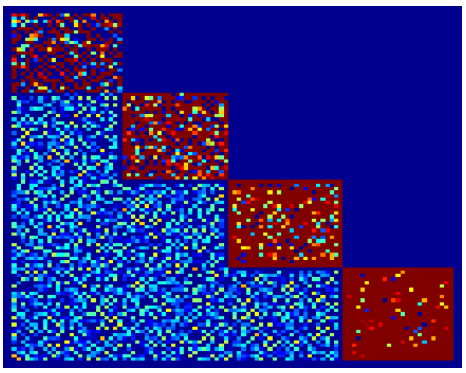


Figura. 4 - Planilha Convertida – Matriz PNA – Bidimensional

A seguir foi criado o Núcleo Inicial – Ni, com base na definição pela transformada de Laplace, conforme [1]:

Ni =

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -1 | 0 | 0 |
| 0 | -1 | 5 | -1 | 0 |
| 0 | 0 | -1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Com a aplicação do Núcleo Inicial na Matriz de dados foi obtido o resultado em imagem conforme Figura 5 Bidimensional e Figura 6 Tridimensional.

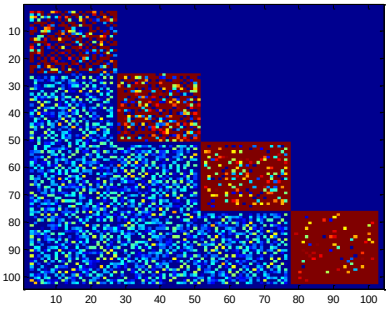



Figura. 5 – Resultado Bidimensional da Convolução

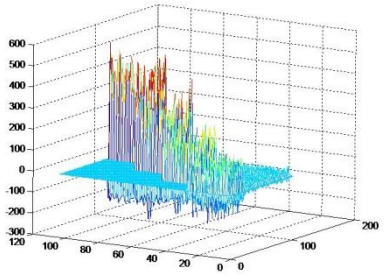



Figura. 6 – Resultado Tridimensional da Convolução

Realizando alterações no Núcleo de Convolução, observa-se alteração do comportamento dos padrões identificados:

Para o Núcleo N1 onde

$Ni_{(3,3)} = 9$

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 9 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Tem-se então o seguinte resultado conforme Figura 7, bidimensional e Figura 8 tridimensional :

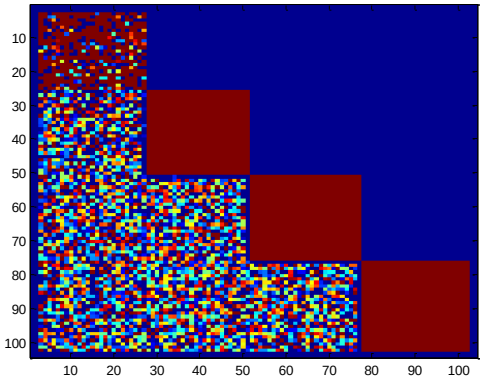



Figura 7 - Resultado bidimensional Núcleo N1

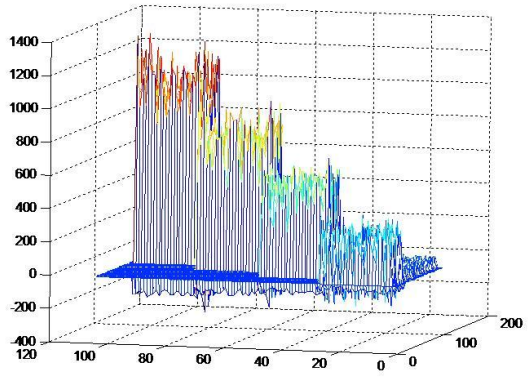



Figura 8 – Resultado Tridimensional Núcleo N1

Pelos experimentos realizados podem-se notar as representações gráficas da convolução, aplicada à Matriz de transição. Comparando os resultados obtidos com a aplicação dos dois diferentes núcleos (Ni e N1) de convolução pode-se perceber que N1, já permite a identificação de regiões no banco de dados com padrões específicos (retângulos marrons na figura 7), comprovando a efetividade da ferramenta proposta.

A continuidade dos experimentos permitirá a comparação de novos resultados da aplicação da convolução com níveis pré-estabelecidos que indicarão a necessidade da utilização de novos núcleos, tal que novos coeficientes sejam gerados demonstrando a adaptatividade do algoritmo.

## VII. Conclusão

Neste artigo são apresentados mecanismos adaptativos de identificação de padrões, objetivando a criação de um aplicativo baseado nos conceitos da convolução associado a conceitos de Adaptatividade, essa é uma proposta para criar um dispositivo adaptativo para aplicação em caráter genérico em processos de tomada de decisão [2].

## Referências Bibliográficas

[1] Gonzalez, Rafael C; Words, Richard E. Processamento Digital de Imagens – Ed. Edgard Blücher Ltda., 2000.

[2] O'BRIEN, J. A. *Sistemas de Informação e as Decisões Gerenciais na Era da Internet*. 2ed. Saraiva, São Paulo, 2004.

[3] 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, Springer-Verlag, 2001, pp. 234-250.

[4] HU, Osvaldo; RAUNHEITTE, Luís, *Processamento e Compressão Digital de Imagens.* Editora Mackenzie, São Paulo, 2004.

[5] PISTORI, H. *Tecnología Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações*. Tese de Doutorado – Escola Politécnica da USP, 2003.

*[6]* NETO, J.J., *Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa.* IEEE Latin America Transactions, Vol. 5, No. 7, Nov. 2007.

[7] Kimbal, R., *The Data Warehouse Toolkit*. 2 ª Ed.Wiley Computer Publishing, USA, 2002.

[8] Moxon, B (1998). *Defining Data Mining-DBMS, Data Warehouse Supplement*, Aug.1996. HTTP//:dbmsmag.com/9608d53.html (30.Out.2008).

[9] RAMAKRISHNAN, R. *Sistema de Gerenciamento de Banco de Dados.* 3ª Ed. McGraw-Hill, São Paulo, 2008.

**Rubens de Camargo** é bacharel em Administração de Empresas, com especialização em Análise de Sistemas pela Faculdade Associadas de São Paulo e mestre em Administração de Empresas pela Universidade Presbiteriana Mackenzie. Atualmente é professor de cursos de graduação na Faculdade de Computação e Informática da Universidade Presbiteriana Mackenzie. É Doutorando do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

**Luís Tadeu Mendes Raunheitte** é Engenheiro Eletrônico pela Universidade Presbiteriana Mackenzie, mestre e doutor em Engenharia Elétrica pela mesma Universidade. Atualmente é professor de cursos de graduação na Faculdade de Computação e Informática da Universidade Presbiteriana Mackenzie.

# Considerações sobre o desenvolvimento de um filtro adaptativo para validação de dados em redes de sensores (18 Outubro 2010)

Osvaldo Gogliano Sobrinho, Renata Maria Marè,
Carlos Eduardo Cugnasca, Brenda Chaves Coelho Leite

***Resumo*— Amplia-se a cada dia o uso de redes de sensores, dada a redução de custos decorrente de avanços na área de microeletrônica. Erros de leitura decorrentes de diversos motivos como, por exemplo, indução de ruído elétrico em redes cabeadas, podem levar a interpretações incorretas sobre os fenômenos observados. O uso de critérios estatísticos para validação de conjuntos de dados pode não ser adequada. Dado o caráter sazonal da variabilidade dos dados, o desenvolvimento de um filtro sensível a esta característica é altamente desejável. A utilização de tecnologias adaptativas pode levar a um modelo computacional capaz de alterar seu comportamento de maneira dinâmica, tornando possível a criação de um filtro de validação de dados com estas características.**

**O trabalho apresenta considerações sobre o desenvolvimento de um filtro de validação de dados obtidos em uma rede Modbus de sensores, instalada em duas salas de aula climatizadas no Edifício de Engenharia Civil da Escola Politécnica da USP. Por tratar-se de pesquisa recém iniciada, o presente trabalho trata apenas de questões genericas levantadas até o momento para o desenvolvimento da solução.**

***Palavras chave*— Tecnologia adaptativa, árvores de decisão, autômatos adaptativos, conforto ambiental, monitoramento, internet, redes de sensores.**

## I. Introdução

Como parte do projeto de pesquisa intitulado "Sistema Seguro para Monitoramento Remoto da Qualidade do Ambiente Interno", patrocinado pela Fundação de Amparo à Pesquisa do Estado de São Paulo, Fapesp, dentro do programa PIPE, instalou-se uma rede de sensores padrão Modbus em duas salas de aula climatizadas do Edifício de Engenharia Civil, da Escola Politécnica da USP. Um dos objetivos da pesquisa é o monitoramento remoto de variáveis ligadas à qualidade do ar interno e seu envio a um servidor *web* remoto, onde são armazenadas e apresentadas aos usuários do sistema, por meio de um portal Internet. Assim, incorporaram-se à rede sensores analógicos (saída 0 a 5V) para medição da temperatura do ar, umidade relativa do ar e teor de $CO_2$.

As medições, iniciadas em setembro de 2009, são efetuadas a cada 30 segundos. No entanto, devido a problemas de latência e indisponibilidade da rede computacional do departamento, utilizada para comunicação com o servidor *web* remoto, com alguma freqüência, o intervalo entre leituras podem chegar a intervalos de 1 a 5 minutos. Em caso de queda da rede, estes intervalos podem chegar à ordem de algumas horas.

Tratando-se de uma rede cabeada, constata-se a indução de ruído elétrico em algumas medições efetuadas, levando à distorções em alguns dos valores obtidos. Faz-se necessária a filtragem dos dados de maneira a descartar estes valores. O uso de filtros estatísticos pode levar à tomada de decisões incorretas.

Discute-se neste trabalho a possibilidade de uso de um filtro baseada em uma árvore de decisão construída com o emprego de um autômato finito adaptativo.

## II. Caracterização do problema

Exibe-se na figura 1 o gráfico da variação da temperatura obtida por 4 sensores e as distorções causada por dados não filtrados, que se apresentam como "picos" instantâneos de valor, que não ocorrem no ambiente monitorado.

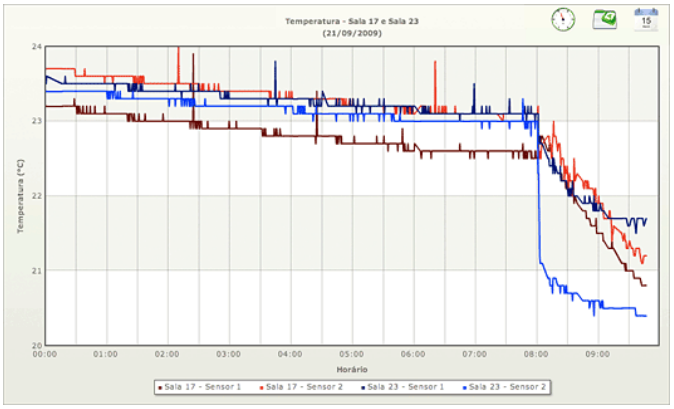



Fig. 1. Gráfico da variação de temperatura sem o uso de filtro.

A aplicação de critérios estatísticos como os de Chauvenet [1], ou Peirce [2] pode eliminar parte destes valores.

No entanto, sua utilização é conceitualmente incorreta, pois tais métodos foram desenvolvidos para a identificação de erros de leitura em conjuntos de medições distintas de uma mesma grandeza. Já no sistema de monitoramento em questão, os valores obtidos correspondem a medições efetuadas em instantes distintos, de parâmetros naturalmente variáveis no tempo.

Como é baixa probabilidade de variações significativas destes parâmetros em curtos intervalos de tempo, na prática, sua utilização pode levar a resultados aceitáveis. Observa-se na figura 2 o mesmo gráfico apos a aplicação do critério de Chauvenet, que levou à eliminação dos picos observados.

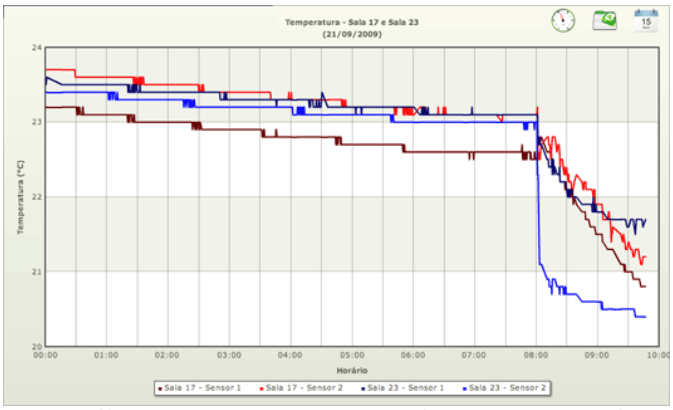



Fig. 2. Gráfico de variação da temperatura após a aplicação do critério de Chauvenet.

Outro problema que decorre do uso de filtros estatísticos é sua incapacidade de refletir as peculiaridades do ambiente monitorado. Por exemplo, nas duas salas do projeto, o sistema de climatização é ligado por volta de 08:00 de 2ª a 6ª feira, o que causa uma variação relativamente rápida da temperatura nestes momentos. Aos finais de semana, as salas não são ocupadas e o sistema de climatização permanece desligado, o que torna menor a variabilidade das condições monitoradas. Em ambas as salas, as aulas ocorrem normalmente no período da tarde, com geração de carga térmica devido à presença dos alunos e das 24 estações de trabalho que são ligadas. O efeito da presença dos alunos é notadamente marcante no teor de $CO_2$ medido.

Idealmente, o filtro utilizado deveria levar em conta a maior ou menor probabilidade de ocorrência destas variações em função do dia da semana e horário.

Para tentar sanar estas deficiências, imaginou-se a utilização de um filtro adaptativo que pudesse alterar dinamicamente seu critério de aceitação das medições segundo condições observadas.

## III. Solução proposta

Por estar em operação desde setembro de 2009, existe uma massa de dados já coletados considerável – próximo ao final de setembro de 2010, cerca de 700.000 medições de cada parâmetro estavam registradas no banco de dados do sistema.

Imaginou-se, a princípio, a utilização de uma árvore de decisão a ser treinada com a massa de dados previamente coletada. No entanto, um requisito essencial do projeto precisa ser respeitado:

- *Possibilidade de utilização dos dados armazenados como prova em eventuais disputas jurídicas envolvendo a qualidade do ar interno dos ambientes monitorados*

Para que isso ocorra, nenhum dado pode ser descartado das medições efetuadas, pois o critério de remoção adotado certamente causará discussões sobre sua validade no âmbito jurídico. Mesmo assim, é importante que se possa visualizar os dados obtidos, por exemplo na forma de gráficos como o da figura 2, aplicando-se o filtro desenvolvido.

A conseqüência deste requisito é que o filtro não poderá ser aplicado durante a coleta de dados e sim, opcionalmente, durante a exibição dos mesmos, o que torna critico seu desempenho: note-se que para a geração do gráfico de variação de uma única grandeza (coletada a cada 30s) ao longo de um dia inteiro, cerca de 2.800 pontos por grandeza deverão ser plotados, implicando no mesmo numero de utilizações do filtro.

Com esta condição, optou-se pela utilização de uma árvore de decisão, dado seu curto tempo de resposta.

Embora a utilização do filtro deve ser feita durante a consulta aos dados, a construção da árvore pode ser feita durante a leitura dos dados já que o intervalo não menor que 30s entre medições é mais que suficiente a execução da rotina de consulta/inserção de novos ramos na árvore de decisão.

## IV. Árvores de decisão

Segundo a definição de Pistori [4], "árvores de decisão (ADs) são mecanismos para a representação de funções discretas sobre múltiplas variáveis (contínuas ou discretas) com características hierárquicas que facilitam a inspeção e a utilização por seres humanos". De fato, considere-se a representação gráfica de uma árvore de decisão apresentada em  exibida na figura 3.

QuickTime™ and a
decompressor
are needed to see this picture.

Fig. 3. Exemplo de árvore de decisão. Extraído de Pistori [].

A árvore apresentada representa a função

$$f : N \text{ X } L \rightarrow C$$

onde:

$$N = \{0,1,2\}$$
$$L = \{a,b\} \; e$$
$$C = \{sim, não\}$$

com os valores de C estão definidos na tabela I.

Os vértices internos da árvore representam testes efetuados sobre variáveis que conduzem, a cada possível valor a uma nova aresta representando o resultado obtido. Parte-se da raiz da árvore submetendo-se exemplos de conjuntos de atributos, chegando-se a resultados da classificação representados nas folhas da árvore.

No desenvolvimento do filtro proposto, pretende-se trabalhar com conjuntos de 4 variáveis conforme descrição no item seguinte deste trabalho.

TABELA I
DEFINIÇÃO DA FUNÇÃO $f: NXL \rightarrow C$.

| N | L | $f$ |
|---|---|---|
| 0 | a | sim |
| 0 | b | sim |
| 1 | a | não |
| 1 | b | não |
| 2 | a | sim |
| 2 | b | não |

## V. DISCRETIZAÇÃO DOS DADOS

Para os testes iniciais decidiu-se trabalhar com as medições efetuadas com o sensor de temperatura nº 1, instalado na Sala 17.

A seguir, definiram-se critérios para discretização dos dados. Decidiu-se pela adoção de vetores de atributos contendo quatro variáveis: dia da semana da medição, hora cheia da medição, intervalo de tempo decorrido desde a última medição e variação percentual do valor lido com relação ao último resultado. Para cada atributo consideraram-se os aspectos descritos a seguir.

### A. *Dia da semana*

Para cada dia da semana, atribuiu-se um *token* $t_d \in T_d$

$$T_d = \{d0, d1, d2, d3, d4, d5, d6\}$$

para cada dia da semana, onde 'd0' corresponde à 2ª feira, 'd1' à 3ª feira e assim por diante.

### B. *Hora da medição*

Decidiu-se dividir as medições em faixas baseadas em sua hora cheia, atribuindo-se a cada uma um *token* $t_h \in T_h$

$$T_h = \{h0, h1, h2, h3, ..., h22, h23\}$$

onde 'h0' corresponde ao intervalo entre 00:00 e 01:00 e 'h23' corresponde ao intervalo entre 23:00 e 24:00.

### C. *Intervalo de tempo decorrido desde a última medição*

Inspecionando-se a massa de dados existentes constatou-se que 99,68% das medições ocorrem em intervalos iguais ou inferiores a cinco minutos. Desta forma, decidiu-se considerar apenas intervalos nesta faixa, atribuindo-se a cada medição um *token* $t_t \in T_t$

$$T_t = \{t1, t2, t3, t4, t5, t6\}$$

onde 't1' corresponde a intervalos iguais ou inferiores a um minuto, 't2' corresponde a intervalos iguais ou inferiores a 2 minutos e assim sucessivamente. Associou-se o *token* 't6' para quaisquer intervalos superiores a cinco minutos, situação na qual se optou pela aceitação do valor medido, ou seja, admite-se que após um intervalo de tempo superior a cinco minutos qualquer variação da leitura é admissível.

### D. *Valor da medição*

Decidiu-se trabalhar com a variação percentual entre cada medição efetuada e a medição prévia. Desta maneira, o critério pode ser aplicado à medição de qualquer grandeza, o que não ocorreria se os valores fossem expressos em suas grandezas reais. Optou-se por trabalhar com intervalos percentuais de 1% já que os sensores empregados possuem uma resolução de 0,1ºC que corresponde a cerca de 0,5% na faixa usual de temperaturas observadas (20 a 30ºC).

A próxima decisão foi a escolha do numero de intervalos a considerar. Novamente, inspecionando-se a massa de dados existente, constatou-se que variações fora do intervalo -5% a +5% correspondem a apenas 0,01% da medições. Dado o equilíbrio entre variações positivas e negativas, considerou-se apenas o valor absoluta dos percentuais de variação. Assim, definiram-se 6 *tokens* $t_v \in T_v$ para os valores das medições

$$T_v = \{v0, v1, v2, v3, v4, v5, v6\}$$

onde 'v0' corresponde a variações de 0% (sem variação entre a medição atual e a anterior), 'v1' corresponde a uma variação de ±1% e assim por diante. Associou-se o *token 'v6'* a variações fora da faixa -5% a +5%. Outra decisão tomada foi o de aceitação automática de valores correspondentes a $t_v = 0$, pois é razoável a suposição que variações nulas dos parâmetros medidos sejam as mais freqüentes, dados os curtos intervalos de tempo entre medições. De fato, observando-se a massa de dados disponíveis, em 96,4% das medições efetuadas obtêm-se variações nulas.

## VI. GERAÇÃO DA ÁRVORE DE DECISÃO E TREINAMENTO

A construção da árvore de decisão poderá ser feita durante a utilização do filtro. A submissão de cada vetor de atributos obtidos a um autômato adaptativo pode reconhecer uma seqüência já obtida ou, caso nova, acrescentá-la à árvore.

Para seu treinamento, árvores de decisão são submetidos a conjuntos de dados para os quais o valor da classe é conhecido o que permite seu registro. Através de algoritmos de indução da árvore de decisão é possivel a classificação de casos não presentes no conjunto de treinamento.

No caso do sistema em questão, a classificação de medições como válidas ou inválidas não pode ser feita por outro critério que não seja a probabilidade estatística de sua ocorrência. Assim, associaram-se a cada folha da árvore de decisão dois *tokens* representando o numero total de ocorrências do conjunto de dados e o numero de total de ocorrências do conjunto de dados obtido no vértice anterior. Esses *tokens* são facilmente atualizados a cada passo da rotina de geração da árvore.

Ao final, a probabilidade de ocorrência do conjunto

examinado pode ser facilmente calculada e a amostra classificada, adotando-se um valor limite para esta probabilidade.

Para a construção da árvore de decisão utilizou-se uma versão modificada do autômato classificador apresentado por Pistori e José Neto [3].

O alfabeto aceito pelo autômato modificado é definido por

$$\sum = \{t_v t_t t_d t_h\}$$

A linguagem aceita pelo autômato é:

$$t_v t_t t_d t_h$$

No autômato proposto em [3] a presença de um símbolo especial 'S' ao final da cadeia servia como seu validador, permitindo sua incorporação condicional à árvore. Em nosso caso, qualquer seqüência é reconhecida (caso previamente incorporada) ou acrescentada à árvore. Em ambos os casos os *tokens* associados aos números de ocorrência são atualizados.

## VII. Resultados e pontos a serem investigados

Por apresentar resultados iniciais de uma trabalho em estágio inicial de desenvolvimento, não existem resultados a serem divulgados. Procura-se focar as primeiras averiguações em alguns pontos específicos. Dentre eles, podemos destacar:

### *A. Critérios de discretização*

Verificar os efeitos do aumento e da diminuição dos níveis de discretização dos dados medidos, tentando balancear performance do filtro e o nível de discretização adotado. Uma menor granulação dos intervalos levará a resultados mais precisos, à custa de uma performance inferior e vice-versa.

### *B. Verificação da relevância de variáveis*

A adoção de tokens relacionados ao dia da semana e hora de medição como atributos parece, a priori, uma idéia correta. Porém, como a diminuição do número de variáveis nos vetores de atributos leva a uma melhor performance do modelo, os efeitos de sua possível não utilização deverão ser avaliados.

### *C. Aprimoramento do modelo*

O modelo adotado leva a resultados que podem ser incorretos. Para citar dois problemas já identificados: a primeira medição após um intervalo longo sempre é considerada correta; a medição seguinte a um ponto descartado tem probabilidade razoável de também ser indevidamente descartada, pois poderá apresentar variação semelhante (de sinal contrário) à da medição anterior.

### *D. Refinamento de limites para inferência estatística de aceitação dos dados*

O limite adequado a ser utilizado para o descarte de medições deverá ser buscado.

### *Referências*


[1] BOL'SHEV, L. N.; UBAIDULLAEVA, M. Chauvenet's Test in the Classical Theory of Errors. **Theory Probab. Appl**, v. 19, n. 4, p. 683-692, 1975.

[2] PEIRCE, B. Criterion for the rejection of doubtful observations. **The Astronomical Journal**, v. 2, n. 21, p. 161-163, 1852.

[3] PISTORI, H.; JOSÉ NETO, J. Adaptree - Proposta de um Algorítmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas. In: Anais Conferência Latino Americana de Informática-CLEI, 2002, Montevideo. **Proceedings…** Montevideo: 2002.

[4] PISTORI, H. Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações. 2003. 174 p. Tese de Doutorado - São Paulo.



**Osvaldo Gogliano Sobrinho** possui mestrado em Engenharia Elétrica, modalidade Sistemas Digitais pela Escola Politécnica da USP. Graduado em Engenharia Civil pela Escola Politécnica da USP (1976). Tem experiência na área de Ciência da Computação, com ênfase em Tecnologia Web e Banco de Dados. Atualmente é aluno de doutorado na Escola Politécnica da USP, na área de Sistemas Digitais e Diretor técnico da empresa Abili Assessoria Técnica Comercial e Tecnologia da Informação Ltda.

**Renata Maria Marè** possui mestrado em Engenharia de Construção Civil e Urbana pela Escola Politécnica da Universidade de São Paulo (2010), com ênfase em qualidade do ar de interiores em ambientes com sistemas de climatização com distribuição de ar pelo piso. Graduação em Engenharia Civil pela Escola de Engenharia Mauá (1985). É diretora comercial da Abili Assessoria Técnica Comercial e Tecnologia da Informação Ltda. desde 1996. Tem experiência nas áreas de Engenharia Civil, Design de Interiores e Paisagismo.

**Carlos Eduardo Cugnasca** é graduado em Engenharia de Eletricidade (1980), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Elétrica (1993). É livre-docente (2002) pela Escola Politécnica da Universidade de São Paulo (EPUSP). Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo, e pesquisador do LAA - Laboratório de Automação Agrícola do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Supervisão e Controle de Processos e Instrumentação, aplicadas a processos agrícolas e Agricultura de Precisão, atuando principalmente nos seguintes temas: intrumentação inteligente, sistemas embarcados em máquinas agrícolas, monitoração e controle de ambientes protegidos, redes de controle baseados nos padrões CAN, ISO11783 e LonWorks, redes de sensores sem fio e computação pervasiva. É editor da Revista Brasileira de Agroinformática (RBIAgro).

**Brenda Chaves Coelho Leite** possui graduação em Engenharia Civil pela Universidade Federal de Minas Gerais (1979), mestrado em Arquitetura e Urbanismo pela Universidade de São Paulo (1997) e doutorado em Engenharia Mecânica pela Universidade de São Paulo (2003). Atualmente é professor doutor da Universidade de São Paulo. Tem experiência em Conforto Térmico e Qualidade do ar nas edificações, Sistemas de Ar-Condicionado, Tecnologias de distribuição de ar pelo piso e painéis radiantes, Eficiência energética das edificações, Simulação computacional de edifícios, Dinâmica dos Fluidos Computacional, Automação e controle aplicados a sistemas de climatização, Eficiência Energética, avaliação pós-ocupação.

# Um Sistema para Ensino da Tecnologia Adaptativa

M. R. F. Santibanez, J. J. Neto

***Abstract*— This paper presents a software system that supports the teaching of adaptive technology of adaptive devices based on rules, to facilitate their learning through a friendly interface. The system was developed in layers: deterministic devices, devices, non-deterministic and adaptive devices, enabling the creation, training and validation to test different scenarios.**

***Keywords*— Adaptive Technology, Adaptive Rule-Driven Devices, Deterministic Devices, Non-deterministic Devices, Adaptive Devices, Friendly Interface, Learning Process.**

## I. INTRODUÇÃO

OS avanços tecnológicos têm auxiliado na popularização dos computadores em várias áreas do conhecimento humano. Na área educacional o computador possui um grande potencial para ser uma ferramenta altamente eficaz, no apoio ao processo de ensino-aprendizagem, isto permite que o aprendiz se transforme em participante ativo do processo de aprendizagem e construa seu próprio conhecimento.

Entendemos por tecnologia ao uso do conhecimento científico na resolução de problemas práticos, sendo a adaptatividade, a propriedade que um modelo tem de alterar seu próprio comportamento, de forma espontânea, sem auxílio externo, logo, a tecnologia adaptativa refere-se às técnicas, métodos e disciplinas que estudam as aplicações práticas da adaptatividade [1].

Dessa forma, a tecnologia adaptativa compreende qualquer modelo de representação formal que tenha capacidade de mudar dinamicamente seu próprio comportamento, em resposta direta a um estímulo de entrada, sem qualquer intervenção externa. Num sistema definido por um conjunto de regras, a incorporação de algum mecanismo, através do qual possa se auto-modificar durante sua execução, o torna adaptativo [1].

Na literatura podemos encontrar uma ampla variedade de aplicações da tecnologia adaptativa de dispositivos adaptativos guiados por regras, assim como também poderíamos visualizar muitas outras, mais para a ampla difusão desta tecnologia é necessário contar com ferramentas computacionais que possuam interfaces amigáveis que apóiem o ensino da tecnologia adaptativa de dispositivos adaptativos guiados por regras, assim como o desenvolvimento de aplicações.

Este trabalho apresenta um sistema desenvolvido em camadas para o ensino da tecnologia adaptativa de dispositivos adaptativos guiados por regras. As camadas consideradas são: Apresentação, Dispositivo Determinístico, Dispositivo Não Determinístico e Dispositivo Adaptativo.

As próximas seções deste artigo estarão organizadas da seguinte maneira: na seção 2 serão abordados os conceitos das teorias educacionais. A seção 3 apresenta os fundamentos da tecnologia adaptativa de dispositivos adaptativos guiados por regras. Na seção 4 descreve o sistema desenvolvido para o ensino da tecnologia adaptativa para dispositivos adaptativos guiados por regras. Finalmente, na seção 5 serão abordadas as considerações finais deste trabalho.

## II. TEORIAS EDUCACIONAIS

O uso das tecnologias computacionais no ensino implica na utilização do computador como ferramenta didático-pedagógica que auxilia no processo de construção do conhecimento nos diversos níveis educacionais, seja para educação continuada, ensino presencial ou à distância, e possibilita múltiplas formas de tratar o conhecimento e criar ambientes mais dinâmicos de aprendizagem. Nesse sentido, o computador transforma-se em um poderoso recurso de suporte à aprendizagem, com inúmeras possibilidades pedagógicas [2].

Também, é importante compreender o modo como as pessoas aprendem e as condições necessárias para a aprendizagem, bem como identificar o papel do professor no processo de ensino, nesse sentido as teorias educacionais são importantes porque possibilita ao professor adquirir conhecimentos, atitudes e habilidades que lhe permitirão alcançar melhor os objetivos do ensino.

As teorias de aprendizagem buscam reconhecer a dinâmica envolvida nos atos de ensinar e aprender, e tentam explicar a relação entre o conhecimento pré-existente e o novo conhecimento. Estas teorias de aprendizagem têm em comum o fato de assumirem que indivíduos são agentes ativos na busca e construção de conhecimento, dentro de um contexto significativo.

Entre as principais teorias de aprendizagem temos:

- **Construtivismo:** Segundo Piaget [3], o desenvolvimento cognitivo ocorre a partir da ação do sujeito sobre o meio ambiente. A partir dessa interação a pessoa constrói ou transforma estruturas mentais (esquemas) adquirindo maneiras de fazê-las funcionar. O eixo central, portanto, é a interação organismo-meio e essa interação acontece através de dois processos simultâneos: a organização interna e a adaptação ao meio, funções exercidas pelo organismo ao longo da vida.
- **Aprendizagem Significativa**, A idéia central da teoria de Ausubel é a diferenciação entre *aprendizagem significativa* e *aprendizagem mecânica*. Aprendizagem significativa ocorre quando uma nova informação é

M. R. F. Santibanez, Universidade de São Paulo (USP), São Paulo SP, Brasil, miguel@usp.br

J. J. Neto, Universidade de de São Paulo (USP), São Paulo SP, Brasil, joao.jose@poli.usp.br

relacionada a um aspecto relevante, já existente, na estrutura cognitiva do aprendiz. Em contrapartida, a aprendizagem mecânica ocorre quando a nova informação não se relaciona a conceitos já existentes na estrutura cognitiva sendo arbitrariamente armazenada e, portanto, pouca ou nenhuma interação ocorre entre a nova informação adquirida e aquela já armazenada [4]. Ausubel refere-se à estrutura cognitiva como o conteúdo total das informações, fatos, conceitos, princípios etc., sendo esta altamente organizada e hierarquizada, na qual elementos menos importantes são incorporados a conceitos maiores, mais gerais e inclusivos. Aprendizagem é definida como sendo o processo pelo qual o novo conteúdo se organiza e se integra à estrutura cognitiva [5]. Dentre as principais implicações da teoria da aprendizagem significativa está a ênfase de que uma nova informação deve ser significativa para o aprendiz. Cabe ao professor conseguir torná-la significativa, fazer o aluno relacionar a nova informação com outros conceitos relevantes já existentes em sua estrutura cognitiva [6].

- **Abordagem Sócio – Interacionista de Lev Seminovitch Vygotsky**: Considera o sujeito um ser ativo que se desenvolve num ambiente histórico e social e aprende na interação com o outro. Segundo Vygotsky [7], a aprendizagem tem um papel fundamental para o desenvolvimento do saber, do conhecimento. Todo e qualquer processo de aprendizagem é ensino-aprendizagem, incluindo aquele que aprende, aquele que ensina e a relação entre eles. Ele explica esta conexão entre desenvolvimento e aprendizagem através da zona de desenvolvimento proximal (distância entre os níveis de desenvolvimento potencial e nível de desenvolvimento real), um "espaço dinâmico" entre os problemas que um aprendiz pode resolver sozinho (nível de desenvolvimento real) e os que deverá resolver com a ajuda de outro sujeito mais capaz no momento, para em seguida, chegar a dominá-los por si mesmo (nível de desenvolvimento potencial). Vygotsky valoriza o trabalho coletivo, cooperativo. A colaboração entre aprendizes pressupõe um trabalho de parceria conjunta para produzir algo que não poderiam produzir individualmente.

As teorias de aprendizagem fundamentam o processo de elaboração do conhecimento mediado por computador, a linguagem, a mediação, a interação, a apropriação e os conceitos também devem nortear as ações e procedimentos na área de tecnologia educacional, definindo o papel do aprendiz como agente participativo e autônomo, o professor como agente mediador e articulador do processo de aprendizagem e o computador como instrumento produzido no decorrer da história humana com a finalidade de realizar atividades produtivas de acordo com os objetivos educacionais.

## III. Tecnologia Adaptativa de Dispositivos Adaptativos Guiados por Regras

Define-se a tecnologia adaptativa como o conjunto de técnicas, métodos, ferramentas e aplicações práticas dos dispositivos e sistemas que apresentam comportamento autonomamente auto-modificável.

Um dispositivo ou dispositivo não adaptativo é qualquer artefato abstrato cujo comportamento é determinado por um conjunto finito e explícito de regras também chamadas de regras não adaptativas. Tais regras especificam que a partir de uma configuração inicial conhecida e fixa, o dispositivo muda sucessivamente de uma configuração para outra em resposta a sucessivos estímulos de entrada recebidos e no conjunto de regras condicionais, do tipo "**se ... então**", que define completamente o comportamento do dispositivo. O conjunto de todas as regras é que determina de forma completa o comportamento do dispositivo. Estas regras têm a seguinte forma "ao receber um estímulo X na configuração Y, a nova configuração passa a ser Z" [8].

Um dispositivo adaptativo ou dispositivo adaptativo guiado por regras é conduzido por um conjunto de cláusulas da forma "ao receber um estímulo X na configuração Y, a nova configuração passa a ser Z, e também o conjunto de regras se altera pela remoção/substituição/inserção as regras R1, ... , Rn" [9].

As regras adaptativas especificam, para cada possível configuração do dispositivo, uma nova configuração, assim como, as eventuais alterações de comportamento, a ela associadas, as quais caracterizam a adaptatividade do dispositivo, e são representadas pelos mecanismos de automodificação expressos nas regras adaptativas, que determinam o comportamento dinâmico do dispositivo.

## IV. Sistema para o Ensino da Tecnologia Adaptativa de Dispositivos Adaptativos Guiados por Regras

O sistema foi implementado com base nos fundamentos da tecnologia adaptativa de dispositivos adaptativos guiados por regras e nos conceitos das teorias de aprendizagem, tendo como propósito prover suporte para o ensino desta tecnologia. Para alcançar este objetivo foi necessário desenvolver uma estrutura de classes para cada componente da estrutura do sistema.

Para a implementação do sistema foi utilizado a linguagem de programação Java. Esta linguagem permite a criação de versões portáteis, com facilidade adicional de integração na WWW, por dispor de código portátil executável sobre ambientes distribuídos. Esta asserção foi principalmente baseada nas características de um conjunto de propriedades de instruções intermediárias chamadas "bytecodes", que constitui a pedra angular do Java na tentativa de satisfazer a promessa de código "compilado uma vez, executado em qualquer lugar" [10].

Esta seção apresenta e descreve os componentes do sistema, com base na funcionalidade provida ao nível da interface interativa e amigável do usuário.

- Arquitetura do Sistema

Com base no estudo dos requisitos e funcionalidades do sistema, foram definidos os componentes do sistema. A arquitetura do sistema é dividida em quatro componentes principais e cada uma delas possui uma interface amigável com o usuário, como ilustrado na Figura 1: Apresentação, Editores de Dispositivo Determinístico, Dispositivo Não Determinístico e Dispositivo Adaptativo. À continuação discorre-se sobre a interface com o usuário e os componentes da arquitetura do sistema.

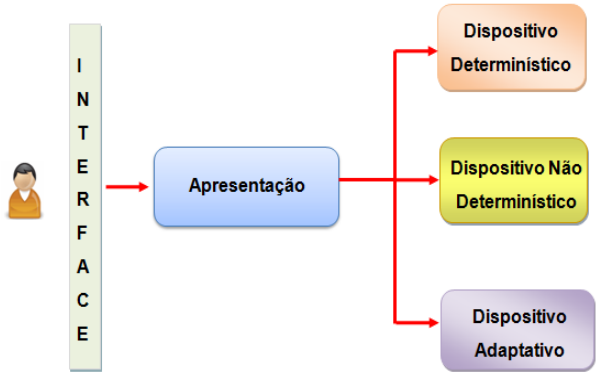



Figura 1. Arquitetura do Sistema de Ensino da Tecnologia Adaptativa de Dispositivos Guiados por Regras.

- Interface com o usuário. As interfaces são geralmente descritos como canais de informação que permitem comunicação entre usuários e computadores. No projeto de interface do sistema levaram-se em conta muitos princípios baseados nas teorias cognitivas, com o objetivo de proporcionar ao usuário um alto grau de eficiência e eficácia com relação a sua utilização. Uma das considerações principais no projeto da interface foi a tentativa de reduzir ao máximo a carga cognitiva do usuário, através de uma interface mais amigável, que se estende aos componentes da arquitetura do sistema.

Essa preocupação refletiu-se em muitas melhorias e recursos disponibilizados pelo sistema, tais como: mecanismo de armazenamento automático, bem como com seus formatos e respectivas cores, uso de caixas de diálogo, barras de rolagens e botões correspondentes, forma de apresentar e editar o banco de regras, banco de funções adaptativas entre outros recursos.

No projeto de interface considerou-se que a qualidade da interface de um sistema interativo com seu usuário influencia seu sucesso independentemente de sua funcionalidade. Outro aspecto importante considerado no projeto de interface foi minimizar o número de golpes no teclado e operações mentais (decisões) requerido para intentar alcançar uma tarefa do usuário, isto permitirá garantir um bom desenho da interface, pois, para os usuários, a interface é o próprio sistema.

-Apresentação. Este componente apresenta o sistema e permite o acesso aos outros componentes que são os dispositivos guiados por regras.

- Dispositivos: Determinístico, Não Determinístico e Adaptativo. Estes componentes são editores com interfaces interativas e amigáveis que permitem manipular e determinar as regras de cada dispositivo.

- Tela de Apresentação

Esta tela apresenta as definições da tecnologia adaptativa, dispositivos guiados por regras, dispositivo determinístico, dispositivo não determinístico e dispositivo adaptativo, assim como também mostram as *tabs* de acesso aos editores dos dispositivos determinístico, não determinístico e adaptativo.

A Fig. 2 mostra a tela de apresentação com as respectivas definições.

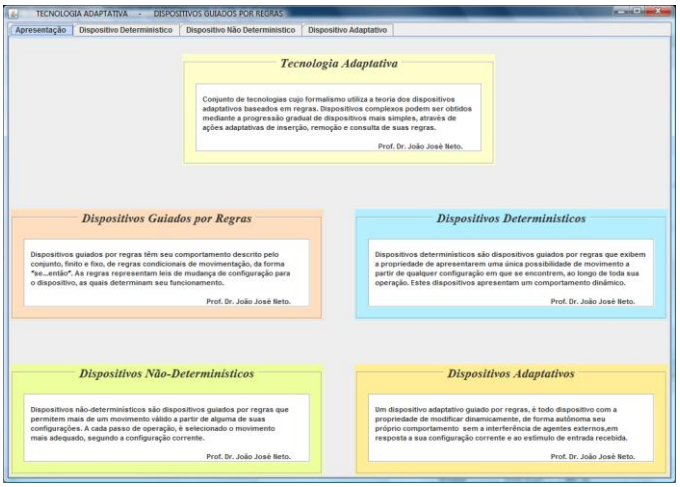



Figura 2. Tela de Apresentação do Sistema de Ensino da Tecnologia Adaptativa de Dispositivos Guiados por Regras.

### A. *Dispositivos Determinísticos*

Dispositivos determinísticos guiados por regras apresentam uma única possibilidade de movimento a partir de qualquer configuração em que se encontrem, ao longo de toda sua operação. Os dispositivos determinísticos apresentam um eficiente comportamento dinâmico, pois fluem livremente entre configurações, até que esgotem os estímulos recebidos, quando encerram então sua operação [11]. As regras têm o seguinte formato: "**Se** condição(ções) **então** ação(ções)".

- Arquitetura do Dispostivo Determinístico

Esta arquitetura possui quatro componentes implementados por classes que manipulam, selecionam e armazenam as regras.

A Fig. 3 mostra a arquitetura do dispostivo determinístico.

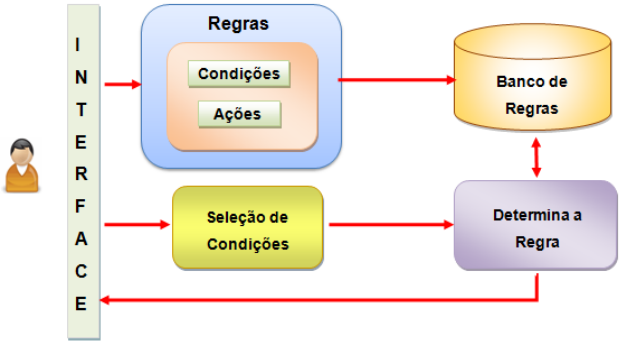



Figura 3. Arquitetura do Dispostivo Determinístico.

- Editor para o Dispositivo Determinístico

A Fig. 4 apresenta o editor do dispositivo determinístico.

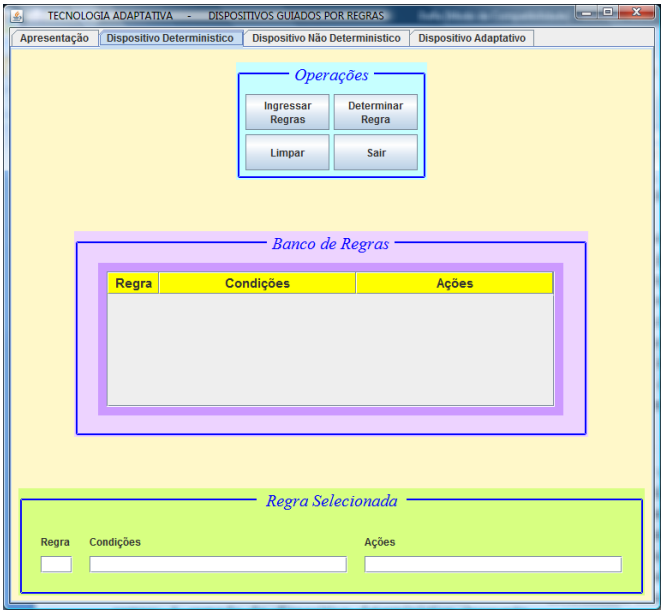



Figura 4. Editor do Dispostivo Determinístico.

Inicialmente esta tela apresenta 3 paineis, a saber: "Operações", "Banco de Regras" e "Regra Selecionada". O painel de "Operações" apresenta os botões: "Ingressar Regras" para digitar as condições e ações das regras. Automaticamente será criado os paineis de "Selecionar Condições" e "Ações" mostrando as condições em botões e as ações em caixas de texto. As regras ingressadas serão mostradas no painel "Banco de Regras". O botão "Determinar Regra" tem como funcionalidade mostrar no painel "Regra Selecionada" a regra que corresponde as condições selecionadas pelo usuário no painel "Selecionar Condições". Estas comparações são realizadas no "Banco de Regras". O botão "Limpar" tem como funcionalidade de deixar todos os botões do painel "Selecionar condições" na condição de não selecionados, assim como limpar as caixas de textos do painel "Regra Selecionada". A funcionalidade do botão "Sair" é sair do sistema. Pode-se selecionar outras *tabs* para acessar aos editores dos outros dispositivos ou á tela de apresentação.

- Teste

A seguir ilustra-se um exemplo de uso do editor do dispositivo determinístico guiado por regras. No painel "Operações" ao clicar no botão "Ingressar Regras" será mostrada uma caixa de diálogo com o cabeçalho "Digitar as Condições e Ações da Regra 1". O número da regra será seqüencial e numerada automaticamente. A caixa de diálogo mostra os rótulos de "Condições:" e "Ações:" assim como duas caixas de texto para digitar números que correspondem às condições e ações da regra. Cada número possui um significado semântico. A Fig. 5 mostra a caixa de diálogo para digitar as condições e ações das regras.

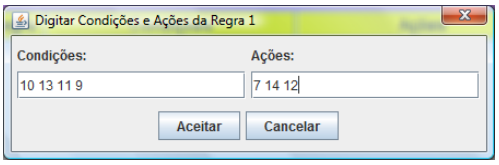



Fig. 5. Caixa de Diálogo para Digitar as Condições e Ações da Regra.

Cada vez que são ingressadas as condições e ações serão criados botões no painel de "Selecionar Condições", onde cada botão possui um número que corresponde a uma condição e no painel de "Ações" serão criadas caixas de texto contendo cada uma um número que corresponde a uma ação. Também será apresentada no "Banco de Regras" cada regra ingressada. A Fig. 6 mostra um exemplo de uso do Editor de Dispositivo Determinístico.

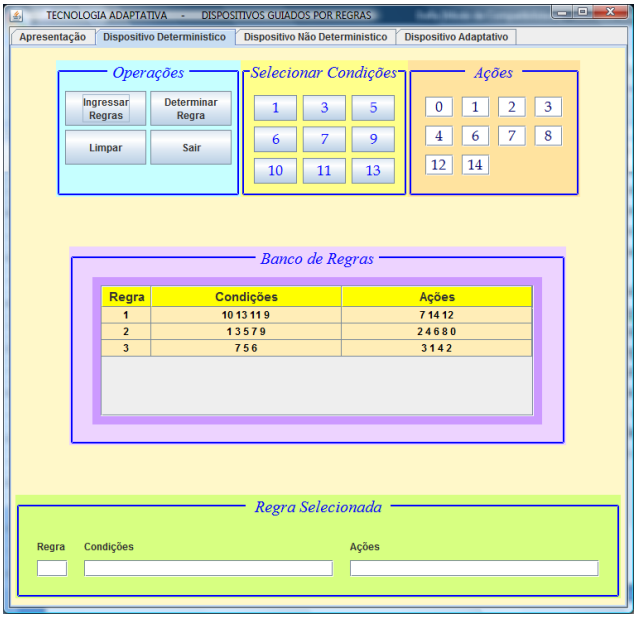



Fig. 6. Editor do Dispositivo Determinístico Mostrando um Exemplo.

Para determinar uma regra, selecionamos as condições clicando nos botões do painel "Selecionar Condições", caso ocorra um erro na seleção, pode-se corrigir clicando novamente no respectivo botão. A continuação clica-se no botão "Determinar Regra" e será mostrado no painel "Regra Selecionada" o número da regra, as condições e as ações da regra em questão obtida por um algoritmo a partir do banco de regras. Se o algoritmo não encontra uma regra no banco de regras para as condições selecionadas será mostrado "erro" nas caixas de texto do painel "Regra Selecionada". A Fig. 7 mostra um exemplo.

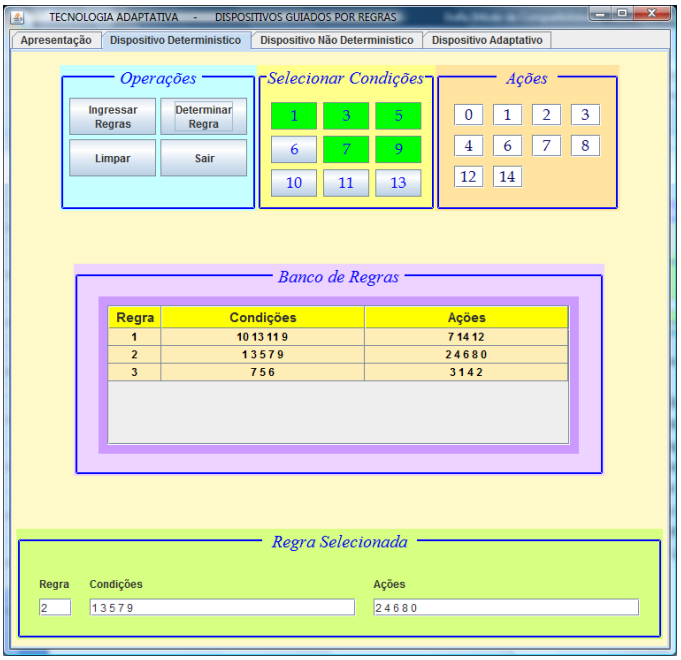



Fig. 7. Editor Mostrando as Condições e Regra Selecionada.

### *B. Dispositivo Não Determinístico*

Dispositivo não determinístico guiados por regras permitem nenhum ou mais de um movimento válido a partir de alguma de suas configurações em que se encontrem no percurso de sua operação.

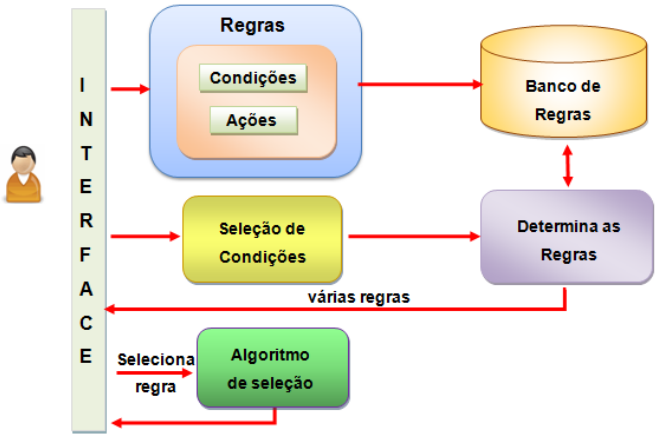
Uma maneira de implementar um dispositivo não determinístico guiada por regras é através do uso de processadores em paralelo onde cada movimento seja processado independentemente aquela em que os possíveis movimentos sejam processados em paralelo, independentemente uns dos outros.

Os dispositivos não-determinísticos exigem, a cada passo de operação, a execução de uma tomada de decisão, para que seja selecionado, dentre todas as possibilidades válidas de movimentação, um particular movimento, aquele que se mostrar mais adequado, a partir da particular configuração corrente.

Tais escolhas nem sempre são triviais, e sua aplicação sistemática durante a operação dos dispositivos não-determinísticos costuma prejudicar, em geral sensivelmente, a eficiência da operação do mesmo, constituindo isso uma das razões mais sérias pelas quais na prática se recomende que, na medida do possível, seja evitado o uso de não-determinismos.

- Arquitetura do Dispostivo Não Determinístico

Esta arquitetura possui cinco componentes implementados por classes que manipulam, selecionam e armazenam as regras. A Fig. 8 mostra a arquitetura do dispositivo não determinístico.




A Fig. 8 Arquitetura do Dispostivo Não Determinístico.

- Editor para o Dispositivo Não Determinístico

Esta tela apresenta o editor para o dispositivo não determinístico cujas funcionalidades dos paineis "Operações", "Selecionar Condições", "Ações", "Banco de Regras" e "Regra Selecionada" são similares ao editor de Dispositivo Determinístico. O painel "Selecione Regra" contem botões de radio com as regras que correspondem às condições selecionadas, o usuário pode selecionar um dos botões de radio o qual será mostrado no painel "Regra Selecionada". A Fig. 9 mostra o editor do dispositivo não determinístico.

- Teste

A continuação mostra-se um uso prático do editor do dispositivo não determinístico guiado por regras. Para o ingresso das condições e ações, assim como a seleção das condições procedemos como no caso do editor de dispositivo determinístico. Após a seleção das condições clica-se no botão "Determinar Regra" e será mostrado no painel "Selecione Regra" o subconjunto de regras que possuem as mesmas condições das condições selecionadas (essa comparação é realizada no "Banco de Regras"), cada regra tem um botão de radio que permitirá ao usuário selecionar um deles. A regra selecionada pelo usuário será mostrada no painel "Regra Selecionada". Se não existem regras no banco de regras com as mesmas condições que as condições selecionadas serão mostradas a mensagem "Erro" nas caixas de texto do painel "Regra Selecionada". A Fig. 10 mostra um exemplo de uso do editor do dispositivo não determinístico.

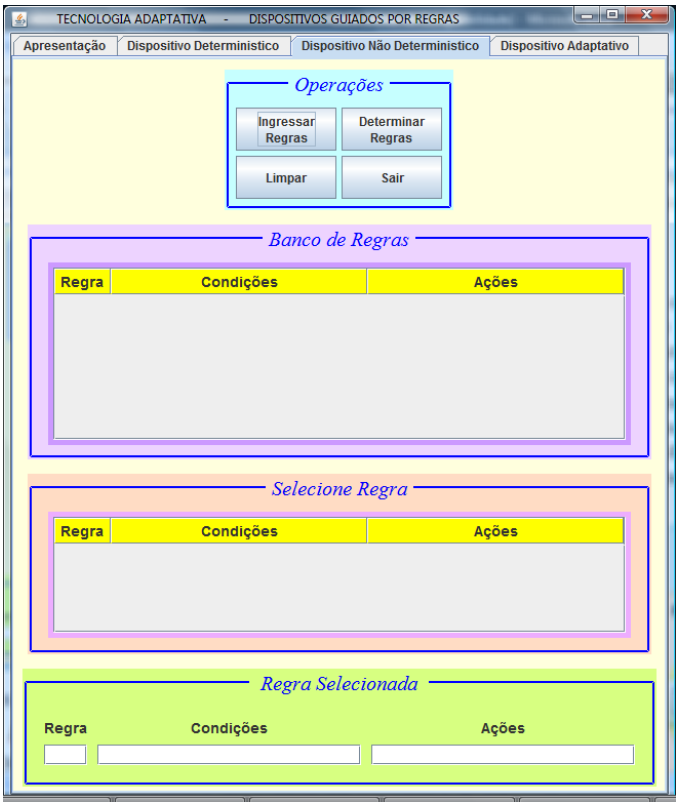



Figura 9. Editor do Dispostivo Não Determinístico.

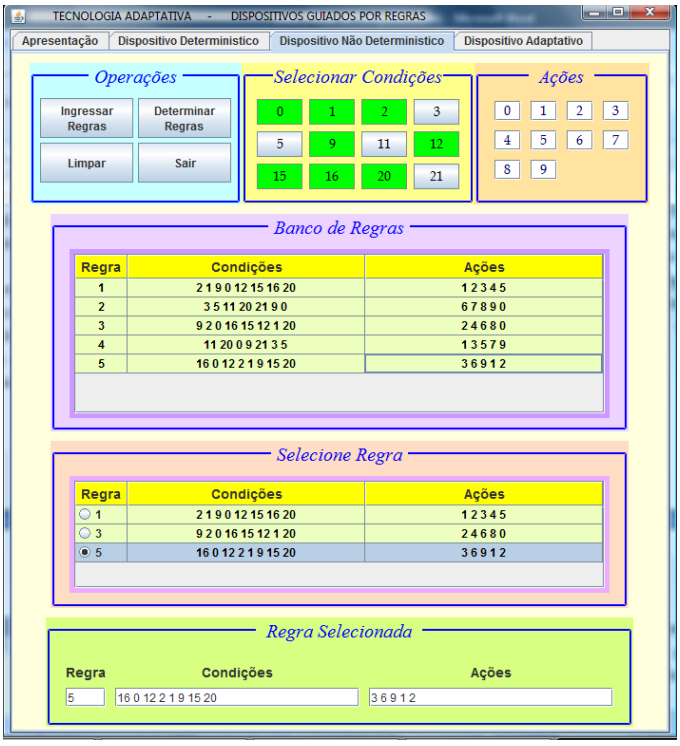



Figura 10. Exemplo de uso do Editor do Dispostivo Não Determinístico.

## C. *Dispositivo Adaptativo*

Um dispositivo adaptativo está composto por um dispositivo não-adaptativo subjacente (determinístico ou não determinístico), acrescentado de um mecanismo formado por um conjunto de funções adaptativas capaz de alterar o conjunto de regras que define seu comportamento.

As funções adaptativas proporcionam ao dispositivo adaptativo uma forma de alterar o seu próprio conjunto das regras, pois os valores associados aos elementos variáveis (parâmetros, geradores) que definem dispositivos adaptativos podem ser alterados a cada passo da sua operação [12].

Assim, resultam dispositivos adaptativos cujos comportamentos são descritos como conjuntos de regras, tendo à disposição uma variedade de funções adaptativas (incluindo a função adaptativa nula), as quais, convenientemente acopladas às regras do dispositivo subjacente, são executadas sempre que for necessário alterar o conjunto de regras.

- Arquitetura do Dispostivo Adaptativo

Esta arquitetura possui seis componentes implementados por classes que manipulam, selecionam e armazenam as regras adaptativas. A Fig. 11 mostra a arquitetura do dispositivo adaptativo.

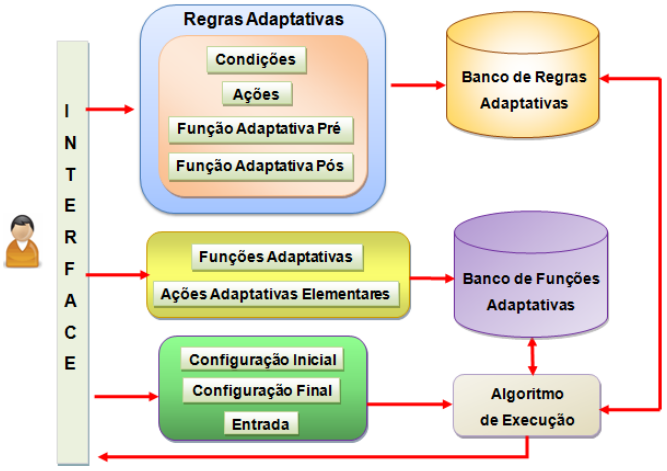



Figura 11. Arquitetura do Dispostivo Adaptativo.

- Editor para o Dispositivo Adaptativo

A seguir apresenta-se o editor para o dispositivo adaptitivo. O painel "Operações" contem 10 botões a saber: "Novo" para criar uma nova aplicação, "Abrir" para abrir uma aplicação existente, "Salvar" salva a aplicação corrente, "Executar" para executar a aplicação corrente, "Limpar" para limpar as caixas de texto do painel "Regra Selecionar" e por os botões do painel "Selecionar Condições" na condição de não selecionados, "Sair" para sair do sistema, "Inserir" ou "Alterar" ou "Remover" ou "Determinar" e logo "Regra" para inserir, alterar, remover ou determinar regras adaptativas; "Inserir", "Alterar" ou "Remover" e logo clica-se numa função adaptativa, para inserir, alterar ou remover uma função adaptativa. As regras são armazenadas no banco de regras e mostradas no painel "Banco de Regras". O painel de "Função Adaptativa" será mostrado as ações adaptativas elementares de cada função adaptativa. Os paineis "Selecionar Condições", "Ações", "Banco de Regras" e "Regra Selecionada" são similares ao outros editores. A Fig. 12 mostra o editor do dispositivo adaptativo.

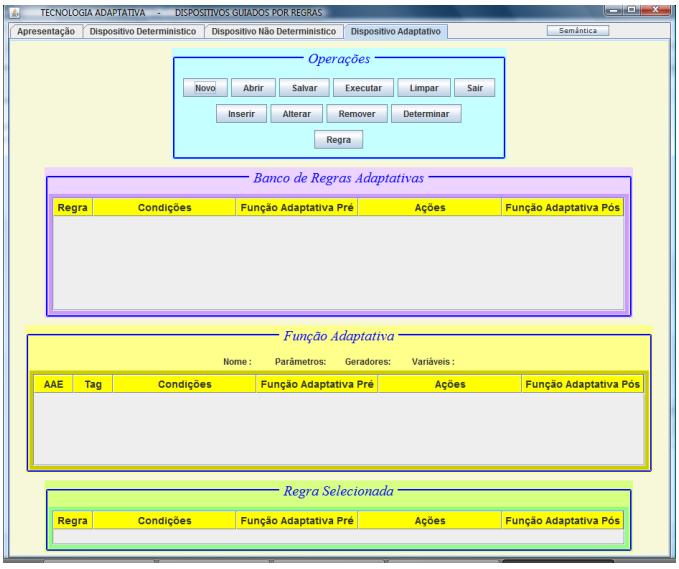



Figura 12. Editor do Dispostivo Adaptativo.

- Teste

A seguir mostra-se um uso prático do editor do dispositivo adaptativo guiado por regras. Para ingressar as regras adaptativas clica-se nos botões "Inserir" e "Regra" o editor abre uma caixa de diálogo como mostrado na Fig. 13.

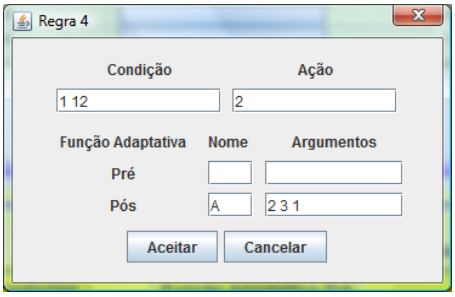



Figura 13. Caixa de Diálogo para o Ingresso das Regras.

Na caixa de diálogo ingressamos as condições e ações. Assim como o nome da função adaptativa pré e/ou pós com seus argumentos respectivos. A numeração das regras adaptativas será mostrada de maneira seqüencial e automática. Ao fechar a caixa de diálogo, a regra ingressada será mostrada no "Banco de Regras Adaptativas". As funções adaptativas pré e/ou pós com seus argumentos respectivos serão mostradas no "Banco de Regras Adaptativas" na forma de botões, de maneira automática será criado um botão por cada função adaptativa, contendo seu nome no painel "Operações". Os painéis de "Condições" e "Ações" são similares ao outros dispositivos. Após da seleção das condições no painel "Condições" clica-se nos botões "Determinar" e "Regra", o editor mostrará no painel "Regra Selecionada" a regra adaptativa que possua as mesmas condições das condições selecionadas (essa comparação é realizada no "Banco de Regras"). Se não existe regra adaptativa no banco de regras com as mesmas condições que as condições selecionadas serão mostradas a mensagem "Erro" nas caixas de texto do painel "Regra Selecionada".

Para ingressar as ações adaptativas elementares de uma função adaptativa clica-se no botão "Inserir" e no botão que tem o nome da função adaptativa respectiva. Na primeira vez será mostrada uma caixa de diálogo com um cabeçalho com o nome da função adaptativa e caixas de texto para ingressar os parâmetros, geradores e variáveis da função adaptativa, como mostrada na Fig. 14.

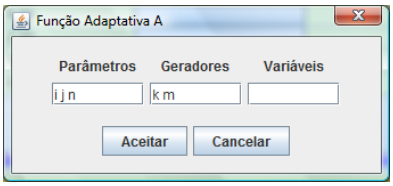



Figura 14. Caixa de Diálogo para o Ingresso da Função adaptativa.

A partir da segunda e posteriores vezes será apresentada a caixa de diálogo com um cabeçalho com o nome da função adaptativa e caixas de texto para ingressar os componentes da ação adaptativa elementar a saber: tag, condição, ação, função adaptativa pré e/ou pós com seus respectivos argumentos, como mostra a Fig. 15.

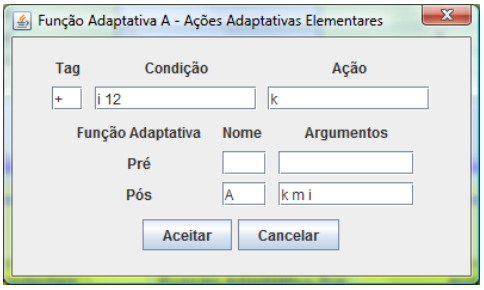



Figura 15. Caixa de Diálogo para Ingresso das Ações Adaptativas Elementares.

Ao fechar as caixas de diálogo das Fig. 14 e 15, os valores ingressados serão mostrados no painel "Função Adaptativa", como ilustra a Fig. 16.

Para visualizar os parâmetros, geradores, variáveis e ações adaptativas elementares de uma função adaptativa cujos valores já foram ingressados, clica-se no botão que tenha o nome e argumentos de aquela função adaptativa que se encontra no painel "Banco de Regras Adaptativo".

Para salvar a aplicação corrente clica-se no botão "Salvar" e aparece uma caixa de diálogo permitindo ingressar o nome do arquivo cuja extensão será ".da" (dispositivo adaptativo).

Os números dos paineis "Condições" e "Ações" possuem um significado semântico, isto pode ser visualizado a qualquer momento ao clicar no botão "Semântica" que apresenta uma janela desplegável com duas tabelas contendo a numeração e seu significado para as condições e ações. O texto do botão muda para "Ocultar" para fechar a janela desplegável como mostra a Fig. 17.

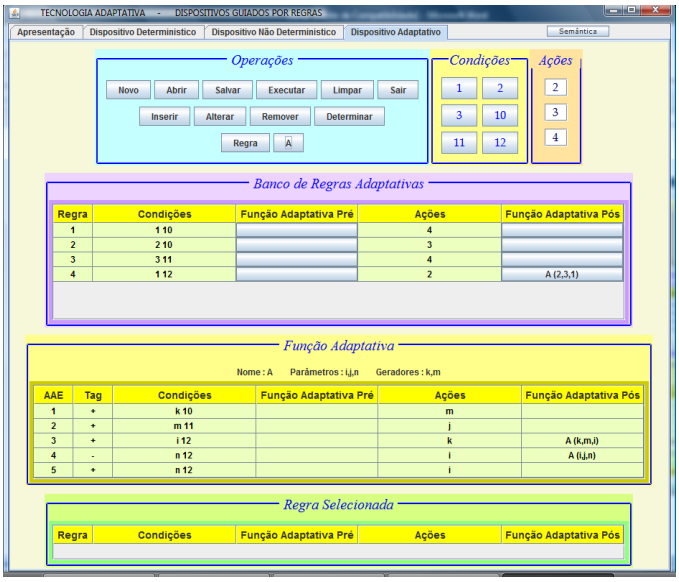



Figura 16. Exemplo de usos do Editor do Dispositivo Adaptativo Guiado por Regras.

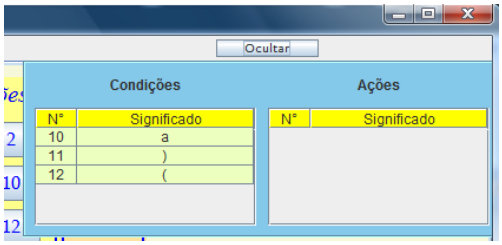



Figura 17. Numeros com o Respectivo Significado Semântico das Condições e Ações.

Ao clicar no botão "Executar" o editor mostra uma caixa de diálogo para ingressar a configuração inicial, configuração final e a cadeia de entrada. Como ilustra a Fig. 18. Após clicar no botão "Aceitar" o sistema executa o exemplo onde as regras do "Banco de Regras Adaptativas" transformam-se de acordo com as funções adaptativas como ilustrado na Fig. 19.

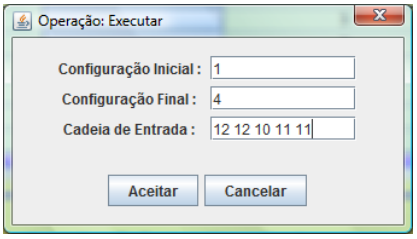



Figura 18. Execução do exemplo.

Finalmente a Fig. 20 ilustra as camadas dos dispositivos guiados por regras: O dispositivo determinístico apresenta as regras com as condições e ações com uma única possibilidade de movimento em qualquer operação. A Camada sobreposta à camada anterior é a do dispositivo não determinístico que permite nenhum ou mais de um movimento valido a partir de alguma de suas configurações. E finalmente a camada sobreposta às duas anteriores é do dispositivo adaptativo, onde além das propriedades anteriores ele pode ter funções adaptativas pré e/ou pós.

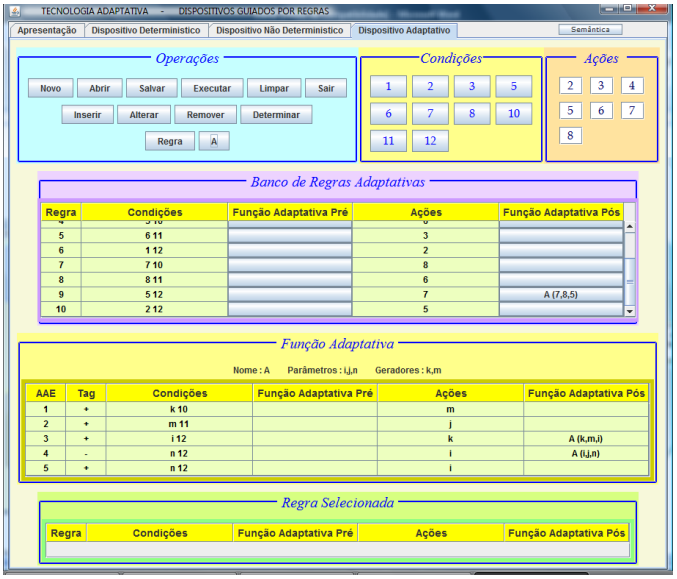



Figura 19. Exemplo de usos do Editor do Dispositivo Adaptativo, Após a Execução do Programa.

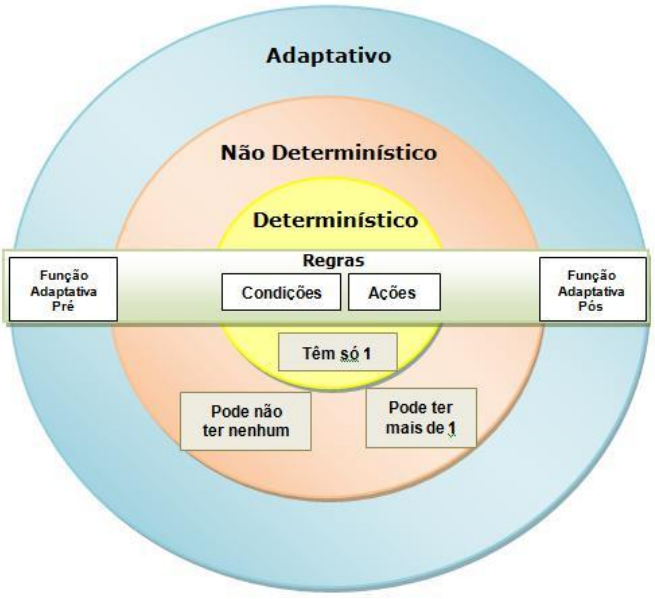



Figura 20. Dispositivos Guiados por Regras.

## V. Conclusão

Diante do exposto, conclui-se que, com base nas teorias de aprendizagem e nos fundamentos da tecnologia adaptativa de dispositivos guiados por regras é possível construir sistemas computacionais que possuam interfaces amigáveis com o propósito de ensino dos dispositivos guiados por regras.

Professores e aprendizes podem usar o sistema de software para construir modelos adaptativos e experimentar interativamente os conceitos para melhor entender a tecnologia adaptativa de dispositivos guiados por regras.

O interesse pela parte prática dos conceitos ocupa o centro das atenções dos aprendizes apontando a necessidade de testar

os conhecimentos teóricos adquiridos sejam aplicados e as estratégias pedagógicas na utilização do computador como ferramenta para o ensino/aprendizagem, específicos da tecnologia adaptativa de dispositivos guiados por regras, sejam exploradas.

Deseja-se que o sistema desenvolvido incentive e colabore na disseminação dos conceitos da Tecnologia Adaptativa de dispositivos guiados por regras, assim como apóie o desenvolvimento de novas aplicações.

## REFERÊNCIAS


[1] J. J. Neto, *Contribuições à metodologia de construção de compiladores.* Tese de Livre Docência, EPUSP, São Paulo, 1993.

[2] W. J. Dizeró, J. J. Neto. Uso de adaptatividade na modelagem de cursos para software educacional. *4º Workshop de Tecnologia Adaptativa – WTA*, 2010.

[3] J. Piaget. A equilibração das estruturas cognitivas. Rio de Janeiro : Zahar, 1975.

[4] J. D. Novak, Progress in application of learning theory. *Theory Into Practice*, **19**(1): 58-65, 1980.

[5] D. Ausubel, J. Novak & H. Hanesian. Educational Psychology: A Cognitive View (2nd Ed.). New York: Holt, Rinehart & Winston, 1978.

[6] M. A. Moreira. *Teoria de aprendizagem*. São Paulo. EPU, 1999.

[7] L.Vygotsky. *A Formação Social da Mente*. SP, Martins Fontes, 1999.

[8] J. J. Neto, Adaptatividade: Generalização Conceitual. 3º *Workshop de Tecnologia Adaptativa – WTA*, 2009.

[9] J. J. Neto. Solving complex problems with Adaptive Automata. *Lecture Notes in Computer Science.* S. Yu, A. Paun (Eds.): Implementation and Application of Automata 5th International Conference, CIAA, Vol.2088, London, Canada, Springer-Verlag, pp.340, 2000.

[10] P. Deitel, H. Deitel. *Java como Programar*. Pearson Education do Brasil 8º Edição, São Paulo, 2010.

[11] J. J. Neto, Adaptive Rule-Driven Devices – General Formulation and Case Study. (B. W. Watson, & D. Wood, Eds.) *Implementation and Application of Automata 6th International Conference* - CIAA, Vol. 2497, pp. 234-250, 2001.

[12] H. Pistori, H. *Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.* Tese de Doutorado. Escola Politécnica da USP. São Paulo, Brasil, Dezembro, 2003.


**Miguel R. Flores Santibanez**, graduado em Ciência da Computação pela Universidad Nacional Mayor de San Marcos (UNMSM), Lima, Perú em 1982, mestrado em Engenharia Eletrônica e Computação pelo Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, São Paulo, Brasil em 1999, e atualmente é aluno de doutorado na Politécnica da Universidade de São Paulo (USP). Suas principais áreas de pesquisas são: tecnologia adaptativa, processamento de linguagem natural, informática na educação, redes bayesianas, teoria da computação e modelos probabilísticos.

**João José Neto** graduado em Engenharia de Eletricidade (1971), mestrado em Engenharia Elétrica (1975) e doutorado em Engenharia Elétrica (1980), e livre-docência (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente é professor associado da Escola Politécnica da Universidade de São Paulo, e coordena o LTA - Laboratório de Linguagens e Tecnologia Adaptativa do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

# Mejora De La Calidad De Voz En Castellano Para El Sintetizador *Festival* Utilizando El Método De Autómatas Adaptativos: Números Arábigos Y Fechas (18 de octubre de 2010)

C. Ruíz, R. Caya , *Member IEEE* y C. Zapata, *Member IEEE*

***Abstract*— The project presented in this article carries on a study of the improvement in the quality of the synthetic voice generated by Festival using the adaptive automata technique. These automatons are designed trying to solve the issues found during the speech synthesis of Arabic numbers and dates, which were identified according to the linguistic rules for Spanish language. Once the development were finished will be possible execute formal tests to evaluate the voice quality generated by the version of Festival that includes the implemented improvements.**

***Keywords*— *speech synthesis, natural language interfaces, adaptive automata.***

## I. Nomenclatura

- AA: Autómata adaptativo.
- TTS: Texto to speech, en español texto a voz.
- Inteligibilidad: la facilidad para comprender la señal de voz producida.
- Naturalidad: un indicador de la semejanza de los sonidos producidos artificialmente con los naturales.

## II. Introducción

ES un hecho ampliamente aceptado que por medio de la computadora es posible resolver múltiples operaciones en tiempos muy cortos en comparación con el que le tomaría al cerebro humano. Del mismo modo, es capaz de procesar grandes cantidades de información de acuerdo a criterios específicos, y buscar información en muchos casos de manera altamente eficiente. Sin embargo, es necesario que las computadoras no sólo sean capaces de realizar estas tareas, sino que sean capaces de realizarlas por medio de una comunicación natural y efectiva por medios auditivos. La razón fundamental de ello es la naturaleza social del ser humano, y sobre todo el uso primordial de la comunicación oral y escrita como medio para satisfacer sus necesidades.

Actualmente, la tecnología ha propuesto vías para establecer esta interacción con las computadoras. Una de ellas es la síntesis del habla, por medio de la cual se busca generar artificialmente voz similar a la humana. Sin embargo la diferencia entre la voz sintética y la humana es claramente perceptible y en muchos casos llega a generar rechazo por parte de los usuarios.

*FESTIVAL* es uno de los sistemas de síntesis disponibles, el cual opera en la plataforma Linux y es de licencia open source. Dentro de él se han desarrollado módulos para diversos idiomas, uno de ellos es el castellano. Ciertamente, la calidad de la voz sintética varía de acuerdo al idioma en cuestión, en inglés y francés se encuentra bastante más avanzada que en el caso del castellano, idioma en el cual se reporta aún una voz de muy poca natural, entendida de acuerdo con **Erro! Fonte e referência não encontrada.** como la similitud lograda entre la voz artificial generada y el discurso humano.

Una de las causas que impiden el desarrollo de una voz natural es la falta de identificación de algunos formatos en el texto a ser sintetizado. Particularmente, los números arábigos y fechas, se presentan en patrones idiomáticamente establecidos y estandarizados para el idioma castellano. No obstante, Festival no es capaz de identificar los patrones que los representan y realiza la síntesis transformando cada carácter de estos números y fechas. Por ello es necesario establecer las reglas gramaticales correspondientes para el castellano que permitan transformar exitosamente las frases que contengan números arábigos y fechas a voz sintética.

## III. Marco conceptual

La síntesis de voz es el proceso en el cual se busca la producción artificial de señales de voz muy semejantes al habla humana. Tal como se señala en **Erro! Fonte de eferência não encontrada.** a síntesis de voz es usualmente realizada a partir de texto, por lo que se le denomina, en inglés, text-to-speech (TTS). Sin embargo, también es posible realizar este proceso a partir de alguna representación lingüística simbólica que los sintetizadores de voz son capaces de reconocer.

C. Zapata, Pontificia Universidad Católica del Perú, Lima, Perú, zapata.cmp@pucp.edu.pe.
R. Caya, Pontificia Universidad Católica del Perú, Lima, Perú, rosalia.caya@pucp.edu.pe.
C. Ruiz, Pontificia Universidad Católica del Perú, Lima, Perú, cruizv@pucp.edu.pe.

Asimismo, en **Erro! Fonte de referência não encontrada.** e menciona que actualmente existen dos técnicas o métodos principales que se utilizan para la síntesis: Síntesis concatenativa y síntesis por formantes. Para poder establecer la calidad de la voz obtenida se suelen tener en cuenta dos parámetros fundamentales: la naturalidad y la inteligibilidad **Erro! Fonte de referência não encontrada.**

Para conseguir un sintetizador de voz ideal es necesario que sea natural e inteligible al mismo tiempo. Sin embargo la proporción ideal entre estas características la definirá el contexto y las condiciones en las cuales se requiera aplicar la síntesis.

Tal como se mencionó en el acápite anterior un aspecto en el cual se puede aportar para la mejora de la calidad de la síntesis de voz es la identificación de los números arábigos y las fechas. En general, un sistema de numeración es el conjunto de símbolos y reglas que permiten construir todos los números válidos en dicho sistema. Estas reglas son diferentes para cada sistema de numeración considerado, pero una regla común a todos es que para construir números válidos en un sistema de numeración determinado sólo se pueden utilizar los símbolos permitidos en dicho sistema. En el caso de este proyecto se trabajará bajo las especificaciones del sistema de numeración decimal pues es el más utilizado.

Dentro de todo sistema existen numerales que representan la cantidad de elementos de un conjunto en relación con la serie de números dentro del sistema, o de acuerdo con **Erro! onte de referência não encontrada.** también llamados números cardinales. Y existen los numerales que expresan orden o sucesión en relación con los números naturales e indican el lugar que ocupa, dentro de una serie ordenada, el elemento al que se refieren. Estos últimos se denominan de acuerdo con **Erro! Fonte de referência não encontrada.** úmeros ordinales

Para el caso de las fechas, es necesario seguir un estándar para definir claramente el concepto de día, así como la representación del paso de los mismos en agrupaciones mayores como mes y año. El Calendario Gregoriano, si bien es un calendario originario de Europa, actualmente es utilizado de manera oficial en todo el mundo occidental. Es por esta globalidad que se tendrán como estándar para los conceptos antes mencionados lo que dicta el calendario gregoriano.

## IV. DESCRIPCIÓN DE LA SOLUCIÓN

Este proyecto propone implementación de una mejora de la calidad del sintetizador de voz *FESTIVAL*, utilizando el método de autómatas adaptativos en la representación de las normas gramaticales aplicadas para los números arábigos y fechas. Cada una de ellas en el contexto que se presenten y afecten la calidad de la voz sintética. La evaluación de la mejora de la calidad se hará, siguiendo el modelo utilizado en **Erro! Fonte de referência não encontrada.** desde as perspectivas de: naturalidad e inteligibilidad.

Para el alcance de este proyecto es importante recalcar que no se implementará un nuevo sintetizador de voz. Lo que se realizará es la modificación de las reglas que utiliza el sintetizador *FESTIVAL* a fin de adicionar el reconocimiento de los patrones necesarios para solucionar los problemas mencionados anteriormente. Dichas reglas se representarán usando el método de autómatas adaptativos.

Tal como se mencionó anteriormente el sintetizador con el cual se trabaja es *FESTIVAL* Speech Synthesis System, esta plataforma está dedicada por completo a la síntesis de voz, fue creada por el Centro de Investigación de Tecnologías del Lenguaje de la Universidad de Edimburgo **Erro! Fonte de eferência não encontrada.** y actualmente es mantenida por el Instituto de Tecnologías del Lenguaje de la Universidad Carnegie Mellon. Su elección corresponde a que se trata de un software de licencia open source y por lo tanto accesible, cuenta con documentación oficial,, ha sido utilizado anteriormente por otros trabajos de investigación con resultados satisfactorios como en **Erro! Fonte de referência ão encontrada.** y actualmente cuenta con una comunidad de mantenimiento oficial dentro de las principales distribuciones de Linux, Además de ello se han desarrollado múltiples aplicaciones que hacen uso de su voz sintética para fines de brindar accesibilidad, el caso más conocido es la herramienta Orca. Debido a estas razones su uso académico es no solo viable sino también recomendado.

Siguiendo la experiencia anterior, detallada en **Erro! Fonte e referência não encontrada.**, el proyecto ha dividido en dos fases: la investigación de las reglas gramaticales del español para la identificación y lectura de números arábigos y los formatos de fechas, y la implementación del aporte producto de la investigación. Los objetivos establecidos para la primera fase son: identificar los factores que impiden la producción de una voz sintética natural por la presencia de números arábigos y/o fechas; formular las reglas dependientes del contexto usando Autómatas Adaptativos que solucionen algunos de los factores identificados en el punto anterior para el idioma castellano; Analizar la estructura del sintetizador de voz *FESTIVAL* para decidir dónde es más conveniente realizar la modificación. La fase de desarrollo del aporte tiene como objetivos: implementar los autómatas adaptativos en la plataforma de síntesis elegida; diseñar y realizar experimentos con los autómatas adaptativos implementados en casos reales de síntesis de voz para el castellano, y por último la realización de pruebas para obtener indicadores que respalden la mejora de la calidad de la voz sintética. Actualmente el proyecto se encuentra finalizando la primera fase.

### *A. Sustento de la solución*

En los últimos años se han obtenido grandes avances en lo que respecta a la síntesis de voz, ello se debe a que se han desarrollado investigaciones que han dado como resultado nuevas técnicas utilizadas parar la mejora en los resultados de la voz sintética. Sin embargo, aún se evidencian problemas de naturalidad en los resultados. Esto se debe principalmente a los errores en el reconocimiento de frases del castellano por la incorrecta identificación de parámetros y patrones que permitan identificar determinados fragmentos de frases como formas particulares de las mismas, que tendrán una pronunciación distinta a la que puede ser determinada en forma literal por el sintetizador.

En el año 2009 se presentó al WTA la primera fase de este conjunto de proyectos **Erro! Fonte de referência não ncontrada.**, donde se trataba de mejorar la calidad de voz realizada en FESTIVAL modificando los patrones melódicos como efecto de los signos de puntuación, utilizando autómatas adaptativos. Entonces quedaron pendientes el estudio de la síntesis de números arábigos y fechas. El presente proyecto retoma dicha investigación para resolver el problema de la incorrecta lectura de números arábigos y de fechas.

Cuando un sintetizador como FESTIVAL procesa la lectura de números reales en formato arábigo y de fechas muchas veces, y sobre todo al sintetizar castellano, reproduce una síntesis incorrecta. Esto es debido a que los patrones son leídos como un símbolo más dentro del texto. Por ejemplo no se considera el separador decimal ni los separadores de fechas. La forma en la que se leen los números y las fechas depende del formato que se ha utilizado para representarlos por lo tanto son palabras léxicas que para su síntesis dependen del contexto. Esta dependencia justifica el uso de autómatas adaptativos para la solución planteada.

### *B. Resultados esperados*

El aporte práctico del proyecto es la implementación de una mejora de la inteligibilidad en la voz sintética producida por FESTIVAL para el castellano, lograda mediante AA **Erro! onte de referência não encontrada.**. Esta mejora en inteligibilidad por consecuencia da una mejora en la naturalidad del sintetizador al leer números y fechas.

En relación con los objetivos específicos del proyecto los resultados esperados son:

- Elaborar las reglas de detección de frases de acuerdo al contexto que permitan solucionar el problema.

- Diseñar de algunas reglas de producción de voz natural enfocadas en el problema planteado de acuerdo al método de Autómatas Adaptativos.

- Someterá el sintetizador a las pruebas convenientes para poder observar la mejora.

## V. APOYO TEÓRICO

La identificación de distintas formas de fragmentos de frases hace posible que, de acuerdo al contexto en el que se encuentran, determinadas frases representadas gráfica o textualmente de la misma forma puedan tener distinto significado.

Para ello, es necesaria la evaluación del contexto en el que se encuentran las frases para poder determinar el significado que se les dará y, a partir de esto, seleccionar la forma en la que se desarrollará la síntesis de voz.

En el caso específico de números y fechas, estas pueden encontrarse en distintas formas o posiciones dentro de frases de mayor longitud. De acuerdo a su posición en la frase o por la presencia de otros fragmentos de frases en el contexto general se puede reconocer la utilización de estos elementos para un determinado significado, lo que permite determinar la forma en la que el sintetizador de voz trabajará, acercándolo a las formas del lenguaje humano y ganando naturalidad.

Mediante la utilización de máquinas de estados finitos es posible representar las distintas reglas gramaticales de los lenguajes naturales. Además, mediante autómatas pueden representarse subconjuntos de reglas, las cuales pueden aplicarse para la mejora en la detección de los fragmentos de frases que permitirán adaptar la pronunciación hacia formas mejoradas por el análisis contextual mediante técnicas adaptativas.

Como se menciona en **Erro! Fonte de referência não ncontrada.**, la técnica de Autómatas Adaptativos es aplicable principalmente para problemas de lenguajes contexto-dependientes, por lo que la aplicación de esta solución vendría a ser la más adecuada para los problemas presentados en este proyecto.

## VI. MÉTODOLOGÍA APLICADA

El proyecto de investigación se encuentra guiado por la metodología propuesta en **Erro! Fonte de referência não ncontrada.** para proyectos de investigación. La adaptación de las líneas generales descritas en **Erro! Fonte de referência não encontrada.** para este proyecto dieron como resultado las siguientes etapas:

- Identificación de los parámetros presentes en la sintaxis que afectan la naturalidad del discurso.
- Identificación de casos en los cuales se encuentra presente la dependencia del contexto.
- Identificación de las reglas que solucionan los casos seleccionados.
- Construcción de los autómatas adaptativos que implementan dichas reglas.
- Selección y realización de pruebas que evalúen la calidad del sintetizador.
- Análisis de resultados.
- Elaboración de conclusiones.

## VII. DISEÑO E IMPLEMENTACIÓN DE LOS AUTÓMATAS ADAPTATIVOS

Una vez realizada la identificación de los factores que afectan la mejora de las características de la voz sintética, pueden definirse las reglas de los autómatas adaptativos.

La mejora en la síntesis de voz para números y fechas se debe realizar mediante modificaciones en distintos puntos de la arquitectura de Festival. A continuación se detallan el diseño e implementación de los Autómatas Adaptativos para cada uno de los casos planteados.

### *A. Autómata Adaptativo para la formación de números arábigos: enteros y reales*

El sistema numérico arábigo es un sistema posicional que consta de 10 glifos diferentes para representar los 10 símbolos del sistema. El valor de un dígito varía según la posición que ocupa dentro del número multiplicándose por la base elevada a la posición. Así, el primer dígito comenzando por la derecha tiene el valor que representa su símbolo multiplicado por $10^0$(=1). El dígito situado a su izquierda tiene el valor que representa su símbolo multiplicado por $10^1$(=10), y así sucesivamente.

El sistema "arábigo" se representa utilizando muchos

conjuntos de glifos diferentes. Estos glifos pueden dividirse en dos grandes familias, los numerales arábigos occidentales y los orientales. Los que estudiaremos en este proyecto serán los occidentales, representados por:

0 1 2 3 4 5 6 7 8 9

Para el análisis de los números *Enteros* y *Reales* se han definido los siguientes autómatas que ha definido los siguientes grupos de elementos:

- # = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- separadorDecimal = {",”}
- signoPuntuacion = {“.” “,” “;” “:” “”” “’” “)” “]” “}” “!” “?”}

Siguiendo la notación formal descrita en **Erro! Fonte de eferência não encontrada.**, el autómata adaptativo encargado de la construcción de números enteros, nombrado como $M_F$, se define formalmente como una 8-tupla:

M = (Q, §, $q_0$, F, $\delta$, Q,¡,¦), donde:

Q: {$a_0$, $a_f$}

§: {#, signoPuntuacion, ε}

$q_0$ : {$a_0$}

F : { $a_f$ }

$\delta$:

[1] $P_0$ : $a_0$, ε, $a_f$

[2] $P_1$ : $a_0$, #, $a_f$

Q : {a1, a2, a3,...}

¡: (−,a,ε,$a_f$) (+,a,#,$a_I$) (+,$a_I$,#,$a_{II}$) (+,$a_I$,ε,$a_{IV}$)
(+,$a_I$,signoPuntuacion,$a_f$) (+,$a_{II}$,#,$a_{III}$) (+,$a_{II}$,ε,$a_{IV}$)
(+,$a_{II}$,signoPuntuacion,$a_f$) (+,$a_{III}$,ε,$a_{IV}$)
(+,$a_{III}$,signoPuntuacion,$a_f$) (+,$a_{IV}$,#,$a_V$)(+,$a_{IV}$,¬#,$a_f$)
(+,$a_V$,#,$a_{VI}$) (+,$a_{VI}$,#,$a_{VII}$) (+,$a_{VII}$,ε,$a_{VIII}$)
(+,$a_{VII}$,signoPuntuacion,$a_f$) (+,$a_{VIII}$,#,$a_V$)
(+,$a_{VIII}$,¬#,$a_f$)

¦ ((a$_i$,#,$a_f$)) = {(−,a,ε,$a_f$), (+,a,#,$a_I$),
(+,$a_I$,#,$a_{II}$), (+,$a_I$,ε,$a_{IV}$),
(+,$a_I$,signoPuntuacion,$a_f$),
(+,$a_{II}$,#,$a_{III}$), (+,$a_{II}$,ε,$a_{IV}$),
(+,$a_{II}$,signoPuntuacion,$a_f$), (+,$a_{III}$,ε,$a_{IV}$),
(+,$a_{III}$,signoPuntuacion,$a_f$), (+,$a_{IV}$,#,$a_V$),
(+,$a_{IV}$,¬#,$a_f$), (+,$a_V$,#,$a_{VI}$), (+,$a_{VI}$,#,$a_{VII}$),
(+,$a_{VII}$,ε,$a_{VIII}$), (+,$a_{VII}$,signoPuntuacion,$a_f$),
(+,$a_{VIII}$,#,$a_V$), (+,$a_{VIII}$,¬#,$a_f$)}

La cadena inicial del AA denotada como ω corresponde al carácter inicial de cada token que ya ha sido separado por Festival en la etapa de tokenizing. Es a partir de este carácter que comienza la validación del autómata. En caso cumpla con las condiciones iniciales se continua con el análisis de los demás caracteres, continuando con la ejecución del autómata hasta que cumpla el estado final o termine en alguno de los estados no finales al no cumplir las condiciones planteadas.

La operación de $M_F$ se encuentra descrita en la Fig. 1

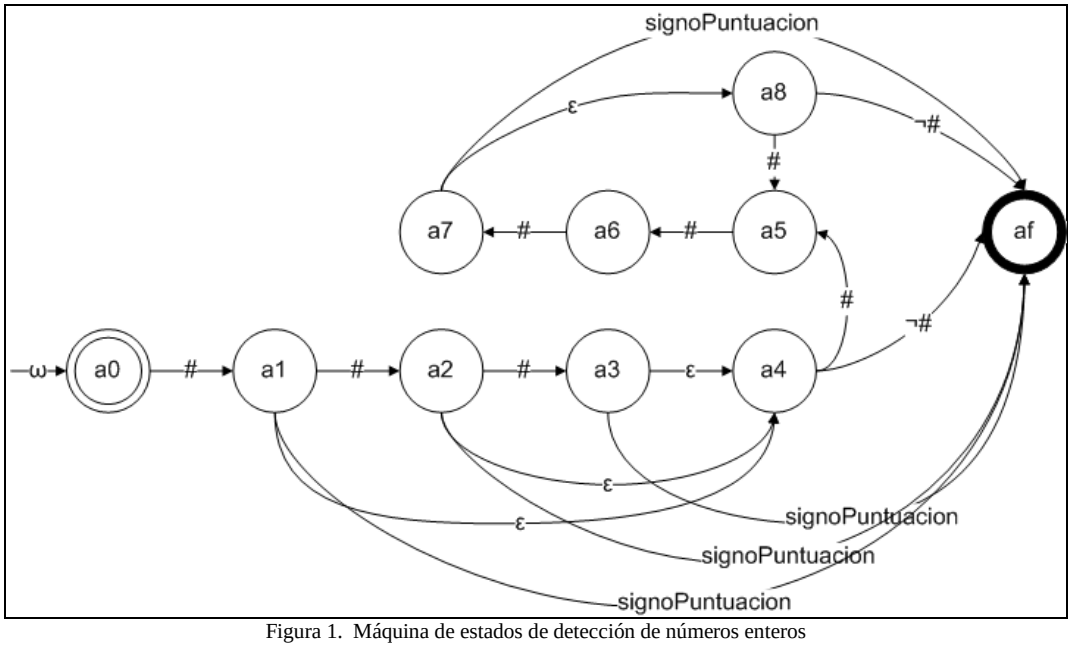



Figura 1. Máquina de estados de detección de números enteros

Para la construcción de números reales se define formalmente la 8-tupla $M_f$ de la siguiente manera:

M = ( Q, §, $q_0$, F, $\delta$, Q,¡,¦), donde:

Q: {$b_0$, $b_f$}

§: {#, signoPuntuacion, separadorDecimal, ε}

$q_0$ : {$b_0$}

F : { $b_f$ }

$\delta$:

[3] $P_0$ : $b_0$, ε, $b_f$

[4] $P_1$ : $b_0$, #, $b_f$

Q : {b1, b2, b3,...}

¡: ( −,b,ε,$b_f$ ) ( +,b,#,$b_I$ ) ( +,$b_I$,#,$b_{II}$ ) ( +,$b_I$,ε,$b_{IV}$ )
( +,$b_I$,separadorDecimal,$b_{IX}$ ) ( +,$b_{II}$,#,$b_{III}$ )
( +,$b_{II}$,ε,$b_{IV}$ )
( +,$b_{II}$,separadorDecimal,$b_{IX}$ ) ( +,$b_{III}$,ε,$b_{IV}$ )
( +,$b_{III}$,separadorDecimal,$b_{IX}$ ) ( +,$b_{IV}$,#,$b_V$ )
( +,$b_V$,#,$b_{VI}$ ) ( +,$b_{VI}$,#,$b_{VII}$ ) ( +,$b_{VII}$,ε,$b_{VIII}$ )
( +,$b_{VII}$,separadorDecimal,$b_{IX}$ ) ( +,$b_{VIII}$,#,$b_V$ )
( +,$b_{IX}$,#,$b_X$ ) ( +,$b_X$,#,$b_X$ ) ( +,$b_X$,ε,$b_f$ )
( +,$b_X$,signoPuntuacion,$b_f$ ) )

¦ ( ( $b_i$,#,$b_f$ ) ) = {( −,b,ε,$b_f$ ), ( +,b,#,$b_I$ ),
( +,$b_I$,#,$b_{II}$ ),
( +,$b_I$,ε,$b_{IV}$ ), ( +,$b_I$,separadorDecimal,$b_{IX}$ ),
( +,$b_{II}$,#,$b_{III}$ ), ( +,$b_{II}$,ε,$b_{IV}$ ),
( +,$b_{II}$,separadorDecimal,$b_{IX}$ ), ( +,$b_{III}$,ε,$b_{IV}$ ),
( +,$b_{III}$,separadorDecimal,$b_{IX}$ ),
( +,$b_{IV}$,#,$b_V$ ), ( +,$b_V$,#,$b_{VI}$ ), ( +,$b_{VI}$,#,$b_{VII}$ ),
( +,$b_{VII}$,ε,$b_{VIII}$ ),
( +,$b_{VII}$,separadorDecimal,$b_{IX}$ ),
( +,$b_{VIII}$,#,$b_V$ ), ( +,$b_{IX}$,#,$b_X$ ), ( +,$b_X$,#,$b_X$ ),
( +,$b_X$,ε,$b_f$ ), ( +,$b_X$,signoPuntuacion,$b_f$ )}

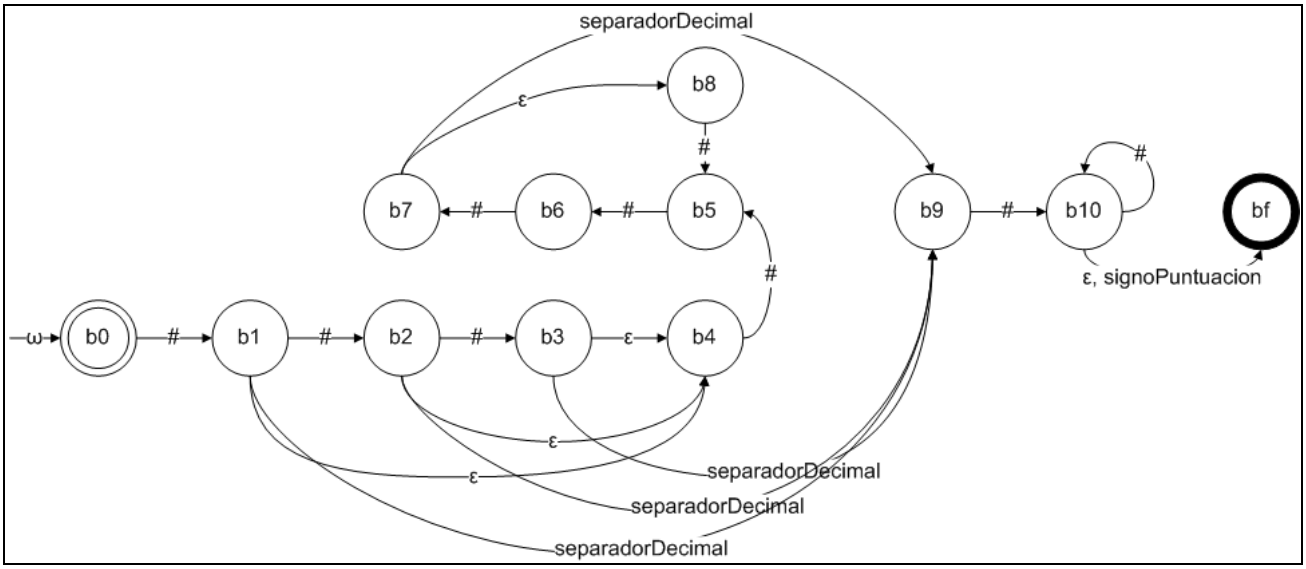



Figura 2. Máquina de estados de detección de números reales

*B. Autómata Adaptativo formación de fechas: estándar ISO y personalizadas*

Existen distintos formatos de fecha que han sido definidos por ISO o que son de utilización general a nivel de Latinoamérica. Es la norma que define el estándar para la representación de instantes, intervalos e intervalos recurrentes de tiempo evitando ambigüedades. Esta notación facilita la migración entre distintas plataformas. Se especifican en orden primeramente los periodos de tiempo más largos y posteriormente los más cortos.

Para representar fechas de calendario se usarán 4 cifras para el año, dos para el mes y dos para el día del mes, rellenando con ceros si es necesario. Por ejemplo, para representar la fecha 9 de octubre de 2009 se utilizará '2009' para representar el año, '10' para representar el mes de octubre y '09' para representar el día 9 de ese mes.

En la siguiente tabla se muestran las posibles representaciones y formatos recogidos en la norma para representar fechas de calendario, tomando como base para los ejemplos la fecha 9 de octubre de 2009. En las representaciones expandidas se representan los años con 6 cifras.

| | **formato básico (ejemplo)** | **formato extendido (ejemplo)** |
|---|---|---|
| **representación completa** | YYYYMMDD (20091009) | YYYY-MM-DD (2009-10-09) |
| **representación de precisión reducida** | YYYY-MM (2009-10)<br>YYYY (2009)<br>YY (20) | no se aplica |
| **representación expandida** | ±YYYYYMMDD (+0020091009)<br>±YYYYY-MM (+002009-10)<br>±YYYYY (+002009)<br>±YYY (+0020) | ±YYYYY-MM-DD (+002009-10-09)<br>no se aplica<br>no se aplica<br>no se aplica |

TABLA I: Formatos de fecha según norma ISO 8601

La representación completa permite especificar un día concreto del calendario. La representación de precisión reducida permite especificar un mes (YYYY-MM), un año (YYYY) o un siglo concretos. La representación expandida permite especificar un día (±YYYYYMMDD), un mes (±YYYYY-MM), un año (±YYYYY) o un siglo (±YYY) concretos (en estos tres últimos casos, se trataría de una representación expandida y de precisión reducida). La representación expandida requiere que haya un acuerdo previo de las partes que intercambian la información acerca del número de dígitos que requiere la representación del año, siendo como mínimo cinco.

Además de la forma estándar descrita anteriormente, existen formas de representación de las fechas del calendario de manera personalizada, con variaciones en forma y separadores, aunque tomando como base el estándar ISO.

Estos cambios consisten en:

- Cambio de separador guión ("-") por una barra oblicua ("/").
- Cambio de separador guión ("-") por un punto (".").
- Cambio de separador guión ("-") por un espacio (" ").
- Utilización del formato del tipo "DD-MM-YYYY".

Para el análisis de fechas se presentan los siguientes autómatas que han definido los siguientes grupos de elementos:

- # = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- separadorFecha = {"/", "-", "."}
- signoPuntuacion = {"." "," ";" ":" "”" "’" ")" "]" "}" "¡" "?"}

Para la construcción de fechas siguiendo el estándar ISO y su personalización se define formalmente la 8-tupla $M_f$ de la siguiente manera:

$M = (Q, \S, q_0, F, \delta, Q, ¡, ¦)$, donde:

$Q$: $\{c_0, c_f\}$

§: {#, signoPuntuacion, separadorFecha ε}

$q_0$ : $\{c_0\}$

F : $\{ c_f \}$

$\delta$:

[5] $P_0 : c_0, \varepsilon, c_f$

[6] $P_1 : c_0, \#, c_f$

Q : {c1, c2, c3,...}

¡: $(-,c,\varepsilon,c_f)$ $(+,c,\#,c_I)$ $(+,c_I,\#,c_{II})$ $(+,c_{II},\#,c_{III})$
$(+,c_{II},\text{separadorFecha},c_V)$ $(+,c_{III},\#,c_{IV})$
$(+,c_{IV},\text{separadorFecha},c_V)$ $(+,c_V,\#,c_{VI})$ $(+,c_{VI},\#,c_{VII})$
$(+,c_{VI},\text{separadorFecha},c_{VIII})$
$(+,c_{VII},\text{separadorFecha},c_{VIII})$ $(+,c_{VIII},\#,c_{IX})$ $(+,c_{IX},\#,c_X)$
$(+,c_{IX},\varepsilon,c_{VII})$ $(+,c_{IX},\text{signoPuntuacion},c_f)$ $(+,c_X,\varepsilon,c_f)$
$(+,c_X,\text{signoPuntuacion},c_f)$

¦ $((c_i,\#,c_f))$ = $\{(-,c,\varepsilon,c_f)$, $(+,c,\#,c_I)$,
$(+,c_I,\#,c_{II})$, $(+,c_{II},\#,c_{III})$, $(+,c_{II},\text{separadorFecha},c_V)$,
$(+,c_{III},\#,c_{IV})$, $(+,c_{IV},\text{separadorFecha},c_V)$,
$(+,c_V,\#,c_{VI})$, $(+,c_{VI},\#,c_{VII})$,
$(+,c_{VI},\text{separadorFecha},c_{VIII})$,
$(+,c_{VII},\text{separadorFecha},c_{VIII})$,
$(+,c_{VIII},\#,c_{IX})$, $(+,c_{IX},\#,c_X)$, $(+,c_{IX},\varepsilon,c_{VII})$,
$(+,c_{IX},\text{signoPuntuacion},c_f)$, $(+,c_X,\varepsilon,c_f)$,
$(+,c_X,\text{signoPuntuacion},c_f)\}$

De manera similar a los autómatas anteriores, la cadena inicial del AA denotada como ω corresponde al carácter inicial de cada token que ya ha sido separado por Festival en la etapa de tokenizing. Es a partir de este carácter que comienza la validación del autómata. En caso cumpla con las condiciones iniciales se continua con el análisis de los demás caracteres, continuando con la ejecución del autómata hasta que cumpla el estado final o termine en alguno de los estados no finales al no cumplir las condiciones planteadas.

La configuración de Mp y su cambio con cada transición adaptativa se representa en la Fig. 3.

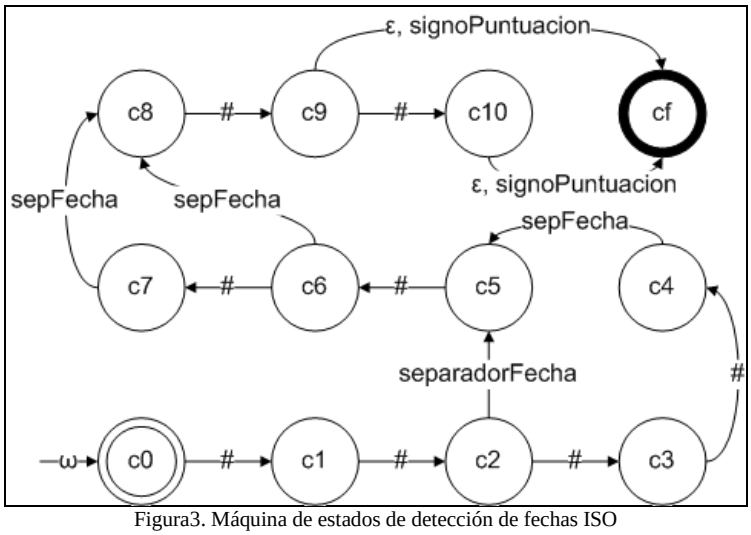



Figura3. Máquina de estados de detección de fechas ISO

Para la construcción de fechas personalizadas se define formalmente la 8-tupla Mf de la siguiente manera:

M =(Q, §, $q_0$, F, $\delta$, Q,¡,¦), donde:

Q: {$d_0$, $d_f$}

§: {#, signoPuntuacion,separadorFecha ε}

$q_0$ : {$d_0$}

F : { $d_f$ }

$\delta$:

[7] $P_0$ : $d_0$, ε, $d_f$

[8] $P_1$ : $d_0$, #, $d_f$

Q : {d1, d2, d3,...}

¡: (−,d,ε,$d_f$) (+,d,#,$d_I$) (+,$d_I$,#,$d_{II}$)
(+,$d_I$,separadorFecha,$d_{III}$) (+,$d_{II}$,separadorFecha,$d_{III}$)
(+,$d_{III}$,#,$d_{IV}$) (+,$d_{IV}$,#,$d_V$) (+,$d_{IV}$,separadorFecha,$d_{VI}$)
(+,$d_V$,separadorFecha,$d_{VI}$) (+,$d_{VI}$,#,$d_{VII}$)
(+,$d_{VII}$,#,$d_{VIII}$) (+,$d_{VIII}$,#,$d_{IX}$) (+,$d_{VIII}$,ε,$d_f$)
(+,$d_{VIII}$,signoPuntuacion,$d_f$) (+,$d_{IX}$,#,$d_X$) (+,$d_X$,ε,$d_f$)
(+,$d_X$,signoPuntuacion,$d_f$)

¦ (((($d_i$,#,$d_f$)) = {(−,d,ε,$d_f$), (+,d,#,$d_I$),
(+,$d_I$,#,$d_{II}$), (+,$d_I$,separadorFecha,$d_{III}$),
(+,$d_{II}$,separadorFecha,$d_{III}$), (+,$d_{III}$,#,$d_{IV}$),
(+,$d_{IV}$,#,$d_V$), (+,$d_{IV}$,separadorFecha,$d_{VI}$),
(+,$d_V$,separadorFecha,$d_{VI}$),(+,$d_{VI}$,#,$d_{VII}$),
(+,$d_{VII}$,#,$d_{VIII}$), (+,$d_{VII}$l,#,$d_{IX}$), (+,$d_{VIII}$,ε,$d_f$),
(+,$d_{VIII}$,signoPuntuacion,$d_f$),(+,$d_{IX}$,#,$d_X$), (+,$d_X$,ε,$d_f$),
(+,$d_X$,signoPuntuacion,$d_f$)}}

La descripción de su funcionamiento se presenta en la Fig. 4.

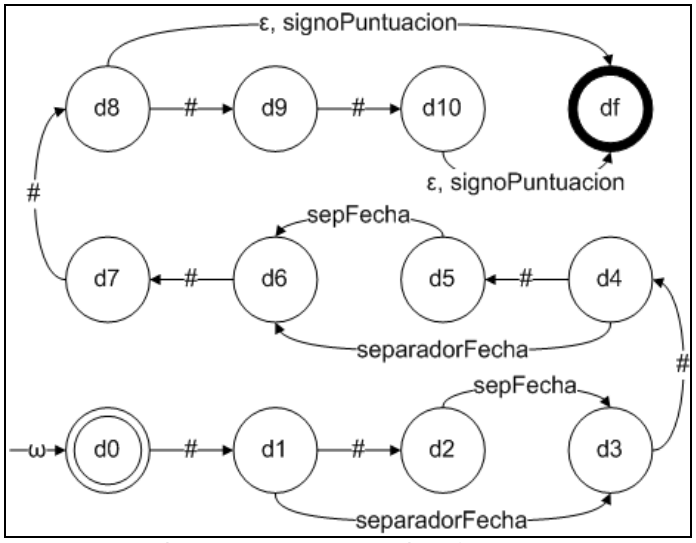



Figura4. Máquina de estados de detección de fechas personalizadas.

## VIII. Implementación

Con respecto a la implementación de la mejora, se espera desarrollar librerías dinámicas que permitan la selección en tiempo de ejecución, y de acuerdo a parámetros establecidos, de los autómatas que se aplicarán durante el desarrollo de la síntesis de voz, además del orden de ejecución.

Las configuraciones antes mencionadas contendrán, además, definiciones hacia los elementos del autómata, haciendo posible que puedan definirse más adelante nuevos casos o variantes a los estados posible del autómata, que permitirán mayor dinamicidad a la ejecución de los autómatas.

Este no es un trabajo concluido pues aún no se ha terminado la implementación y las pruebas.

## IX. Evaluación de la Calidad

La calidad de voz es evaluada por su inteligibilidad y su naturalidad. Además, dado que se han seleccionado casos específicos para frases que se encuentren en un entorno mayor, se puede definir una nueva perspectiva de evaluación en base a la adaptación de la voz sintética generada al contexto general.

En vista que la calidad es un concepto subjetivo, en el que el resultado de la evaluación depende de la persona que responda a la prueba planteada, se tendrán valores aproximados para la explotación de los datos que permitan obtener características generales sobre el producto desarrollado y la mejora obtenida luego de la implementación del autómata, ya que se desarrollarán pruebas que permitan al usuario escoger entre un determinado número de rangos de calidad, disminuyendo la variedad de respuestas y agrupando los resultados de acuerdo a características similares para un grupo de personas encargadas de resolver las pruebas planteadas.

Como la evaluación de la calidad será subjetiva, lo que la mayoría de personas califiquen lo que oyen como "perfecto" tendrá mayor calidad, lo que nos llevará a la síntesis de conclusiones en base a las respuestas de los evaluados.

Los métodos de evaluación planteados servirán para evaluar la calidad de la voz sintética en base a sus características generales: naturalidad e inteligibilidad. Cada una de las características mencionadas anteriormente tendrá métodos de evaluación diferentes, los cuales se definen a continuación.

La prueba MOS (Mean Opinion Score) **Erro! Fonte de referência não encontrada.** es de aplicación general para obtener índices que permitan validar la calidad de la voz sobre telefonía IP. En este caso, aplicaremos las pruebas MOS para poder obtener estos resultados relacionados a la calidad de la voz sintética de manera más ordenada y que haga posible la medición, a través de valores numéricos, de indicadores generales relacionados a las características de la voz sintética que el evaluado escucha. Con los datos obtenidos podremos obtener estadísticas más elaboradas, gracias a que con el cuestionario se pudo dejar de lado la calificación subjetiva por una

Esta prueba MOS consiste en preguntas que relacionan determinadas preguntas del cuestionario, que se puede encontrar en el anexo D, con la voz sintética que se le brinda al usuario. Como alternativas de respuesta presenta 5 niveles, que van del 1 (peor) al 5 (mejor).

La naturalidad, en el contexto de la mejora en la síntesis de números y fechas, estará relacionada con la forma en la que la voz sintética producida con la mejora pueda ser similar a la voz humana. Estas pruebas están más relacionadas con el

aspecto supra-segmental del texto, debido a que es necesario ubicar la naturalidad en un contexto general, el cual se da al desarrollar un texto completo y donde se muestren las dependencias de las frases que contengan números arábigos y fechas en el texto completo a sintetizar. Para la evaluación de esta característica se aplicará el método de evaluación MOS explicado anteriormente. Sin embargo, las preguntas a evaluar serán las relacionadas a la voz sintética generada y si esta fue de agrado del oyente.

La inteligibilidad del texto presentado al evaluado será una característica muy importante a evaluar, ya que lo que se desea es que se pueda comprender perfectamente el texto que se planteó sintetizar. Estas pruebas están más relacionadas con el ámbito segmental del texto, ya que para poder entender el mensaje es necesaria la audición correcta de los números arábigos o fechas, según corresponda, para que se comprenda específicamente la porción del texto que los representa.

Nuevamente la evaluación MOS será aplicable para esta prueba, esta vez dirigiendo las preguntas al aspecto de entendimiento de los números o fechas de manera concreta.

## X. Conclusiones

Como resultado del trabajo realizado hasta hoy, se pueden obtener las siguientes conclusiones:

*1) La síntesis de voz como medio de interacción entre el humano y el computador es cada día más utilizada por lo que las investigaciones sobre síntesis de voz están concentradas en la calidad de la voz producida para garantizar que los sintetizadores sean usados por las personas y no rechazados.*

*2) Las técnicas de autómatas adaptativos siguen siendo una de las más simples y prácticas formas para resolver aspectos de la síntesis de voz que son dependientes del contexto.*

*3) No sólo la melodía que utiliza un sintetizador de voz es importante para la naturalidad de la voz pues si un texto es mal reproducido existirá mala inteligibilidad y por consiguiente el receptor detectará que la voz es robótica al no obtener la salida esperada.*

*4) Existen distintos formatos, como los de fechas, que son de uso común pero no son parte de un estándar o simplemente son una adaptación de uno. Es importante considerar estos formatos para la creación de sintetizadores más naturales.*

## Agradecimientos

Los autores reconocen las contribuciones de la profesora Beatriz Mauchi y al profesor Jorge Pérez del Departamento de Humanidades de la Pontificia Universidad Católica del Perú, por su aporte en el aspecto lingüístico del conjunto de proyectos del cual forma parte el aporte descrito en este artículo. Asimismo, al Doctor Joao José Neto de la Universidad Politécnica de Sao Paulo (Brasil) por su colaboración respecto a los Autómatas Adaptativos, y al Doctor Joaquim Llisterri de la Universidad Autónoma de Barcelona (España) por su colaboración en el tema de métodos de evaluación de TTS.

## Referencias

[1] J.J. Neto, "Adaptive Automata for Context-Sensitive Languages", *ACM Sigplan Notices*, vol. 29, n° 9, pp. 115-124, 1994.

[2] A. Black, P. Taylor y R. Caley, *Festival Speech Synthesis System*, 1.4 ed., University of Edinburh, 2002. [Online] Available: http://www.cstr.ed.ac.uk/projects/festival/

[3] E. Da Silva and E. Muszkat, *Metodologia da Pesquisa e Elaboração de Dissertação*, 3th ed., UFSC - Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2001.

[4] D. Jurafsky and J. H. Martin, *Speech and language processing: An Introduction to Natural Language Processing*, Computational Linguistics, and Speech Recognition, 2nd ed. New York: Prentice-Hall, 2000, p.p. 30-50.

[5] Real Academia Española y Asociación de Academias de la Lengua Española, *Diccionario Panhispánico de dudas*, Madrid: Santillana, 2005.

[6] R. Caya. "Estudio de la mejora de la calidad de un Sintetizador de voz para el castellano utilizando El método de autómatas adaptativos". Trabajo de fin de carrera de la especialidad de Ingeniería Informática de la Pontificia Universidad Católica del Perú. Perú, 2009.

[7] Sapiensman, Matemáticas: Números Decimales. [En línea]. http://www.sapiensman.com/matematicas/matematicas9.htm

[8] Nadeem Unuth, Mean Opinion Score (MOS) - A Measure Of Voice Quality. [En línea]. http://voip.about.com/od/voipbasics/a/MOS.htm

[9] R. Caya, C. Zapata. "Estudio de la mejora de la calidad de voz para un sintetizador en idioma castellano usando el método de Autómatas Adaptativos" en la memoria de Workshop de Tecnologías Adaptativas. Sao Paulo, Brasil. Enero, 2009

[10] R. Caya, C. Zapata. "Results of the improvement on synthesis system's speech quality for Spanish using Adaptive Automatas", en la memoria de the Third International Conference on Advances in Computer-Human Interactions - ACHI 2010. St. Maarten, 2010.

[11] S. Lemmety, "Review of Speech Synthesis Technology", Master's thesis, supervised by M. Karjalainen, Helsinki University of Technology, University, Helsinki, March 1999.

[12] H. Pistori, "Tecnologia adaptativa en Engenharia de computação: estado da arte e aplicações", Dissertação de doutorado, orientada por J. J. Neto, Departamento de Engenharia de Comparação e Sistemas Digitais, Escola Politecnica da Universidad de Sao Paulo, 2003.

[13] International Organization for Standardization, ISO – FAQs – Date and time format. [En línea]. http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/date_and_time_format.htm

[14] Schollaris, Aritmética - Números naturales - Conceptos. [En línea]. http://schollaris.com.mx/020101nnconceptos.php

**Rosalia Edith Caya Carhuanina**(M'2010) nació en Lima, Perú, el 27 de febrero de 1986. Se recibió de Bachiller en Ciencias con mención en Ingeniería Informática de la Pontificia Universidad Católica del Perú. Obtuvo el título profesional de Ingeniero en Informática mediante trabajo de tesis. Ejerció profesionalmente en BCTS y actualmente es profesor TPA de la Pontificia Universidad Católica del Perú. Entre sus campos de interés están el procesamiento de lenguaje natural y las interfaces humano-computador.

**Claudia Zapata Del Río** (M'2007) nació en Lima, Perú, el 03 de octubre de 1978. Se recibió de Bachiller en Ciencias con mención en Ingeniería Informática de la Pontificia Universidad Católica del Perú y concluyó en la misma los estudios de Maestría en Ciencias de Computación.

Obtuvo el título profesional de Ingeniero en Informática mediante trabajo de tesis y es miembro del Colegio de Ingenieros del Perú.

Ejerció profesionalmente en Synopsis S.A. y actualmente es profesor auxiliar de la Pontificia Universidad Católica del Perú

**Ruiz Vergara, César Augusto** nació en Ica, Perú, el 26 de setiembre de 1988. Se recibió de Bachiller en Ciencias con mención en Ingeniería Informática de la Pontificia Universidad Católica del Perú.

Ejerce profesionalmente en Novatronic S.A.C y actualmente es asistente de docencia de la Pontificia Universidad Católica del Perú.

# Aprendizagem Incremental usando Tabelas Decisão Adaptativas

R. L. Stange, J. J. Neto

***Resumo* — A aprendizagem incremental requer que o mecanismo de aprendizagem seja capaz de acumular experiência dinamicamente. Os dispositivos adaptativos que possuem a capacidade de reter em suas regras informações extraídas de suas entradas, podem acumular essas informações e utilizá-las quando for necessário. Este trabalho propõe a utilização de dispositivos adaptativos para representar o conhecimento adquirido através da aprendizagem incremental. Alguns dos problemas do processo de aprendizagem em classificadores são apresentados. Um exemplo ilustrativo utilizando tabelas de decisão adaptativa, como forma de representar o conhecimento, é mostrado.**

***Palavras chaves* — Aprendizagem de Incremental, Adaptatividade, Tabela de Decisão Adaptativa, Representação do conhecimento.**

## I. Introdução

APRENDIZAGEM incremental consiste de técnicas para fornecer ao mecanismo de aprendizagem a capacidade de extrair informações adicionais a partir dos dados, sempre que surgirem mais dados, ou quando se julgar conveniente. Além disso, o conhecimento adquirido anteriormente é preservado e existe a capacidade de incorporar novos conhecimentos, possivelmente introduzidos por novos dados [1].

Em muitas aplicações, novas informações podem surgir ao longo do tempo. Por exemplo, quando um sistema de reconhecimento facial é construído é difícil recolher todas as variações de imagens de face com antecedência, pois rostos são facilmente alterados. Neste caso, pode existir a necessidade de implementar processos dinâmicos baseados em aprendizagem incremental.

As técnicas de aprendizagem incremental envolvem a adaptação gradual das estruturas de aprendizagem.

A adaptatividade é a capacidade que um dispositivo adaptativo tem de modificar a sua estrutura em resposta ao seu histórico de operações e aos dados de entrada [7].

A tecnologia adaptativa é o conjunto de ferramentas, métodos e técnicas que permitem modelar problemas práticos utilizando adaptatividade.

A tecnologia adaptativa pode ser uma solução alternativa para os problemas de aprendizagem computacional. Com a incorporação da adaptatividade é possível capturar um aspecto fundamental da aprendizagem, a adaptação dinâmica das regras de aprendizagem em função de sua interação com o ambiente [8].

O objetivo deste trabalho é apresentar uma proposta de utilização de adaptatividade em mecanismos de aprendizagem incremental. As tabelas de decisão adaptativas são utilizadas como forma de representação do conhecimento adquirido durante o processo de aprendizagem incremental.

## II. Conceitos

A aprendizagem de máquina está relacionada à correta manipulação de conhecimento prévio e a novas observações que possam levar a novos conhecimentos [10]. A manipulação de conhecimento está relacionada aos métodos de inferência. Em particular, um método de inferência lógica é a aprendizagem indutiva.

A indução é uma forma de inferência lógica que permite obter conclusões genéricas a partir de um conjunto particular de exemplos [16].

A aprendizagem indutiva pode ser dividida em aprendizagem supervisionada e não supervisionada [18].

Na aprendizagem supervisionada é fornecido ao algoritmo de aprendizado, ou indutor, um conjunto de exemplos de treinamento, para os quais o rótulo da classe é conhecido. Em geral, cada exemplo de treinamento é descrito por um conjunto de características e pelo rótulo da classe que o exemplo de treinamento pertence. Quando o rótulo da classe é uma valor discreto, esse problema é conhecido como classificação. Quando o rótulo da classe é um valor contínuo, o problema é denominado regressão [2].

Na aprendizagem não supervisionada, o algoritmo de aprendizado analisa um conjunto de exemplos, não rotulados, e forma agrupamentos ou *clusters* desse conjunto de exemplos.

Neste trabalho é considerada a aprendizagem supervisionada.

R. L. Stange é estudante de Mestrado em Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Computação e Sistemas Digitais, Av. Prof. Luciano Gualberto, travessa 3, número 158 - Cidade Universitária - São Paulo - SP - CEP: 05508-900 (e-mail: rlstange@usp.br).

J. J. Neto atua no Departamento de Engenharia de Computação e Sistemas Digitais (PCS) da Escola Politécnica (POLI) da Universidade de São Paulo (USP), Av. Prof. Luciano Gualberto, Trav. 3, N. 158 - CEP: 05508-900 - São Paulo/SP, Brasil. (e-mail: joao.jose@poli.usp.br).

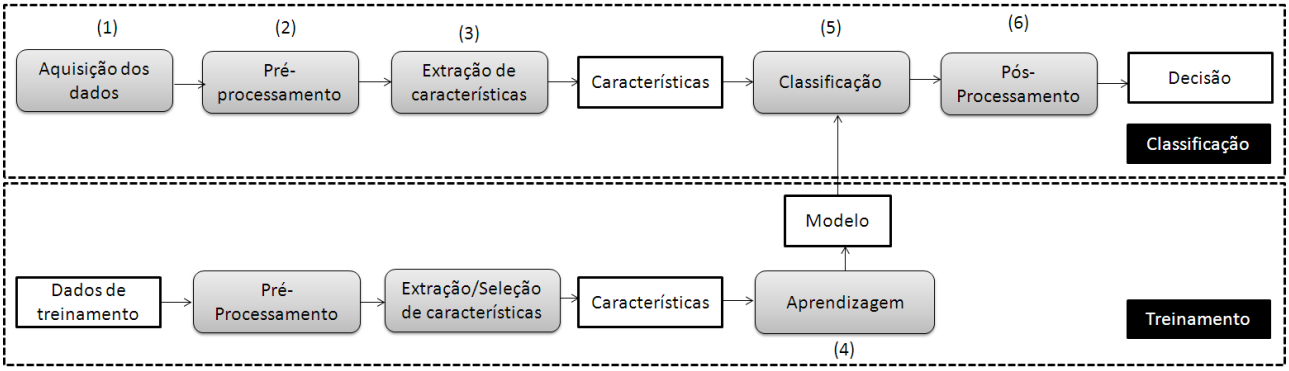



Figura 1. Processo de reconhecimento de padrão (Adaptada de [2])

Conforme a disponibilidade dos exemplos de treinamento, os algoritmos de aprendizagem podem ser classificados em incrementais e não incrementais [10].

Na aprendizagem não incremental é necessário que todos os exemplos de treinamento estejam disponíveis simultaneamente para a indução do modelo de aprendizagem do classificador. Algorimtos não incrementais são apropriados para problemas de classificação com dados de treinamentos estáticos, ou seja, quando não haverá mudanças no conjunto de treinamento.

Na aprendizagem incremental o modelo de aprendizagem é modificado, se necessário, quando novos dados são disponibilizados. A vantagem em utilizar algoritmos incrementais é que o conhecimento pode ser atualizado de forma flexível. Além disso, em alguns casos é mais eficiente revisar um modelo de aprendizagem do que construir um novo modelo, cada vez que um novo exemplo é disponibilizado [14].

O reconhecimento de padrão de acordo com [14] é a descoberta automática de regularidades em dados, através de algoritmos computacionais, e uso dessas regularidades para classificar os dados em categorias. O termo genérico "padrão" é utilizado para referir-se a esses dados ou objetos.

A Fig. 1 mostra o diagrama geral para o processo de reconhecimento de padrão na aprendizagem supervisionada. O processo de reconhecimento de padrão é dividido basicamente em treinamento e classificação. Na componente de treinamento a finalidade é inferir, a partir de um conjunto de exemplos de treinamento, um conjunto de regras que define um padrão. O resultado final do treinamento é um modelo que descreve os padrões. Na componente de classificação, o modelo é utilizado com a finalidade de classificar novos indivíduos de acordo com critérios definidos durante sua construção.

A seguir são descritas as etapas do processo de reconhecimento de padrão de acordo com [2], identificadas na Fig. 1.

(1) Aquisição de dados e sensoriamento*:* Nesta etapa são realizadas as medições das variáveis físicas dos indivíduos (eg.: câmera para capturar imagens, microfone para capturar som, etc.).

(2) Pré-processamento: A etapa de pré-processamento ocorre tanto na fase de treinamento quanto na fase de classificação. O objetivo é remover eventuais ruídos nos dados ou realizar qualquer outro tratamento sobre os dados capturados, para simplificar as operações subsequentes (eg.: ajustar distorção de imagem). No pré-processamento pode ocorrer um processo denominado segmentação. Na segmentação, os padrões devem ser separados para evitar sobreposições.

(3)

(4) Extração e seleção de características: A seleção de características é a tarefa de identificar e selecionar um subconjunto de características discriminantes, ou correlacionadas. Na extração de características novas características são geradas a partir de transformações ou combinações de características do conjunto original.

(5) Aprendizagem do modelo: Algoritmos de aprendizagem mapeiam as características entre grupos de padrões.

(6) Classificação: A classificação utiliza as características e o modelo de aprendizagem para atribuir categorias ou classes aos novos indivíduos.

(7) Pós-processamento: Após a classificação, uma avaliação de confiança nas decisões tomada pelo classificador é realizada. Na etapa de pós-processamento também podem ocorrer combinações de classificadores.

## III. Tecnologia Adaptativa

As pesquisas em adaptatividade investigam soluções para diversos problemas complexos de teoria da computação [5], de aprendizagem de máquina [8][9], de tomada de decisão [15], de engenharia da computação [8], entre outros.

### *A. Dispositivos Adaptativos*

O modelo geral para um dispositivo adaptativo é definido como um conjunto finito de regras que pode sofrer modificações dinamicamente [6].

Um dispositivo dirigido por regras tem seu comportamento completo descrito na forma de um conjunto finito de regras. As regras usualmente são expressas como cláusulas condicionais do tipo "*Se* $C_i$ *então* $A_i$".

O conjunto de regras $R = \{r_1, r_2, ... r_k\}$, onde k é o número de regras, determina o funcionamento do dispositivo adaptativo, o qual muda sucessivamente de configuração em resposta a estímulos externos de entrada. Cada regra $r_i$ , para $0<i\leq k$ pode ser representada no formato $r_i = (C_i, A_i)$, onde C é conjunto de condições $C = \{condição_1 , condição_1, ..., condição_n\}$, onde $n \geq 1$ é número de condições e A é o conjunto de ações $A = \{ação_1, ação_2, .., ação_m\}$, onde $m \geq 1$ é o número de ações.

Uma formulação para um dispositivo guiado por regras não adaptativo encontra-se resumida na Tabela 1, e sua a formulação completa, em [6].

Um dispositivo adaptativo AD é dito com adaptatividade básica quando reduze reduzido a um dispositivo não adaptativo subjacente ND, acoplado a um mecanismo adaptativo AM.

O mecanismo adaptativo incorpora um conjunto de funções adaptativas, as quais são acopladas às regras do dispositivo subjacente e, são capazes de alterar convenientemente o seu próprio conjunto das regras provocando uma mudança de comportamento no dispositivo AD.

TABELA 1
FORMULAÇÃO DE UM DISPOSITIVO GUIADO POR REGRAS.

| FORMULAÇÃO - DISPOSITIVO NÃO ADAPTATIVO | |
|---|---|
| ND = (C, R, S, $c_0$, A) | |
| C | Conjunto de todas as possíveis configurações do dispositivo; |
| R | Relação de mudança de configuração, $R \subseteq C \times (S \cup \{\varepsilon\}) \times C$. |
| S | Conjunto de estímulos de entrada do dispositivo; |
| $c_0$ | Configuração inicial única do dispositivo, pertence ao conjunto C |
| A | Conjunto das configurações de aceitação do dispositivo; |

Um dispositivo AD pode ser descrito como uma sêxtupla, conforme a formulação apresentada na Tabela 2.

As funções adaptativas são descritas basicamente por um cabeçalho e um corpo, similar às funções em linguagens de programação, conforme Tabela 3.

As funções adaptativas, em seu núcleo, referem-se a operações básicas de edição do conjunto de regras que definem o dispositivo adaptativo, representadas pelas ações adaptativas elementares de consulta, remoção e inserção de regra. As ações adaptativas elementares são apresentadas na Tabela 4.

Têm sido alvos frequentes de pesquisa os seguintes dispositivos adaptativos: os autômatos adaptativos [7], as gramáticas adaptativas [4], as árvores de decisão adaptativas [11] e as tabelas de decisão adaptativa [6] [15].

TABELA 2
FORMULACÃO DE UM DISPOSITIVO ADAPTATIVO.

| FORMULAÇÃO - DISPOSITIVO ADAPTATIVO BÁSICO | |
|---|---|
| AD = (C, RA, S, AA, $c_0$, $RA_0$, A) | |
| C | Conjunto de todas as possíveis configurações do dispositivo adaptativo; |
| RA | Conjunto de todas as regras adaptativas do dispositivo adaptativo; |
| S | Conjunto de estímulos de entrada do dispositivo adaptativo; |
| AA | Conjunto de funções adaptativas φ, incluindo a função nula λ; |
| $c_0$ | Configuração inicial $c_0 \in C$ do dispositivo adaptativo, que deve ser única; |
| $RA_0$ | Conjunto inicial de regras adaptativas do dispositivo adaptativo; |
| A | Conjunto de todas as configurações de aceitação do dispositivo adaptativo; |

TABELA 3
DECLARAÇÃO DAS FUNÇÕES ADAPTATIVAS.

| DECLARAÇÃO - FUNÇÃO ADAPTATIVA φ | |
|---|---|
| CABEÇALHO | |
| Nome φ | Uma função adaptativa é referenciada por um nome; |
| Parâmetros | Ênupla ordenada ($\rho_1$, $\rho_2$, ..., $\rho_p$), de p≥0 parâmetros formais; |
| Variáveis | Um conjunto {$\upsilon_1$, $\upsilon_2$, ..., $\upsilon_v$}, de v ≥ 0 variáveis. Os valores das variáveis são preenchidos uma única vez pelas ações adaptativas de consulta; |
| Geradores | Um conjunto {$\chi_1$, $\chi_2$, ..., $\chi_g$}, de g ≥ 0 geradores. Os valores dos geradores são preenchidos com novos valores a cada chamada da função adaptativa. |
| CORPO (βμ, Δ, αν) | |
| Função adaptativa anterior | Chamada βμ de uma função adaptativa anterior (μ) executada antes da aplicação das ações adaptativas elementares definidas no conjunto Δ. |
| Função adaptativa posterior | Chamada αν de uma função adaptativa posterior (ν) executada após a aplicação das ações adaptativas elementares definidas no conjunto Δ. |
| Núcleo da função adaptativa | Núcleo da função adaptativa φ consiste do conjunto $\Delta = \{\delta_1, \delta_2, ..., \delta_e\}$, contendo e ≥ 0 ações adaptativas elementares $\delta_i$, para $0 \le i < e$. |

TABELA 4
AÇÕES ADAPTATIVAS ELEMENTARES.

| AÇÕES ADAPTATIVAS ELEMENTARES $\delta_i$ | |
|---|---|
| $\delta_i = \otimes$ [ r ] | |
| ⊗ | Representa um operador aplicado à regra r e ao conjunto $RA_t$ de regras que definem o comportamento do dispositivo adaptativo AD. |
| $RA_t$ | Conjunto RA no passo t de operação do dispositivo AD. |
| r | Padrão para a regra sobre a qual deve incidir a ação adaptativa elementar. |
| OPERADORES ⊗ | |
| Inserção + | Inclui no conjunto $RA_t$ uma nova regra r. |
| Remoção - | Elimina do conjunto $RA_t$ a regra r. |
| Consulta ? | Pesquisa em $RA_t$ o padrão r e preenche variáveis com os resultados da busca. |

### *B. Tabelas de Decisão Adaptativas*

Nas tabelas de decisão adaptativas o dispositivo subjacente é uma tabela de decisão convencional. Uma tabela de decisão convencional é composta por colunas que representam conjuntos de regras associadas a condições e ações [3]. A primeira coluna, partindo da primeira linha da tabela, representa um conjunto de condições e, a seguir, encontra-se um conjunto de ações. A tabela de decisão opera verificando as condições de acordo com os valores definidos nas colunas de regras. Quando a condição é satisfeita de acordo com uma determinada regra, essa regra é considerada válida e todas as ações a ela associadas são executadas.

A formulação da tabela de decisão é apresentada na Tabela 5.

TABELA 5
FORMULAÇÃO DA TABELA DE DECISÃO CONVENCIONAL.

| FORMULAÇÃO – TABELA DE DECISÃO CONVENCIONAL | |
|---|---|
| | TD = (CT, R, CV, $t_0$, AT, A) |
| CT | Conjunto de todas as configurações possíveis da tabela de decisão; |
| R | Conjunto finito de regras de decisão: R = {$r_i$, 1≤i≤ n}. Cada regra $r_i$ = ($c_{ij}$, $a_{ik}$) ∈ R, onde: $c_{ij}$: representa um valor para a condição $c_i$ na regra $r_i$; $a_{ik}$ e representa um valor para a alternativa ou ação $a_k$ para a regra $r_i$; |
| CV | Conjunto finito dos valores $c_{ij}$ válidos para as condições $C_i$ |
| $t_0$ | Configuração inicial da tabela de decisão; |
| AT | Conjunto de configurações aceitas da tabela de decisão; |
| A | Conjunto finito de ações; |

Uma versão adaptativa de uma tabela de decisão convencional pode ser obtida adicionando-se a ela linhas, onde são incluídas as funções adaptativas. Além disso, a cada coluna que representa uma regra simples, deve ser adicionada uma chamada para uma função adaptativa associada à execução de uma regra em particular. Com isso, sempre que uma regra adaptativa é aplicada, uma ação adaptativa é invocada, permitindo que sejam feitas mudanças no conjunto de regras.

Uma tabela de decisão adaptativa ou TDA, cujo formalismo está apresentado em [6][15], é um dispositivo guiado por regras que permite modificações dinâmicas no conjunto de regras que compõe a tabela de decisões, através de ações adaptativas. As tabelas de decisão adaptativas são formadas por um conjunto de condições, ações, regras e funções adaptativas. A estrutura geral gráfica de uma tabela de decisão adaptativa é apresentada na Tabela 6, baseada no formato descrito em [6] e na tabela de decisão convencional descrita em [3].

Na Tabela 6, as linhas que compõem a tabela de decisão TD correspondem ao dispositivo não adaptativo subjacente ND. O mecanismo adaptativo é obtido acrescentando uma camada adaptativa no dispositivo ND. A camada adaptativa composta por um conjunto de linhas para a definição das funções adaptativas.

Os elementos que compõe a tabela de decisão adaptativa são descritos a seguir:

*1. Linha Cabeçalhos →(Tags):* especifica o cabeçalho das colunas. O título de cada coluna pode ser:

a. "H" que representa o cabeçalho da função adaptativa;
b. "?", "-" ou "+" que indicam respectivamente uma ação elementar de consulta, remoção e inserção;
c. "S", "E" ou "R" que indicam respectivamente uma regra delimitadora inicial, final ou uma regra da tabela de decisão convencional.

*2. Linhas das Condições:* cada linha de condição (condição$_1$,..., condição$_n$) corresponde a um rótulo de condição;

*3. Valores das condições:* em suas células, são indicados os valores das condições ($c_{11}$,...$c_{kn}$). Cada valor pode corresponder a entradas limitadas ("S", "N" ou branco) ou a entradas estendidas;

*4. Linhas das Ações:* cada linha representa uma ação (ação$_1$,...,ação$_m$), que pode ser executado em resposta ao conjunto de condições;

*5. Valores das Ações:* devem ser marcadas as células das ações ($a_{11}$, ...,$a_{kn}$) que serão executadas quando as regras são avaliadas;

*6. Funções adaptativas:* as linhas nome da função adaptativa (Nome φ), parâmetros ($\rho_1$,... $\rho_p$), variáveis ($\nu_1$,..., $\nu_v$) e geradores ($\chi_1$,..., $\chi_g$) são definidas, convenientemente;

*7. Chamadas às ações adaptativas:* em suas células assinalam-se aquelas que serão chamadas antes ou depois da aplicação da regra.

Na execução de uma regra em uma TDA, primeiramente são verificadas as regras não adaptativas e, se uma única delas se aplica, as ações correspondentes são executadas. Caso mais de uma regra não adaptativa satisfaça a condição, as ações correspondentes às mesmas devem ser aplicadas em paralelo, como previamente definido para tabelas de decisão convencionais. Porém, se nenhuma regra não adaptativa satisfizer a condição, trata-se de uma condição não prevista e, no caso de uma tabela de decisão convencional, não haveria como prosseguir.

TABELA 6
TABELA DE DECISÃO ADAPTATIVA.

| | | | | $r_1$ | $r_2$ | ... | $r_k$ |
|---|---|---|---|---|---|---|---|
| | Cabeçalhos (Tags)→ | | (H,+, -,?) | S | R | R | E |
| Tabela de decisão TD | Condições | condição$_1$ | Declaração das funções | $c_{11}$ | $c_{21}$ | ... | $c_{k1}$ |
| | | ... | | ... | ... | ... | ... |
| | | condição$_n$ | | $c_{1n}$ | $c_{2n}$ | ... | $c_{kn}$ |
| | Ações | ação$_1$ | | $a_{11}$ | $a_{21}$ | ... | $a_{k1}$ |
| | | ... | | ... | ... | ... | ... |
| | | ação$_m$ | | $a_{1m}$ | $a_{2m}$ | ... | $a_{kn}$ |
| Funções Adaptativas | Anterior | Nome φ | | Chamadas às ações adaptativas | | | |
| | Posterior | Nome φ | | | | | |
| | Parâmetros | $\rho_1$ | | | | | |
| | | ... | | | | | |
| | | $\rho_p$ | | | | | |
| | Variáveis | $\upsilon_1$ | | | | | |
| | | ... | | | | | |
| | | $\upsilon_v$ | | | | | |
| | Geradores | $\chi_1$ | | | | | |
| | | ... | | | | | |
| | | $\chi_g$ | | | | | |

## IV. TABELAS DE DECISÃO ADAPTATIVAS EM PROBLEMAS DE RECONHECIMENTO DE PADRÕES.

Neste trabalho, o modelo de aprendizagem é representado por um conjunto de regras do tipo "*Se...então*". O conjunto de regras é utilizado para classificar novos indivíduos. A Fig. 2 mostra o processo de aprendizagem incremental baseado em regras, onde $D_1$, $D_2$,...,$D_t$ são novos dados disponíveis em diferentes instantes no tempo. A base de regras R armazena as regras do modelo.

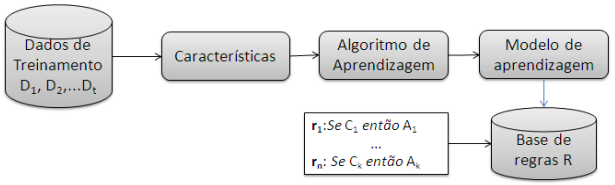



Figura 2. Modelo de aprendizagem incremental baseado em regras.

Na apredizagem incremental utilizando adaptatividade, as regras de classificação podem ser alteradas, em resposta à estímulos de entrada, tais como quando novos exemplos de treinamento estão disponíveis. A adaptatividade é incorporada na base de regras através das regras adaptativas, que quando aplicadas alteram o modelo de aprendizagem ou base de regras.

A seguir, apresenta-se um estudo de caso, para explorar a utilização das tabelas de decisão adaptativas, na resolução de problemas de classificação.

Considere-se o conjunto de atributos A = {$\alpha_1$, $\alpha_2$, $\alpha_3$, $\alpha_4$, $\alpha_5$, $\alpha_6$, $\alpha_7$}, onde:

- $\alpha_1$ = "temperatura corporal", com temperatura corporal = {quente, frio};
- $\alpha_2$ = "cobertura da pele", com cobertura da pele = {cabelos, escamas, pelos, penas, espinhos, nada};
- $\alpha_3$ = "ovíparo", com ovíparo = {sim, não};
- $\alpha_4$ = "criatura aquática", com criatura aquática = {sim, não, semi};
- $\alpha_5$ = "criatura aérea", com criatura aérea = {sim, não};
- $\alpha_6$= "possui pernas", com possui pernas = {sim, não};
- $\alpha_7$ = "hiberna", com hiberna = {sim, não}.

Considere-se o conjunto de classes distintas Ω = {$\omega_1$, $\omega_2$, $\omega_3$, $\omega_4$, $\omega_5$}, onde:

- $\omega_1$ = "mamífero";
- $\omega_2$ = "pássaro";
- $\omega_3$ = "peixe";
- $\omega_4$ = "réptil";
- $\omega_5$ = "anfíbio".

Dado o conjunto de dados de treinamento de vertebrados D com amostras rotuladas, conforme Tabela 7.

TABELA 7
DADOS DE TREINAMENTO [13]

| Animal | Temperatura Corporal | Cobertura da pele | Ovíparo | Criatura Aquática | Criatura Aérea | Possui Pernas | Hiberna | Classe |
|---|---|---|---|---|---|---|---|---|
| Humano | Quente | Cabelos | Não | Não | Não | Sim | Não | Mamífero |
| Cobra | Fria | Escamas | Sim | Não | Não | Não | Sim | Réptil |
| Salmão | Fria | Escamas | Sim | Sim | Não | Não | Não | Peixe |
| Baleia | Quente | Cabelos | Não | Sim | Não | Não | Não | Mamífero |
| Sapo | Fria | Não possui | Sim | Semi | Não | Sim | Sim | Anfíbio |
| Morcego | Quente | Cabelo | Não | Não | Sim | Sim | Sim | Mamífero |
| Pomba | Quente | Penas | Sim | Não | Sim | Sim | Não | Pássaro |
| Gato | Quente | Pelos | Não | Não | Não | Sim | Não | Mamífero |
| Tartaruga | Fria | Escamas | Sim | Semi | Não | Sim | Não | Réptil |
| Pinguim | Quente | Penas | Sim | Semi | Não | Sim | Não | Pássaro |
| Porco Espinho | Quente | Espinhos | Não | Não | Não | Sim | Sim | Mamífero |
| Enguia | Fria | Escamas | Sim | Sim | Não | Não | Não | Peixe |
| Salamandra | Fria | Não possui | Sim | Semi | Não | Sim | Sim | Anfíbio |

Considere que $\mu_j$ é um método de indução de árvore. Esse método é implementado pelo algoritmo ID3 [11]. A saída do algoritmo de aprendizagem é um classificador τ, neste caso, representado por um dispositivo do tipo árvore de decisão. Dado um novo padrão χ, o classificador deve predizer o valor correspondente da classe Ω associada ao padrão.

Com a finalidade de melhorar a legibilidade as informações contidas na árvore, são transformadas em um conjunto de regras do tipo *"Se...então"* e codificadas em uma tabela de decisão convencional para representar o classificador. O resultado é o conjunto de regras R = {$r_1$,$r_2$,...$r_k$}, onde k é o número de regras. R é descrito na Tabela 8.

As tabelas de decisão, assim como as árvores de decisões ou redes neurais, são dispositivos capazes de representação do conhecimento para o reconhecimento de padrões.

Conforme a Fig. 1, na fase de avaliação do processo de aprendizagem, os dados de testes são submetidos ao classificador. Como resultado, quatro casos resultantes são possíveis.

TABELA 8
REGRAS DE CLASSIFICAÇÃO NA TABELA DE DECISÃO.

| | Regras → | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ |
|---|---|---|---|---|---|---|---|
| Condições | Temperatura corporal | fria | - | - | - | - | - |
| | Cobertura da pele | - | cabelo | escamas | - | Não possui | - |
| | Ovíparo | - | - | - | sim | - | - |
| | Criatura aquática | sim | não | não | não | - | - |
| | Criatura aérea | - | - | - | - | - | - |
| | Possui pernas | - | - | - | - | - | - |
| | Hiberna | - | - | - | - | - | - |
| Ações | Classe | peixe | mamífero | réptil | pássaro | anfíbio | - |

*Caso 1:* Apenas uma regra do conjunto R é aplicável ao dado de entrada e o classificador rotula o padrão corretamente. Por exemplo, para χ = {fria, escamas, não, sim, não, não, não} a regra aplicável é $r_1$. Neste caso, o padrão é classificado como peixe. O exemplo χ é um salmão, portanto, a classificação está correta.

*Caso 2:* Nenhuma regra do conjunto R é aplicável. Isso significa que o classificador não é capaz de classificar χ. Os motivos podem ser diversos, tais como: χ não é um padrão, neste caso, χ deve ser descartado; χ é um padrão, porém o classificador não aprendeu as regras necessárias para classificá-lo, neste caso, este problema ocorreu na fase de aprendizagem e os motivos podem ser: o método de aprendizagem não é adequado, por exemplo, os atributos selecionados pelo método para discriminar as classes são insuficientes, ou não são bons discriminantes; ocorrência de problemas nas amostras de treinamento (dados ruidosos, dados ausentes, quantidade de dados insuficiente); χ é um novo padrão que deverá ser aprendido, a *priori*, não pertencia ao conjunto de treinamento D.

*Caso 3:* Existe mais de uma regra aplicável no conjunto R. Trata-se, portanto, de uma situação não determinística. Isso pode ocorrer quando os dados de treinamento são ambíguos ou quando o conjunto de treinamento é pequeno, entre outros motivos.

*Caso 4:* Existe uma regra aplicável no conjunto R, porém o classificador está rotulando os padrões de maneira incorreta. Isso pode ocorrer por várias razões, entre as quais: o método de aprendizagem é inadequado para as características dos dados de treinamento; os dados de treinamento são insuficientes para uma boa taxa de aprendizagem; ocorreu *overfitting*[2][2].

A seguir, é ilustrado um exemplo de utilização das tabelas de decisão adaptativas para o Caso 2.

Considere $\chi$ = {quente, pelos, não, não, não, sim, não}. Neste caso, não existe uma regra aplicável. Porém, é sabido que $\chi$ é um exemplo de gato e deveria ser classificado como mamífero. O algoritmo ID3 não foi capaz de aprender a regra que classifica um gato como mamífero.

Neste exemplo, utilizando uma tabela de decisão adaptativa uma regra adaptativa é adicionada ao conjunto de regras R da tabela de decisão. Essa regra é satisfeita toda vez que nenhuma regra não adaptativa é aplicável. A regra $r_6$ é uma regra adaptativa e possui a seguinte função adaptativa associada:

Exemplo do pseudocódigo para declaração de uma função adaptativa.

```
Função Adaptativa f1(Parâmetros: p1, p2,
p3, p4, p5, p6, p7){
Variáveis: v1, v2, v3, v4, v5, v6, v7
Geradores: Geradores de rótulos *gt
Ações elementares Δ {
    δ1:?[v1,v2,v3,v4,v5,v6,v7](Existe
    uma regra na área temporária que
    satisfaça as condições χ).
    δ2:-[v1,v2,v3,v4,v5,v6,v7](Se
     existir, remover).
    δ3:+[p1,p2,p3,p4,p5,p6,p7,*g1](Inse
     rir uma regra rt)
    .}
 }.
```

Para fins didáticos, o conjunto □□ é divido em $R_p$, o conjunto de regras principais do dispositivo TDA e $R_t$, conjunto de regras temporárias do dispositivo TDA. Assim, R = $R_p \cup R_t$.

No caso de aplicação da regra $r_6$ em $\chi$, a ação elementar de consulta irá pesquisar se existe uma regra em $R_t$ que satisfaça $\chi$. Se existir a ação elementar de remoção irá removê-la. Em seguida a ação elementar de inserção inclui uma regra $r_{ti}$, para i = 1, 2 ,... n, onde n é o número de regras temporárias. Por exemplo, ao receber as entradas $\chi_1$ = {quente, pelos, não, não, não, sim, não}, $\chi_2$={quente, pelos, não, não, não, sim, não} e $\chi_3$ = {quente, penas, sim, semi, não, sim, não}, a regra $r_6$ é aplicada e o conjunto de regras $R_t$ é modificado.

A Tabela 9 mostra a TDA após sucessivas transformações. As transformações do dispositivo são representadas por $TDA_j$ para j = 0, 1, 2,..., n, onde n é o número de modificações do dispositivo.

Posteriormente, as regras temporárias deverão ser utilizadas para inferir novas regras de classificação e assim aprender futuramente a classificar corretamente novos padrões não cobertos pelas regras iniciais.

É importante destacar que no exemplo da Tabela 2, apenas uma regra adaptativa foi inserida, porém é possível incluir outras regras adaptativas com funções adaptativas capazes de realizar diferentes modificações na tabela de decisão adaptativa.

## V. Considerações Finais

A área de aprendizagem de máquina tem-se mostrado uma rica fonte de pesquisa para a exploração prática das aplicações dos fundamentos da tecnologia adaptativa.

Em problemas de classificação diversas soluções são propostas na tentativa de construir classificadores eficientes, na maioria dos casos, as melhorias ocorrem nas fases de treinamento e validação. Porém, muitos problemas só ocorrem na fase de classificação, quando os padrões a serem rotulados são submetidos ao classificador.

A aprendizagem incremental é uma solução desejável para adaptar gradualmente o modelo aprendido. A adaptatividade pode ser útil para detectar os problemas que podem ocorrer na fase de classificação e modificar o modelo, de maneira incremental. Embora existam maneiras de refazer a estrutura das tabelas de decisão após a obtenção de novos dados, em alguns casos isso pode ser ineficiente, podendo levar à perda, parcial ou total, de informações que haviam sido anteriormente aprendidas.

No contexto da aprendizagem incremental e mecanização da aprendizagem, o exemplo ilustrativo da seção IV mostra que a utilização de dispositivos adaptativos em classificadores é conveniente, pois sua operação poder ser descrita de forma incremental.

O equilíbrio entre a obtenção de modelos de aprendizagem compreensíveis e expressivos é adquirido com a utilização das tabelas de decisão adaptativas. A utilização da tabela de decisão como dispositivo subjacente para representar os modelos de classificação facilita o entendimento do modelo. Muitos modelos de classificação são como caixas pretas, as tabelas de decisão têm uma estrutura simples e compreensível, o que transforma os classificadores, vistos como caixas pretas, em ferramentas úteis para a descoberta do conhecimento.

A proposta do modelo de aprendizagem incremental baseado em regras adaptativas mostra as tabelas de decisão adaptativas como uma proposta atraente para mecanizar o processo de aprendizagem incremental em classificadores. Também permite estabelecer de forma satisfatória o domínio do problema, bem como identificar melhorias importantes para o processo de aprendizagem. Uma das vantagens da utilização dos dispositivos adaptativos é a facilidade de uso e simplicidade na aprendizagem incremental.

A grande vantagem da adaptatividade sobre muitas outras técnicas correntemente utilizadas para a formulação de modelos de representação e de manipulação do conhecimento

[2] *Overfitting*: aprendizagem superajustada ao conjunto de treinamento.

reside no fato de que: (a) a informação total, encerrada no dispositivo adaptativo, está representada integralmente no conjunto de regras; (b) a aprendizagem angariada em cada passo adaptativo do dispositivo encontra-se integralmente confinada à variação sofrida pelo conjunto de regras; (c) a observação, a identificação, a qualificação e a quantificação da evolução da aprendizagem, bem como a relação de causa e efeito entre a sequência de entradas processadas, a das regras aplicadas e a das variações sofridas pelo conjunto de regras, tudo isso pode ser efetuado única e exclusivamente pela análise do conjunto de regras do dispositivo adaptativo e das suas variações ao longo da operação do dispositivo.

TABELA 9
TABELA DE DECISÃO ADAPTATIVA APÓS 3 MODIFICAÇÕES $TDA_3$.

| Cabeçalhos (Tags) | | | H | ? | - | + | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_{t1}$ | $r_{t2}$ | $r_{t3}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Condições | Temperatura corporal | | | $p_1$ | $v_1$ | $v_1$ | fria | - | - | - | - | - | quente | quente | quente |
| | Cobertura da pele | | | $p_2$ | $v_2$ | $v_2$ | - | cabelo | escamas | - | Não possui | - | pelos | espinhos | penas |
| | Ovíparo | | | $p_3$ | $v_3$ | $v_3$ | - | - | - | sim | - | - | não | não | sim |
| | Criatura aérea | | | $p_4$ | $v_4$ | $v_4$ | - | - | - | - | - | - | não | não | semi |
| | Criatura aquática | | | $p_5$ | $v_5$ | $v_5$ | sim | não | não | não | - | - | não | não | não |
| | Possui pernas | | | $p_6$ | $v_6$ | $v_6$ | - | - | - | - | - | - | sim | sim | sim |
| | Hiberna | | | $p_7$ | $v_7$ | $v_7$ | - | - | - | - | - | - | não | sim | não |
| Ações | Classe | | | | | $*g_1$ | peixe | mamífero | réptil | pássaro | anfíbio | ? | $*g_1$ | $*g_2$ | $*g_3$ |
| Funções adaptativas | Anterior | $f_1$ | B | √ | √ | √ | | | | | | √ | | | |
| | Posterior | | | | | | | | | | | | | | |
| | Parâmetros | $p_1$ | P | | | | | | | | | $p_1$ | | | |
| | | ... | ... | | | | | | | | | ... | | | |
| | | $p_7$ | P | | | | | | | | | $p_7$ | | | |
| | Variáveis | $v_1$ | V | | | | | | | | | | | | |
| | | ... | ... | | | | | | | | | | | | |
| | | $v_7$ | V | | | | | | | | | | | | |
| | Geradores | $g_1$ | G | | | | | | | | | | | | |

Em trabalhos futuros, deseja-se incluir no modelo uma memória para armazenar as operações na base de regras adaptativas. Essa memória representa o histórico das operações e as informações contidas na memória, que podem ser utilizadas futuramente para melhorar o modelo de aprendizagem através da experiência. Outros experimentos devem ser realizados para comparar os resultados com modelos de aprendizagem incremental que não utilizam adaptatividade, tais como o C4.5 [12]. Outras funções adaptativas devem ser declaradas e associadas às regras para tratar de outros problemas de classificação tais como dados de entradas ausentes ou com ruídos.

## REFERÊNCIAS


[1] CAVALIN, P.R., SABOURIN, R., SUEN, C.Y., BRITTO JR., A.S.: "Evaluation of Incremental Learning Algorithms for An HMM-Based Handwritten Isolated Digits Recognizer". In: The 11th International Conference on Frontiers in Handwriting Recognition (ICFHR 2008), Montreal, 19-21, 2008.

[2] DUDA, R. O.; HART, P. E.; STORK, D. G.: *Pattern Classification*. 2ª Edição. Wiley, 2001. ISBN: 0471056693.

[3] HUGHES, M. L., SHANK, R. M., STEIN, E. S.: *Decision Tables*. Midi Publications, Management Development Institute, Divisions of Information, Ind"ustries, Inc., Wayne, Pennsylvania, 1968.
[4] IWAI, M. K.: "Um formalismo gramatical adaptativo para linguagens dependentes de contexto". Tese (Doutorado), EPUSP. São Paulo, 2000.
[5] NETO, J. J.: "Solving complex problems with Adaptive Automata". Lecture Notes in Computer Science. S. Yu, A. Paun (Eds.): Implementation and Application of Automata, CIAA 2000, Vol.2088, London, Canada, Springer-Verlag, pp.340, 2000.
[6] NETO, J. J. "Adaptive Rule-Driven Devices – General Formulation and Case Study". Revista de Engenharia de Computação e Sistemas Digitais, São Paulo, v.1, n.1, 2001.
[7] NETO, J. J.: "Contribuição à metodologia de construção de compiladores". São Paulo, 272p. Tese (Livre-Docência), Escola Politécnica, Universidade de São Paulo, 1993.
[8] PISTORI, H. "Tecnologia Adaptativa em Engenharia de Computação: estado da arte e aplicações". Tese (Doutorado), Escola Politécnica da USP, 2003.
[9] PISTORI, H.; NETO, J. J.: "Decision Tree Induction using Adaptive FSA". CLEI Eletronic Journal. Volume 6, Number 1, 2003
[10] PRATI, R. C.: "Novas abordagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos". Tese, ICMC-USP, 2006.
[11] QUINLAN, J. R.: "Induction of decision trees". Machine Learning, 1, 81-106, 1986
[12] QUINLAN, J. R.: "C4.5: Programs for Machine Learning". Morgan-Kaufmann, San Francisco, 1993.
[13] TAN, P. STEINBACH, M.; KUMAR, V.: *Introduction to Data Mining*, Boston, MA, USA Addison-Wesley Longman Publishing Co., Inc., 2005.
[14] THEODORIDIS, S.; KOUTROUMBAS, K.: *Pattern Recognition*. 3º Edição. Academic Press, 2006.
[15] TCHEMRA, A. H.: "Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério". Tese (Doutorado), EPUSP, São Paulo, 2009.
[16] RUSSEL, S. J.; NORVIG, P.: *Artificial Intelligence: A Modern Approach*. 2º Edição. Prentice-Hall, 2002. ISBN - 10: 0137903952

**Renata Luiza Stange** é mestranda em Engenharia Elétrica (Área de concentração: Computação e Sistemas Digitais) pela Escola Politécnica da Universidade de São Paulo. Especialista em Administração de Sistemas de Informação pela Universidade Federal de Lavras e Bacharel em Análise de Sistemas pela Universidade Estadual do Centro-Oeste do Paraná. Atualmente é professora da Universidade Federal Tecnológica do Paraná.

**João José Neto** é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo (EPUSP) e coordena o LTA – Laboratório de Linguagens e Tecnologia Adaptativa do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

# Técnica de Controle Flexível para o Projeto e Desenvolvimento de Equipamentos Automáticos Programáveis

Alexandre dos Santos Roque, Alexandre Dias da Silva
*Programa de Pós-Graduação em Engenharia de Produção, Núcleo de Automação e Projetos de Fabricação - NAFA, Universidade Federal de Santa Maria – UFSM, Santa Maria, RS, Brasil.*

***Resumo*—A parametrização de configurações de equipamentos diversifica os modos de controle com menores custos e agilidade nas fases de projeto. Este trabalho propõe uma técnica de controle adaptativo/configurável, que possibilita o projeto e controle de equipamentos que são amplamente usados em aplicações dedicadas nas indústrias. A técnica é apresentada por meio de fluxogramas e detalhada com o auxílio de diagramas comportamentais da linguagem UML (Linguagem de Modelagem Unificada) para facilitar aplicações futuras. Os resultados são o desenvolvimento de um conjunto *software/hardware* de controle, com características parametrizáveis. A comunicação entre computador e equipamento realizada pelo protocolo USB, aumenta a portabilidade do sistema de controle e permite o acionamento de equipamentos por computadores pessoais.**

***Palavras Chave:* Sistemas de controle (*control systems*), tecnologia adaptativa (*adaptive technology*), sistemas reconfiguráveis (*reconfigurable systems*), equipamentos automáticos programáveis (*programmable automatic machines*).**

## I. INTRODUÇÃO

S sistemas de controle (*hardware e software*) devem ser cada Ovez mais susceptíveis a adaptações ou configuráveis as necessidades de controle dos projetos. A agilidade e flexibilidade são conceitos reconhecidos como requisitos para os sistemas de controle aplicados em ambiente industrial [1]. Os equipamentos programáveis que fazem uso de motores de passo como meio de acionamento são os que mais se beneficiam de tais conceitos, pois, permitem um controle simplificado e de fácil compreensão, aliando movimentos precisos com o seu baixo custo. Freqüentemente, pesquisadores desenvolvem ou projetam equipamentos de pequeno porte, acionados por motores de passo, com os mais variados objetivos e aplicados em diferentes áreas [2], [3].

Os diversos equipamentos programáveis projetados possuem particularidades que muitas vezes não são atendidas em *softwares* comerciais, pois estes são sistemas fechados, e não permitem ao usuário realizar modificações ou adaptações, fato que faz com que os equipamentos e projetos se adaptem aos *softwares* de controle e não o contrário.

O controle de equipamentos com objetivos dedicados tomam como referência exemplos e características consolidadas na indústria, dentre os quais, as máquinas de comando numérico computadorizado – CNC se destacam, por possuírem características parametrizáveis em seu sistema de controle para se adaptar a um projeto ou processo de fabricação [4]. Nesse sentido, em muitos projetos existem características que podem ser benéficas ao controle de diversos equipamentos e configuradas de acordo com suas aplicações.

Outro aspecto importante para o controle de equipamentos diz respeito ao modo de funcionamento do seu sistema de controle. O sistema de controle que estabelece uma relação comparativa entre saída e entrada de referência, utilizando a diferença como meio de controle, caracteriza-se como sistema de controle com realimentação ou controle em malha fechada. De outra forma, o sistema de controle onde a saída não exerce ação sobre a entrada, ou seja, não é usada como referência para uma nova entrada, é denominado sistema de controle em malha aberta [5].

Dentre as características ou funcionalidades que um sistema de controle pode possuir, a forma de comunicação entre o equipamento e o computador é um fator crucial, pois, a evolução das portas de comunicação propiciou ao longo do tempo maior velocidade e confiabilidade na transmissão de dados, provendo portabilidade a diferentes equipamentos. Por ser um padrão de comunicação reconhecido e popularizado, o protocolo USB (*Universal Serial Bus*) constitui um diferencial para o desenvolvimento de uma técnica de controle flexível.

De acordo com [6], o desenvolvimento de uma tecnologia própria gera flexibilidade para a pesquisa, de modo que, diminui as limitações ou restrições impostas por um equipamento ou software comercial, sendo possível realizar alterações de acordo com as necessidades encontradas no decorrer do projeto ou do seu uso em campo.

Desta forma, este trabalho apresenta uma metodologia para o acionamento de equipamentos programáveis por meio da

---

Este trabalho consolida os resultados de pesquisa recentemente realizada no Núcleo de Automação e Projetos de Fabricação – NAFA, da Universidade Federal de Santa Maria, sendo parte integrante de dissertação de mestrado focada em Automação e Informática Industrial.

Alexandre dos Santos Roque possui graduação em Ciência da Computação, e concluiu recentemente Mestrado em Engenharia de Produção com Ênfase em Automação e Informática Industrial, pela Universidade Federal de Santa Maria - UFSM.
Alexandre Dias da Silva é Doutor em Engenharia Mecânica e Professor Adjunto da Universidade Federal de Santa Maria - UFSM.

técnica de controle em malha aberta, com comunicação através do protocolo USB. O sistema de controle é destinado ao projeto e desenvolvimento de equipamentos automáticos com aplicações dedicadas nas indústrias.

O artigo está dividido da seguinte maneira: na seção 2 são enfocados aspectos essenciais para a parametrização e flexibilidade de controle dos equipamentos; na seção 3 aspectos metodológicos são abordados; a seção 4 descreve a técnica de controle proposta; na seção 5 são abordados os resultados da pesquisa; por fim a seção 6 apresenta as conclusões e perspectivas do trabalho.

## II. REQUISITOS PARA AUTOMAÇÃO FLEXÍVEL

Para agregar características de flexibilidade aos equipamentos, alguns aspectos são importantes e considerados fundamentais, permitindo o controle de equipamentos programáveis por computadores pessoais [7]. O uso de motores de passo, a porta de comunicação usada, a visualização com simulação gráfica do projeto, e características como resolução (distância percorrida a cada comando de movimentação), aceleração, desaceleração e o modo de codificação (linguagem usada) dos passos que representam a trajetória do equipamento, podem ser parametrizáveis por meio de um sistema de controle (conjunto *hardware e software*).

Os motores de passo são um exemplo de movimentadores que agregam características fundamentais, tais como, grande precisão de movimentos, controle de aceleração e desaceleração, entre outras. Por conseguinte, são amplamente usados em sistemas robóticos e no controle de equipamentos, onde estes têm que realizar operações de posicionamento de alta precisão [8]. De acordo com [9], um motor de passo atua como um preciso acionador mecânico, tendo um dispositivo elétrico dividindo a rotação do motor em um grande número de passos, proporcionando movimentação precisa.

Diversos trabalhos utilizam estes motores como atuadores, sendo projetos com as mais variadas características e complexidades, como por exemplo, equipamentos que fazem a obtenção de propriedades mecânicas em dutos de petróleo [10]; um projeto de ambiente de aprendizagem remoto (via WEB) que permite a prática laboratorial e aprendizado de física [11]; um posicionador eletrônico para estruturas micrométricas, utilizado para posicionar objetos com passos tridimensionais e micrométricos [12]. Outro exemplo que destaca a importância da parametrização do controle é um dispositivo programável que permite a mobilização dos membros inferiores imediatamente no pós-operatório (exoesqueleto), promovendo a manutenção dos tecidos que envolvem a articulação, destacando o uso da tecnologia de comando numérico e propondo um modelo que opere de maneira flexível, fornecendo possibilidades de programação por parte do usuário [13].

A quantidade crescente de projetos e sistemas robóticos diferenciados alavanca a necessidade da existência de alternativas que possam acelerar o projeto destes equipamentos, focando na flexibilidade e portabilidade dos sistemas de controle. No âmbito de sistemas de controle para equipamentos programáveis é notável a necessidade de adaptação as mudanças de projeto, sendo necessário possuir características configuráveis.

## III. MOTIVAÇÃO E METODOLOGIA

Este trabalho objetiva apresentar uma técnica de controle focada na flexibilidade e parametrização de configurações, para auxiliar no projeto e desenvolvimento de equipamentos programáveis.

O estudo iniciou com uma pesquisa bibliográfica sobre controle de equipamentos automáticos programáveis em âmbito industrial, constatando a realidade atual dos mesmos e explorando as suas sistemáticas de controle. Desta forma, do ponto de vista de seus objetivos, esta pesquisa é classificada como bibliográfico-exploratória, atuando em área onde existem estudos, mas ainda com possibilidades de pesquisa, que podem ser exploradas visando à obtenção de novas soluções e expandir os horizontes da área. Quanto à natureza, esta pesquisa é qualitativa, porque identifica características fundamentais para a melhoria dos sistemas de controle para equipamentos programáveis, focando na possibilidade de parametrização de configurações do equipamento, tornando-o assim, um sistema adaptativo aos projetos. Assim, é proposta e apresentada uma metodologia de controle flexível aplicando estas características, facilitando a sua compreensão e aplicabilidade, para agregar alternativas no processo de controle de equipamentos programáveis. Com tais características, a metodologia também objetiva integrar pesquisadores de outras áreas promovendo assim atividades interdisciplinares.

Para testes da metodologia proposta foi desenvolvido um circuito elétrico de controle, como interface de comunicação entre o computador e o equipamento programável. A sua composição permite o envio de comandos ao equipamento por meio da porta de comunicação USB.

Por fim, um *software* foi desenvolvido seguindo as características da metodologia proposta com capacidade para simular graficamente a trajetória do equipamento controlado, permitindo ainda correções e ajustes prévios, para em seguida enviar os comandos (pulsos elétricos) do computador para os atuadores do equipamento controlado.

Os testes para validação do sistema de controle foram realizados no Núcleo de Automação e Projetos de Fabricação - NAFA na Universidade Federal de Santa Maria – UFSM.

## IV. TÉCNICA DE CONTROLE EMPREGADA

A metodologia de controle flexível proposta neste trabalho proporciona o projeto e desenvolvimento de equipamentos programáveis, aliando baixo custo com alta portabilidade. O foco é agregar valor e propor melhorias para os sistemas de controle desenvolvidos atualmente, aumentando as possibilidades de controle dos equipamentos, impulsionando atividades interdisciplinares e o aprendizado de automação e controle.

A informação de entrada do sistema proposto é um arquivo contendo uma codificação de pulsos que representa a trajetória do equipamento no ambiente de atuação. Por meio desta codificação, o sistema de controle faz a leitura e interpretação de cada código, que possui uma letra do alfabeto

representando um determinado movimento nos eixos/motores do equipamento. Esta forma de entrada de dados (Tabela 1) pode ser inserida manualmente no programa, representando uma trajetória conhecida e simplificada, movimentando os motores em sentido horário ou anti-horário. Outra forma de entrada de dados é o uso de um programa que converte a trajetória de um equipamento para estes códigos de movimentação [14], por meio de técnicas de interpolação linear e circular, gerando um arquivo que pode ser carregado no sistema de controle.

A Tabela 1 mostra a codificação da movimentação de um equipamento programável. O sinal de "+" e "-" na descrição do eixo controlado indica o sentido (horário ou anti-horário) de movimentação do eixo.

TABELA 1
CODIFICAÇÃO DOS PULSOS PARA MOVIMENTAÇÃO DOS EIXOS

| Cod | Descrição do Movimento | Eixo |
|---|---|---|
| A | Avanço no Eixo X | +X |
| B | Avanço no Eixo Y | +Y |
| C | Avanço no Eixo Z | +Z |
| D | Retorno no Eixo X | -X |
| E | Retorno no Eixo Y | -Y |
| F | Retorno no Eixo Z | -Z |
| G | Avanço no Eixo X e Y | +X e +Y |
| H | Avanço no Eixo X e Z | +X e +Z |
| I | Avanço no Eixo Y e Z | +Y e +Z |
| J | Retorno no Eixo X e Y | -X e -Y |
| K | Retorno no Eixo X e Z | -X e -Z |
| L | Retorno no Eixo Y e Z | -Y e -Z |
| M | Avanço em X e Retorno em Y | +X e -Y |
| N | Retorno em X e Avanço em Y | -X- e +Y |
| O | Avanço em X e Retorno em Z | +X e -Z |
| P | Retorno em X e Avanço em Z | -X e +Z |
| Q | Avanço em Y e Retorno em Z | +Y e -Z |
| R | Retorno em Y e Avanço em Z | -Y e +Z |
| S | Avanço nos Eixos X, Y e Z | +X, +Y,+Z |
| T | Retorno nos Eixos X, Y e Z | -X, -Y, -Z |
| U | Avanço Eixo X e Retorno Y e Z | +X, –Y, -Z |
| V | Avanço Eixo Y e Retorno X e Z | -X, +Y, -Z |
| W | Avanço Eixo Z e Retorno X e Y | -X, -Y, +Z |
| X | Retorno Eixo X e Avanço Y e Z | -X, +Y, +Z |
| Y | Retorno Eixo Y e Avanço X e Z | +X, -Y, +Z |
| Z | Retorno Eixo Z e Avanço X e Y | +X, +Y, -Z |

Esta codificação serve de parâmetro para que sistemas de controle futuros possam ser desenvolvidos de maneira mais amigável, melhorando a característica de usabilidade, possibilitando maior flexibilidade de controle, bem como também a realização de atividades interdisciplinares.

Ao executar, por exemplo, a informação "AAAAAAAAAA BBBBB GGGGGGGGGG" o sistema de controle envia dez pulsos de movimentação, no eixo X, cinco pulsos, no eixo Y, e depois dez pulsos em ambos os eixos X e Y. Neste caso, a tabela trabalha com sistemas atuando com três eixos de coordenadas, cada eixo tendo um motor de passo como atuador.

A Fig. 1 apresenta um diagrama representativo da metodologia de controle proposta neste trabalho, representada por meio de um fluxograma dinâmico, mostrando o fluxo de dados de controle da metodologia, dividindo os seus principais componentes. No diagrama os atores principais são o operador ou usuário do sistema e o equipamento a ser controlado. Assim, um fluxo de informações de controle é intermediado por vários processos, como a entrada de dados, interpretação, simulações, correções e envio dos comandos para a porta USB. Estes são responsáveis por traduzir a entrada de dados feita pelo operador do sistema em movimentos nos atuadores do equipamento.

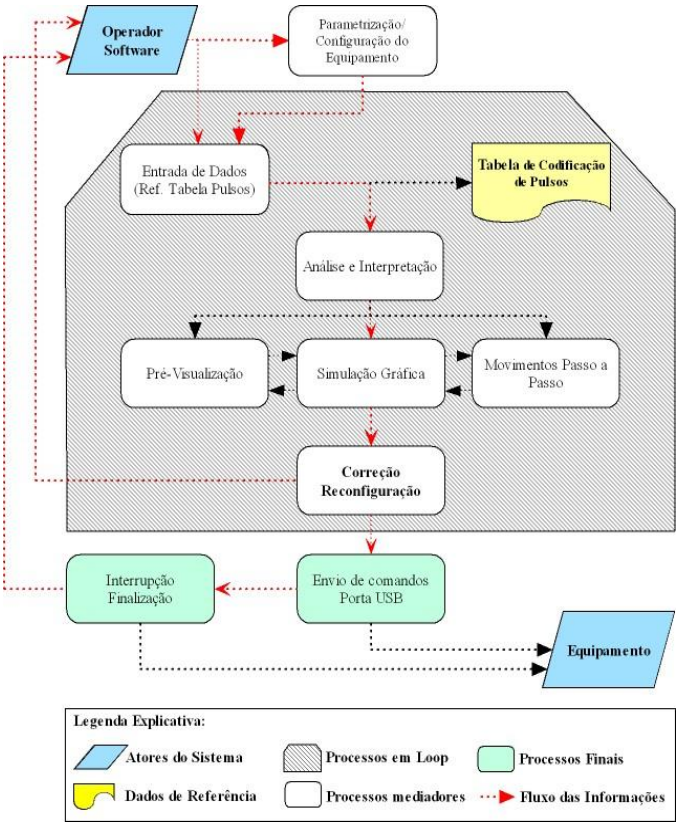


Fig. 1: Diagrama da metodologia em forma de modelo dinâmico.

Para contextualizar a metodologia proposta, descrevendo a sua dinâmica comportamental e mostrando a interação entre os atores do sistema, foi utilizada a modelagem UML – *Unified Modeling Language*, por meio de alguns de seus diagramas, diagrama de casos de uso e o diagrama de atividades. A linguagem UML utiliza representações gráficas para descrever os vários elementos que compõem um sistema, permitindo sua visualização sob diferentes perspectivas. O objetivo principal da UML é fornecer uma linguagem padrão que permita modelar um sistema, possibilitando a troca natural destes modelos entre os desenvolvedores de *softwares* [15]. Com a UML é possível descrever eficazmente requisitos de *software*, caracterizar a arquitetura de um sistema e direcionar programadores, aumentando a produtividade e diminuindo os riscos [16] [17].

O objetivo da diagramação é descrever a dinâmica comportamental da metodologia de controle proposta. Deste modo, dois diagramas da visão dinâmica da UML são utilizados, o diagrama de casos de uso e o de atividades.

O diagrama de ***casos de uso***, apresentado na Fig. 2, é usado para representar os atores, que desempenham papéis fundamentais no sistema, mostrando o fluxo entre os

processos ou funções que nesta técnica o sistema de controle deve realizar. O operador (ator inicial), por intermédio do *software* de controle, inicializa o processo com a configuração do equipamento e leitura do arquivo contendo os códigos de pulsos, este processamento pode ser simulado graficamente antes do envio dos comandos para a realização da movimentação do equipamento (ator final).

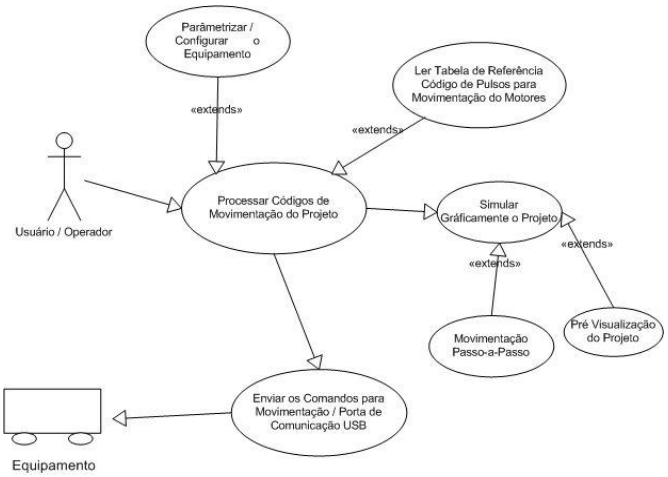



Fig. 2: Diagrama de Casos de Uso para a metodologia de controle.

O ***diagrama de atividades*** ilustra a seqüência de atividades realizadas no sistema, demonstrando o fluxo de trabalho de um ponto inicial até o ponto final, detalhando as decisões tomadas durante o caminho percorrido durante a execução das tarefas.

Dois pontos formados por círculos preenchidos na cor preta representam o estado inicial e o estado final do sistema. Entre eles os processos e as tomadas de decisão formam o fluxo de controle (*workflow*). No diagrama da Fig. 3, uma atenção especial deve ser dada ao processo de correção ou reconfiguração do projeto, pois, esta tomada de decisão gera um novo processamento no sistema de controle para permitir a adaptação da trajetória inicial especificada, fazendo assim o movimento esperado no equipamento.

A seguir, a Fig. 3 apresenta este diagrama.

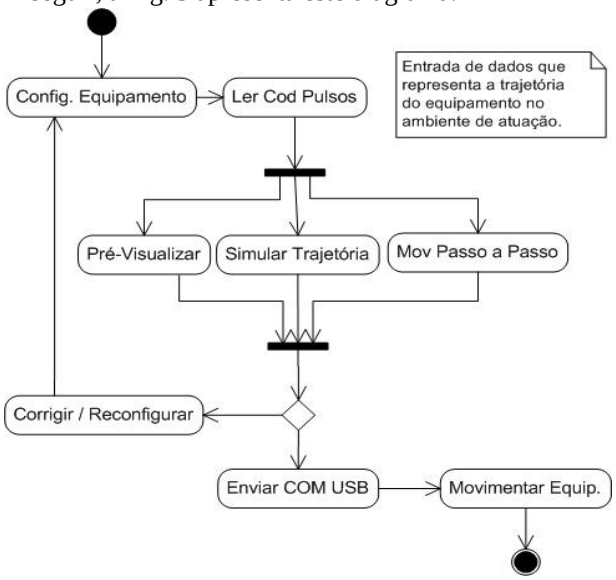



Fig. 3: Diagrama de Atividades da metodologia de controle.

O processo envia os comandos para o equipamento por meio do protocolo USB, utilizando uma biblioteca de funções de controle, tendo como resultado o movimento dos motores do equipamento. Na seção seguinte são detalhados os resultados da pesquisa, descrevendo o *hardware* e o *software* desenvolvido de acordo com a metodologia de controle.

## V. RESULTADOS

O trabalho foi validado com o desenvolvimento de um *hardware* e um *software* para controle de equipamentos programáveis, os quais são apresentados a seguir.

### *A. Hardware de controle desenvolvido*

Para efetivar o controle é necessário um dispositivo de *hardware* que faça o papel de interface de comunicação entre o *software* de controle instalado no computador e o equipamento a ser controlado. O *hardware* é composto de atuadores controlados e alimentados por meio de um circuito elétrico.

Com o advento da comunicação USB, desenvolvedores e projetistas obtiveram ganhos na conectividade ou portabilidade dos equipamentos projetados com a velocidade de transmissão de dados, conexão simplificada e suporte a software que este protocolo de comunicação permite.

O modelo apresentado possibilita o controle de um equipamento via *software*, seguindo as características enfocadas na técnica. O circuito desenvolvido permite a comunicação entre o computador e o equipamento por meio do protocolo USB [18].

O circuito de controle é baseado em um módulo *USB-to-FIFO* de interface paralela, dispondo de um barramento de dados bidirecional de 8 bits e sinais de controle, que são usados para controlar o fluxo de dados entre o computador e qualquer circuito externo utilizando o protocolo de comunicação USB.

O módulo utilizado é o componente modelo DLP-USB245M-G, visualizado na Fig. 4, da FTDI (*Future Technology Devices International*), que possui em seu *firmware* (programa controlador) toda a camada do protocolo USB. São fornecidos pelo fabricante os *drivers* para a instalação no computador, tendo sua biblioteca de funções e documentação detalhada para desenvolvimento na linguagem de programação *C++ Builder*® ou *Visual C++*®.

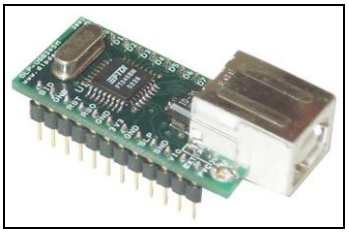


Fig. 4: Chip FTDI DLP-USB245M-G.

As principais características do componente são:

• Velocidade de transmissão e recebimento de dados de 1 MBps;
• Comunicação USB gerenciado pelo próprio chip;
• Compatível com padrão USB 1.1 e 2.0;
• 384 bytes reservados em *buffer* para transmissão;
• 128 bytes reservados em *buffer* para recepção;
• Montagem em soquete de 24 pinos, o que facilita a conexão em circuitos;
• Modo de transferência *Bulk* (em grandes quantidades e simultaneamente) e Isossíncrona (quantidade pré-negociável da banda de transmissão do barramento);
• *Drivers* de comunicação compatíveis com Windows 98/2000/CE/XP, MAC OS-8 e OS-9 e Linux 2.40 ou superior.

O dispositivo *USB-toFIFO*, DLP-USB245M-G, é usado como meio de comunicação em alguns trabalhos atuais [19], [20], pois, agrega um meio simples e de baixo custo para permitir a comunicação dos computadores atuais com o hardware de controle.

Durante os testes do circuito de controle, realizado no NAFA da UFSM, foi detectado que do modo em que o circuito elétrico está constituído, é necessário o envio de um ***pulso de controle*** que seleciona um motor diferente do atual, mas sem ativar bobinas, entre os pulsos de uma seqüência no mesmo eixo/motor (ex: pulsos "AAAAA"). Esse pulso é enviado para gerar uma transição de *clock* (sinal que habilita o envio de pulsos) entre comandos enviados no mesmo eixo. Quando pulsos são enviados alternadamente em diferentes eixos (ex: pulsos "ABABABCACA"), este pulso de controle não é necessário, pois a simples troca de eixo gera o *clock* necessário.

### *B. Software de controle desenvolvido*

O *software* para controle desenvolvido neste trabalho foi chamado de "SCEAP – Sistema de Controle para Equipamentos Automáticos Programáveis". Este *software* foi desenvolvido focando o projeto, controle e testes de equipamentos com aplicações dedicadas nas indústrias que fazem o uso de motores de passo como atuadores, como mesas XY, XYZ, pequenos robôs, e outros equipamentos.

O *software* desenvolvido para testes da técnica compreende algumas etapas que foram seguidas para torná-lo simples e funcional facilitando o seu uso. Estas etapas são:
• Escolha da linguagem de programação;
• Estruturação da interface gráfica;
• Programação do ambiente gráfico de simulação;
• Programação da comunicação por meio da porta USB; e
• Testes com o *hardware* desenvolvido.

A base para seu desenvolvimento são características apresentadas em [7], as quais foram agregadas à técnica apresentada, para flexibilizar o controle dos equipamentos. A linguagem de programação utilizada foi a *C++*, por ser uma linguagem que permite a programação orientada a objetos, com amplo uso no meio acadêmico e a boa documentação do módulo DLP-USB245M-G agregado ao *hardware* de controle.

A possibilidade de parametrização de variáveis que compõem a planta de atuação do equipamento é importante, pois, permite aos pesquisadores, o projeto e testes de equipamentos em sua fase inicial. Assim, evita-se o gasto dispendioso de tempo no desenvolvimento de software de controle específico, contribuindo para obtenção de resultados em menor tempo. A seguir são apresentadas as principais telas do sistema desenvolvido, enfocando as suas particularidades de programação e controle.

A Fig. 5 mostra a tela inicial, onde é dada a entrada de um arquivo contendo a codificação da movimentação ou trajetória do equipamento. Usando a opção "Código de Pulsos - Abrir", o sistema faz a leitura e interpretação do código de pulsos, armazenando esta informação para uso na simulação e na movimentação efetiva do equipamento. Em seguida, o usuário pode escolher qual o tipo de simulação que deseja realizar.

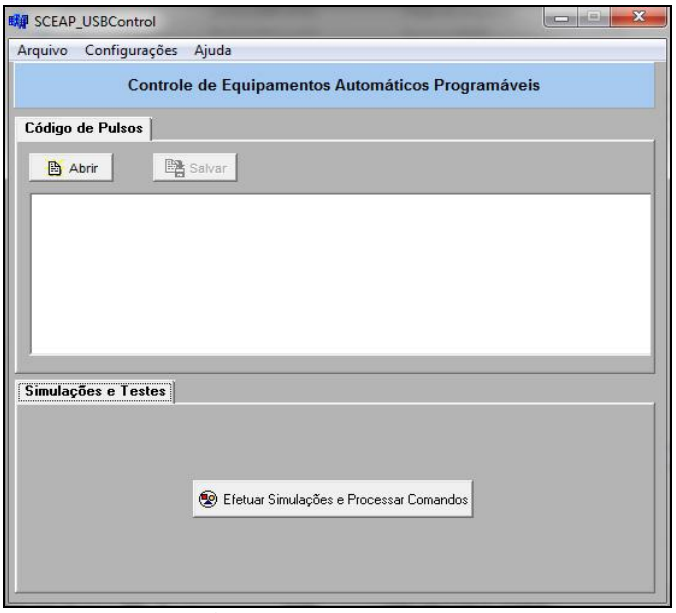


Fig. 5: Tela inicial do software desenvolvido.

Descrição da tela:
• Menu Arquivo – Opção de acesso a leitura do arquivo contendo o código de pulsos;
• Menu Configurações – Parâmetros que são configuráveis;
• Menu Ajuda – Descrição geral do sistema;
• Abrir Código de Pulsos – Códigos de pulsos que representam a trajetória que o equipamento irá realizar;
• Efetuar Simulações e Processar comandos – Esta opção possibilita simular e executar o controle de equipamentos que usam sistemas de coordenadas com dois eixos. Outra opção de simulação também mostra em uma linha de tempo a movimentação de um equipamento com três eixos;

Após a leitura do arquivo contendo o código de pulsos, a pré-visualização e simulação da movimentação do equipamento podem ser realizadas. A simulação gráfica foi programada usando a linguagem OpenGL, que é uma biblioteca de rotinas gráficas que permite a programação de diversas formas geométricas em 2D e 3D, a qual é muito utilizada em jogos e sistemas de visualização gráfica. Optou-se pela linguagem OpenGL, por ser facilmente integrada ao ambiente de programação *C++ Builder*®, onde permitiu a

programação e simulação gráfica da trajetória do equipamento controlado.

A programação realizada no sistema permite a visualização e correção pró-ativa do equipamento que está utilizando o sistema de controle. A Fig. 6 mostra a tela com a simulação gráfica de uma trajetória exemplificada em um sistema com dois eixos, podendo-se observar a esquerda do eixo Y do plano cartesiano as informações da simulação que está em andamento.

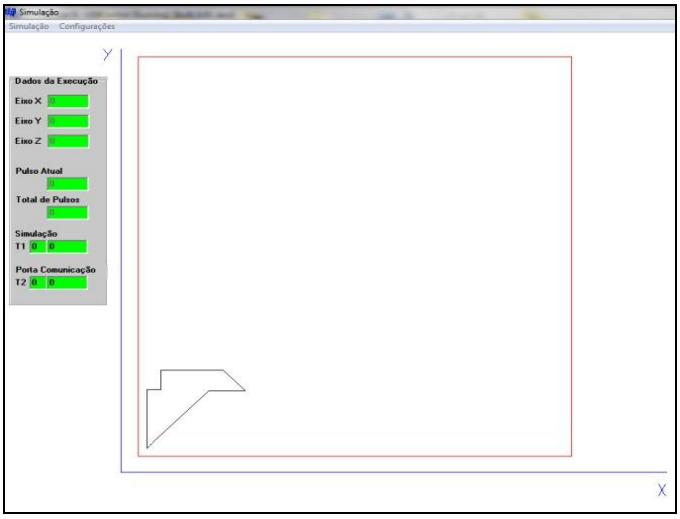



Fig. 6: Tela que faz a simulação da trajetória do equipamento.

Os dados atualizados durante a simulação são:

• Posição atual do Eixo X e do Eixo Y;

• Pulso atual – de acordo com o arquivo de codificação da trajetória; Total de pulsos que serão executados;

• T1 – Tempo de simulação;

• T2 – Tempo de envio dos comandos para o equipamento.

Os tempos de execução (*timers*) T1 e T2 são equivalentes e proporcionais a quantidade de pulsos que serão executados, permitindo executar a simulação e o envio dos comandos para a porta USB ao mesmo tempo, para assim, visualizar em tempo de execução a movimentação no equipamento.

Outra opção de simulação é a execução do controle de três eixos/motores de um equipamento sendo visualizada em uma linha de tempo. A Fig. 7 ilustra este modo de simulação.

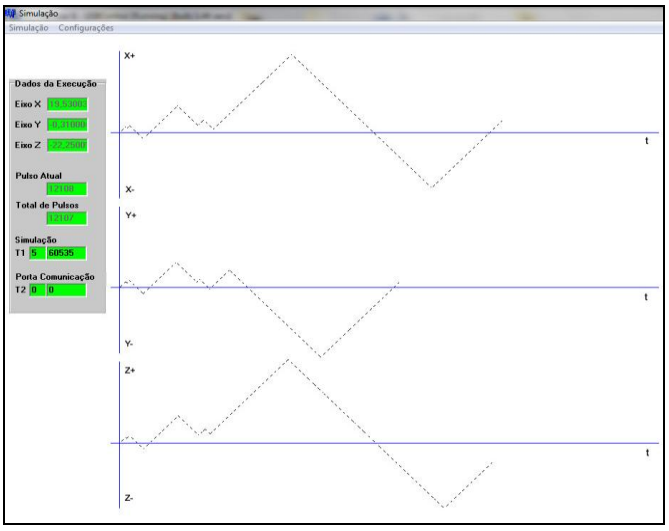



Fig. 7: Simulação do processamento dos pulsos em linha do tempo.

Esta opção pode ser usada em aplicações diversas, onde o monitoramento do envio dos comandos para os motores pode ser visualizado passo a passo. Pode-se citar como exemplo do uso deste tipo de simulação, o projeto de um equipamento para reabilitação passiva contínua em cotovelo, onde foram utilizados dois motores de passos realizando a movimentação de duas articulações [21]. Este tipo de equipamento não usa sistema de coordenadas, assim, pode-se testar e visualizar em uma linha do tempo os pulsos enviados aos motores.

Quando ajustes na trajetória forem necessários, algumas configurações podem ser alteradas, podendo-se simular várias vezes o projeto antes de enviar o comando para a porta USB.

São elas:

• **Intervalo de envio entre pulsos:** É o intervalo de tempo entre o envio de pulsos/comandos para a porta USB, o qual é informado em milissegundos. Este intervalo deve ser ajustado de acordo com as características de funcionamento do equipamento, onde o incremento e decremento deste parâmetro representam a velocidade dos motores.

• **Resolução no equipamento:** representa a distância que o equipamento percorre a cada pulso enviado, definido em milímetros por pulso (mm/pulso). Esta configuração é importante para a representação gráfica, ao simular a movimentação do equipamento, auxiliando assim na detecção de erros.

• **Distância para aceleração:** esta informação é usada para o cálculo da aceleração dos motores, onde o usuário pode definir de acordo com características físicas do equipamento, qual a distância que o motor percorrerá até chegar à velocidade constante (intervalo de envio de pulsos).

• **Aceleração inicial e desaceleração:** configuração por eixo controlado no sistema, onde é informada a aceleração inicial do equipamento. São configurações importantes e usadas de acordo com o tamanho do equipamento e características físicas dos motores, sendo informada pelo projetista, de acordo com variantes como inércia da carga e tamanho/limites da planta/ambiente de atuação.

• **Folga *backlash*:** configurar um número de pulsos para compensar a folga do equipamento em movimentos inversos;

• **Limites de atuação:** tamanho do ambiente ou planta de atuação, definida em centímetros. Importante para a representação gráfica durante a simulação da trajetória, verificando assim possíveis erros de movimentação.

• **Movimentos manuais:** realização de movimentos com o envio passo a passo (ou pulso a pulso) na porta de comunicação USB, de acordo com os eixos da primeira opção de simulação.

A seguir são descritas as principais funções do módulo DLP-USB245M-G que foram usadas na programação do *software* de controle desenvolvido, detalhando como as mesmas são processadas e como ocorre o envio dos comandos para a porta de comunicação USB. Maiores detalhes e também outros exemplos de uso das funções podem ser encontrados na sua biblioteca de funções.

***FT_ListDevices*** *(PVOID pvArg1, PVOID pvArg2, DWORD dwFlags)*

A função *FT_ListDevices* retorna informações dos dispositivos conectados, como por exemplo, o número de dispositivos conectados, descrição do componente e identificador usado na conexão.

***FT_Open*** *(int iDevice, FT_HANDLE *ftHandle)*

A função *FT_Open* é usada para abertura de um ou mais dispositivos, deixando-o pronto para os acessos seguintes.

Esta função possui apenas dois parâmetros, o primeiro indica o número do dispositivo e o segundo é a variável usada para acessar o dispositivo nas operações seguintes. Se apenas um dispositivo estiver conectado, o primeiro parâmetro (*iDevice*) deve ser 0, quando se estiver usando múltiplos dispositivos, estes serão referenciados por números inteiros, como 1,2 e etc.

***FT_Write*** *(FT_HANDLE ftHandle, LPVOID lpBuffer, DWORD dwBytesToWrite, LPDWORD lpdwBytesWritten)*

As operações de escrita no dispositivo que resultarão em movimentos nos motores de passo, são processadas pela função *FT_Write*, a qual tem sua estrutura detalhada e definida de acordo com quatro parâmetros importantes. Estes parâmetros são definidos de acordo com as características do hardware a controlar.

• *ftHandle*: identificador do dispositivo em que será efetuada a operação de escrita. O mesmo identificador usado na operação de abertura na função anterior.

• *lpBuffer*: apontador para o *buffer* (variável que armazena em memória os dados que serão usados na operação) que contém os dados que serão escritos no dispositivo.

• *dwBytesToWrite*: número de bytes que serão escritos no dispositivo.

• *lpdwBytesWritten:* apontador para a variável que retorna o número de bytes escrito no dispositivo.

Na Fig. 8 é visualizada uma sequencia de seis pulsos (AAAADD), os quais são representados por códigos hexadecimais, que no circuito de controle são convertidos em códigos de 8 bits, para assim representar o motor selecionado e acionado.

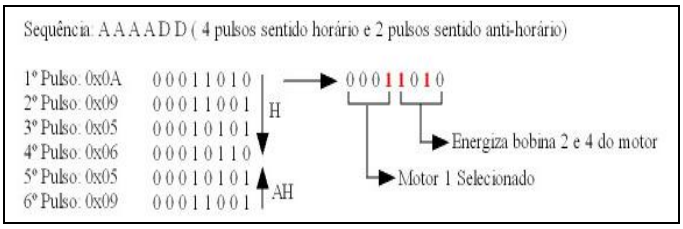


Fig. 8: Envio de pulsos com a composição do *byte* enviado para a porta USB.

Neste trabalho, o circuito de controle implementado permite o controle de até quatro motores. Para a movimentação correta dos motores, os dados devem ser enviados para a porta USB seguindo uma tabela de dados de controle, a qual é apresentada a seguir pela Fig. 9.

| Data byte to USB-to-FIFO | Stepper motor selected | Coils energized L1 | L2 | L3 | L4 | Direction Forward | Reverse |
|---|---|---|---|---|---|---|---|
| 0x0A | SM1 | 1 | 0 | 1 | 0 | | |
| 0x09 | SM1 | 1 | 0 | 0 | 1 | | |
| 0x05 | SM1 | 0 | 1 | 0 | 1 | | |
| 0x06 | SM1 | 0 | 1 | 1 | 0 | | |
| 0x1A | SM2 | 1 | 0 | 1 | 0 | | |
| 0x19 | SM2 | 1 | 0 | 0 | 1 | | |
| 0x15 | SM2 | 0 | 1 | 0 | 1 | | |
| 0x16 | SM2 | 0 | 1 | 1 | 0 | | |
| 0x2A | SM3 | 1 | 0 | 1 | 0 | | |
| 0x29 | SM3 | 1 | 0 | 0 | 1 | | |
| 0x25 | SM3 | 0 | 1 | 0 | 1 | | |
| 0x26 | SM3 | 0 | 1 | 1 | 0 | | |
| 0x3A | SM4 | 1 | 0 | 1 | 0 | | |
| 0x39 | SM4 | 1 | 0 | 0 | 1 | | |
| 0x35 | SM4 | 0 | 1 | 0 | 1 | | |
| 0x36 | SM4 | 0 | 1 | 1 | 0 | | |

Fig. 9: Dados de controle dos motores [18].

A sequência de envio dos comandos define a direção de rotação do motor, e as configurações de velocidade, aceleração e desaceleração são especificadas no software de controle. Esta tabela contém os valores em notação hexadecimal que são usados para energizar as bobinas de um determinado motor, sendo que a sequência de ativação destas bobinas fará com que o motor gire em determinado sentido.

O arquivo de entrada contendo o código de pulsos é resultado de ***técnicas de interpolação linear e circular***, como já destacado anteriormente. Na Fig. 10 é visualizada uma seqüência gráfica que exemplifica o processamento de um arquivo de pulsos contendo uma técnica de interpolação circular [22].

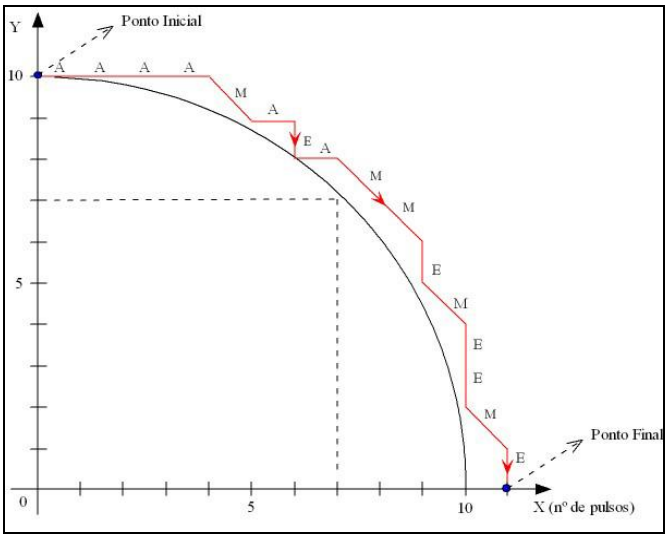



Fig. 10: Sequência de pulsos exemplificada na técnica de interpolação circular. Adaptado de [22].

Desta forma, o *software* faz a leitura dos códigos de envio para movimentar o equipamento em determinada trajetória. Todas as funções utilizadas são importantes na programação do software de controle, pois possibilitam o controle e parametrização de características de um equipamento.

## VI. CONCLUSÕES E CONSIDERAÇÕES FINAIS

Esta pesquisa apresentou uma técnica de controle flexível/adaptativa para equipamentos automáticos programáveis, dando ênfase à flexibilidade de controle, com a possibilidade de configuração e parametrização de características dos equipamentos.

A parametrização destas características é delimitada pela complexidade dos projetos, que é resultado da criatividade dos projetistas. Muitas vezes esta criatividade é limitada às possibilidades de controle oferecidas por *softwares* comerciais fechados. Sendo assim, as características configuráveis destacadas e usadas, diversificaram as funcionalidades de controle.

A técnica proposta é aplicável em ambiente industrial como alternativa de sistema de controle, pois suas características permitem o desenvolvimento de equipamentos com funções dedicadas e peculiares. Considerando que para se fazer uso da técnica, é necessário que a sequência de pulsos para cada tarefa programada esteja de acordo com a codificação especificada, o sistema de controle exige processamento prévio desses dados. Porém, o uso de técnicas de interpolação linear e circular, implementados em trabalhos recentes [14], representam um meio eficiente de conversão dos movimentos desejados na codificação de pulsos usada (Tabela 1).

É importante destacar na técnica, a comunicação e o controle entre equipamento e computador pessoal, usando o protocolo USB, realizado por meio de um dispositivo eletrônico (chip USB245BM-G) programável via *software*. A biblioteca de funções desse componente traz possibilidades de controle por esta porta de comunicação, tornando o sistema de controle portável (usado em qualquer computador pessoal) e de acordo com os requisitos de comunicação atuais.

Os testes do conjunto *software/hardware* desenvolvidos foram realizados em laboratório, com motores de passo comumente usados em diversos projetos, sendo também usado para testes de trabalhos recentes realizados no Núcleo de Automação e Projetos de Fabricação – NAFA/UFSM [14] [21], demonstrando a sua viabilidade no projeto e desenvolvimento de novos equipamentos.

Por fim, conclui-se que a técnica apresentada permite aos projetistas maior liberdade nas operações de acionamento, diminuindo o tempo gasto no desenvolvimento de *software e hardware* de controle específico (usados nos testes e projeto de novos equipamentos), possibilitando assim, diminuir os custos relacionados.

Este trabalho possibilita a realização de outras pesquisas na área de sistemas de controle flexíveis destinados ao projeto e desenvolvimento de equipamentos programáveis. Desta forma, algumas atividades podem ser realizadas para dar esta continuidade, como as descritas a seguir:

- Integrar no sistema a conversão de programas CN para a codificação de pulsos especificada na metodologia, através de técnica de interpolação linear e circular;
- Desenvolver outra forma de simulação da trajetória do equipamento, adicionando a parte tridimensional de movimentação;
- Acrescentar a possibilidade de leitura de sensores no sistema, para controlar e monitorar eventos;
- Acrescentar um código a tabela de pulsos para acionamento e desligamento de dispositivos, a partir de um determinado ponto de execução;
- Acrescentar à metodologia, a possibilidade de controlar *servomotores* (motores com alto torque e larga faixa de rotação), para acionamento de equipamentos de maior porte;
- Implementar uma rotina para efetuar a leitura reversa do arquivo de pulsos, a partir de qualquer ponto de execução. Esta característica permitirá a execução reversa da trajetória do equipamento.

## REFERÊNCIAS

[1] M. A. B. De Paula, E. A. P. Santos, "Uma Abordagem Metodológica para o Desenvolvimento de Sistemas Automatizados e Integrados de Manufatura" *Revista Produção*, 2008.

[2] L. S. Magalhães, "Abastecimento de Estoque por Robô Microcontrolado", *Anhanguera Educacional S. A.*, São Paulo, 2009.

[3] M. Y. Kamogawa, M. A. Teixeira, "Autoamostrador de Baixo Custo para Análise por Injeção em Fluxo", *Revista Química Nova*, 2009.

[4] A. S. Ribeiro, "Modelagem e Especificação de Controle de Sistemas Flexíveis de Manufatura, Utilizando Redes de Petri de Alto Nível", Programa de Pós-Graduação em Mecatrônica, Universidade Federal da Bahia , Dissertação de Mestrado, 2009.

[5] K. Ogata, *Engenharia de Controle Moderno*. Pearson Prentice Hall, 4ª ed., SP, 2003.

[6] J. C. Dutra, L. Felippe, R. S. Carvalho, "Sistema robótico de quatro graus de liberdade e processos de soldagem dedicados para o revestimento de tubos de caldeiras", Universidade Federal de Santa Catarina – UFSC, *V Congresso de Inovação Tecnológica em Energia Elétrica – V CITENEL*, Belém/PA, 2009.

[7] A. S. Roque, A. D. Silva. "Sistemas de Controle para Equipamentos Automáticos Programáveis: Características Fundamentais para Compor uma Metodologia de Controle Flexível", *4º Workshop de Robótica Aplicada e Automação, Robocontrol*, 2010.

[8] P. Melin, O. Castilho, "Intelligent Control of a Stepping Motor Drive Using Adaptive Neuro-fuzzy Inference System", *Journal Information Sciences*, 170 133-151, Elsevier – ScienceDirect, 2004.

[9] C. Muñiz, R. Levi, M. Benkrid, F. B. Rodriguez, P. Varona, "Real-time control of stepper motors for mechano-sensoryvstimulation", *Journal of Neuroscience Methods*, Elsevier – ScienceDirect, 2008.

[10] J. V. De Souza, A. B. Soares, S. D. Franco, "Desenvolvimento de um Sistema de Hardware e Software para Otimização das Unidades de Controle, Monitoração, Coleta e Processamento de Dados de um Macroidentador Portátil para Avaliação In-Situ de Propriedades Mecânicas de Dutos Metálicos", *Revista Horizonte Científico*, Volume 1, N. 9, 2008.

[11] C. R. Oliveira, I. Oliveira, H. Santos, A. Pereira, "Um Ambiente para a Prática Remota de Aulas Laboratoriais de Física (determinação da viscosidade de líquidos)", *Revista Brasileira de Informática na Educação - RBIE*, v.17, N. 1 – 2009.

[12] G. M. Calixto, "Desenvolvimento de um posicionador eletrônico para estruturas micrométricas", Universidade Estadual de Campinas - UNICAMP, Dissertação de Mestrado, 2009.

[13] D. Q. Sperb, "Desenvolvimento de dispositivo programável de movimento passivo contínuo para membros inferiores", Universidade Federal de Santa Maria - UFSM, Dissertação de Mestrado, 2006.

[14] E. F. De Cristo, "Implementação de Técnicas de Controle de Motor de Passo em Aplicações CNC", Universidade Federal de Santa Maria - UFSM, Dissertação de Mestrado, 2009.

[15] R. P. Silva, *UML2 em Modelagem Orientada a Objetos*, Florianópolis, Visual Books, 2007.

[16] T. Pender, *UML a Bíblia*. Elsevier, RJ, 2004.

[17] R. S. Moura, L. A. Guedes, "Modelagem de aplicações de automação industrial usando diagramas UML e reuso de componentes", Universidade Federal do Piauí – UFPI, *Simpósio Brasileiro de Automação Inteligente – SBAI*, 2007.

[18] E. LOGASHANMUGAM, K. SURESH, "Control Multiple Stepper Motors Through A PC's USB Port", Ed. Online – Id 16125, *Eletronic Design*, http://eletronicdesign.com, 2007.

[19] J. M. Nageb, A. Ruhullah, S. Salleh, "12 Channel USB Data Acquisition System For QT Dispersion Analysis", Proceedings of the International Conference on Robotics, Vision, *Information and Signal Processing, ROVISP*, 2005.

[20] P. Giassi Junior, "Etofisiógrafo: Protótipo de um Sistema Telemétrico para Registro de Variáveis Fisiológicas em Pequenos Animais", Universidade Federal de Santa Catarina – UFSC, Dissertação de Mestrado, 2006.

[21] A. M. Callegaro, "Desenvolvimento de um Equipamento Computadorizado de Movimentação Passiva Contínua para Cotovelo e Antebraço", Universidade Federal de Santa Maria – UFSM, Dissertação de Mestrado, 2010.

[22] M. Weck, *Werkzeugmaschinen Band 3 – Autommatisierung und Steuerungstechnik*. Düsseldorf: VDI-Verl, Germany, 1989.

# Análise de Gramáticas Inferidas usando Adaptatividade aplicadas em Linguagens Naturais

(17 de janeiro de 2011)

G. C. Januário, L. A. F. Leite, M. L. Koga, J. J. Neto

***Resumo* — Para o processamento de linguagens naturais, em aplicações como tradutores, faz-se necessário a modelagem da língua em questão como uma linguagem formal, bem como a análise sintática do texto processado. Essas tarefas apresentam grandes desafios devido a grande complexidade das linguagens naturais, o que nos leva a fazer uso de recursos como a adaptatividade para superar esses obstáculos. Neste artigo propomos um processo de validação de um algoritmo indutor de gramáticas no contexto de linguagens naturais, sendo o algoritmo a ser validado um indutor adaptativo de gramáticas livre de contexto.**

***Palavras-chave* — adaptatividade, indução de gramática, linguagens formais, linguagens naturais**

## I. Introdução

Este trabalho está inserido no contexto do projeto Poli-libras, um tradutor de português para LIBRAS, a língua de sinais utilizados pelos surdos no Brasil.

O Poli-libras está sendo construído não como um mero tradutor palavra por palavra, mas considerando a necessária adaptação da sintaxe da sentença entre as línguas em questão, sendo o primeiro passo para essa adequação a análise sintática da frase em português.

A análise sintática é feita com base em uma gramática livre de contexto, sendo inicialmente adotada a gramática apresentada por **Erro! Fonte de referência não encontrada.**; o entanto já se verificou as limitações deste modelo.

Sendo a qualidade das transformações sintáticas altamente dependente do reconhecimento dos padrões sintáticos a serem transformados, iniciamos dentro de nosso trabalho uma pesquisa sobre como obter uma gramática que melhor modele a língua portuguesa.

A ideia central é produzir essa gramática de forma automática a partir do processo de inferência, o que consiste na análise de um texto para a geração de uma gramática que reconheça completamente este texto base. Uma vez que a gramática gerada gera perfeitamente o texto base, espera-se que modele razoavelmente a língua portuguesa. Há vários trabalhos desenvolvidos na área de indução de gramáticas; no entanto, nos trabalhos analisados como em **Erro! Fonte de eferência não encontrada.**, não há um foco na validação da qualidade das gramáticas geradas para linguagens naturais.

Dessa forma objetivamos estruturar um processo para realizar tal validação, sendo que adotaremos como algoritmo a ser validado o algoritmo adaptativo de indução de gramáticas livre de contexto apresentado por **Erro! Fonte de referência ão encontrada.**.

O artigo encontra-se organizado da seguinte forma: na seção II, discorre-se sobre linguagens formais e gramáticas; na seção III é dada uma visão sobre adaptatividade; a seção IV apresenta o algoritmo que pretendemos utilizar para induzir a gramática; a seção V versa sobre os métodos de avaliação da gramática induzida e a seção VI apresenta as conclusões.

## II. Gramáticas para linguagens naturais

Uma linguagem formal é um conjunto de cadeias, onde cada cadeia pode ser formada combinando-se os símbolos de um alfabeto Σ, formando então um subconjunto de Σ* (todas as combinações possíveis de símbolos). Uma gramática é um conjunto de regras que definem a formação dessas cadeias, portanto uma gramática define uma linguagem. Assim sendo, existe uma relação de equivalência entre gramáticas e linguagens, dado um deles existe sempre pelo menos um correspondente do outro.

O linguista Noam Chomsky, em 1959, definiu uma hierarquia com 4 tipos de gramáticas (linguagens), sendo a de tipo 0 (linguagens recursivamente enumeráveis) a que não apresenta nenhuma restrição e cada tipo subsequente: 1 (linguagens sensíveis ao contexto), 2 (linguagens livres de contexto) e 3 (linguagens regulares) mais restritivo que o anterior. Ou seja, tanto para gramáticas como para linguagens:

$$tipo3 \subseteq tipo2 \subseteq tipo1 \subseteq tipo0$$ **Erro! Fonte de eferência não encontrada.**.

Além da correspondência entre linguagens e gramáticas, há ainda a correspondência desses com os reconhecedores, que aceitam sentenças pertencentes à linguagem correspondente e rejeitam as sentenças não correspondentes à linguagem; as

G. C. Januário, L. A. F. Leite e M. L. Koga são engenheiros formados de Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo.
João José Neto é o responsável pelo LTA – Laboratório de Linguagens e Tecnologia Adaptativa, vinculado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo– Av. Prof. Luciano Gualberto, trav. 3, No. 158 – Cidade Universitária – São Paulo - SP - Brasil. (joao.jose@poli.usp.br) fone: (11) 3091-5402

correspondências são: linguagem regular ↔ autômato finito; linguagem livre de contexto ↔ autômato de pilha; linguagem recursivamente enumerável ↔ máquina de Turing.

Uma gramática deve ser capaz de produzir todas as sentenças sintaticamente possíveis da linguagem correspondente e deve ser incapaz de gerar sentenças sintaticamente inválidas para a mesma linguagem. No entanto, devido à enorme complexidade das linguagens naturais, não se consegue encontrar uma gramática perfeita que representa uma língua, sendo qualquer gramática definida no contexto de linguagens naturais uma aproximação da realidade, ou seja, gerará e reconhecerá algumas sentenças inválidas.

Trabalhos de linguistas têm procurado analisar a estrutura sintática de sentenças através de árvores sintáticas **Erro! onte de referência não encontrada.**, o que nos leva ao uso das gramáticas livres de contexto, pois essas correspondem a autômatos de pilha, que são capazes de reconhecer estruturas em árvores. O próprio Chomsky também desenvolveu a teoria de *gramáticas transformativas* para efetuar análises e transformações sintáticas de linguagens naturais baseando-se em gramáticas livre de contexto **Erro! Fonte de referência ão encontrada.**.

Além disso, o uso de gramática livre de contexto para descrever uma linguagem natural, que a princípio parece corresponder a uma gramática irrestrita, se dá pelo fato de que o uso dos tipos superiores implicaria numa complexidade computacional muito elevada para o reconhecimento de sentenças, tornando impraticável seu uso em aplicações em que o tempo do processamento é relevante, caso da maioria delas.

Em **Erro! Fonte de referência não encontrada.** temos um rabalho de um linguista que procura gerar uma descrição do português com o uso de regras de produção de gramáticas livres de contexto, também apresentando as estruturas sintáticas analisadas na forma de árvores.

A proposta deste artigo é sugerir uma forma automatizada de gerar gramáticas livre de contexto que representem linguagens naturais. Espera-se que essas gramáticas geradas sejam melhores do que as gramáticas feitas de modo manual pelo esforço intelectual dos linguistas.

## III. Adaptatividade

Tecnologias adaptativas se referem a dispositivos que possuem um comportamento de acordo com uma dada entrada e que podem sofrer alterações nesse comportamento de acordo com a entrada fornecida. Essas alterações no comportamento são denominadas *ações adaptativas*.

Um desses dispositivos é o *autômato adaptativo* **Erro! onte de referência não encontrada.**, que consiste em um autômato finito com ações adaptativas associadas às suas transições. Essas ações adaptativas consistem na inclusão e remoção de transições e estados no autômato.

Com essas modificações, o autômato passa a ter o poder de uma Máquina de Turing, o que nos permite resolver problemas complexos com um reconhecedor mais simples (mais próximo de um autômato finito) em situações em que seria necessário um reconhecedor tradicional mais complicado.

Exemplos de problemas tradicionais que podem ser resolvidos mais eficientemente com o uso de adaptatividade podem ser encontrados em **Erro! Fonte de referência ão encontrada.**.

## IV. Algoritmo

O algoritmo que utilizaremos para inferir a gramática é o descrito em **Erro! Fonte de referência não ncontrada.**, que é uma extensão voltada para gramáticas livres de contexto do algoritmo proposto por **Erro! Fonte e referência não encontrada.** para inferência de gramáticas regulares. Aplicar-se-á esse algoritmo para análise sintática de linguagens naturais, especificamente o português. Vale ressaltar também que este artigo trabalha na criação de uma gramática no nível sintático, i.e., o alfabeto será constituído por um conjunto de classes morfológicas da língua, como "substantivo", "verbo", "artigo", etc., onde cada classe representa um símbolo do alfabeto.

O algoritmo se baseia em aprendizado positivo, ou seja, recebe um conjunto S+ de sentenças-exemplo que são sabidamente válidas na sintaxe portuguesa a partir do qual ele deve inferir um reconhecedor. A Fig.1 ilustra a funcionamento básico dele.

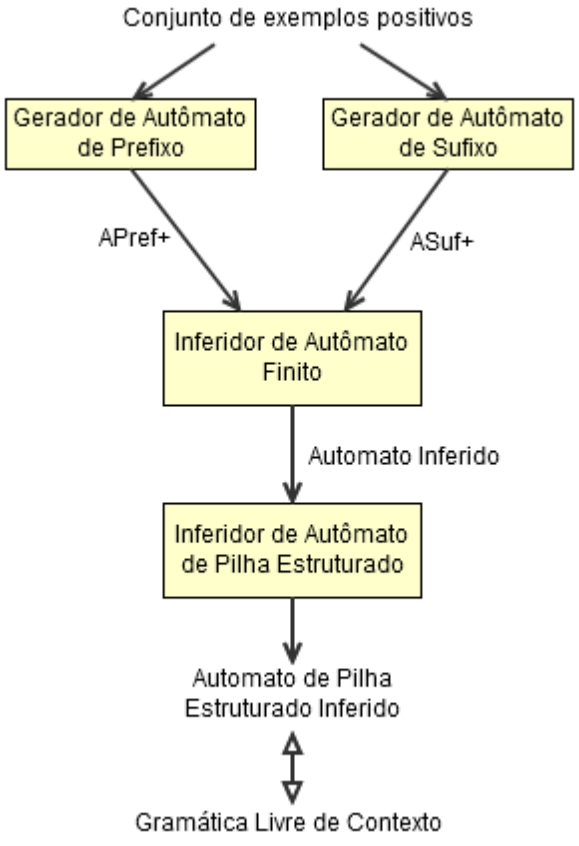



Fig. 1. Diagrama ilustrando o funcionamento geral do algoritmo adotado para a inferencia da gramática

Adaptativamente o autômato inicial evolui conforme a cadeia de entrada para um autômato finito (AF) reconhecedor

apenas da primeira sentença do conjunto S+. Esse passo é repetido para as demais sentenças, obtendo-se assim um reconhecedor exclusivo para o conjunto S+, ou seja, não há sobre-reconhecimento (reconhecimento de sentenças que não pertencem a linguagem). Esse mesmo passo é repetido com as mesmas cadeias de entrada, porém dessa vez invertidas, gerando assim um outro AF que reconhece as cadeias lidas de trás para frente. O primeiro autômato podemos chamar de autômato de prefixo (Apref+) e este segundo de autômato de sufixo (Asuf+).

Em posse destes dois autômatos, um novo reconhecedor é inferido, num algoritmo que une os dois através de suas semelhanças (se Apref+ possui uma transição para o símbolo 'a' entre o estado inicial e o estado $s_1$, então toda transição partindo do estado inicial de Asuf+ para entrada 'a' será equivalente, ou seja, todo estado destino dessas transições será equivalente a $s_1$).

Até esse passo, o autômato inferido é finito, ou seja, reconhece somente linguagens regulares. A extensão para ele aceitar linguagens livres de contexto se baseia na ideia de inferir ciclos presentes no autômato finito, construindo então um autômato de pilha estruturado (APE). Isso porque basicamente a diferença entre linguagens regulares e livres de contexto é que as livres de contexto conseguem representar a ideia de aninhamentos sintáticos. Resumidamente, o que o algoritmo faz é: primeiro procura o caminho mínimo entre o estado inicial e final e replica ele no APE final. Em seguida, procura por ciclos que se iniciam no caminho mínimo e voltam para ele. Esses ciclos formarão novas sub-máquinas do APE, e após a análise completa do do autômato inferido, o APE estará formado.

Aplicaremos então esse algoritmo descrito para um conjunto de sentenças em português que terão suas devidas classes morfológicas analisadas por alguma entidade confiável, verificando assim o seu funcionamento ao inferir a gramática da língua portuguesa. Após gerado o autômato, ele será avaliado como explicado na seção seguinte.

É importante salientar que embora estejamos interessados em analisar a gramática obtida, o algoritmo apresentado gera na verdade um autômato reconhecedor, que naturalmente equivale à gramática em questão. Os procedimentos que faríamos para testar a gramática seria submetê-la a um processo que a transformaria em um reconhecedor, portanto basta utilizar diretamente o dispositivo obtido pelo algoritmo nas etapas de reconhecimentos na avaliação.

## V. Implementação

Esse algoritmo descrito na seção anterior foi implementado utilizando-se a linguagem de programação Java, assim como um sistema validador de gramáticas[3]. Como subsídio para modelar autômatos e gramáticas, gerar e executar os reconhecedores usamos a biblioteca poli-sin[4], desenvolvido como parte do projeto Poli-Libras**Erro! Fonte de eferência não encontrada.**.

Para realizar a análise das gramáticas, o sistema implementado recebe uma descrição textual de uma gramática libre de contexto e faz a conversão para um autômato de pilha estruturado, gerando portanto seu reconhecedor.

Esse reconhecedor então, quando alimentado por uma cadeia de entrada (sequência de clases morfológicas no nosso caso), responderá se a sentença é aceita ou não, formando também sua árvore sintática. Desse modo, se montarmos reconhecedores a partir de diferentes gramáticas, podemos observar os resultados obtidos pro cada uma para efeitos de comparação.

Na nossa implementação do validador de gramáticas, as entradas são os textos puros (e não as classes gramaticais), passando por um analisador morfológico automático[5] que alimenta o analisador sintático com as classes gramaticais de cada palavra.

## VI. Avaliação da Gramática Induzida

Feita a inferência da gramática é preciso realizar testes que procurem medir a qualidade da gramática obtida.

Como já foi explicado, uma gramática ideal deveria ser capaz de produzir todas as sentenças sintaticamente possíveis da linguagem correspondente e ser incapaz de gerar sentenças sintaticamente inválidas para a mesma linguagem.

Logo, olhando o problema do ponto de vista de aceitação/rejeição de cadeias, um possível teste é a aplicação de um conjunto de frases sabidamente válidas e inválidas para se checar o quanto a gramática obtida se aproxima da gramática ideal.

Na literatura encontramos referências de *benchmarks* para a avaliação de gramáticas induzidas [10,11], porém estas avaliações não eram feitas com o foco de linguagens naturais, o que as torna impróprias para nossos propósitos.

Dessa forma adotaremos nossos próprios textos como treinamento para o algoritmo de indução; acreditamos que para esse propósito pequenos textos jornalísticos sejam mais adequados, por serem normalmente considerados como "português correto" e ao mesmo tempo estarem mais próximos da linguagem coloquial, uma vez que acabam não seguindo a risca a norma culta da língua.

Já os testes de aceitação/rejeição não serão feitos com textos, mas apenas com sentenças mais curtas, devido à complexidade e alto nível de ambiguidade que as linguagens naturais geram. Embora o objetivo do teste seja na verdade verificar a capacidade de a gramática gerar (ou não) as sentenças de testes, esse teste será efetuado na prática através de testes de aceitação/não-aceitação de dispostivos reconhecedores, uma vez que toda gramática livre de contexto possui um autômato de pilha correspondente capaz de aceitar todas as sentenças geradas pela gramática e de recusar todas as sentenças não geradas pela gramática.

A aplicação da avaliação se torna mais interessante quando os resultados são comparados com resultados de uma gramática de referência, sendo que no caso propomos o uso da

[3] *Grammar-analyser*. Disponível em: http://code.google.com/p/grammar-analyser/ (acesso em 17/01/2011)

[4] Projeto *Poli-sin*. Disponível em: www.polilibras.com.br/desenvolvimento/analisador-sintatico (acesso em 17/01/2011)

[5] *JJSpell*. Disponível em: http://code.google.com/p/jjspell/ (acesso em 17/01/2011)

gramática de **Erro! Fonte de referência não ncontrada.**, para mostrarmos a contraposição de gramáticas feitas manualmente e geradas automaticamente.

De início é melhor aplicar textos do mesmo estilo do texto que serviu de base para a inferência, mas podemos também testar o alcance da validade da gramática para estilos diferentes (ex: gramática induzida com texto jornalístico e sentenças mais literárias para serem reconhecidas);

Outra diferença a ser analisada é o tamanho do texto base para a inferência, sendo interessante comparar textos pequenos contra textos maiores.

Aplicando esses testes teremos como resultados tabelas que mostram os parâmetros dos testes (variações propostas) e os resultados em termo de porcentagem de sentenças corretamente aceitas ou rejeitadas.

## VII. Resultados

Nos nossos testes, decidimos a princípio avaliar somente considerando o universo das orações absolutas (períodos simples), acreditando que elas já forneceriam resultados significativos e deixando a extensão para períodos compostos para trabalhos futuros.

Uma vez implementados os algoritmos, o primeiro passo é realizar a indução da gramática. Para essa tarefa, utilizamos um conjunto de 100 frases analisadas morfologicamente manualmente (análise supervisionada por humanos), ou seja, a cada palabra atribuiu-se a sua classe gramatical correspondente pertinente. A análise foi manual pois deve-se garantir que os exemplos de treinamento estejam corretos (o que não se pode certificar com analisadores automáticos). Somente exemplos positivos foram utilizados.

Em posse da gramática inferida e também da gramática produzida por especialistas (recorte e interpretação pelos autores das regras apresentadas por Luft **Erro! Fonte de eferência não encontrada.**, doravante chamada de gramática baseada Luft), ambas foram submetidas a um teste de aceitação/rejeição no validador de gramáticas. Foram usadas 50 sentenças sintaticamente corretas e 50 sentenças incorretas para alimentar o validador. Os resultados podem ser observados na Tabela I.

TABELA I
RESULTADOS DO TESTE DE ACEITAÇÃO/REJEIÇÃO.

| | **Gramática baseada Luft** | **Gramática inferida** |
|---|---|---|
| Taxa de aceitação das sentenças corretas | 24,1 % | 90,7 % |
| Taxa de rejeição das sentenças erradas | 82,7 % | 0 % |

Pode-se observar que a taxa de aceitação foi bem alta na gramática inferida ao passo que a taxa de rejeição foi bem melhor na gramática baseada em Luft. Isso se explica pelo fato do treinamento ter sido feito somente com exemplos positivos, que atribuiu à gramática inferida uma natureza muito mais receptiva. Além disso, nosso analisador sintático foi construído para buscar a aceitação, pois verifica todas as combinações possíveis dentre as possibilidades de categorias gramaticais retornadas pelo analisador morfológico, ou seja, uma frase em linguagem natural que estivesse sintaticamente errada poderia gerar algumas possibilidades corretas pelo fato do analisador morfológico ser automático. A gramática baseada em Luft tem menor taxa de aceitação pois de fato ela era bem simplificada, e só aceitava sentenças na ordem direta (sujeito-verbo-objeto).

## VIII. Conclusões

Neste trabalho verificamos a necessidade de realizar uma avaliação de gramáticas que modelem linguagens naturais, uma vez que gramáticas livre de contextos são apenas modelagens imperfeitas das estruturas sintáticas das linguagens naturais.

Salientamos que essa validação não deve considerar apenas a capacidade de geração de sentenças corretas (assim como a não-geração de sentenças incorretas), mas também como a capacidade de ser utilizada para a criação de uma estrutura sintática coerente com os conceitos linguísticos, i.e., gerar uma árvore sintática adequada, dada uma sentença.

Após a análise dos testes, verificamos que na questão de aceitação de sentenças a gramática inferida desempenhou bem, mas falta aprimorar com o uso de exemplos negativos.

Além do teste de aceitação/rejeição, para a aplicação no campo de linguagens naturais, deixamos para trabalhos futuros um teste qualitativo da capacidade da gramática induzida representar uma estrutura sintática condizente com a classificação da linguística, o que quer dizer que em posse de uma gramática inferida devemos ser capazes de relacionas os não-terminais da gramática com estruturas sintáticas convencionais, tais como sintagmas nominais, sintagmas verbais etc.

A adaptadividade mostra-se promissora como uma nova alternativa no campo do processamento de linguagem natural e estudos com uso dessas técnicas ainda estão no começo.

## Referências


[1] C. Luft. "Moderna gramática brasileira" - 2a edição, 2002.

[2] I. P. Matsuno. "Um Estudo do Processo de Inferência de Gramáticas Regulares e Livres de Contexto Baseados em Modelos Adaptativos". Dissertação de Mestrado, EPUSP, São Paulo, 2006.

[3] N. Chomsky. "On certain formal properties of grammars". *Information and Control* **2** (2): 137–167

[4] J. L. Fiorin, (org.). "Introdução à Lingüística II: Princípios de análise". 4. ed. – São Paulo: Contexto, 2005.

[5] J. Friedman, "A Computer Model of Transformational Grammar". 1971.

[6] J. J. Neto. "Adaptive automata for context-dependent languages". ACM SIGPLAN Notices, 1994.

[7] J. J. Neto. "Solving complex problems with Adaptive Automata". Lecture Notes in Computer Science. S. Yu, A. Paun (Eds.): *Implementation and Application of Automata 5th International Conference, CIAA 2000*, Vol.2088, London, Canada, Springer-Verlag, 2000, pp.340.

[8] J. J. Neto.; IWAI, M. K. "Adaptative Automata for Syntax Learning". Anais Conferência Latino Americana de Informática – CLEI 1998 MEMORIAS. pp. 135-149, Quito, Equador, 1998.

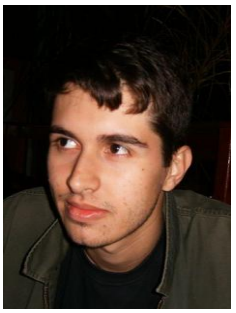
[9] M. L. Koga, G. C. Januário, L. A. F. Leite. "Poli-Libras: um tradutor de português para LIBRAS". Trabalho de Formatura, EPUSP, São Paulo, 2010.

[10] M. Tomita. "Dynamic construction of finite-automata from examples using hill-climbing". In: 4th Annual Cognitive Science Conference, pages 105-108, 1982.

[11] P. Dupont. "Regular grammatical inference from positive and negative samples by genetic search : the GIG method". In: Grammatical Inference and Applications, ICGI'94, number 862. In: Lecture Notes in Artificial Intelligence, pages 236-245. Springer Verlag, 1994.

**Guilherme Carvalho Januário** nasceu em Jundiaí, em setembro de 1987, tendo se graduado em Engenharia Elétrica com ênfase em Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP) em 2010. Atualmente participa do programa de pós-graduação da mesma instituição e atua no mercado projetando e desenvolvendo sistemas voltados à georreferência.

De 2008 a 2009 trabalhou na Orientação Pedagógica da EPUSP, concebendo aplicativos no estilo *applets*. No último ano também se dedicou ao desenvolvimento do Poli-Libras, sob a orientação do prof. João José Neto.



**Leonardo Leite** Nascido em Sorocaba, em 1987, graduou-se em Engenharia Elétrica – ênfase Computação pela Escola Politécnica da Universidade de São Paulo em 2010. Atualmente está no programa de mestrado em Ciências da Computação no Instituto de Matemática e Estatística – USP, cujo estudo versa sobre a área de composição de serviços.

De 2008 a 2009 desenvolveu trabalhos no LTS – Laboratório de Tecnologia de Software do PCS – Departamento de Computação e Sistemas Digitais na área de engenharia de software. No último ano se dedicou ao desenvolvimento do Poli-Libras, um tradutor de Português para LIBRAS (Língua de Sinais Brasileira), sob a orientação do prof. João José Neto, paralelamente a atuação no mercado em empresa desenvolvedora de sistemas M2M (machine to machine).

O engenheiro publicou, dentre outros, artigo no I SIMTVD - Simpósio Internacional de TV Digital (Unesp, Bauru) sobre arquitetura de componentes.

**Marcelo Li Koga** nasceu em São Paulo, Brasil, em 1986. É graduado em Engenharia Elétrica com ênfase em Computação (2010) e atualmente é estudante de Mestrado em Inteligência Artificial, ambos pela Escola Politécnica da Universidade de São Paulo.

Já atuou profissionalmente na área de tecnologia da informação na Claro e hoje trabalha com desenvolvimento de sistemas na Touch Tecnologia. Como trabalho de conclusão do curso de engenharia, desenvolveu o Poli-Libras, um tradutor de Português para LIBRAS.

O engenheiro Koga teve em 2007 um artigo publicado no "ISA SHOW 2007 – XI Congresso Internacional e Exposição Sul-Americana de Automação, Sistemas e Instrumentação" sobre seu trabalho de iniciação científica em sistemas de monitoramento cardíaco. Em 2008 atuou como representante discente do PCS (Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP).

**João José Neto** Graduado em Engenharia de Eletricidade (1971), mestrado em Engenharia Elétrica (1975) e doutorado em Engenharia Elétrica (1980), e livre-docência (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente é professor associado da Escola Politécnica da Universidade de São Paulo, e coordena o LTA - Laboratóirio de Linguagens e Tecnologia Adaptativa do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseados em tecnologia adaptativa.

# Tecnologia Adaptativa Aplicada ao Processamento da Linguagem Natural

Ana Contier, Djalma Padovani, João José Neto

***Resumo*— Este trabalho faz uma breve revisão dos conceitos de Tecnologia Adaptativa, apresentando seu mecanismo de funcionamento e seus principais campos de aplicação, destacando o forte potencial de sua utilização no processamento de linguagens naturais. Em seguida são apresentados os conceitos de processamento de linguagem natural, ressaltando seu intricado comportamento estrutural. Por fim, é apresentado o Linguístico, uma proposta de reconhecedor gramatical que utiliza autômatos adaptativos como tecnologia subjacente.**

***Palavras Chave*— *Autômatos Adaptativos, Processamento de Linguagem Natural, Reconhecedores Gramaticais, Gramáticas Livres de Contexto***

## I. Autômatos Adaptativos

O autômato adaptativo é uma máquina de estados à qual são impostas sucessivas alterações resultantes da aplicação de ações adaptativas associadas às regras de transições executadas pelo autômato [1]. Dessa maneira, estados e transições podem ser eliminados ou incorporados ao autômato em decorrência de cada um dos passos executados durante a análise da entrada. De maneira geral, pode-se dizer que o autômato adaptativo é formado por um dispositivo convencional, não-adaptativo, e um conjunto de mecanismos adaptativos responsáveis pela auto-modificação do sistema.

O dispositivo convencional pode ser uma gramática, um autômato, ou qualquer outro dispositivo que respeite um conjunto finito de regras estáticas. Este dispositivo possui uma coleção de regras, usualmente na forma de cláusulas if-then, que testam a situação corrente em relação a uma configuração específica e levam o dispositivo à sua próxima situação. Se nenhuma regra é aplicável, uma condição de erro é reportada e a operação do dispositivo, descontinuada. Se houver uma única regra aplicável à situação corrente, a próxima situação do dispositivo é determinada pela regra em questão. Se houver mais de uma regra aderente à situação corrente do dispositivo, as diversas possíveis situações seguintes são tratadas em paralelo e o dispositivo exibirá uma operação não determinística.

Os mecanismos adaptativos são formados por três tipos de ações adaptativas elementares: consulta (inspeção do conjunto de regras que define o dispositivo), exclusão (remoção de alguma regra) e inclusão (adição de uma nova regra). As ações adaptativas de consulta permitem inspecionar o conjunto de regras que definem o dispositivo em busca de regras que sigam um padrão fornecido. As ações elementares de exclusão permitem remover qualquer regra do conjunto de regras. As ações elementares de inclusão permitem especificar a adição de uma nova regra, de acordo com um padrão fornecido.

Autômatos adaptativos apresentam forte potencial de aplicação ao processamento de linguagens naturais, devido à facilidade com que permitem representar fenômenos linguísticos complexos tais como dependências de contexto. Adicionalmente, podem ser implementados como um formalismo de reconhecimento, o que permite seu uso no pré-processamento de textos para diversos usos, tais como: análise sintática, verificação de sintaxe, processamento para traduções automáticas, interpretação de texto, corretores gramaticais e base para construção de sistemas de busca semântica e de aprendizado de línguas auxiliados por computador.

Diversos trabalhos confirmam a viabilidade prática da utilização de autômatos adaptativos para processamento da linguagem natural. É o caso, por exemplo, de [2], que mostra a utilização de autômatos adaptativos na fase de análise sintática; [3] que apresenta um método de construção de um analisador morfológico e [4], que apresenta uma proposta de autômato adaptativo para reconhecimento de anáforas pronominais segundo algoritmo de Mitkov.

## II. Processamento da Linguagem Natural: Revisão da Literatura

O processamento da linguagem natural requer o desenvolvimento de programas que sejam capazes de determinar e interpretar a estrutura das sentenças em muitos níveis de detalhe. As linguagens naturais exibem um intricado comportamento estrutural visto que são profusos os casos particulares a serem considerados. Uma vez que as linguagens naturais nunca são formalmente projetadas, suas regras sintáticas não são simples nem tampouco óbvias e tornam, portanto, complexo o seu processamento computacional. Muitos métodos são empregados em sistemas de processamento de linguagem natural, adotando diferentes paradigmas, tais como métodos exatos, aproximados, pré-definidos ou interativos, inteligentes ou algorítmicos [5]. Independentemente do método utilizado, o processamento da linguagem natural envolve as operações de análise léxico-morfológica, análise sintática, análise semântica e análise pragmática [6].

A análise léxico-morfológica procura atribuir uma classificação morfológica a cada palavra da sentença, a partir das informações armazenadas no léxico [7]. O léxico ou dicionário é a estrutura de dados contendo os itens lexicais e as informações correspondentes a estes itens. Entre as informações associadas aos itens lexicais, encontram-se a categoria gramatical do item, tais como substantivo, verbo e adjetivo, e os valores morfo-sintático-semânticos, tais como gênero, número, grau, pessoa, tempo, modo, regência verbal ou nominal. Um item lexical pode ter uma ou mais

representações semânticas associadas a uma entrada. É o caso da palavra "casa", que pode aparecer das seguintes formas:

Casa: substantivo, feminino, singular, normal. Significado: moradia, habitação, sede

Casa: verbo singular, 3a pessoa, presente indicativo, 1ª conjugação. Significado: contrair matrimônio

Dada uma determinada sentença, o analisador léxico-morfológico identifica os itens lexicais que a compõem e obtém, para cada um deles, as diferentes descrições correspondentes às entradas no léxico. A ambiguidade léxico-morfológica ocorre quando uma mesma palavra apresenta diversas categorias gramaticais. Neste caso existem duas formas de análise: a tradicional e a etiquetagem. Pela abordagem tradicional, todas as classificações devem ser apresentadas pelo analisador, deixando a resolução de ambiguidade para outras etapas do processamento. Já pela etiquetagem (POS *Tagging*), o analisador procura resolver as ambiguidades sem necessariamente passar por próximas etapas de processamento. Nesta abordagem, o analisador recebe uma cadeia de itens lexicais e um conjunto específico de etiquetas como entrada e produz um conjunto de itens lexicais com a melhor etiqueta associada a cada item. Os algoritmos para etiquetagem fundamentam-se em dois modelos mais conhecidos: os baseados em regras e os estocásticos. Os algoritmos baseados em regras usam uma base de regras para identificar a categoria de um item lexical, acrescentando novas regras à base à medida que novas situações de uso do item vão sendo encontradas. Os algoritmos baseados em métodos estocásticos costumam resolver as ambiguidades através de um corpus de treino marcado corretamente, calculando a probabilidade que uma palavra terá de receber uma etiqueta em um determinado contexto.

O passo seguinte é a análise sintática. Nesta etapa, o analisador verifica se uma sequência de palavras constitui uma frase válida da língua, reconhecendo-a ou não. O analisador sintático faz uso de um léxico e de uma gramática, que define as regras de combinação dos itens na formação das frases. A gramática adotada pode ser escrita por meio de diversos formalismos. Segundo [7] destacam-se as redes de transição, as gramáticas de constituintes imediatos (PSG ou phrase structure grammar), as gramáticas de constituintes imediatos generalizadas (GPSG) e as gramáticas de unificação funcional (PATR II e HPSG). As gramáticas de constituintes imediatos (PSG), livres de contexto, apresentam a estrutura sintática das frases em termos de seus constituintes. Por exemplo, uma frase (F) é formada pelos sintagmas nominal (SN) e verbal (SV). O sintagma nominal é um agrupamento de palavras que tem como núcleo um substantivo (Subst) e o sintagma verbal é um agrupamento de palavras que tem como núcleo um verbo. Substantivo e verbo representam classes gramaticais. O determinante (Det) compõe, junto com o substantivo, o sintagma nominal. O sintagma verbal é formado pelo verbo, seguido ou não de um sintagma nominal. O exemplo apresentado ilustra uma gramática capaz de reconhecer a frase: O menino usa o chapéu.

F → SN SV.
SN → Det Subst.
SV → Verbo SN.
Det → o
Subst → menino, chapéu
Verbo → usa

Considerando o processamento da direita para esquerda e de baixo para cima, a frase seria analisada da seguinda forma:

O menino usa o chapéu
1. chapéu = Subst: O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo: O menino verbo SN
5. verbo SN=SV: O menino SV
6. menino = Subst: O subst SV
7. O = Det: Det subst SV
8. Det subst= SN: SN SV
9. SN SV= F: F (Aceita)

No entanto, este formalismo não consegue identificar questões de concordância de gênero e número. Por exemplo, se fossem incluídos no léxico o plural e o feminino da palavra menino, frases como: "O meninos usa o chapéu." e "O menina usa o chapéu." seriam aceitas. Por exemplo:

O meninos usa o chapéu
1. chapéu = Subst: O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo: O menino verbo SN
5. verbo SN=SV: O menino SV
6. meninos = Subst: O subst SV
7. O = Det: Det subst SV
8. Det subst= SN: SN SV
9. SN SV= F: F (Aceita)

Para resolver este tipo de problema existem outros formalismos, tais como o PATR II:

F → SN, SV
  <SN numero> = <SV numero>
  <SN pessoa> = <SV pessoa>
SN → Det, Subst
  <Det numero> = <Subst numero>
  <Det genero > = <Subst genero>
SV → Verbo, SN
o
  <categoria> = determinante
  <genero> = masc
  <numero> = sing
menino
  <categoria> = substantivo
  <genero> = masc
  <numero> = sing
chapéu
  <categoria> = substantivo
  <genero> = masc
  <numero> = sing
usa
  <categoria> = verbo
  <tempo> = pres
  <numero> = sing
  <pessoa> = 3
  <argumento 1> = SN
  <argumento 2> = SN

Neste formalismo, a derivação leva em consideração outras propriedades do léxico, além da categoria gramatical, evitando os erros de reconhecimento apresentados anteriormente. Segundo [7], esse formalismo gramatical oferece poder gerativo e capacidade computacional, e tem sido usado com sucesso em ciência da computação, na especificação de linguagens de programação. Aplicando este formalismo ao exemplo acima, o erro de concordância seria identificado e a frase não seria aceita:

O meninos usa o chapéu
1. chapéu = Subst masc sing : O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo pres 3a. Pessoa sing: O menino verbo SN
5. verbo SN=SVsing: O menino SVsing
6. meninos = Subst masc plural: O subst SVsing
7. O = Det: Det subst SVsing
8. Det subst= SNplur: SNplur SVsing
9. SNplur SVsing = Não aceita

Certas aplicações necessitam lidar com a interpretação das frases bem formadas, não bastando o conhecimento da estrutura, mas sendo necessário o conhecimento do significado dessas construções. Por exemplo, quando é necessário que respostas sejam dadas a sentenças ou orações expressas em língua natural, as quais, por exemplo, provoquem um movimento no braço de um robô. Ou quando é necessário extrair conhecimentos sobre um determinado tema a partir de uma base de dados textuais. Nos casos nos quais há a necessidade de interpretar o significado de um texto, a análise léxico-morfológica e a análise sintática não são suficientes, sendo necessário realizar um novo tipo de operação, denominada análise semântica [7].

Na análise semântica procura-se mapear a estrutura sintática para o domínio da aplicação, fazendo com que a estrutura ganhe um significado [8]. O mapeamento é feito identificando as propriedades semânticas do léxico e o relacionamento semântico entre os itens que o compõe. Para representar as propriedades semânticas do léxico, pode ser usado o formalismo PATR II, já apresentado anteriormente. Para a representação das relações entre itens do léxico pode ser usado o formalismo baseado em predicados: cada proposição é representada como uma relação predicativa constituída de um predicado, seus argumentos e eventuais modificadores. Um exemplo do uso de predicados é apresentado para ilustrar o processo de interpretação da sentença "O menino viu o homem de binóculo". Trata-se de uma sentença ambígua da língua portuguesa, uma vez que pode ser interpretada como se (a) O menino estivesse com o binóculo, ou (b) O homem estivesse com o binóculo. Uma gramática para a análise do exemplo acima é dada pelas seguintes regras de produção:

F → SN SV
SN → Det Subst
SN → SN SP
SV → V SN
SV → V SN SP
SP → Prep Subst

Uma possível representação semântica para as interpretações da sentença seria:

I. Sentença de interpretação (a):
agente(ação(ver), menino)
objeto(ação(ver), homem)
instrumento(ação(ver), binóculo)

II. Sentença de interpretação (b):
agente(ação(ver), menino)
objeto(ação(ver), homem)
qualificador(objeto(homem), binóculo)

Existem casos em que é necessário obter o conteúdo não literal de uma sentença, ligando as frases entre si, de modo a construir um todo coerente, e interpretar a mensagem transmitida de acordo com a situação e com as condições do enunciado [7]. Por exemplo, para uma compreensão literal da sentença: "O professor disse que duas semanas são o tempo necessário", é possível recorrer aos mecanismos de representação expostos até aqui, porém para uma compreensão aprofundada, seria necessário saber a que problema se refere o professor, já que o problema deve ter sido a própria razão da formulação dessa sentença. Nestes casos, é necessária uma nova operação denominada análise pragmática.

A análise pragmática procura reinterpretar a estrutura que representa o que foi dito para determinar o que realmente se quis dizer [2]. Dois pontos focais da pragmática são: as relações entre frases e o contexto. À medida que vão sendo enunciadas, as sentenças criam um universo de referência, que se une ao já existente. A própria vizinhança das sentenças ou dos itens lexicais também constitui um elemento importante na sua interpretação. Assim, alguns novos fenômenos passam a ser estudados, como fenômenos pragmático-textuais. Inserem-se nessa categoria as relações anafóricas, co-referência, determinação, foco ou tema, dêiticos e elipse [7]. Por exemplo, nem sempre o caráter interrogativo de uma sentença expressa exatamente o caráter de solicitação de uma resposta. A sentença "Você sabe que horas são?" pode ser interpretada como uma solicitação para que as horas sejam informadas ou como uma repreensão por um atraso ocorrido. No primeiro caso, a pergunta informa ao ouvinte que o falante deseja obter uma informação e, portanto, expressa exatamente o caráter interrogativo. Entretanto, no segundo caso, o falante utiliza o artifício interrogativo como forma de impor sua autoridade. Diferenças de interpretação desse tipo claramente implicam interpretações distintas e, portanto, problemáticas, se não for considerado o contexto de ocorrência do discurso [9]. As questões relacionadas à análise pragmática são objetos de estudos de modo a prover mecanismos de representação e de inferência adequados, e raramente aparecem em processadores de linguagem natural [7].

Em [10] são apresentados trabalhos de pesquisas em processamento de linguagem natural para a Língua Portuguesa tais como o desenvolvido pelo Núcleo Interinstitucional de Linguística Aplicada (NILC) no desenvolvimento de ferramentas para processamento de linguagem natural; o projeto VISL – Visual Interactive Syntax Learning, sediado na Universidade do Sul da Dinamarca, que engloba o desenvolvimento de analisadores morfossintáticos para diversas línguas, entre as quais o português; e o trabalho de resolução de anáforas desenvolvido pela Universidade de Santa Catarina. A tecnologia adaptativa também tem contribuído com trabalhos em processamento da linguagem natural. Em [11], são apresentadas algumas das pesquisas desenvolvidas pelo Laboratório de Linguagens e Tecnologia

Adaptativa da Escola Politécnica da Universidade de São Paulo: um etiquetador morfológico, um estudo sobre processos de análise sintática, modelos para tratamento de não-determinismos e ambigüidades, e um tradutor texto-voz baseado em autômatos adaptativos.

## III. Reconhecedor Adaptativo: Suporte Teórico Linguístico

A Moderna Gramática Brasileira de Celso Luft [12] foi escolhida como suporte teórico linguístico do reconhecedor aqui proposto. A escolha foi feita em função da forma clara e precisa com que Luft categoriza os diversos tipos de sentenças de língua portuguesa, se diferenciando das demais gramáticas que priorizam a descrição da língua em detrimento da análise estrutural da mesma.

Luft diz que a oração é moldada por padrões denominados frasais ou oracionais. Estes padrões são compostos por elementos denominados sintagmas. Sintagma é qualquer constituinte imediato da oração, podendo exercer papel de sujeito, complemento (objeto direto e indireto), predicativo e adjunto adverbial. É composto por uma ou mais palavras, sendo que uma é classificada como núcleo e as demais como dependentes. As palavras dependentes podem estar localizadas à esquerda ou à direita do núcleo. Luft utiliza os seguintes nomes e abreviaturas:

1. Sintagma substantivo (SS): núcleo é um substantivo;
2. Sintagma verbal (SV): núcleo é um verbo;
3. Sintagma adjetivo (Sadj): núcleo é um adjetivo;
4. Sintagma adverbial (Sadv): núcleo é um advérbio;
5. Sintagma preposicional (SP): é formado por uma preposição (Prep) mais um SS.
6. Vlig: verbo de ligação
7. Vi: verbo intransitivo
8. Vtd: verbo transitivo direto
9. Vti: verbo transitivo indireto
10. Vtdi: verbo transitivo direto e indireto
11. Vt-pred: verbo transitivo predicativo

A Tabela 1 apresenta os elementos formadores dos sintagmas, e a sequência em que aparecem, de acordo com Luft.

TABELA 1.
Elementos formadores de sintagmas [12]

| Sintagmas | |
|---|---|
| Substantivo | Quantitativos+Pronomes Adjetivos+<br>Sintagma Adjetivo1+Substantivo+<br>Sintagma Adjetivo2+<br>Sintagma Preposicional+<br>Oração Adjetiva |
| Verbal | Pré-verbais+<br>Verbo Auxiliar+<br>Verbo Principal |
| Adjetivo | Advérbio de Intensidade+<br>Adjetivo+<br>Sintagma Preposicional |
| Adverbial | Advérbio de Intensidade+<br>Adverbio+<br>Sintagma Preposicional |
| Preposicional | Preposição+<br>Sintagma Substantivo |

Um padrão oracional é determinado pelos tipos de sintagmas e pela sequência em que aparecem. Por exemplo, o padrão oracional SS Vlig SS, indica que a frase é composta por um sintagma substantivo, seguido de um verbo de ligação e de outro sintagma substantivo. A Tabela 2 apresenta a relação de todos os padrões oracionais propostos por Luft.

Os padrões são classificados em 5 tipos:

1. Padrões pessoais nominais: Neste caso, existe sujeito e o núcleo do predicado é um nome (substantivo, adjetivo, advérbio) ou um pronome (substantivo, adjetivo, advérbio). O verbo, nesses casos, é chamado de verbo de ligação (Vlig).

TABELA 2
Padrões oracionais de Luft [12]

| | | | | |
|---|---|---|---|---|
| Padrões Pessoais Nominais | | | | |
| SS | Vlig | SS | | |
| SS | Vlig | Sadj | | |
| SS | Vlig | Sadv | | |
| SS | Vlig | SP | | |
| Padrões Pessoais Verbais | | | | |
| SS | Vtd | SS | | |
| SS | Vti | SP | | |
| SS | Vti | Sadv | | |
| SS | Vti | SP | SP | |
| SS | Vtdi | SS | SP | |
| SS | Vtdi | SS | Sadv | |
| SS | Vtdi | SS | SP | SP |
| SS | Vi | | | |

2. Padrões pessoais verbais: São aqueles nos quais existe o sujeito e o núcleo do predicado é um verbo. O verbo pode ser transitivo direto (Vtd), transitivo indireto (Vti), transitivo direto e indireto (Vtdi), e intransitivo (Vi). Se o verbo for transitivo direto (Vtd), o complemento será um objeto direto; se o verbo for transitivo indireto (Vti), o complemento será um objeto indireto; se o verbo for transitivo direto e indireto (Vtdi), o complemento será um objeto direto e um indireto; se o verbo for intransitivo (Vi), não há complemento.

TABELA 2 - CONTINUAÇÃO

| | | | | |
|---|---|---|---|---|
| Padrões Pessoais Verbo-Nominais | | | | |
| SS | Vtpred | SS | SS | |
| SS | Vtpred | SS | Sadj | |
| SS | Vtpred | SS | SP | |
| SS | Vtpred | SS | Sadv | |
| SS | Vtpred | SS | | |
| SS | Vtpred | Sadj | | |
| SS | Vtpred | SP | | |
| Padrões Impessoais Nominais | | | | |
| | Vlig | SS | | |
| | Vlig | Sadj | | |
| | Vlig | Sadv | | |
| | Vlig | SP | | |
| Padrões Impessoais Verbais | | | | |

| | | | | |
|---|---|---|---|---|
| | Vtd | SS | | |
| | Vti | SP | | |
| | Vi | | | |

3. Padrões Pessoais Verbo-Nominais: Neste caso, existe o sujeito e o núcleo do predicado é um verbo transitivo predicativo (Vt-pred), cujo complemento é um objeto direto e um predicativo do objeto.

4. Padrões Impessoais Nominais: Ocorrem quando não existe sujeito e o núcleo do predicado é um nome (substantivo, adjetivo, advérbio) ou um pronome (substantivo, adjetivo, advérbio).

5. Padrões Impessoais Verbais: Neste caso, não existe sujeito e o núcleo do predicado é um verbo.

Luft apresenta uma gramática usada para análise sintática da Língua Portuguesa no modelo moderno, em que as frases são segmentadas o mais binariamente possível: Sujeito+Predicado; Verbo+Complemento; Substantivo+Adjetivo, etc. Neste modelo, a descrição explicita somente as classes analisadas; as funções ficam implícitas. Querendo explicar estas, Luft sugere que sejam escritas à direita das classes: SS:Sj (Sujeito), V:Núc (Núcleo do Predicado), PrA:NA (Adjunto Adnominal), etc.
A gramática proposta por Luft é a seguinte:

F → [Conec] [SS] SV [Conec]
Conec → F
SS → [Sadj] SS [Sadj | SP]
SS → [Quant | PrA] (Sc | Sp | PrPes)
SV→ [Neg] [Aux | PreV] (Vlig | Vtd| Vti | Vtdi| Vi)
[SS | Sadj | Sadv | SP] [SS| Sadj| Sadv| SP] [SP]
SP → Prep (SS | Sadj)
Sadj → Sadj [SP]
Sadj →[Adv] Adj
Sadv → Sadv [SP]
Sadv → [Adv] Adv
PrA → Ind | ArtDef | ArtInd| Dem| Pos
Sendo:
F – Frase
SS – Sintagma substantivo
SV – Sintagma verbal
SP – Sintagma preposicional
SN – Sintagma nominal
Sadv – Sintagma adverbial
Sadj – Sintagma adjetivo
Adv – adverbio
Adj – adjetivo
ArtDef – artigo definido
ArtInd – artigo indefinido
Aux – Partícula auxiliar (apassivadora ou pré-verbal)
Conec – Conector (conjunção ou pronome relativo)
Dem – pronome demonstrativo indefinido
Ind – pronome indefinido
Neg – partícula (negação)
PrA – pronome adjetivo
PrPes – pronome pessoal
Prep – preposição
Quant – numeral
Sc – substantivo comum
Sp – substantivo próprio
V– verbo
Vlig – verbo de ligação
Vi – verbo intransitivo
Vtd – verbo transitivo direto
Vti – verbo transitivo indireto
Vtdi – verbo transitivo direto e indireto

## IV. Proposta de um Reconhecedor Gramatical

O Linguístico é uma proposta de reconhecedor gramatical composto de 5 módulos sequenciais que realizam cada qual um processamento especializado, enviando o resultado obtido para o módulo seguinte, tal como ocorre em uma linha de produção, até que o texto esteja completamente analisado.

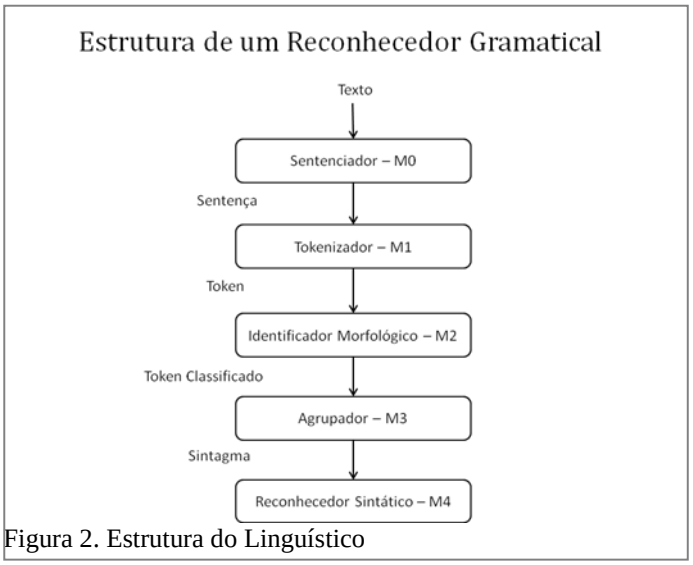


Figura 2. Estrutura do Linguístico

A Fig.2 ilustra a estrutura do Linguistico. O primeiro módulo, denominado Sentenciador, recebe um texto e realiza um pré-processamento, identificando os caracteres que possam indicar final de sentença, palavras abreviadas e palavras compostas, e eliminando aspas simples e duplas. Ao final, o Sentenciador divide o texto em supostas sentenças, para análise individual nas etapas seguintes.

O segundo módulo, denominado Tokenizador, recebe as sentenças identificadas na etapa anterior e as divide em *tokens*, considerando, neste processo, abreviaturas, valores monetários, horas e minutos, numerais arábicos e romanos, palavras compostas, nomes próprios, caracteres especiais e de pontuação final. Os *tokens* são armazenados em estruturas de dados (*arrays*) e enviados um a um para análise do módulo seguinte.

O terceiro módulo, denominado Identificador Morfológico, recebe os *tokens* da etapa anterior e os identifica morfologicamente, utilizando, como biblioteca de apoio, os textos pré-anotados do corpus Bosque[13], os verbos, substantivos e adjetivos que fazem parte da base de dados do TeP2.0 – Thesauro Eletrônico para o Português do Brasil [14] e as conjunções, preposições e pronomes disponíveis no Portal São Francisco[15], cujas informações provém da Wikipedia [16]. O Bosque é um conjunto de frases anotadas morfossintaticamente (conhecido por *treebank)*, composto por 9368 frases retiradas dos primeiros 1000 extratos dos corpora

CETEMPublico (Corpus de Extractos de Textos Electrónicos MCT/Público) e CETENFolha (Corpus de Extractos de Textos Electrónicos NILC/Folha de S. Paulo ). A Fig. 3 apresenta um fragmento do Bosque.

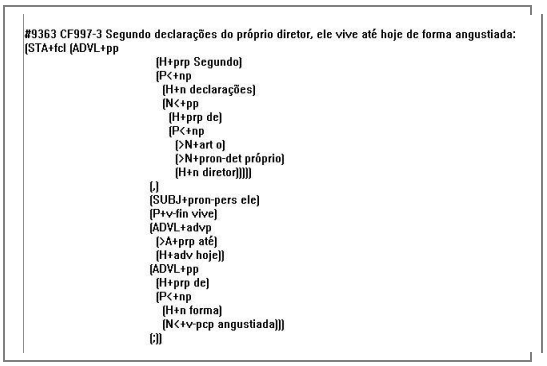



Figura 3. Exemplo de frase etiquetada do corpus Bosque [11].

O TeP2.0 é um um dicionário eletrônico de sinônimos e antônimos para o português do Brasil, que armazena conjuntos de formas léxicas sinônimas e antônimas. É composto por 19.888 conjuntos de sinônimos, 44.678 unidades lexicais, e 4.276 relações de antonímia, correspondendo a 22% da base [14]. A Fig 4 apresenta um fragmento da base de dados do TeP2.0.

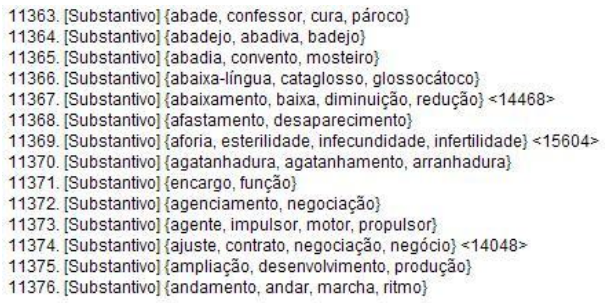



Figura 4. Fragmento da Base de Dados do TeP2.0[15].

O Portal São Francisco apresenta um curso online da Lingua Portuguesa e, entre seus módulos, encontra-se um sumário das classes morfológicas, no qual são encontrados exemplos de palavras e locuções mais comuns de cada classe.

Inicialmente, o Identificador Morfológico procura pela classificação morfologica dos tokens no léxico do Bosque, caso não a encontre, então ele procura na base de dados do TeP2.0 e no léxico do Portal São Francisco.

O padrão de etiquetas usado pelo Linguístico é o mesmo do Bosque, que apresenta, além da classificação morfológica, o papel sintático que o token exerce na sentença pré-anotada. Por exemplo, a etiqueta >N+art indica que o token é um artigo que está à esquerda de um substantivo (>N). A notação usa como gramática subjacente a Gramática Constritiva proposta por Fred Karlsson [17].

O léxico do Bosque foi organizado de modo a relacionar todas as classificações de um tokens ordenadas por frequencia em que ocorrem no texto pre-anotado. Por exemplo, o token "acentuada" está classificado com as seguintes etiquetas: N<+v-pcp e P+v-pcp, o que significa que, no texto pré-anotado, ele foi classificado como verbo no particípio (+v-pcp) antecedido de um substantivo (N<) e como verbo no particípio no papel de predicador (P).

No caso de ambiguidade, o Identificador Morfológico assume a classificação mais frequente como inicial e verifica se a classificação mais frequente do token seguinte é consistente com o que indica a etiqueta do token analisado. Se for, vale a classificação mais frequente, senão o Identificador analisa a próxima classificação, repetindo o algoritmo. Caso o algoritmo não retorne uma classificação única, o Identificador passa todas as classificações encontradas para os módulos seguintes, para que ambiguidade seja resolvida pelas regras gramaticais do reconhecedor.

O quarto módulo, denominado Agrupador é composto de um autômato, responsável pela montagem dos sintagmas a partir de símbolos terminais da gramática e um bigrama, responsável pela montagem dos sintagmas a partir de não-terminais (Figura 5). Inicialmente, o Agrupador recebe do Identificador as classificações morfológicas dos *tokens* e as agrupa em sintagmas de acordo com a gramática proposta por Luft. Neste processo são identificados sintagmas nominais, verbais, preposicionais, adjetivos e adverbiais Para isso, o Agrupador utiliza um autômato adaptativo cuja configuração completa é definida da seguinte forma:

Estados = {1, 2, 3, 4, SS, SP, V, Sadj, Sadv, A},
Onde:
1,2,3 e 4 = Estados Intermediários
SS, SP, V Sadj, Sadv = Estados nos quais houve formação de sintagmas, sendo:
SS= Sintagma substantivo
SP = Sintagma preposicional
V = Verbo ou locução verbal
Sadj = Sintagma adjetivo
Sadv = Sintagma adverbial
A = Estado após o processamento de um ponto final
*Tokens* = {art, num, n, v, prp, pron, conj, adj, adv, rel, pFinal, sClass}, onde:
art = artigo, num = numeral
n = substantivo, v = verbo
prp = preposição, pron = pronome
conj = conjunção, adj = adjetivo
adv = advérbio, rel = pronome relativo
pFinal = ponto final, sClass = sem classificação
Estados de Aceitação = {SS, SP, V, Sadj, Sadv, A}
Estado Inicial = {1}
Função de Transição = {(Estado, *Token*)→Estado}, sendo:
{(1, art)→2, (2, art)→2, (3, art)→3
(1, num)→Sadv, (2, num)→2, (3, num)→3
(1, n)→SS, (2, n)→SS, (3, n)→SP
(1, v)→SV, (2, v)→ SV, (3, v)→SP
(1, prp)→3, (2, prp)→2, (3, prp)→3

(1, prop)→SS, (2, prop)→SS, (3, prop)→SP
(1, pron)→SS, (2, pron)→SS, (3, pron)→SP
(1, conj)→conj, (2, conj)→Ø, (3, conj)→ Ø
(1, adj)→Sadj, (2, adj)→Sadj, (3, adj)→3
(1, adv)→Sadv, (2, adv)→2, (3, adv)→3
(1, rel)→conj, (2, rel)→ Ø, (3, rel)→conj
(1, pFinal)→A, (2, pFinal)→ Ø, (3, pFinal)→ Ø}

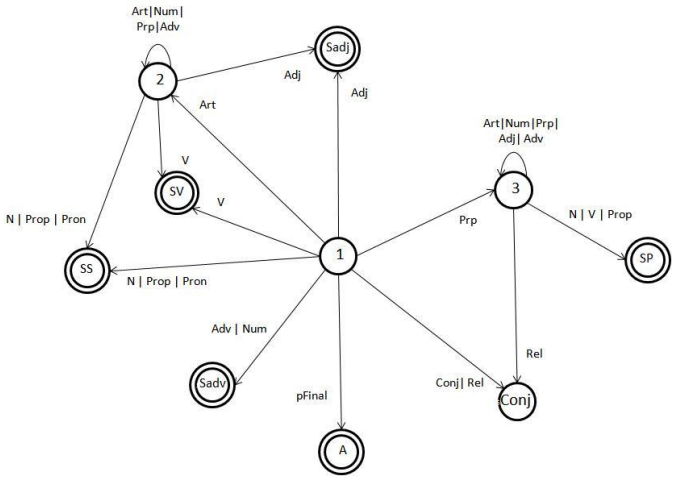



Figura 5. Configuração Completa do Autômato Construtor de Sintagmas.

Por exemplo, segundo a gramática de Luft, os sintagmas substantivos são obtidos através da seguinte regra:

SS → [Quant | PrA] (Sc | Sp | PrPes)

Pela regra acima, o conjunto de *tokens* "A" e "casa" formam um sintagma substantivo, da seguinte forma:

PrA = "A" (artigo definido)

Sc = "casa" (substantivo comum)

Da direita para esquerda, são realizadas as seguintes derivações:

PrA Sc → SS; A Sc → SS; A casa → SS

Já o Agrupador recebe o *token* "A", identificado pelo Tokenizador como artigo definido, e se movimenta do estado 1 para o estado 2. Ao receber o *token* "casa", identificado como substantivo comum, ele se movimenta do estado 2 para o estado SS, que é um estado de aceitação. Neste momento o Agrupador armazena a cadeia "A casa" e o símbolo "SS" em uma pilha e reinicializa o autômato preparando-o para um novo reconhecimento.

Em um passo seguinte, o Agrupador usa o bigrama para comparar um novo sintagma com o último sintagma formado, visando identificar elementos mais altos na hierarquia da gramática de Luft. Para isso ele usa a matriz apresentada na Tabela 3, construída a partir da gramática de Luft. A primeira coluna da matriz indica o último sintagma formado (US) e a primeira linha, o sintagma atual (SA). A célula resultante apresenta o novo nó na hierarquia da gramática.

TABELA 3
Matriz de agrupamento de sintagmas

| US \ SA | SS | SP | V | Sadv | Sadj | Conj |
|---|---|---|---|---|---|---|
| SS | SS | SS | - | - | SS | - |
| SP | SP | - | - | - | - | - |
| V | - | - | V | - | - | - |
| Sadv | - | - | - | Sadv | Sadj | - |
| Sadj | SS | Sadj | - | - | Sadj | - |
| Conj | - | - | - | - | - | Conj |

Esta técnica foi usada para tratar as regras gramaticais nas quais um sintagma é gerado a partir da combinação de outros, como é o caso da regra de formação de sintagmas substantivos: SS → [Sadj] SS [Sadj | SP]. Por esta regra, os sintagmas substantivos são formados por outros sintagmas substantivos precedidos de um sintagma adjetivo e seguidos de um sintagma adjetivo ou um sintagma preposicional. No exemplo anterior, supondo que os próximos 2 *tokens* fossem "de" e "madeira", após a passagem pelo autômato, o Agrupador formaria um sintagma SP. Considerando que na pilha ele tinha armazenado um SS, após a passagem pelo bigrama, e de acordo com a Tabela 3, o sintagma resultante seria um SS e o conteúdo que o compõe seria a combinação dos textos de cada sintagma que o originou. Caso não haja agrupamentos possíveis, o Agrupador envia o último sintagma formado para análise do Reconhecedor Sintático e movimenta o sintagma atual para a posição de último sintagma no bigrama, repetindo o processo com o próximo sintagma.

O quinto e último módulo, denominado Reconhecedor Sintático, recebe os sintagmas do módulo anterior e verifica se estão sintaticamente corretos de acordo com padrões gramaticais de Luft. O Reconhecedor Sintático utiliza um autômato adaptativo que faz chamadas recursivas sempre que recebe conjunções ou pronomes relativos, armazenando, em uma estrutura de pilha, o estado e a cadeia de sintagmas reconhecidos até o momento da chamada. Caso o Reconhecedor Sintático não consiga se movimentar a partir do sintagma recebido, ele gera um erro e retorna o ponteiro para o último sintagma reconhecido, finalizando a instância do autômato recursivo e retornando o processamento para aquela que a inicializou. Esta, por sua vez, retoma posição em que se encontrava antes da chamada e continua o processamento até o final da sentença ou até encontrar uma nova conjunção, situação na qual o processo se repete.

A configuração completa do autômato é definida da seguinte forma:

Estados = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27}

*Tokens* = {SS, SP, Vli, Vi, Vtd, Vti, Vtdi, Sadj, Sadv, Conj, A}

Estados de Aceitação = { 4, 5, 6, 9, 12, 13,14, 15, 17, 18, 19, 21, 22, 24, 25, 26, 27}

Estado Inicial = {1}

Função de Transição = {(Estado, *Token*)→Estado}, sendo:
{(1, SS)→2, (2, Vti)→3, (3, SP)→4, (4, SP)→4,
(3, Sadv)→5, (2, Vi)→6, (2, Vtdi)→7
(7, SS)→8, (8, SP)→9, (9, SP)→9, (8, Sadv)→10,
(2, Vlig)→11, (11, SP)→12, (11, Sadv)→13,
(11, Sadj)→ 14, (11, SS)→15, (2, Vtd)→16, (16, SS)→17,

(2, Vtpred)→18, (18, SP)→19, (18, Sadj)→20
(18, SS)→ 21, (21, SS)→22, (21, Sadj)→23, (21, Sadv)→24, (21, SP)→25}

Pilha = {[Texto, Sintagma, Estado]}

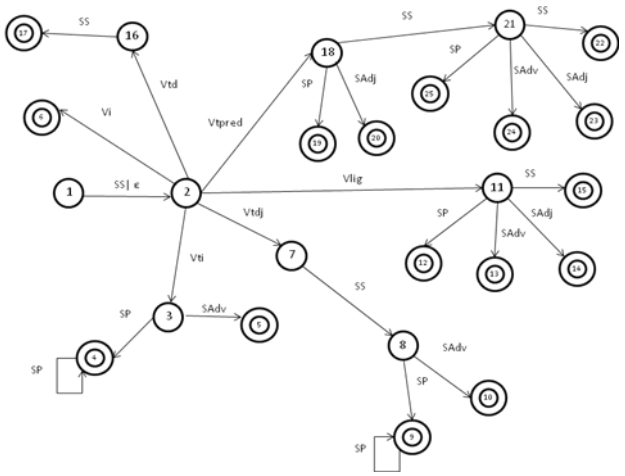



Figura 6.1. Configuração Completa do Reconhecedor Sintático.

No entanto, para que a análise sintática seja feita, não são necessárias todas as ramificações da configuração completa do autômato. Por exemplo, quando se transita um verbo de ligação a partir do estado 2, o autômato vai para o estado 11 e todas as demais ramificações que partem deste estado para os estados 3, 7, 16 e 18, não são usadas. Com a tecnologia adaptativa, é possível criar dinamicamente os estados e transições do autômato em função dos tipos de verbos, evitando manter ramificações que não são usadas.

A Fig. 6.2 apresenta a configuração inicial do autômato adaptativo equivalente ao autômato de pilha apresentado anteriormente. No estado 1, o autômato recebe os *tokens* e transita para o estado 2 quando processa um sintagma substantivo (SS) ou quando transita em vazio. No estado 2, o autômato transita para si mesmo quando recebe qualquer tipo de verbo: Vi, Vtd, Vlig, Vtpred, Vtdi e Vti. Todas as outras ramificações são criadas por meio de funções adaptativas chamadas em função do tipo de verbo processado.

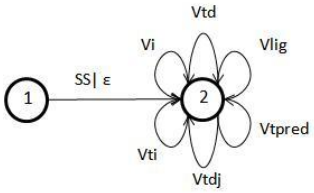



Figura 6.2. Configuração Inicial do Reconhecedor Gramatical.

Por exemplo, se o verbo é de ligação (Vlig), o autômato utiliza as funções adaptativas α (j) e β (o), definidas da seguinte forma:

α (j): { o* :
- [ (j, Vlig)]
+[ (j, Vlig) :→ o, β (o)]
}

β (o): { t*u*v*x*:
+ [ (o, SP) :→ t]
+ [ (o, Sadv) :→ u]
+ [ (o, Sadj) :→ v]
+ [ (o, SS) :→ x]

A função adaptativa α (j) é chamada pelo autômato antes de processar o *token*, criando o estado 11 e a produção que leva o autômato do estado 2 ao novo estado criado. Em seguida, o autômato chama a função β (o), criando os estados 12, 13, 14 e 15 e as produções que interligam o estado 11 aos novos estados. A Fig. 6.3 mostra a configuração do autômato após o processamento do verbo de ligação. Neste exemplo, o autômato criou apenas os estados 11, 12, 13, 14 e 15 e as respectivas transições, evitando alocar recursos que seriam necessários para criar o autômato completo, conforme apresentado na Fig. 6.1.

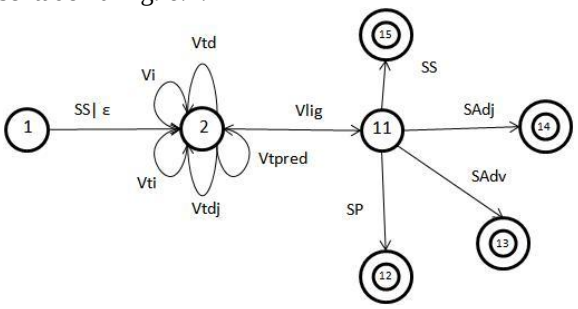



Figura 6.3. Configuração do autômato após o processamento do verbo de ligação.

Toda movimentação do autômato, assim como os sintagmas identificados em cada passagem e a classificação morfológica dos termos das sentenças, são armazenados em arquivos que podem ser acessados por um editor. Se o Linguistico não consegue reconhecer a sentença, ele registra os erros encontrados e grava uma mensagem alertando para o ocorrido.

## V. Considerações Finais

Este artigo apresentou uma revisão dos conceitos de Tecnologia Adaptativa e de Processamento da Linguagem Natural. Em seguida, foi apresentado o Linguístico, uma proposta de reconhecedor gramatical que utiliza autômatos adaptativos como tecnologia subjacente.

O Linguístico encontra-se em fase de contrução, dividida em etapas em função da estrutura do reconhecedor. A primeira versão do sentenciador e do tokenizador foram finalizadas e estão em fase de testes. Na próxima etapa, está prevista a construção do analisador morfológico que vai utilizar as sentenças e tokens gerados pelos módulos anteriores.

## VI. Referências

[1] NETO, J.J. Apresentação LTA-Laboratório de Linguagens e Técnicas Adaptativas. Disponível em: http://www.pcs.usp.br/~lta. Acesso 01/11/2009.

[2] [2] TANIWAKI, C. Formalismos adaptativos na análise sintática de linguagem natural . Dissertação de Mestrado, EPUSP, São Paulo, 2001.

[3] [3] MENEZES, C. E. Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos. Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo, 2000

[4] [4] PADOVANI, D. Uma proposta de autômato adaptativo para reconhecimento de anáforas pronominais segundo algoritmo de Mitkov. Workshop de Tecnologias Adaptativas – WTA 2009, 2009.

[5] [5] MORAES, M. de Alguns aspectos de tratamento sintático de dependência
de contexto em linguagem natural empregando tecnologia adaptativa, Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, 2006.

[6] [6] RICH, E.; KNIGHT, K. Inteligência Artificial, 2. Ed. São Paulo: Makron Books, 1993.

[7] [7] VIEIRA, R.; LIMA, V Linguística computacional: princípios e aplicações. IX Escola de Informática da SBC-Sul, 2001.
[8] [8] FUCHS, C., LE GOFFIC, P. Les Linguistiques Contemporaines.
[9] [9] NUNES, M. G. V. et al. Introdução ao Processamento das Línguas Naturais. Notas didáticas do ICMC Nº 38, São Carlos, 88p, 1999.
[10] Paris, Hachette, 1992. 158p.
[11] [10] SARDINHA, T. B. A Língua Portuguesa no Computador. 295p. Mercado de Letras, 2005.
[12] [11] ROCHA, R.L.A. Tecnologia Adaptativa Aplicada ao Processamento Computacional de Língua Natural. Workshop de Tecnologias Adaptativas – WTA 2007, 2007.
[13] [12] LUFT, C. Moderna Gramática Brasileira. 2ª. Edição Revista e Atualizada. 265p. Editora Globo, 2002.
[14] [13] Linguateca : http://www.linguateca.pt/
[15] [14]Tep2. Thesauro Eletrônico para o Português do Brasil. Disponível em: <http://www.nilc.icmc.usp.br/tep2/>
[16] [15]Portal São Francisco. Materiais de Língua Portuguesa. Disponível em: <http://www.portalsaofrancisco.com.br/alfa/materias/index-lingua-portuguesa.php/>
[16]Wikipedia. Disponível em:
<http://pt.wikipedia.org/wiki/P%C3%A1gina_principal/>
[17] KARLSSON, F. Constraint Grammar as a Framework for Parsing Running Text. 13o. International Conference on Computational Linguistics, Helsinki (Vol.3, pp.168-173).
[17]

# Gramática Adaptativa para Análise Sintática da Língua Portuguesa

J. M. N. dos Santos

***Resumo*—Este artigo apresenta uma proposta de geração de gramática da lingua Portuguesa através de formalismo adaptativo. O dispositivo adaptativo que será apresentado neste artigo tem dois módulos principais. O primeiro módulo do dispositivo é um reconhecedor sintático, que indica se uma frase fornecida foi ou não reconhecida como uma frase válida pela Gramática representada no momento. O segundo é o módulo de aprendizagem onde novas regras são aprendidas pelo dispositivo, abrangendo a sua capacidade de reconhecimento. Este trabalho está em andamento e faz parte da tese de doutorado do autor.**

***Palavras Chave*— Sistema em Linguagem natural, Tecnologia Adaptativa, Gramática, Análise Sintática em português, Processamento em Linguagem Natural.**

## I. NOMENCLATURA

*PLN:* significa processamento em linguagem natural.

Sintagma: "é qualquer constituinte imediato da oração, podendo exercer a função de sujeito, complemento, predicativo ou adjunto adverbial" [2]. Pode ser composto de uma ou mais palavras. Como exemplo podemos citar que a expressão "o poeta" é um sintagma substantivo, enquanto que a expressão "de muito valor" é um sintagma preposicional.

## II. INTRODUÇÃO

O processamento de linguagem natural é uma área de pesquisa da computação e sua característica principal é o tratamento de uma língua nativa como por exemplo o português, inglês e francês. É uma área multidisciplinar por envolver aspectos da ciência da computação, lingüística, lingüistica computacional e das ciências cognitivas.

Um dos grandes desafios da área de PLN é o desenvolvimento de sistemas e ferramentas que permitam que a comunicação do usuário com os aparelhos domésticos e computadores em geral possa ser efetuada na língua nativa de cada pessoa.

Uma das primeiras tarefas para entender o significado de uma frase em uma língua natural é realizada através da análise sintática, com base em uma descrição formal de uma gramática para a língua. A análise sintática é o processo de análise de um texto para determinar sua estrutura gramatical conforme uma determinada definição formal de gramática.

Este artigo apresentará uma forma de se obter uma gramática para a língua portuguesa, utilizando tecnologia adaptativa, posibilitando-se efetuar a análise sintática.

## III. PROCESSAMENTO EM LINGUAGEM NATURAL

Os avanços das pesquisas em *PLN* podem oferecer no futuro um conjunto de benefícios às pessoas. A seguir seguem alguns destes benefícios.

**Maior facilidade no uso de aparelhos eletrônicos domésticos.**

Com o avanço tecnológico e a diminuição do custo do *chip*, os aparelhos domésticos estão cada vez mais sofisticados: televisão, *DVD*, telefone celular, câmera fotográfica digital, etc. Os recursos destes aparelhos são muitas vezes sub-utilizado por uma parcela da população, devido à dificuldade de interação, que é feita através de menus das opções. Uma interação com os aparelhos utilizando uma língua nativa, ao invés de teclas e menus de funções, facilitaria o uso destes aparelhos.

**Maior acessibilidade às informações da rede mundial de computadores (internet)**

Outro aspecto a ser considerado é o crescimento mundial do uso da rede mundial (internet), inclusive nas camadas sociais menos favorecidas. Como boa parte das informações está em inglês, este fator dificulta o acesso à muita informação para a parcela da população com menos acesso à educação em países em desenvolvimento, como o Brasil. O uso de recursos de *PLN* poderá facilitar o uso da internet, sob os seguintes aspectos:

- tradução dos textos para a língua nativa das pessoas,
- buscas mais refinadas, facilitando o acesso à informação desejada e
- interface com usuário através da língua nativa, em substituição ao teclado e mouse utilizados atualmente.

**Otimização no serviço de atendimento das empresas à clientes ("*Call-Center*")**

Atualmente é comum as empresas oferecerem serviços de atendimento aos seus clientes por telefone. Porém com a diversidade de produtos ou serviços oferecidos, para direcionar a ligação do usuário ao departamento desejado é necessário que o cliente responda a várias perguntas de um sistema através das teclas do telefone. Várias pesquisas realizadas mostram a insatisfação destes clientes, devido à demora e dificuldade de compreensão na utilização destes serviços. A comunicação de uma forma mais natural, através

J. M. N. dos Santos é aluno de Doutorado da Escola Politécnica da Universidade de São Paulo (e-mail: zemaria.novaes@.usp.br).

da língua nativa, facilitará a utilização destes recursos.

**Maior facilidade no uso de terminais de auto atendimento da rede bancária.**

O auto atendimento bancário, feito através dos caixas eletrônicos, também está presente cada vez mais no cotidiano de todos, independentemente da escolaridade e classe social. Uma parcela das pessoas enfrenta muita dificuldade na interação com estas máquinas, devido à enorme quantidade de opções de serviços que são agrupados em opções dos menus, nem sempre de fácil entendimento.

A forma mais intuitiva para se identificar qual função o usuário quer acessar é através de um diálogo em uma língua nativa.

**Tradução**

Através da análise sintática e semântica de uma frase, pode-se realizar a tradução para uma outra língua.

## IV. Dispositivo Adaptativo

Um dispositivo adaptativo pode ser representado por um conjunto de regras expressas em qualquer formato (que podem ser descritas do tipo "se-então") e um conjunto de operações associado a cada uma das regras. Isto implica em se ter uma relação entre um conjunto de regras (condições) e um conjunto de ações.

Aplicamos a tecnologia adaptativa para representar fenômenos que possam ser representados por um modelo computacional na forma de regras, conforme a definição descrita acima. Algumas vezes, após a execução do modelo computacional representando várias situações do fenômeno estudado conclui-se que os conjuntos que representam as regras e ações podem ser modificadas para uma melhor adaptação ou aperfeiçoamento do modelo computacional. Esta alteração pode ser representada nas regras iniciais do dispositivo, como regras adaptativas. As regras adaptativas, quando executadas, alteram o conjunto de regras e ações, representando a experiência e aprendizado do dispositivo.

## V. Gramática Adaptativa da Língua Portuguesa

Este trabalho apresenta uma arquitetura de um dispositivo adaptativo para análise sintática de frases escritas na língua portuguesa.

Para se realizar a análise sintática de uma frase, utiliza-se um conjunto de regras gramaticais que definem a língua em questão. Baseado neste conjunto de regras de derivação, usualmente utilizada para representar uma gramática, o dispositivo será capaz de verificar se uma frase está gramaticalmente correta ou não, conforme sua gramática.

Como a elaboração de uma gramática representada por regras que abranja toda a língua portuguesa é complexa, adotou-se um modelo computacional que permite que novas regras gramaticais possam ser adicionadas a um conjunto inicial, aumentando assim a capacidade de representação da gramática. Desta forma o modelo criado será amplo mas poderá ser aprimorado por um módulo de aprendizagem, aumentando sua capacidade de representação.

Esta característica de possibilitar a inclusão de novas regras gramaticais, torna este modelo computacional um dispositivo adaptativo, conforme conceito apresentado anteriormente no item IV.

As regras iniciais da gramática foram extraídas de [2]. Podemos separar essas regras em dois grupos. O primeiro abrange as regras de primeiro nível (o mais alto da árvore de derivação) e o segundo abrangendo as demais regras, ou seja, desde as intermediárias até as últimas contendo os símbolos terminais. Os simbolos terminais serão representados pelas categorías gramaticais das palavras, como: artigo, pronome, adjetivo ou advérbio. Os simbolos não terminais são utilizados para a definição de grupos de palavras, como por exemplo Sintagma Substantivo que pode ser um substantivo próprio, um substantivo comum, precedidos ou não de artigo. Para mais detalhes, observar as regras de derivações envolvendo estes elementos nas tabelas de regras.

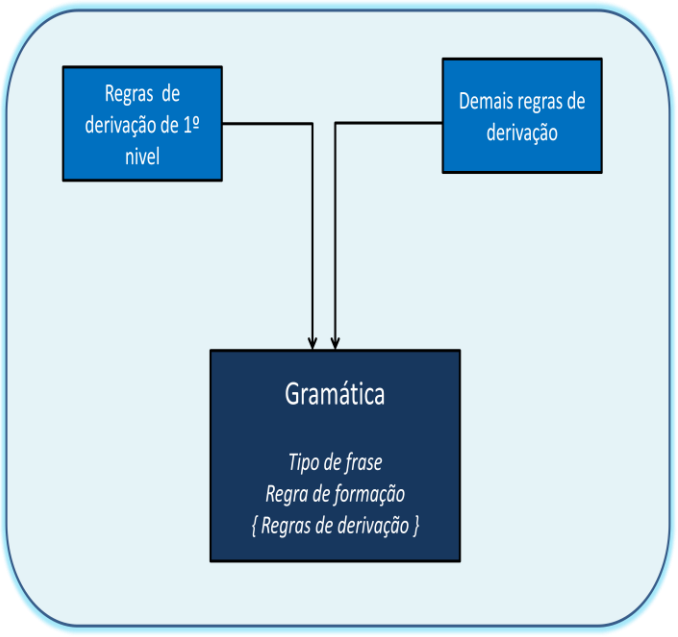



Fig 1 – Tabelas das regras da definição da Gramática.

A Fig. 1 ilustra as tabelas mencionadas na composição das regras da gramática. Baseado em [2], definiu-se o conjunto inicial da gramática, separando-as nos dois grupos de regras mencionados. Estas regras iniciais reconhecem varios padrões de frases, como por exemplo: oração de predicado nominal, oração de verbo transitivo indireto, oração de período composto por subordinação adverbial, etc

A seguir são apresentadas as duas tabelas de regras (a de 1º nível e a das demais regras) e uma tabela contendo as descrições dos simbolos (terminais e não terminais).

| Regras de derivação – 1º nível (tabela 1) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1. | SS | + | Vtd | + | SS | | | | |
| 2. | SS | + | Vlig | + | SS | | | | |
| 3. | SS | + | Vlig | + | SP | | | | |
| 4. | SS | + | Vti | + | SP | | | | |
| 5. | SS | + | Vtdi | + | SS | + | SP | + | Sadv |
| 6. | SS | + | Vi | + | Conj | + | SS | + | Vi |
| 7. | SS | + | Vtdi | + | SS | + | SP | | |
| 8. | SS | + | neg | + | Vti | + | SP | + | SAdv |
| 9. | SS | + | Vtd | + | SS | | | | |
| 10. | Vlig | + | Sadj | + | SS | | | | |
| 11. | SS | + | Vti | + | SP | + | SP | | |
| 12. | SS | + | Vi | + | Sadv | | | | |
| 13. | Sadv | + | SS | + | Vti | + | SP | | |
| 14. | SS | + | Vlig | + | Sadj | | | | |

| Demais regras de derivação (tabela 2) | |
|---|---|
| 1 | *SS →(PrA + Sc\|Sp) ou (Sp) ou (Sc)* |
| 2 | *SP →Prep + SS* |
| 3 | *Sadv → Advt* |
| 4 | *SS →Sb +(Artdef) + Vlig + Sadj* |
| 5 | *Sadj → Adj* |
| 6 | *SS →Sb + SS1 (1) + SS2 (1) + Vtd* |
| 7 | *SS →PrA + Sc + OSAR* |
| 8 | *OSAR → PrRel + Vi* |
| 9 | *OSAR → PrRel + PrPes + Vtd* |
| 10 | *SS → Card + Sc + Sadj* |
| 11 | *SS → PrPes ou elipse* |
| 12 | *OSAR → Sadv + (PrPes) + Vi* |
| 13 | *OSAR → Prep + PrRel PrPes + PrPes + Vti* |
| 14 | *OSAR → Rel + Sc + ArtDef + Sc + Vtd* |
| 15 | *SS → Sadv + GV* |
| 16 | *Sadv → PrRel* |
| 17 | *GV → Vtd + SS + PrA + Sc* |
| 18 | *SS → ArtDef + Sc* |
| 19 | *Sadv → Prep + Sb + SS + Aux + Vi* |
| 20 | *PrA → Artind* |
| 21 | *PrA → Artdef* |
| 22 | *PrA → dem* |

| Simbolos – terminais e não terminais (tabela 3) | |
|---|---|
| *SS* | Sintagma substantivo |
| *Conj* | Conjunção |
| *Sadj* | Sintagma (locução) adjetivo |
| *Sadv* | Sintagma adverbial |
| *SP* | Sintagma preposicional |
| *Vlig* | Verbo de ligação |
| *Vtd* | Verbo transitivo direto |
| *Vtdi* | Verbo transitivo direto e indireto |
| *Vti* | Verbo intransitivo |
| *Neg* | Negação |
| *GV* | Grupo verbal |
| *OSAR* | Oração subj. adjetiva restritiva |
| *PrA* | Pronome adjetivo |
| *Advt* | Advérbio de tempo |
| *Artdef* | Artigo definido |
| *Artind* | Artigo indefinido |
| *Card* | Numeral cardinal |
| *Dem* | Demonstrativo |
| *Prep* | Preposição |
| *Prind* | Pronome indefinido |
| *PrPess* | Pronome pessoal |
| *PrRel* | Pronome relativo |
| *Sb* | Subordinador (conj. subordinativa) |
| *Sc* | Substantivo comum |
| *Sp* | Substantivo próprio |

## VI. ARQUITETURA

O dispositivo adaptativo é composto de dois módulos: o de reconhecimento e o de aprendizagem.

**Módulo de reconhecimento**

Este modulo lê uma frase e verifica se ela é reconhecida por alguma regra da gramática. Caso ela seja reconhecida por alguma regra, o dispositivo emite uma mensagem indicando que a frase foi reconhecida e gera em um arquivo XML a árvore sintática correspondente.

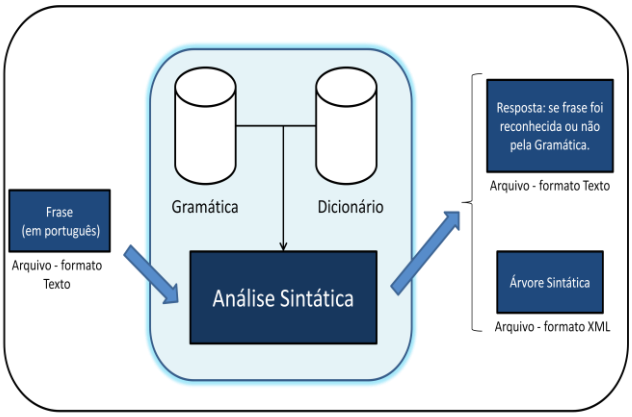



Fig 2 – Análise Sintática de uma frase

**Modulo de aprendizagem**

Este módulo contém a parte adaptativa da gramática. Seu funcionamento ocorre através da leitura de um texto (contendo uma ou mais frases). Este módulo poderá operar de duas maneiras: com ou sem supervisão.

Quando operar com supervisão, os padrões de frases identificados nos exemplos que não forem reconhecidos pela gramática serão submetidos a um especialista para aprovação ou não dessa nova regra.

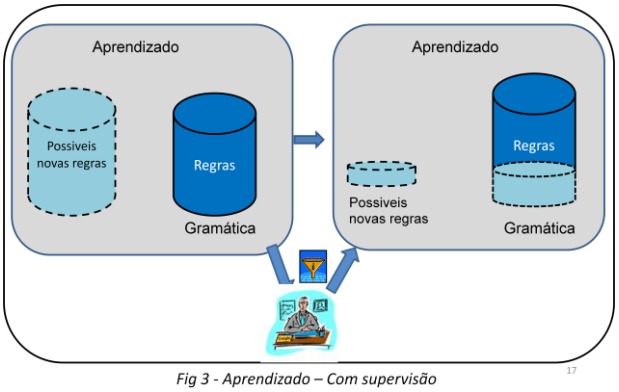



*Fig 3 - Aprendizado – Com supervisão*

Quando operar sem supervisão, o módulo de aprendizagem irá acatar como novas regras os padrões não reconhecidos pela gramática e que foram identificados através dos exemplos.

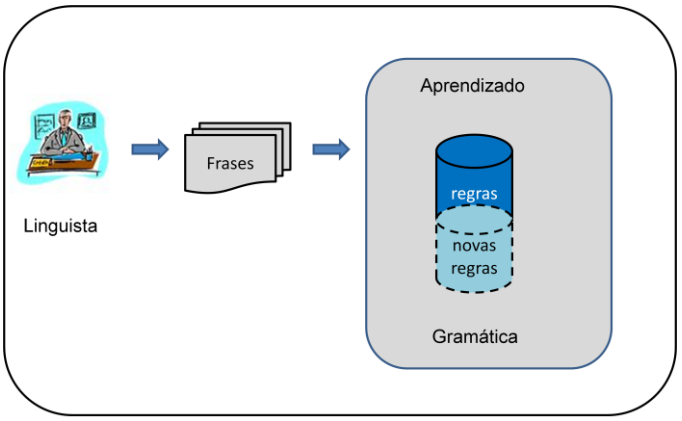



*Fig. 4 - Aprendizado – Sem supervisão*

Abaixo temos alguns exemplos de ilustração. O processo de identificação é o mesmo para todas as situações mencionadas. A diferença é apenas que o módulo de aprendizado inclui uma nova regra de formação quando um novo tipo de frase é encontrado operando no modo sem supervisão, ou quando for confirmado pelo especialista quando estiver operando no modo com supervisão.

Oração Transitiva

Representação de uma oração transitiva direta através de sequência de não-terminais*: SS1 - Vtd - SS2* (símbolos da tabela *SNT-1). Regra 1 da tabela 1*.

Regras de derivação dos termos da oração:

*SS1* → Regras1, 20 e
*SS2* → Regras 1 e 21 da tabela 2.

Exemplo: nenhum aluno conhece o livro

nenhum aluno: *SS1,*
conhece: *Vtd,*
o livro: *SS2.*

Aplicando-se a regra 1 e 20 em *SS1 e regras 1 e 21* em *SS2*, temos:

nenhum: *Artind*
aluno: *Sc*
conhece: *Vtd*
o : *Artdef*
livro*: Sc*

Oração de verbo transitivo direto e indireto

Representação de uma oração transitiva direta através de sequência de não-terminais: *SS1 – Vtdi - SS2 – SP – Sadv*. *Regra 5 da tabela 1.*

Regras de derivação (tabela 2) dos termos da oração:

*SS1* → Regra 1,
*SS2* → Regras 1 e 21,
*SP* → Regras 2, 1,21 e
*Sadv* → Regra 3.

Exemplo: Pedro colocou o livro na biblioteca hoje

Pedro: *SS1,*

colocou: *Vtdi*,
o livro: *SS2*,
em a (na) biblioteca: *SP*.

Aplicando-se as regras, temos:
Pedro: *Sp*,
colocou: *Vtdi*,
o: *Artdef*,
livro: *Sc*,
em: *Prep*,
a : *Artdef*,
biblioteca: *Sc*,
hoje: *Advt*.

## VII. TRABALHOS FUTUROS

Como futuro trabalho podemos citar a elaboração de um corretor gramatical para a lingua portuguesa, baseado na gramática representada pelo dispositivo descrito neste artigo.

Outro trabalho interesante, é adaptar esta gramática e o corretor gramatical para outras linguas como o espanhol e o inglês.

Outro trabalho futuro pode ser realizado através do aprimoramento deste modelo para reconhecimento de padrão (estilo) de escrita. Pode-se fazer inferência em um texto longo, com várias frases de um autor, para tentar se reconhecer um padrão, conforme o tipo de frases utilizadas. Com base nestas informações, pode-se tentar verificar se um outro texto pertence ao mesmo autor ou não.

## VIII. CONCLUSÃO

Foi apresentado um dispositivo adaptativo para representação de gramática para a língua portuguesa. A implementação deste modelo está em desenvolvimento, fazendo parte da tese de doutorado do autor.

Com base nos tipos de frases mapeados, através de [2], temos uma gramática com boa abrangência de representação da língua portuguesa. A sua característica adaptativa permitirá que ele aumente sua capacidade de representação. Porém, deve-se observar que este aumento de regras poderá diminuir significamente a performance de reconhecimento. Então, o ideal é escolher uma gramática representativa para a aplicação em questão, onde se tenha uma performance aceitável.

## REFERÊNCIAS

[1] J. J. Neto, 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, Springer-Verlag, 2001, pp. 234-250.

[2] C. P. Luft, Moderna Gramática Brasileira, São Paulo, Brasil, Editora Globo, 2ª. edição revisada e atualizada.

[3] J. J. Neto, "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, Springer-Verlag, 2001, pp. 234-250.

[4] J. J. Neto, "Contribuições à metodologia de construção de compiladores", Tese de Livre Docência, USP, São Paulo,1993.

[5] J. J. Neto e M. K. Iwai, "Adaptive Automata for Syntax Learning", CLEI 98 - XXIV Conferencia Latinoamericana de Informatica, MEMORIAS. pp. 135-149, Quito, Equador, 1998

**José Maria Novaes dos Santos.** É bacharel em Matemática Aplicada pela Universidade de São Paulo. É mestre em Engenharia da Computação pela Escola Politécnica da Universidade de São Paulo.
Atualmente é Doutorando do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo. Suas áreas de interesse e pesquisa abrangem: Teoria da Computação, Tecnologia Adaptativa e Processamento em Linguagem Natural.

# Adaptatividade em Imagens de Satélite para Previsão do Tempo

L. E. C. Dalla Valle and R. L. A. da Rocha

***Abstract*— Como uma forma alternativa de estudar o clima, com o intuito de melhorar sua previsão usando menos recursos computacionais, um modelo de reconhecimento de padrões em imagens de satélite é abordado. Um gerador de gramática mínima para uma linguagem livre de contexto, cujos símbolos são baseados na própria sequência de imagens é utilizada, e um dispositivo adaptativo é montado sobre esta gramática no intuito de inferir contextos, e, desta forma, prever o comportamento futuro deste sistema.**

***Keywords*— Imagens de Satélite, Adaptatividade.**

## I. INTRODUÇÃO

DESDE os primeiros indícios de civilização na antiguidade, as forças da natureza foram motivo de atenção e preocupação. Durante muito tempo, nas sociedades primitivas politeístas, cada evento diferente estava associado a um deus e seus sentimentos em relação á humanidade, afinal, são estas forças que definem períodos de bonança na agricultura, ou a desgraça humana, sendo diretamente associadas com castigos ou premiações. É muito comum os episódios em que Zeus lança raios forjados por seu filho Hefesto, para demonstrar sua ira contra a humanidade. Assim também acontece nas narrativas egípcias, na mitologia nórdica, na crença dos descendentes da raça vermelha e nas tribos africanas.

Como mostra a epígrafe, a bíblia apresenta vários episódios de previsão através de sonhos ou comunicações, como é o caso de Noé, que foi avisado do dilúvio de 40 dias, e José que sonhou com 7 anos de abundância e 7 anos de fome no Egito.

No livro de Jó, há um trecho: "*Com a sua voz troveja Deus maravilhosamente; faz grandes coisas, que nós não compreendemos. Pois à neve diz: Cai sobre a terra; como também às chuvas e aos aguaceiros: Sede copiosos. Ele sela as mãos de todo homem, para que todos saibam que ele os fez. E as feras entram nos esconderijos e ficam nos seus covis. Da recâmara do sul sai o tufão, e do norte o frio. Ao sopro de Deus forma-se o gelo, e as largas águas são congeladas. Também de umidade carrega as grossas nuvens; as nuvens espalham relâmpagos. Fazem evoluções sob a sua direção, para efetuar tudo quanto lhes ordena sobre a superfície do mundo habitável: seja para disciplina, ou para a sua terra, ou para beneficência, que as faça vir. A isto, Jó, inclina os teus ouvidos; para e considera as obras maravilhosas de Deus. Sabes tu como Deus lhes dá as suas ordens, e faz resplandecer o relâmpago da sua nuvem? Compreendes o equilíbrio das nuvens, e as maravilhas daquele que é perfeito nos conhecimentos; tu cujas vestes são quentes, quando há calma sobre a terra por causa do vento sul? Acaso podes, como ele, estender o firmamento, que é sólido como um espelho fundido?...*"

O primeiro estudo realizado em meteorologia, foi de Aristóteles, onde, apesar de não contar com instrumentos básicos, como termômetro, barômetro, etc, escreveu o livro Meteorologia, onde descreve seus estudos das nuvens, da chuva, do vento, orvalho, trovão, associa as condições de tempo a estes fenômenos, e foi capaz de realizar algumas previsões precisas sobre as chuvas [6].

Com o passar do tempo, várias descobertas contribuíram para o aprimoramento da técnica. A invenção dos instrumentos, dos

meios de armazenamento e transmissão de dados, a descoberta que a atmosfera poderia ser melhor descrita através de modelos físicos numéricos, a invenção do computador, capaz de realizar inúmeros cálculos em uma pequena fração de tempo, os modelos estatísticos propostos por Lorenz [10], tudo contribuiu para melhorar a qualidade das previsões. Porém nunca mudaram o fato de que existe uma certa imprevisibilidade inerente, e que, como humanidade, continuamos vulneráveis a estas forças.

No Brasil, o Instituto Nacional de Pesquisas Espaciais (INPE), dentre outras atribuições, é o órgão responsável pelos estudos meteorológicos. Em seu Centro de Previsão do Tempo e Estudos Climáticos (CPTEC), os dados das diversas estações

meteorológicas espalhadas pelo território nacional são centralizados, e os diversos modelos são processados em supercomputadores para realizar previsões do tempo e clima.

Os modelos mais comuns, processados pelo CPTEC são Global [4,9], Regional (ETA) [2, 12], Ensemble [3, 10] e Oceânico [5, 16]. Todos utilizam simulações numéricas regidas pelas leis físicas e formulações estatística, que representam a realidade aproximada da melhor forma possível dentro do tempo disponível para realizar os cálculos.

O presente trabalho procura investigar, paralelamente, a existência de um novo modelo, não baseado na física, mas na

linguagem que a física gera através das formas e movimentações das massas de vapor, no intuito de melhorar o tempo e diminuir a necessidade de processamento, para, com isso, melhorar a qualidade de previsão do tempo.

## II. MOTIVAÇÃO

O uso de métodos mais baratos de prever o tempo, e, ao mesmo tempo, que apresentem bons resultados (mesmo quando comparados com aqueles obtidos de modelos

L. E. C. Dalla Valle, Universidade de São Paulo (USP), São Paulo, São Paulo, Brasil, luisdallavalle@usp.br
R. L. A. da Rocha, Universidade de São Paulo (USP), São Paulo, São Paulo, Brasil, rlarocha@usp.br
Agradecemos ao CNPq pelo apoio sem a esta pesquisa.

numéricos que funcionam em super-computadores) é a motivação principal. As questões que impulsionam esta pesquisa são:

- É possível prever o comportamento do clima usando apenas uma sequência de imagens de satélite, com resultados similares aos obtidos com super-computadores?
- É possível inferir alguns dados, como pressão atmosférica, temperatura, umidade, apenas observando estas imagens?

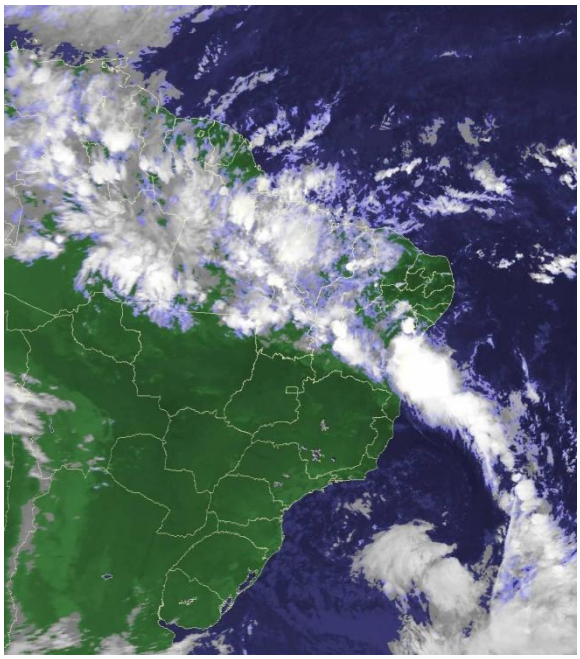


Figura 1: Imagem Original. Fonte: INPE

## III. Bases Teóricas

Para a realização deste trabalho, foram necessário estudos em diversas áreas, sendo abordados aqui os principais itens.

### A. O Alfabeto

Para tornar possível a utilização do sistema de reescrita e da gramática mínima em uma imagem, um alfabeto bidimensional [**Erro! Fonte de referência não encontrada.**] baseado em imagem, $\Sigma^{**}$ foi criado.

A menor unidade de informação de uma imagem é um pixel, no entanto, para uma análise de movimento de padrões, um pixel não é suficiente. A menor unidade mais conveniente é uma matriz 3×3, pois consegue representar o pixel central e seus vizinhos. Assim, fixando-se um ponto de análise no pixel central, é possível entender o movimento geral daquela região analisando seus vizinhos. Com isso em mente, as imagens criadas para representar os símbolos têm este tamanho, três pixeis de largura por três pixeis de altura, representados pela matriz:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

onde $p_{22}$ é o pixel central de referência.

A imagem que deu origem ao alfabeto, é a imagem resultante do tratamento apresentado na seção**Erro! Fonte de referência não encontrada.**, onde apenas a imagem em preto e branco contendo as bordas das nuvens geradas pelo algorítimo de Canny é utilizada. Consequentemente, o alfabeto é configurado neste mesmo es-quema de cores.

Como esta matriz representa uma imagem binária $B : N^2 \rightarrow \{0, 1\}$, e contém ao todo nove pixeis, também pode ser representada como uma cadeia de bits da forma $p_{33}p_{32}p_{31}p_{23}p_{22}p_{21}p_{13}p_{12}p_{11}$, sendo possíveis $2^9$ combinações diferentes de zeros e uns, totalizando 512 imagens, onde 0 é preto e 1 é branco.

Estas imagens são a representação direta dos símbolos da linguagem criada para esta análise. A Figura **Erro! Fonte de referência não encontrada.** mostra alguns exemplos de símbolos gerados para a linguagem.

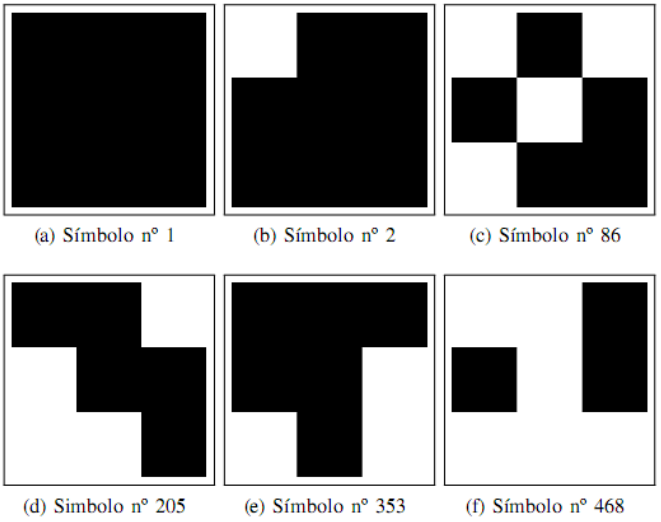



Figura 2: Alguns símbolos criados pela combinação de preto e branco em uma matriz 3x3

### B. Autômatos Adaptativos

Autômatos Adaptativos são dispositivos orientados a regras, derivados de Autômatos Finitos, que é um modelo matemático computacional que representa, de forma sintética, um dispositivo mínimo capaz de realizar computação. É constituído de duas partes, uma fita de entrada que contém símbolos, que o autômato está configurado para reconhecer através de uma cabeça de leitura, e o controle finito, uma "caixa preta" que contém todos os estados finais ou não e suas respectivas transições, que verifica o símbolo atual da fita, dada a posição da cabeça de leitura, analisa se o estado atual possui transição para o símbolo lido e transita de estado conforme o caso, repetindo esta operação enquanto houver elementos na fita. Possui três maneiras diferentes de parar. A primeira ocorre quando a cadeia termina e o estado de

transição é final, parando em acerto. A segunda, é quando lê um símbolo cujo estado atual não o reconhece, não existindo transição correspondente, parando em erro. A terceira ocorre quando a fita acaba, e o estado de transição não é final, parando em erro novamente. Segundo [**Erro! Fonte de referência não encontrada.**], suas utilizações mais importantes são:

1. Software para design e verificação do comportamento de circuitos digitais.
2. O "analisador léxico" de um compilador convencional, que quebra o texto em unidades lógicas, como identificadores, palavras reservadas e pontuação.
3. Software de escaneamento de textos longos, como coleções de páginas web, para encontrar e contabilizar ocorrências de palavras, frases ou outros padrões.
4. Software para verificar sistemas de todos os tipos que possuem número finito de estados, como protocolos de comunicação, ou protocolos para troca segura de informação.

Sua definição matemática segue como apresentado em [13]:

**Definição 1:** *Um automato finito determinístico é uma quín-tupla* $M = (K, \Sigma, \delta, s, F)$*, onde:*

$K$ é um conjunto finito de estados,
$\Sigma$ é um alfabeto reconhecido pelo autômato,
$s \in \Sigma$ é o estado inicial,
$F \subseteq K$ é o conjunto de estados finais,
$\delta$ é a função de transição, $\delta: K \times \Sigma \to K$

Existem duas formas de representação de autômatos mais comumente utilizadas, através de grafos (Figura 3) e tabelas de decisão (Tabela 1).

| Estado | Entrada | Transição |
|---|---|---|
| $\delta_0$ | $a$ | $\delta_1$ |
| $\delta_1$ | $a$ | $\delta_1$ |
| $\delta_1$ | $b$ | $\delta_0$ |

Tabela 1: Representação em tabela de um autômato finito

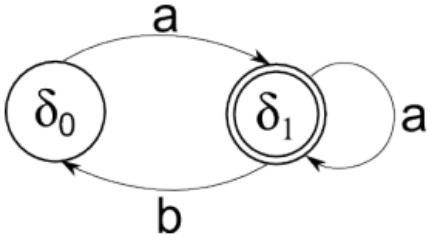



Figura 3: Mesmo autômato da tabela acima, porém representado em um grafo

O autômato finito "determinístico", apresenta a característica de que cada estado possui no máximo uma transição para um determinado símbolo (não vazio $\varepsilon$) do alfabeto, de forma a não haver ambiguidades em seu comportamento.

Segundo [8], adaptatividade é uma característica apresentada por um dispositivo formal quando este possui a capacidade de alterar sua própria estrutura interna de forma autônoma, sem interferência externa, nem mesmo de seus usuários, mudando, em consequência, seu próprio comportamento em resposta direta ao estímulo de entrada.

Pode ser aplicada a virtualmente qualquer dispositivo baseado em regras, como tabelas adaptativas e autômatos finitos. Para isto, basta acrescentar à estrutura original do dispositivo, um conjunto de ações adaptativas que serão executadas quando o dispositivo transitar pelos estados. Baseado em [8], um Autômato Adaptativo nada mais é do que a aplicação de um mecanismo adaptativo ao autômato finito convencional definido em III-B

Como visto em [14]:

**Definição 2:** *Um autômato Adaptativo de Estado Finito (AAF) pode ser definido como uma óctupla* $M = (Q, \Sigma, q_0, F, \delta, Q_\infty, \Gamma, \Pi)$*, onde:*

$Q \subseteq Q_\infty$ é um conjunto finito de estados
$\Sigma$ é o alfabeto
$q_0$ é o estado inicial
$F \subseteq Q$ é o conjunto de estados finais
$\delta \subseteq (Q_\infty \times (\Sigma \cup \{\varepsilon\}) \times Q_\infty)$ é a relação de transição
$Q_\infty$ conjunto teoricamente infinito de estados
$\Gamma \subseteq (\{+,-\} \times (Q_\infty \times (\Sigma \cup \{\varepsilon\}) \times Q_\infty))$ conjunto de ações primitivas de inclusão $(+)$ ou exclusão $(-)$ de transições[6]
$\Pi: (Q_\infty \times (\Sigma \cup \{\varepsilon\}) \times Q_\infty) \to \Gamma^*$ função que associa a cada transição em $\delta$ uma sequência de ações adaptativas primitivas. Pode-se associar uma sequência vazia a uma determinada transição para indicar que é uma transição não adaptativa.

Os autômatos adaptativos são amplamente utilizados em reconhecimento de linguagens, padrões diversos em aprendizado de máquina, etc. Para este trabalho, uma ``Camada" adaptativa será utilizada sobre uma gramática mínima gerada, na tentativa de inferir contextos, já que esta, por sua vez, cria uma linguagem livre de contexto.

### *C. Gramática Mínima*

A base de estudo para esta técnica algorítmica foi o trabalho da Ivone Matsuno [11]

Uma gramática é definida em [13] como a quádrupla:

$$G = (V, \Sigma, R, s)$$

$\forall A \in V - \Sigma$ e $u \in V^*$, $A \to_G u$ quando $(A, u) \in R$

$\forall u, v \in V^*$, onde $u, v$ são sequência de símbolos, e $A \in V - \Sigma | u = xAy$, $v = xv'v$, e $A \to_G v'$

Uma gramática é dita livre de contexto, se todas as regras de produção $R$ são da forma $\alpha \to \beta$, onde $\in N$ tal que $|\alpha| = 1$ e $\beta = wA$ ou $\beta = w$, e $A$ e $w \in \Sigma^*$

Como definido em [11], uma gramática livre de contexto é dita mínima quando gera uma única sentença. Dada uma sentença $x$ de tamanho $n$ sobre um alfabeto qualquer, a linguagem gerada pela gramática mínima $G_x$ é $L = \{x\}$, e o

[6] Pelo mesmo motivo apresentado em [14], a operação de consulta "?" é omitida para simplificar a especificação, bem como a ordem em que uma ação adaptativa será executada, se antes ou depois da transição.

problema da gramática mínima é encontrar $G_x$ que gere somente $x$, de forma que o tamanho de $|G| < n$.

Já uma gramática sensível ao contexto, tem suas regras de produção da forma $\alpha \rightarrow \beta$, onde $\alpha$ é uma palavra de $V^*NV^*$, $\beta$ é palavra de $V^*$ e $|\alpha| < |\beta|$, exceto quando $x \rightarrow \varepsilon$ [11]

### D. Linguagem Bidimensional

Como forma alternativa de se tratar os símbolos, pode-se utilizar também as técnicas e operações sobre linguagem bidimensional [15]. Que consiste na aplicação das cadeias sobre um alfabeto nas duas dimensões, formando imagens.

Dessa forma, o alfabeto de imagens é tratado como uma linguagem. $L$ é o conjunto de imagens $i$, tal que, dado o alfabeto $\Sigma = \{0,1\}$, $i \in \Sigma^{**}$ e o tamanho de linhas é igual ao tamanho de colunas $l(p) = c(p) = 3$, permitindo todas as combinações de zeros e uns limitadas nas dimensões $3 \times 3$

### E. Sistema de Reescrita

O sistema de reescrita aqui utilizado é apenas uma substituição de um símbolo terminal por outro, para realizar a contagem de transições, onde, toda vez que ocorre, o acumulador de transição para este caso é incrementado.

$$S_m \rightarrow S_n$$
$$Acum_{(m,n)} + +$$

## IV. Tecnologias

Para a implementação desta pesquisa, recorreu-se a algumas tecnologias disponíveis para a implementação algorítmica e interface com o usuário para visualização dos processos.

### A. Java

Java é uma Plataforma de Desenvolvimento de Software que inclui a linguagem Java, o compilador, a Máquina Virtual Java, uma extensa biblioteca e um conjunto de ferramentas extras. As grandes vantagens desta plataforma é que é de distribuição e uso livre, e funciona nas principais plataformas computacionais da atualidade, sem exigir reescrita de código ou recompilação para plataformas específicas.

Sua extensa biblioteca oferece classes para criação de interface com o usuário e uma boa quantidade de classes para tratamento e manipulação de imagens utilizadas neste projeto.

### B. Python

Durante o desenvolvimento deste projeto, surgiu a necessidade de se alterar a linguagem para Python, por se tratar de uma linguagem que facilita o uso da adaptatividade, por ser dinamicamente modificável em tempo de execução.

Python é uma linguagem funcional, dinâmica, clara, concisa e independente de plataforma, criada em 1990 por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda (CWI). Pode ser facilmente integrada com outras linguagens e possui uma extensa comunidade de desenvolvedores.

### C. QT

QT é uma biblioteca de componentes utilizados em interfaces gráficas de alto nível, multi-plataforma e de grande aceitação pelo mercado. Atualmente é mantida pela Nokia, e é amplamente utilizada em diversos dispositivos móveis e em muitas outras arquiteturas.

Originalmente é desenvolvida em C++, mas possui bibliotecas de ligação (*Bindings*) em Python.

### D. Tratamento de Imagens

Como este trabalho envolve exclusivamente a análise de dados a partir de imagens, não poderia deixar de ter um tratamento especial para elas. Um conjunto de algorítimos que aplicam transformações sequenciais, de forma a deixá-la própria para análise.

Estas transformações são comumente chamadas de filtros, pois, como num filtro verdadeiro, retêm substâncias (informações), e deixam passar apenas o que se deseja extrair.

Para este trabalho, uma sequência de três filtros foram aplicados, o mediano, o limiar para conversão em preto e branco, e a detecção de bordas de Canny.

## V. O Processo

O sistema criado para realizar esta proposta armazena em memória um conjunto de imagens de satélites copiadas das bases de imagens do INPE. Primeiramente, as linhas dos estados são retiradas, utilizando um filtro mediano, onde as cores de um determinado pixel é calculada usando-se o valor médio dos pixeis vizinhos. Como resultado geral, a imagem aparenta-se um pouco mais desfocada, mas não é um problema, já que o processo seguinte, de conversão para preto e branco não é afetado por esta singularidade.

Depois de remover as linhas, a imagem é convertida para preto e branco, utilizando-se um limiar de cores, onde tudo o que é cinza, ou seja, possuem os valores para R, G e B mais elevados e semelhantes, torna-se branco. Todo o resto é convertido para preto. A figura 4 mostra este processo.

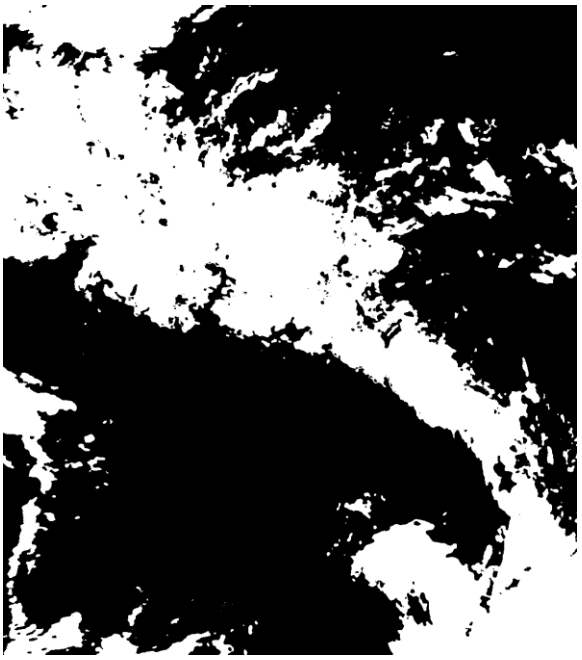


Figura 4: Imagem em P&B

Em seguida, a implementação do algoritmo de Canny [**Erro! Fonte de referência não encontrada.**] é utilizado, gerando uma borda de um pixel de espessura em todas as nuvens que o processo realizado até o momento detectou, como pode ser visto na Figura **Erro! Fonte de referência não encontrada.**.

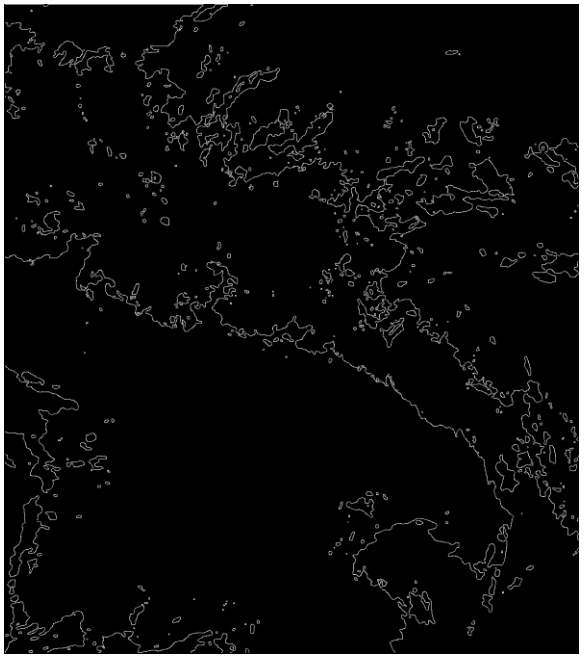


Figure 5: Detecção de Bordas usando Canny

Com as imagens preparadas, inicia-se a busca por símbolos e a contagem das transições. Esta busca varre toda a imagem atual e a próxima (caso houver), pixel a pixel, associando símbolos 3×3, e contando as transições de símbolos de uma imagem para outra.

Os resultados destas contagens são acumulados em uma matriz, como mostra a Tabela **Erro! Fonte de referência não encontrada.**. Nesta tabela, as linhas representam os símbolos de origem, e as colunas os símbolos de destino, e as células armazenam a quantidade de transições que ocorreram em uma determinada sequência de imagens.

| | Símbolo 0 | Símbolo 1 | Símbolo 2 | ⋯ |
|---|---|---|---|---|
| Símbolo 0 | 10.498 | 5.469 | 4.376 | ⋯ |
| Símbolo 1 | 4.623 | 3.984 | 2.445 | ⋯ |
| Símbolo 2 | 4.297 | 3.243 | 2.538 | ⋯ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋱ |

Tabela 2: Representação da Matriz de Acumulação de Transições, onde a linha representa o símbolo de origem e a coluna o símbolo de destino

Para representar de forma mais abrangente as transições de todos os símbolos, tomou-se emprestado o *micro-array* da genética, mudando sua denominação para *macro-array*, que, para este caso, segue a mesma formação da Tabela **Erro! Fonte de referência não encontrada.**, como mostram as Figuras **Erro! Fonte de referência não encontrada.** e **Erro! Fonte de referência não encontrada.**, onde as cores dos pixeis[7] são definidas pela quantidade de transições (*n*), através de uma simples conversão de base, da forma:

$$COR(Simb_i, Simb_j) = \begin{Bmatrix} R = mod(n, 256) \\ G = mod(div(n, 256), 256) \\ B = mod(div(n, 256^2), 256) \end{Bmatrix}$$

Obviamente que com essa representação, não podem haver mais que $256^3$ transições, o que é um número muito além do necessário para o estado atual da pesquisa. Este processo permite duas formas distintas de análise. Uma se baseia no acúmulo de imagens da sequência, como é o caso da Figura **Erro! Fonte de referência não encontrada.**, que acumulou as transições das primeiras 60 imagens analisadas, apresentando cores mais intensas. Outra forma é realizar a análise duas a duas, onde um conjunto de macro-arrays é gerado no modelo da Figura **Erro! Fonte de referência não encontrada.**, acumulando o resultado da análise da primeira imagem com a segunda, em seguida da segunda com a terceira, e assim sucessivamente, permitindo a criação de uma animação de curta duração destes arrays.

Ambas são úteis, pois permitem analisar as transições que ocorreram em períodos determinados, como, por exemplo, o verão deste ano comparando ao verão do ano passado,

[7] Por motivos de impressão, as imagens do macro-array tiveram seu contraste e saturação alterados, de forma a tornar mais simples a interpretação visual dos dados no texto impresso

períodos mais curtos, como a manhã de ontem com a manhã de hoje, ou mesmo fenômenos conhecidos, como comparar o *El-niño* do ano passado e o *El-niño* deste ano.

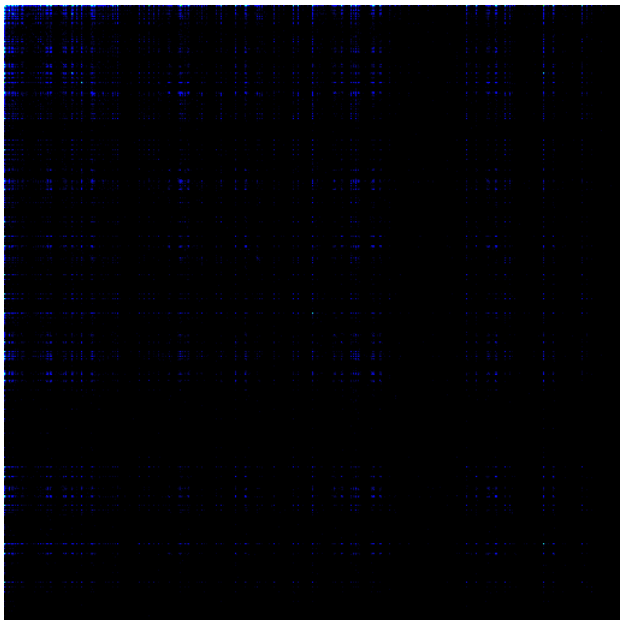


Figure 6: *Match* para duas imagens

Com estas transições armazenadas, um gerador de gramática mínima entra em ação, verificando onde ocorreram as transições, e em seguida, agrupa as repetições de forma a reduzir (compactar) os dados obtidos. Este processo baseia-se no algoritmo LZ77 [**Erro! Fonte de referência não encontrada.**] para realizar a tarefa.

A gramática gerada pelo processo descrito acima é, por definição, livre de contexto. Porém, é perceptível que a linguagem que rege a movimentação aqui analisada é notavelmente sensível ao contexto, e talvez, em alguns casos até irrestrita.

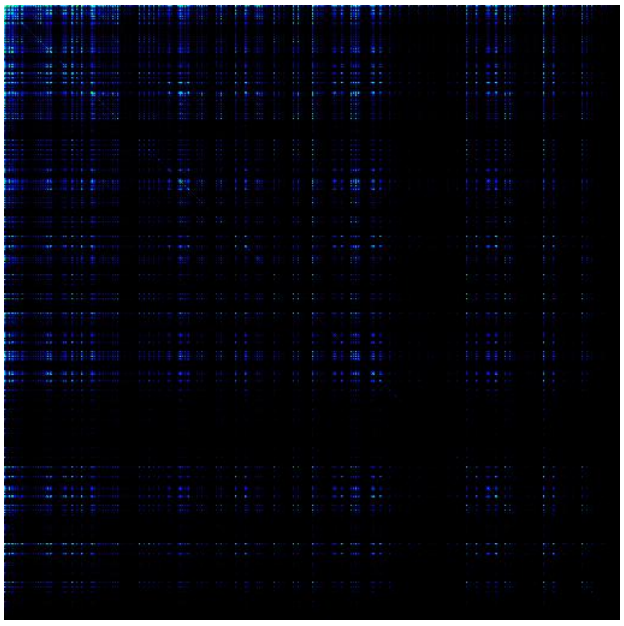


Figure 7: Result of *Match* para sessenta imagens

Para inferir contextos, um dispositivo adaptativo é montado sobre a gramática gerada, de forma a gerar um conjunto de regras bem definidas, às quais o sistema pode se basear para realizar a previsão dos eventos futuros.

## VI. Análises

É possível notar nas Figuras **Erro! Fonte de referência não encontrada.** e **Erro! Fonte de referência não encontrada.**, que as primeiras linhas e colunas possuem grande quantidade de transições, por apresentarem cores mais intensas. Isto se deve ao fato de que a maior parte das imagens analisadas é composta de preto, sendo correspondente ao símbolo nº1, e tanto a transição deste símbolo para ele mesmo, como dele para os outros e de outros para ele, são as que ocorrem com maior frequência, o que implica em sua possível remoção em estudos futuros, pois apenas os símbolos de significância maior em correspondência às bordas detectadas serão mantidos.

É facilmente notada também uma componente diagonal, do pixel (0,0) até o (512,512), o que indica ausência de transição. Este fato se deve ao movimento lento de algumas bordas analisadas, da correspondência ocasional do mesmo símbolo por puro acaso, ou por movimentos circulatórios padronizados da atmosfera. Esta componente diagonal tenderá a ficar menos evidente com a melhor calibração do modelo.

É importante salientar que há uma dissipação de ocorrências dos primeiros símbolos para os últimos. Aparentemente pelo fato de que as bordas se correspondem melhor com os primeiros símbolos, que possuem mais pixeis pretos do que brancos, formando padrões lineares, do que os últimos, que formam massas densas de branco.

Ao final, um grafo é montado contendo os símbolos em seus vértices e os pesos das derivações em suas arestas, onde o tamanho da aresta é inversamente proporcional ao peso que ela contém, para aproximar os vértices com maior peso de transições e afastar os mais leves, no intuito de adquirir uma resposta visual do processo realizado.

## VII. Conclusão

No atual estado da pesquisa, o processo está modelado. O sistema de *match*, para detectar os símbolos nas imagens de satélite está desenvolvido, e contabilizando as transições, exibindo os resultados nos *macro-arrays*.

Entre as tarefas pendentes, estão a migração para Python, criação do gerador de gramática mínima, utilizando o algoritmo LZ77, definir como as imagens resultado servirão de base para a gramática, se é que essa forma de acúmulo será útil para a geração da linguagem livre de contexto, e aplicar a camada adaptativa para inferir contextos.

Na próxima fase do trabalho, partindo de uma imagem de satélite, será utilizada a gramática encontrada e os contextos inferidos para montar as próximas imagens e calcular o quanto elas refletem os acontecimentos futuros.

## REFERÊNCIAS


[1] C. de Previsão do Tempo e Estudos Climáticos (CPTEC-INPE), "Cptec website." www.cptec.inpe.br, Junho 2010. Acesso em 8 de junho de 2010.

[2] E. N. Lorenz, "A study of the predictability of a 28-variable atmospheric model.," *Tellus*, vol. 17, p. 321–333, 1965. doi: 10.1111/j.2153-3490.1965.tb01424.x.

[3] CPTEC, "Modelo global." http://previsaonumerica.cptec.inpe.br/mod_glb.shtml, Setembro 2010.

[4] J. L. Kinter, D. DeWitt, P. A. Dirmeyer, M. J. Fennessy, B. P. Kirtman, L. Marx, S. Edwin K, J. Shukla, and D. M. Strauss, "The cola atmosphere-biosphere general circulation model. volume1: Formulation.," tech. rep., Center for Ocean-Land-Atmosphere Studies., Calverton, USA., 1997. Report n.o 51.

[5] CPTEC, "Eta model." http://etamodel.cptec.inpe.br/, Setembro 2010.

[6] F. Mesinger, "Eta model at ncep: Challenges overcome and lessons learned. lecture notes.," in *Workshop on "Design and Use of Regional Weather Prediction Models"*, (Miramare, Trieste, Italy), p. 42 pp, The Abdus Salam International Centre for Theoretical Physics, 11-19 April 2005.

[7] CPTEC, "Modelo ensemble." http://previsaonumerica.cptec.inpe.br/mod_ens.shtml, Setembro 2010.

[8] CPTEC, "Modelo wwatch." http://previsaonumerica.cptec.inpe.br/mod_wwatch.shtml, Setembro 2010.

[9] H. L. Tolman, "The numerical model wavewatch: a third generation model for the hindcasting of wind waves on tides in shelf seas.," *Communications on Hydraulic and Geotechnical Engineering, Delft Univ. of Techn., ISSN 0169-6548*, vol. 89-2, p. 72 pp., 1989.

[10] G. Rozenberg and A. Saloma, eds., *Handbook of Formal Languages*, vol. 3. Springer, 1997.

[11] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Finite Automata*. Pearson Education and Addison-Wesley, 2001.

[12] C. H. Papadimitriou and H. R. Lewis, *Elements of the Theory of Computation*. Prentice-Hall, 1998.

[13] J. José Neto, "Adaptive rule-driven devices - general formulation and case study," *Lecture Notes in Computer Science*, vol. 2494/2002, pp. 466–470, 2002.

[14] H. Pistori and J. J. Neto, "Adaptree - proposta de um algoritmo para indução de Árvores de decisão baseado em técnicas adaptativas," *Anais Conferência Latino Americana de Informática*, vol. Novembro/2002, pp. 1–10, 2002.

[15] I. P. Matsuno, "Um estudo dos processos de inferência de gramáticas regulares e livres de contexto baseado em modelos adaptativos," Master's thesis, Universidade de São Paulo, 2006.

[16] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679 – 698, 1986.

# Introdução a Árvores de Decisão Adaptativas

F. Catae, R. L. A. Rocha

***Abstract*— A árvore de decisão é utilizada para classificar dados com base nos seus atributos. A estrutura é baseada em regras, que consegue mapear dados previamente observados em possíveis casos futuros. A adaptatividade permite que uma estrutura se auto-modifique como resposta às mudanças de ambiente, incorporando novas informações. No entanto, a adaptatividade não se restringe às modificações estruturais do núcleo. Através do não-determinísmo, adiciona-se possibilidade de enumerar diferentes modelos e determinar o melhor deles usando uma métrica comum, como o conceito da complexidade de Kolmogorov. Em uma árvore de decisão adaptativa, há a possibilidade de criar novos atributos em função dos já existentes. Esses "atributos calculados" estendem sua capacidade para lidar com problemas simples, como paridade e multiplexação, mas que não podem ser bem modelados por uma árvore de decisão tradicional.**

***Palavras-chaves*—Aprendizagem de Máquina, Árvore de Decisão, Tecnologia Adaptativa**

## I. Introdução

Nos últimos anos, observa-se maior disponibilidade de informações por meios digitais acompanhada pelo aumento da capacidade de armazenamento. Sabe-se que o excesso de informação afeta negativamente a produtividade e, por isso, torna-se necessária a utilização de ferramentas de apoio. Um sistema inteligente pode, por exemplo, filtrar as informações desnecessárias e mantê-las ordenadas de acordo com sua relevância.

A automação de processo pode ser realizada através de um dispositivo composto por conjuntos de regras e ações armazenados em uma árvore de decisão. Dessa forma, é possível classificar e ordenar um grande volume de dados, que seriam inviáveis de se realizar manualmente.

No campo de aprendizado de máquina a estrutura de árvore de decisão é utilizada para inferir propriedades e atributos sobre informações até então desconhecidas. Nesse caso, a árvore de decisão mapeia os dados previamente observados em possíveis casos futuros.

A tecnologia adaptativa [1] está diretamente relacionada à questão de aprendizagem de máquina. Um dispositivo adaptativo é baseado em um mecanismo base com a capacidade de apresentar modificações na própria estrutura. Por exemplo, uma árvore de decisão adaptativa [2] pode assimilar novas informações através da inclusão de nós e ramos em sua estrutura.

F. Catae é estudante de Mestrado junto ao Laboratório de linguagens e Técnicas Adaptativas, Escola Politécnica da USP; e-mail: fcatae@usp.br.
R. L. A. Rocha é pesquisador do Laboratório de Linguagens e Técnicas Adaptativas, Escola Politécnica da USP, São Paulo/SP, Brasil; fone: 55 11-3091-5583; e-mail: luis.rocha@poli.usp.br.

## II. Classificação de Dados

Ao observar uma revoada de pássaros voando no céu, imaginamos que sejam todos da mesma espécie. Entretanto, sabemos que existem atributos que os diferenciam entre eles: formato dos olhos, tom das penas, tamanho das asas. Assim como existem outros, que são formas características do grupo: formato do bico e dos pés, proporção do tamanho da asa em relação ao corpo. Nesse cenário, a classificação corresponde ao problema inverso: a partir dos atributos de um indivíduo, seria possível descobrir qual a sua espécie?

O problema apresenta um universo de objetos, os quais podem ser descritos individualmente por meio de seus atributos. Um objeto possui uma série de atributos e pertence a uma única classe. A classificação pode ser feita por meio de regras, associando conjuntos de atributos às classes correspondentes. Por exemplo, o estudo de aves migratórias poderia ser modelado da seguinte forma:

Atributos:
Asas = { atrofiadas, desenvolvidas }
Bico = { curto, longo }
Corpo = { pequeno, grande }

Classes:
Espécie Migratória = { sim, não }

As regras são criadas com base nos estudos realizados sobre as diferentes espécies, buscando associar a morforlogia ao comportamento de migração. Uma hipótese seria afirmar que somente as aves com asas desenvolvidas, bico longo e corpo grande são migratórias.

Regras:
{ desenvolvidas, curto, pequeno } → não
{ desenvolvidas, longo, pequeno } → não
{ desenvolvidas, curto, grande } → não
{ desenvolvidas, longo, grande } → sim
{ atrofiadas, curto, pequeno } → não
{ atrofiadas, longo, pequeno } → não
{ atrofiadas, curto, grande } → não
{ atrofiadas, longo, grande }

→ não

A intenção é usar as regras para classificar indivíduos de acordo com seus atributos (asas, bico, corpo) em aves migratórias ou não.

O processo pode ser aplicado para os diferentes tipos de aves a partir de sua descrição. Por exemplo, observamos que o pelicano é uma ave migratória, enquanto que o avestruz e o pinguim não são.

**indivíduo = (asas, bico, corpo)**
*avestruz = (atrofiadas, curto, grande)*
*pinguim = (atrofiadas, curto, pequeno)*
*pelicano = (desenvolvidas, longo, pequeno)*

As regras foram dispostas semelhante a árvore de decisão, e representadas através de um autômato de estados finitos, como em ilustrado na Fig 1.

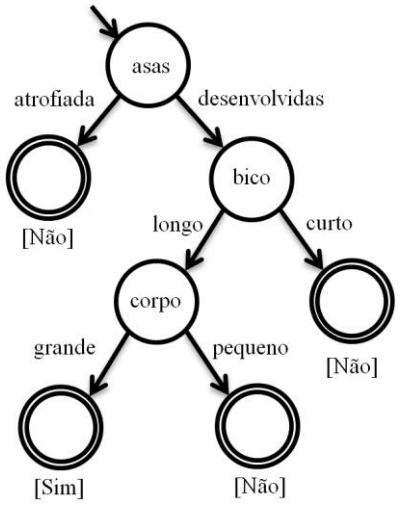



**Figura 1. Representação das regras de classificação na estrutura de árvore de decisão**

Formalmente, define-se um autômato de estados finitos determinístico por um quíntupla ($Q$,$\Sigma$, δ, $s_0$,$F$) [3], no qual:

- $Q$ – Conjunto finito de estados
- $\Sigma$ – Conjunto finito de símbolos (alfabeto)
- δ – Transições entre estados, $\delta: Q \times \Sigma \longrightarrow Q$
- $s_0$ – Estado inicial, $s_0 \in Q$
- $F$ – Conjunto de estados finais, $F \subseteq Q$

Dado uma entrada inicial $w$ composto por uma cadeia de símbolos de $\Sigma$, o autômato processa a informação a partir do estado inicial $s_0$ e realiza transições de estados conforme a função δ. Se a cadeia $w$ apresenta tamanho $n$, então após n iterações o processamento finaliza no estado $q$. Dizemos que um autômato aceita $w$, quando o estado final $q$ pertence ao conjunto F.

A representação de uma árvore de decisão em autômatos apresenta os nós e ramos mapeados para o conjunto de estados $Q$ e transições δ. O alfabeto é definido pela união dos conjuntos de atributos, enquanto que as classes são partições do conjunto de estados finais.

No exemplo do estudo de aves, estamos interessados em determinar se o autômato finaliza em um estado que representa a classe de aves migratórias ou não. Assim, os conjuntos poderiam ser definidos da seguinte forma:

A = { atrofiadas, desenvolvidas }
B = { curto, longo }
C = { pequeno, grande }
$F_{[SIM]}$ = { q ∈ Q | classe de aves migratórias }
$F_{[NAO]}$ = { q ∈ Q | classe de aves não migratórias }

$\Sigma$ = A∪B∪C
F = ($F_{[SIM]}$) ∪ ($F_{[NAO]}$)

Permitindo a classificação de objetos que apresentem a descrição através da cadeia de entrada.

$w = w_a\, w_b\, w_c$ , tal que $w_a \in A$, $w_b \in B$, $w_c \in C$

O autômato classifica a ave em "sim" se o estado final pertencer ao conjunto $F_{[SIM]}$, caso contrário, o processamento termina $F_{[NAO]}$ e a resposta será "não".

## III. Aprendizagem

Com o passar do tempo, novas regras de classificação podem surgir por conta de situações não consideradas inicialmente. Essas mudanças se refletem na mudança do autômato, forçando a criação de uma nova estrutura de classificação. A aprendizagem permite que um autômato incorpore as mudanças em sua própria estrutura de forma incremental, sem a necessidade de ser refeito. Consideramos, como exemplo, a classificação de uma andorinha seguindo as regras anteriormente definidas.

*andorinha = (desenvolvidas, curto, pequeno)*

Apesar de conhecido o fato das migrações sazonais das andorinhas, o classificador por regras define que as características de asas desenvolvidas, bico curto e corpo pequeno correspondem a classe de aves nao-migratórias. Esse comportamento pode ser corrigido durante o estágio de aprendizado, alterando-se os estados e transições do autômato.

Quando uma estrutura reage às mudanças impostas a ela e se auto-modifica, dizemos que houve uma adaptação. Essa adaptatividade incorpora informações do meio externo, permitindo o aprendizado incremental.

Na área de Tecnologia Adaptativa, estudam-se dispositivos que apresentam mudança de comportamento através de alterações em seu próprio núcleo. Um autômato adaptativo é uma extensão do autômato de estados finitos acrescido de regras de auto-modificação [1].

Um resultado relevante é que os autômatos de estados

finitos reconhecem somente linguagens regulares, enquanto que os autômatos adaptativos apresentam poder de Máquina de Turing [4].

O aprendizado do autômato ocorre por meio das operações elementares de consulta e inserção. A implementação da Adaptive-NDD Tree (Árvore Não-Determinística de Decisão Adaptativa) [2] se baseia em aprendizado por instância, na qual todos os dados de treinamento são incorporados na estrutura da árvore. A Fig. 2 apresenta os estados representando atributos e classificação após o treinamento.

Uma derivação da Adaptive-NDD Tree está ilustrada na Fig. 3, na qual os estados são agrupados conforme os atributos e classificações. Quando dados de treinamentos são inseridos, pode ocorrer uma operação denominada "split", na qual são criados estados novos para armazenar as diferentes classes.

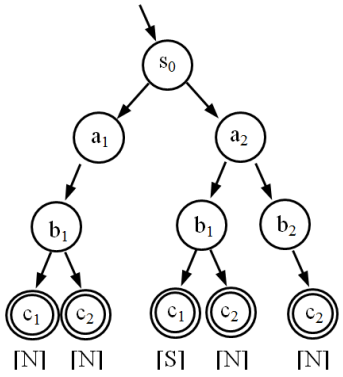



**Figura 4. Em uma Adaptive-NDD Tree, todos os dados de treinamento são armazenados na própria estrutura.**

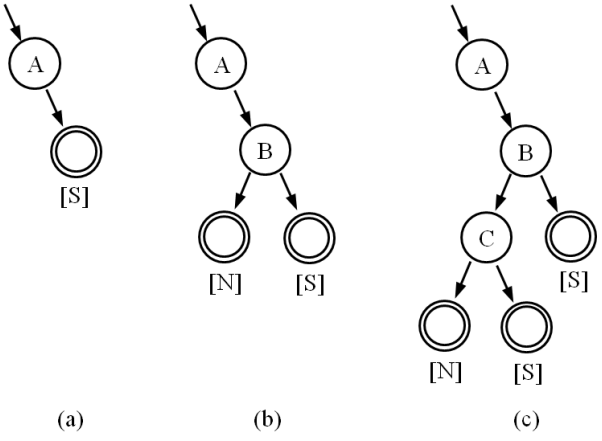



**Figura 5. (a) Estado inicial contendo uma única classe [S]. (b), (c) Objetos com diferentes atributos são inseridos na árvore, forçando operações de split.**

Os autômatos adaptativos são capazes de incorporar novas informações em sua própria estrutura, caracterizando a fase de aprendizado. Os conceitos mais fáceis de serem aprendidos podem ser atingidos por um pequeno número de auto-modificações.

## IV. Árvores de Decisão

A árvore de decisão é uma estrutura bem conhecidas [5], composta por nós e folhas relacionados aos atributos dos dados.

A aprendizagem ou processo de indução permite que uma árvore assimile novas informações e realize inferências sobre a classificação de dados. Para isso, utiliza-se um conjunto de dados de treinamento para ensinar ao dispositivo qual o resultado esperado de acordo com os atributos presentes. Durante o processo inicial de treinamento, a estrutura se automodifica. A operação adaptativa de adição corresponde ao chamado "split" de um nó da árvore.

O problema de aprendizado ótimo é conhecido como da classe NP-completo [6] e, por isso, os algoritmos são heurísticos e com base em melhor decisão local, porém, sem a garantia de retornar a melhor resposta global. Um método tradicional conhecido por ID3 e sua evolução C4.5 apresentados por Quinlan [7][8] utilizam o cálculo de alteração da entropia relacionada à operação de split.

Existem diferentes métodos para determinar a forma de crescimento da árvore, variando entre cálculo de entropia, chi-square ou diferentes medidas, e sempre buscando mínimos locais. Apesar de não ser possível determinar o mínimo global, observa-se que, pelo número de referência aos algoritmos ID3 e C4.5, os resultados são satisfatórios para grande parte dos casos de uso.

Em contrapartida, existem conceitos simples que não são bem expressos pela estrutura de árvore como, por exemplo, as operações de paridade, "ou exclusivo" (XOR), multiplexação. Nesses casos, a estrutura de árvore de decisão cresceria rapidamente sem, no entanto, melhorar sua capacidade de classificação ou predição.

Em algumas implementações de árvore de decisão, existe o mecanismo de poda (prunning), no qual são removidos os ramos e nós em excesso. Dessa forma, o tamanho total da estrutura diminui e fornece como resultado uma árvore de decisão mais concisa.

## V. Erros de Modelagem

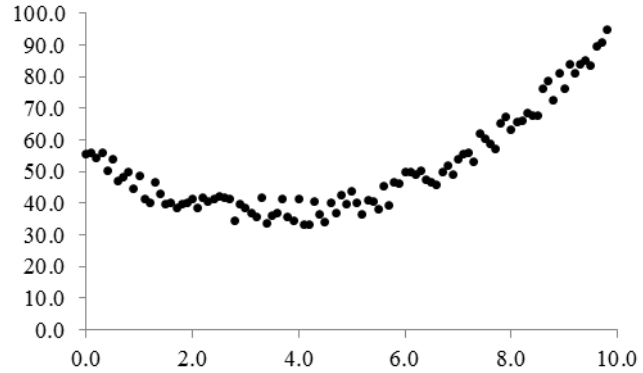
Um problema muito comum na modelagem de dados é a presença do fenômeno de overfitting [9]. Como exemplo, tomamos um conjunto de dados conforme ilustrado na Fig. 4(a), na qual foram disponibilizados os dados coletados durante um experimento. Com o objetivo de determinar uma curva que descreva o comportamento observado, selecionam-se arbitrariamente amostras de dados para serem utilizadas na definição do modelo. Será utilizado o método dos mínimos quadrados em polinômios de graus distintos.

A Fig. 4(b) apresenta a aproximação dos pontos de amostragem por um polinômio de segundo grau e seu resultado é visualmente coerente. Dessa forma, podemos dizer que um polinômio de grau 2 representa o fenômeno observado dado dentro de uma tolerância de erro razoável.

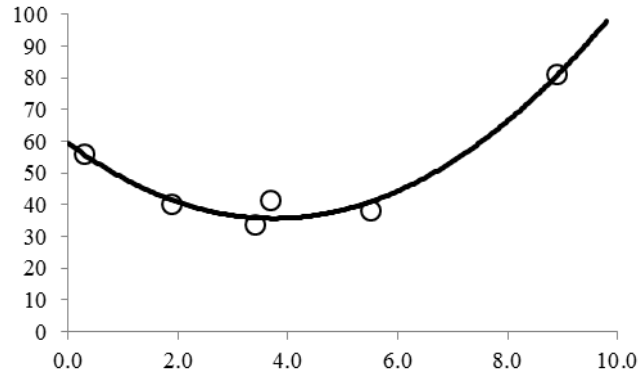
Ao utilizar um polinômio de grau maior, espera-se obter um erro de aproximação igual ou menor. Repetimos o processo de aproximação usando o método dos mínimos quadrados e obtemos um polinômio de grau 6, conforme

observado na Fig. 4(c). Apesar do gráfico do polinômio passar exatamente sobre cada ponto de amostragem e zerar o erro quadrado, a curva diverge sobre o resultado esperado da Fig. 4(a).

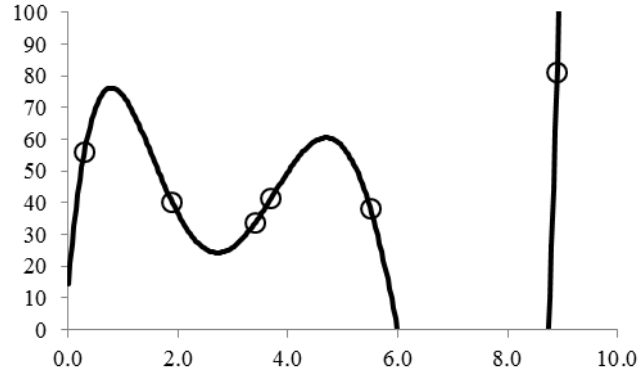
Nesse segundo caso, o modelo não representa bem a realidade e dizemos que o modelo sofreu um overfitting. Apesar do polinômio de grau 6 eliminar o erro de aproximação dos pontos de amostragem, existe um polinômio mais simples que consegue uma boa representação para os dados coletados experimentalmente.




(a)




(b)




(c)

**Figura 4. (a) medição experimental de um determinado comportamento, (b) aproximação adequada usando um polinômio de grau 2 com presença de erro, (c) aproximação usando um polinômio de grau 6, que resultou no fenômeno chamado overfitting.**

Em uma árvore de decisão, esse fenômeno está associado ao desejo de representar usando um numero de nós e ramos maior do que o necessário. Isso significa que o espaço foi subdividido em diversas partes e não há um número suficiente de dados de amostragem para determinar as classes em cada região. O mesmo ocorreria em uma modelagem adaptativa que apresentasse somente operações de adição, mas sem remoção de parte da estrutura.

### *A. Princípio da Navalha de Occam*

"Se há diversas possibilidades, opte pela mais simples. Ela provavelmente é a mais correta". O princípio da navalha de Occam é intuitivamente um ótimo fator de decisão para a escolha de modelos. Por exemplo, se existem polinômios de grau 2 e 6 que representam o mesmo comportamento, então deve-se optar pelo de menor grau.

De fato, existem inúmeras formas de interpretar o significado da palavra "simples". Em uma árvore de decisão, a simplicidade pode estar relacionada com o número de nós, profundidade da árvore ou número de folhas. A decisão se torna subjetiva. A fim de complementar o Princípio da Navalha de Occam e definir o termo "simplicidade", introduzimos os conceitos de Complexidade Algorítmica ou também conhecida como Complexidade de Kolmogorov.

### *B. Complexidade de Kolmogorov ou Algorítmica*

A complexidade de Kolmogorov é denotada por $K_L(x)$ e corresponde ao tamanho da menor descrição da cadeia de entrada x em uma linguagem L. Uma definição equivalente é feita através da formalização de complexidade utilizando uma Máquina de Turing Universal [10]. Nesse caso, define-se $K_M(x)$ como o tamanho da menor representação $<M,w>$ de uma Máquina de Turing *M*, que a partir de uma entrada *w*, gera a cadeia de saída *x*.

Usando a linguagem associada aos computadores podemos dizer, por exemplo, que $K_M(000000000)$ corresponde ao tamanho em bytes do menor programa capaz de imprimir a cadeia de zeros "000000000" na tela. Observa-se que, para cadeias previsíveis, a complexidade de programa se mantém baixa independente da quantidade de símbolos a serem impressos.

$$K_M(00) \sim K_M(000000000) \sim K_M(000...000)$$

O mesmo não ocorre para sequências que apresentam um maior grau de aleatoriedade, como 011000101011. Esse tipo de cadeia requer uma descrição detalhada e o tamanho $K_M$ do programa cresce de acordo com o tamanho da cadeia. De forma independente, Kolmogorov, Solomonoff e Chaitin [11] observaram que o conceito de complexidade está diretamente relacionado com aleatoriedade e compressão de dados.

Um resultado importante da teoria de complexidade algoritmica é ao Teorema da Invariância, demonstrado por Solomonoff [12].

**Teorema da Invariância:** $|K_{M1}(x) - K_{M2}(x)| <$ constante

Isso significa que as complexidades em diferentes máquinas são próximas, apesar da diferença por uma constante. Além disso, esse teorema reforça a idéia de que o conceito de complexidade é universal e não depende da linguagem, máquina utilizada ou estrutura de dados.

Em uma árvore de decisão, a complexidade de Kolmogorov está associada ao tamanho do menor programa necessário para reconstruir a estrutura da árvore. Portanto, uma árvore de decisão "simples" é aquela que possui um menor número de nós, ramos ou folhas.

### C. As subdivisões do espaço

Dizemos que um modelo apresenta overfitting quando a complexidade do modelo é superior a Complexidade de Kolmogorov, K(*x*), associada ao comportamento observado. Na Fig. 5, a aproximação da borda do círculo por retas horizontais e verticais caracteriza uma situação de overfitting. Isso ocorre porque o espaço será dividido em várias regiões menores, que requerem uma quantidade maior de dados de amostragem.

Similar ao comportamento de subdivisão sucessiva da região espacial, a Fig. 6 ilustra o crescimento desbalanceado da estrutura de árvore que pode ser um problema de overfitting. A árvore de decisão possui muitos nós e ramos e, consequentemente, necessitam amostras para classificar cada nó ou folha presente.

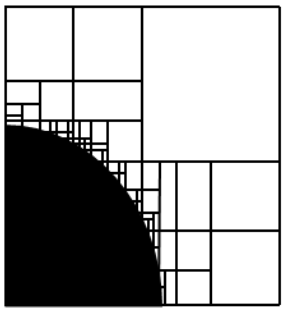


**Figura 5. Aproximação de um círculo por retas horizontais e verticais. Próximo à borda do círculo, há uma série de regiões cada vez menores que são incapazes de aproximar corretamente.**

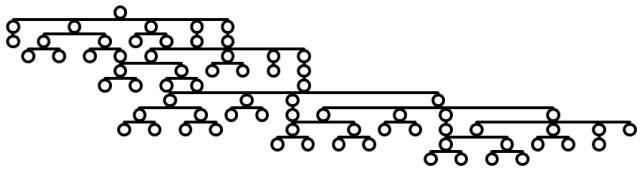


**Figura 6. Árvore de decisão com grande quantidade de nós, representando o comportamento de overfitting.**

Esse problema afeta uma modelagem de autômato adaptativo que apresenta somente operações de adição de dados e não há remoção. A fim de se evitar o overfitting de dados, é importante retirar estruturas redundantes ou com baixa utilização. Esse mecanismo é conhecido, nas árvores de decisão, como poda (prunning) [13].

## VI. Árvores de Decisão Adaptativas

Na área de Tecnologia Adaptativa, os dispositivos apresentam mudança de comportamento para se adequar melhor ao ambiente. Uma árvore de decisão adaptativa é uma extensão da árvore de decisão, que atua como núcleo, acrescido de uma camada adaptativa.

### A. Alteração Estrutural

A camada adaptativa permite que a estrutura hierárquica de nós e ramos seja modificada ao longo da sua utilização a partir das ações elementares de consulta, inserção e remoção. O crescimento da árvore corresponde ao conhecimento que se acumula durante o processo inicial de treinamento.

Usando o formalismo da Árvore Não-Determinística de Decisão Adaptativa, Hemerson [2] propôs uma notação eficiente para descrever as ações adaptativas relacionadas ao modelo da árvore.

? [(no, atributo)]
+ [(no, atributo, (resultado1, filho1), (resultado2, filho2))
- [(no, atributo)]

Uma árvore adaptativa pode ser implementada através de métodos incremental ou completo. O método completo avalia todos os elementos antes de iniciar a criação da árvore de decisão. Após a fase de treinamento, a estrutura continua reagindo aos dados fornecidos e pode se auto-modificar. Esse processo de adquirir novas informações a partir do meio externo é chamado de aprendizagem incremental.

Nem sempre uma abordagem incremental é adequada, uma vez que as atualizações na estrutura da árvore de decisão seguem a ordem dos dados apresentados. A dependência da ordem dos dados é chamada de influência ou tendência (bias, em inglês), na qual resultam diferentes estruturas de acordo com a ordem de apresentação dos dados.

### B. Pode da Árvore

Seguindo o princípio da navalha de Occam, uma estrutura mais simples apresenta uma melhor generalização para as predições de resultados. Sob esse ponto de vista, a remoção de estrutura deve ser vista como parte da aprendizagem da estrutura.

A poda da árvore de decisão remove os ramos e os nós que não acrescentam informação suficiente à estrutura e, portanto, pode ser descartada. A definição sobre relevância é feita pela camada adaptativa usando algum cálculo ou heurística.

De fato, ao simplificar a estrutura da árvore de decisão, o dispositivo requer um menor número de amostras para classificação.

### C. Não-Determinismo

O comportamento não-determinístico permite que o processamento seja feito em paralelo. A vantagem do paralelismo não é em termos de maior poder computacional, mas a possibilidade de explorar outras alternativas. Isso significa que mais de uma hipótese pode ser verificada antes de chegar ao resultado final.

Em uma árvore não-determinística de decisão, a camada adaptativa pode enumerar diferentes modelos para serem comparados com o comportamento a ser previsto.

*D. Uso de atributos contínuos*

A modelagem de árvores de decisão depende do uso de atributos discretos para classificar objetos. Entretanto, uma grande parte dos problemas apresenta atributos contínuos a serem considerados [14]. Uma forma trivial de mapear atributos contínuos em discretos seria utilizar uma correspondência entre intervalos de valores contínuos com atributos discretos. Por exemplo, um animal de 28 metros pode ser caracterizado como de grande porte, enquanto que animais menores que 5 metros seriam considerados de pequeno porte.

Atributos que possuam uma grande quantidade de resultados, como os números de 1 a 1000, podem ser considerados como atributos contínuos. Nesse caso, poderia ser realizada uma divisão pré-definida em regiões de dados, como por exemplo usar as faixas de 1-100, 101-201, e assim por diante, agrupando os valores em um total de 10 conjuntos.

Outra possibilidade é transformar os valores numéricos em expressões booleanas, como por exemplo: predicado[t <= 500] retorna verdadeiro para os primeiros 500 registros e falso para os últimos 500. Assim reduz-se um problema de 1000 possibilidades a apenas dois casos: verdadeiro ou falso.

Um atributo contínuo pode, também, ser associado a mais de um atributo discreto. Por exemplo, usando uma série de atributos para representar intervalos cada vez menores. É possível representar um número na base 2 e utilizar os dígitos mais significativos como atributos discretos. A Fig. 7 ilustra a definição dos atributos discretos $a_1$,..., $a_8$ de acordo com os dígitos, que representam o intervalo dos números inteiros de 0 a 255.

| $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | $a_8$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Figura 7. Número 131 apresenta os atributos discretos $a_1$,..., $a_8$, que apresentam somente os valores 0 ou 1. Dessa forma, representa-se um atributo contínuo através de 8 atributos.**

A utilização de um modelo não-determinístico permite a escolha entre as diferentes formas de discretização de dados, determinando aquela que melhor se adequa ao problema proposto.

*E. Inclusão de novos atributos*

Durante a construção da árvore de decisão, assume-se que a relação entre os atributos é independente, ou seja, um não depende um do outro. Em muitos casos, essa suposição é incorreta e pode levar incapacidade do modelo ser uma boa representação do objeto real.

Uma das causas de ocorrência de overfitting está relacionada com limitações do modelo em relação aos dados obtidos experimentalmente. Um exemplo é a tentativa de modelar uma figura através da leitura de atributos contínuos *x* e *y* como apresentados na Fig. 5, observa-se uma figura circular (primeiro quadrante) sendo aproximado por uma série de quadrados menores. Embora os atributos *x* e *y* sejam tratados independentes, sabemos que seguem a equação $x^2 + y^2 < c$.

A inclusão de novos atributos calculados com base nos existentes cria a interdependência entre eles. Por exemplo, a substituição das coordenadas cartesianas (x,y) por coordenadas polares aumentaria a eficiência da árvore de decisão para modelar o círculo, Portanto, a camada adaptativa pode realizar transformações entre diferentes sistemas de coordenadas, assim como operações de translação, rotação e escala.

Conhece-se a limitação de árvores de decisão não possuirem uma boa representação para operações de paridade ou multiplexação, uma vez que o tamanho da árvore resultante é grande. A utilização de um mecanismo gerador de novos atributos combinados pode favorecer sua representação.

## VII. EXEMPLO

Abordamos o problema de uma figura geométrica em um espaço bidimensional usando as coordenadas cartesianas x e y. A capacidade de representação do objeto por meio de uma árvore de decisão depende da modelagem adotada, na qual formas geométricas mais complexas apresentam maior propensão ao erro. Por exemplo, usando retas é mais difícil mapear a borda de uma figura curva do que os lados de um quadrado.

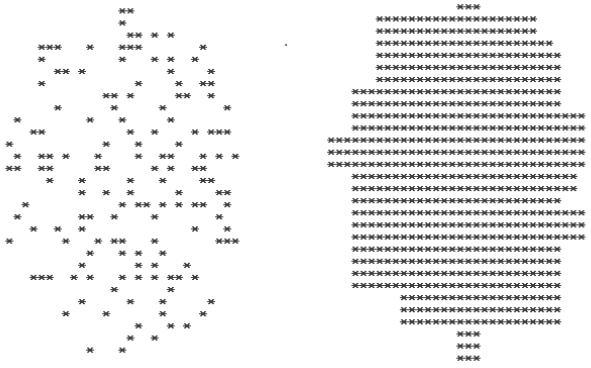


**Figura 8. Mapeamento de figuras**

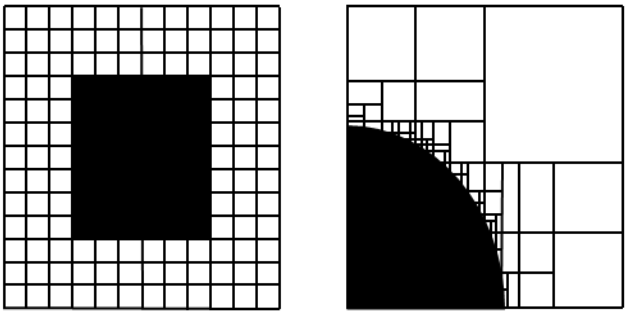


**Figura 9. Diferença no mapeamento de regiões entre um quadrado e um arco**

Em posições distintas em um espaço de 50x50, foram desenhados círculos de diferentes tamanhos. A tabela 1

apresenta as posições do centro (x,y) do círculo seu respectivo raio. O problema proposto é construir uma árvore de decisão que modela a figura proposta. Em outras palavras, é possível determinar se um ponto qualquer está dentro ou fora da região do da figura com base nas suas coordenadas.

Foi escolhida como estrutura núcleo a implementação de árvores de decisão usando autômatos adaptativos, baseado no Adaptive-NDD, com seleção de atributos seguindo uma ordem pré-estabelecida. A partir desse modelo, modificamos a estrutura para apresentar um comportamento determinístico e que ocorresse a poda da árvore em casos triviais, como um conjunto de sub-ramos que pertencem à mesma classe.

### A. *Árvore de Decisão*

Os atributos contínuos foram discretizados usando a representação do número na base binária e associando cada dígito a um novo atributo discreto, caracterizando um algoritmo não-supervisionado. Por exemplo, a representação da posição no eixo X utiliza uma série de atributos booleanos $a_{x1}$, $a_{x2}$,..., $a_{xn}$ que os dígitos da coordenada *x* na base binária.

Na Tabela 1, as colunas *x*, *y* e *raio* representam as características do círculo, enquanto o tamanho resultante da árvore de decisão está apresentado na coluna *tree*. Observamos que os tamanhos das árvores de decisão variaram entre 15 a 200.

| x | y | size | tree |
|---|---|---|---|
| 6 | 6 | 3 | 21 |
| 12 | 12 | 3 | 27 |
| 17 | 17 | 3 | 15 |
| 22 | 22 | 3 | 15 |
| 25 | 25 | 3 | 33 |
| 28 | 28 | 3 | 26 |
| 8 | 8 | 5 | 35 |
| 14 | 14 | 5 | 44 |
| 19 | 19 | 5 | 55 |
| 24 | 24 | 5 | 64 |
| 27 | 27 | 5 | 50 |
| 30 | 30 | 5 | 37 |
| 10 | 10 | 7 | 57 |
| 16 | 16 | 7 | 50 |
| 21 | 21 | 7 | 76 |
| 26 | 26 | 7 | 46 |
| 29 | 29 | 7 | 56 |
| 32 | 32 | 7 | 39 |
| 14 | 14 | 11 | 110 |
| 20 | 20 | 11 | 140 |
| 25 | 25 | 11 | 110 |
| 30 | 30 | 11 | 121 |
| 33 | 33 | 11 | 122 |
| 36 | 36 | 11 | 107 |
| 18 | 18 | 15 | 137 |
| 24 | 24 | 15 | 116 |
| 29 | 29 | 15 | 200 |
| 34 | 34 | 15 | 184 |
| 37 | 37 | 15 | 119 |
| 40 | 40 | 15 | 61 |

Como observado pela variação dos valores na coluna *tree* da Tabela 1, foram geradas árvores de decisão com diferentes complexidades,. Dependendo da posição da figura geométrica, foi necessário um determinado número de nós para representá-la. O método de discretização, portanto, influencia bastante no tamanho da árvore de decisão.

### B. *Árvore de Decisão Adaptativa*

Em uma árvore não-determinística de decisão, torna-se possível enumerar diferentes modelos a serem considerados. A camada adaptativa atua na geração de novos atributos, que são calculados em função dos já existentes. Utilizaremos, nesse exemplo, uma ação adaptaviva que transforma as coordenadas cartesianas em coordenadas polares, adicionado de uma operação de translação.

A representação feita na Fig. 10 ilustra algumas das possibilidades de transformações entre coordenadas, acrescido de um vetor T de translação entre os sistemas. A idéia é utilizar o método adaptativo para fazer a regulagem correta de parâmetros até determinar um valor adequado de T, cujo valor é inicialmente desconhecido.

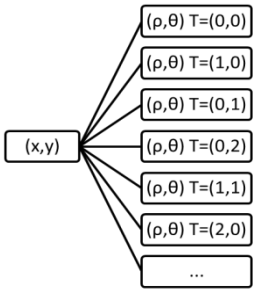



**Figura 10. Enumeração entre sistemas de coordenadas cartesiana em coordenadas polares, com a adição de um vetor de translação T.**

Ao escolher a árvore de decisão com a menor complexidade associada, encontraremos os valores ideais de T. Esse ajuste pode ser feito, inclusive, em tempo real e favorecer a utilização de métodos incrementais.

Após o ajuste do modelo, verifica-se que a árvore de decisão escolhida apresentou de 10 a 12 nós. Essa baixa complexidade é independente da posição do centro do círculo, mas está relacionado com o total de dígitos necessários para representar o raio do círculo na base binária. Esse resultado é muito superior ao caso da árvore de decisão sem a camada adaptativa, no qual o tamanho variava entre 15 a 200 nós.

## VIII. CONSIDERAÇÕES FINAIS

As árvores de decisão adaptativas são dispositivos capazes de incorporar novas informações em sua própria estrutura através das auto-modificações. A camada adaptativa, que corresponde a um segundo nível e independente da estrutura núcleo, é responsável por enumerar diferentes modelos e decidir qual deles possui a melhor representação.

Como contribuição fundamental do artigo, propõe-se que o modelo de árvore de decisão adaptativa seja capaz de associar novos atributos com base nos já existentes. Os atributos calculados permitem obter melhores resultados mantendo-se a mesma estrutura do núcleo. O resultado imediato é a possibilidade de modelar adequadamente operações simples, como paridade, ou exclusivo ("XOR") e multiplexação, que

não apresentavam bons resultados com uma árvore de decisão tradicional.

A complexidade de Kolmogorov, apesar de incomputável, formaliza um conceito importante na escolha do melhor modelo: a simplicidade. Durante o artigo, o tamanho da árvore foi a métrica utilizada para determinar a complexidade do modelo, mas pode ser inadequado para outras situações.

O tamanho final da árvore de decisão adaptativa pode ser visto como uma forma de caracterizar o problema [14], ou seja, transforma-se um problema de classificação em reconhecimento de objeto.

## REFERÊNCIAS


[1] J. J. Neto, "Adaptive rule-driven devices – general formulation and a case study". In CIAA'2001 Sixth International Conference on Implementation and Application of Automata. Springer-Verlag.

[2] H. Pistori, "Adaptive Non-Deterministic Decision Trees: General Formulation and Case Study".

[3] H. Lewis and C. Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall, 1997.

[4] R. Rocha e J. Neto, Autômato adaptativo, limites e complexidade em comparação com máquina de Turing. In: Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000. São Paulo 2001.

[5] S. B. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques, Informatica 31(2007) 249-268, 2007

[6] L. Hyafil, R.L. Rivest, Constructing optimal binary decision trees is NP-Complete. Information Processing Letters, 5(1), 15–17, 1976

[7] J. R. Quinlan, Induction of Decision Trees. Machine Learning (Mar. 1986), 81-106

[8] J. R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993

[9] T. G. Dietterich, Overfitting and under-computing in machine learning. Computing Surveys, 27 (3), 326-327, 1995

[10] Li, Ming and Vitányi, Paul, *An Introduction to Kolmogorov Complexity and Its Applications*, *Springer*, 1997

[11] R. Solomonoff, Complexity-Based Induction Systems: Comparisons and Convergence Theorems, IEEE Trans. on Information Theory, Vol. IT-24, No. 4, pp. 422-432, July 1978

[12] R. Solomonoff, A Formal Theory of Inductive Inference Part I, Information and Control, Part I: Vol 7, No. 1, pp. 1-22, March 1964

[13] J. Mingers, An Empirical Comparison of Pruning Methods for Decision Tree Induction, Machine Learning, Vol. 4, 1989, pp.227-243

[14] P.M. Murphy, An empirical analysis of the benefit of decision tree size biases as a function of concept distribution, TR. 95-29, Department of Information and Computer Science, University of California, Irvine, 1995



**Fabricio S. Catae** é graduado em Engenharia Elétrica (2002) na ênfase de Automação e Controle e aluno regular do programa de mestrado da Escola Politécnica, Universidade de São Paulo. Sua área de atuação está relacionada com Mecanismos de Inferência e Linguagem Natural, trabalhando em linhas de pesquisa do Laboratório de Linguagens e Técnicas Adaptativas.

**Ricardo Luis A. Rocha** é natural do Rio de Janeiro-RJ e nasceu em 29/05/1960. Graduou-se em Engenharia Elétrica modalidade Eletrônica na PUC-RJ, em 1982. É Mestre e Doutor em Engenharia de Computação pela EPUSP (1995 e 2000, respectivamente). Suas áreas de atuação incluem Tecnologias Adaptativas, Fundamentos de Computação e Modelos Computacionais. Dr. Rocha é membro da ACM (Association for Computing Machinery) e da SBC (Sociedade Brasileira de Computação).

# Modelagem de Autômato Adaptativo para a definição dinâmica do intervalo de amostragem em Rede de Sensores Sem Fio

I. M. Santos, M. A. Dota e C. E. Cugnasca

**Resumo — As Redes de Sensores Sem Fio têm potencial para diferentes aplicações, dentre elas destaca-se o monitoramento de ambientes. Este trabalho tem como objetivo apresentar a formalização de uma técnica para minimizar o gasto de energia dos nós sensores em uma Rede de Sensores Sem Fio durante a coleta de dados. Essa técnica considera a utilização de Autômatos Adaptativos e a possibilidade de a rede de sensores variar dinamicamente o intervalo (tempo) entre uma amostragem e outra. Dessa forma, espera-se que a rede de sensores utilize automaticamente intervalos longos enquanto estiver coletando informações dentro de um limite de valores considerado normal ou aceitável. Entretanto, quando os nós coletarem dados considerados fora da normalidade, espera-se que os intervalos de amostragem sejam reduzidos, de modo que a rede de sensores monitore o fenômeno com mais detalhe. Com a aplicação dessa estratégia ocorre uma economia no consumo de energia nos nós sensores da rede, sem que a rede perca eficiência no monitoramento dos fenômenos. A formalização da técnica é feita a partir da modelagem de um Autômato Adaptativo que possui ações adaptativas para adequar dinamicamente o intervalo de amostragem dos nós da rede de sensores.**

**Palavras-chave – Redes de Sensores Sem Fio, Conservação de Energia, Autômato Adaptativo.**

## III. Introdução

Uma Rede de Sensores Sem Fio (RSSF) é um tipo especial de rede *ad hoc* com capacidade de coletar e processar informações de maneira autônoma, estando esses sensores distribuídos em uma determinada área [1, 2, 3]. Esse tipo de rede pode ser utilizado em um grande conjunto de aplicações, como por exemplo, no monitoramento ambiental, na agricultura de precisão, nos processos industriais, nos sistemas embarcados em automóveis e aeronaves, nos sistemas de segurança, entre outros [4, 5, 6].

Dentre as diversas aplicações, pode-se destacar o monitoramento agrícola, que envolve o acompanhamento e a observação contínua de uma área de plantio, com o objetivo de avaliar as mudanças ocorridas nesse ambiente. Esse monitoramento é importante no processo de tomada de decisão e auxilia na solução de problemas, como ataques de pragas e doenças, correção do solo, aplicação de insumos e mudanças climáticas que podem prejudicar a produtividade da plantação, entre outros aspectos. Entretanto, esse tipo de aplicação não necessita de um monitoramento com alta frequência de coleta de dados, pois as condições ambientais se modificam lentamente.

Um dos principais desafios na aplicação das RSSF é a economia de energia dos nós sensores, de modo a propiciar um maior tempo de vida útil da rede. Na prática, recarregar manualmente as baterias dos nós da rede é inconveniente, sendo cada nó responsável por utilizar de maneira eficiente sua carga de energia disponível.

Este trabalho apresenta uma proposta para o ajuste dinâmico dos intervalos entre amostragens efetuadas pelos nós da rede de sensores. Em aplicações nas quais os dados se modificam lentamente, ou o monitoramento esteja obtendo dados dentro de um padrão de normalidade (como na agricultura de precisão), espera-se que os nós utilizem intervalos longos e passem a adotar intervalos curtos caso sejam observados dados inesperados. Com a adoção dessa estratégia, pode-se obter uma maior economia no gasto de energia, visto que o número de medições e envio de dados pelos nós da rede é reduzido, sem que isso signifique perda de eficiência no monitoramento do ambiente pela RSSF.

Para controlar a dinamicidade dos intervalos, é proposto um Autômato Adaptativo (AA) [7] que será executado em cada nó sensor, de modo que cada nó da rede será autônomo na decisão do seu intervalo de amostragem, em função dos fenômenos que estão sendo monitorados.

Este trabalho apresenta na Seção II uma breve contextualização do monitoramento agrícola, as potencialidades de se utilizar RSSF nesse contexto e descreve a estratégia adaptativa que será utilizada pelo AA. A Seção III discute e apresenta o AA para o problema de definição do intervalo de amostragem em RSSF, além de apresentar a Tabela de Decisão Adaptativa e discutir sua verificação. Finalmente na Seção IV são apresentadas as considerações finais do trabalho.

## IV. Monitoramento de ambiente agrícola com RSSF.

A agricultura de precisão consiste em um manejo específico da área cultivada, ou seja, consiste em dividir o terreno em parcelas e tratá-lo de modo diferenciado, dependendo das necessidades de cada parcela. Dessa forma, conseguem-se vantagens econômicas, aumento de produção e benefícios para o meio ambiente [8]. A aplicação das RSSF na agricultura de precisão é uma alternativa promissora, possibilitando um melhor monitoramento da cultura e das propriedades do ambiente de cultivo [9].

No monitoramento agrícola, geralmente os dados

(fenômenos monitorados) variam lentamente. Assim, uma RSSF não precisaria monitorar o ambiente constantemente em intervalos considerados curtos, pois isso representaria desperdício de energia. Outro aspecto inerente ao monitoramento agrícola é o fato de que se um fenômeno que está sendo monitorado apresenta valores dentro de um intervalo considerado normal (a faixa de normalidade é definida sob a orientação de um agrônomo e de acordo com o objetivo do monitoramento), então o sistema de monitoramento continua acompanhando o ambiente, avaliando se ocorre alguma variação que esteja fora desse intervalo de normalidade, para só então atuar no sistema. Dessa forma, pode-se considerar uma estratégia em que os nós da rede de sensores adotem intervalos longos para a coleta de dados enquanto as informações obtidas corresponderem a um padrão de normalidade. Quando ocorrer a obtenção de informações fora do padrão de normalidade, então os nós da rede de sensores deverão reduzir o intervalo de amostragem de dados, de modo a intensificar o monitoramento e registrar os dados da variação.

A Figura 1 ilustra a estratégia, sendo "*a*" correspondente ao valor mínimo, enquanto que "*b*" corresponde ao valor máximo do intervalo de valores considerados como medições normais. No decorrer do tempo (eixo *x*), cada instante de coleta de informação de um sensor é representado por um ponto. Observa-se que enquanto os dados obtidos estão dentro do padrão de normalidade, o intervalo entre uma amostragem e outra é maior do que quando ocorre uma medição fora da normalidade.

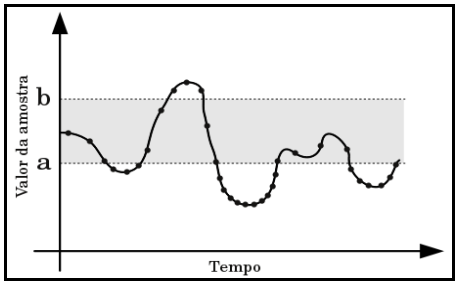



Fig. 1. Exemplo de uma sequência de medidas de um nó sensor ao longo do tempo. Quando ocorrem medidas fora da normalidade (limites a e b) o intervalo entre uma amostra e outra é reduzido.

Com essa estratégia, espera-se economizar energia enquanto o ambiente apresentar uma situação de normalidade e garantir a eficiência do monitoramento do ambiente em situações de anormalidade [10].

## V. DESCRIÇÃO DO MODELO ADAPTATIVO

Um AA é uma máquina de estados à qual são impostas sucessivas alterações resultantes da aplicação de ações adaptativas associadas às regras de transições executadas pelo autômato [7]. Dessa maneira, estados e transições podem ser eliminados ou incorporados ao autômato em decorrência de cada um dos passos executados durante a análise da entrada. De maneira geral, pode-se dizer que o AA é formado por um dispositivo convencional, não-adaptativo, e um conjunto de mecanismos adaptativos responsáveis pela automodificação do sistema.

Para determinar a mudança nos intervalos de amostragem nos nós sensores, é proposto um AA que assume as seguintes considerações:

- Cada estado do AA corresponde a um intervalo de amostragem distinto para o nó sensor.
- Se o valor da amostra coletada pelo nó sensor mantém-se estável, então o AA permanece no mesmo estado.
- Se o valor da amostra coletada pelo nó sensor "piorar", ou seja, extrapolar o padrão de normalidade, então o AA transita para um estado que corresponda a um intervalo de amostragem menor (aumento da frequência das amostras), por meio de uma função adaptativa.
- Se o valor da amostra "melhorar", então o autômato transita para um estado que corresponde a um intervalo de amostragem maior (redução da frequência das amostras).

À medida que os dados obtidos pelos nós sensores atingem o intervalo de normalidade, o AA deve transitar de volta ao estado que corresponde ao maior intervalo, reduzindo, portanto, a frequência de coleta e envio de dados na rede.

Uma proposta de um AA é apresentada na Figura 2. Na figura o símbolo < corresponde a uma leitura de uma amostra com valor abaixo do padrão de normalidade, com variação superior a um limite mínimo, que aqui será referenciado como k%. O símbolo > representa uma leitura superior ao padrão de normalidade. Portanto, espera-se que ocorra uma ação adaptativa, gerando um novo estado no autômato que corresponda a um intervalo mais curto, para o caso de uma leitura abaixo ou superior ao padrão de normalidade.

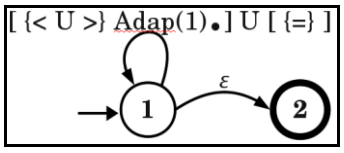



Fig. 2. Proposta de AA para determinar o intervalo de amostragem em RSSF.

A Figura 3 ilustra um possível exemplo de execução do AA proposto. Na Figura 3(a), o AA se encontra em seu estado inicial. Considerando uma suposta leitura, pelo sensor, de um valor abaixo do padrão de normalidade estabelecido, ocorre uma ação adaptativa, criando um novo estado (#1). A nova configuração do AA pode ser observada na Figura 3(b). O novo estado corresponde a um intervalo de amostragem menor do que o intervalo estabelecido pelo estado 1. Nessa nova configuração, caso o nó sensor volte a executar uma nova leitura e a obter um dado "pior", ou seja, com um valor ainda menor (<), irá ocorrer uma nova ação adaptativa, criando um novo estado com intervalo de amostragem menor que o atual. Caso ocorra uma amostragem "melhor", então o autômato transita para um estado com intervalo de amostragem maior (retornando no exemplo para o estado 1).

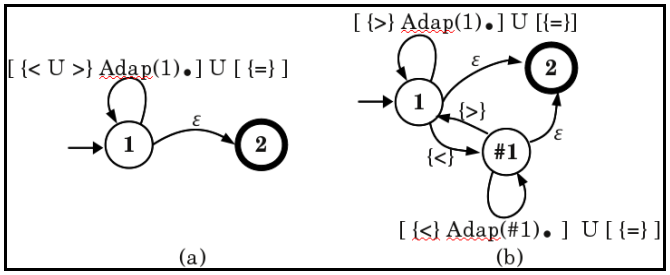



Fig. 3. Exemplo de execução da primeira ação adaptativa, considerando a leitura de um valor abaixo do limite inferior.

O comportamento do AA é análogo para o caso da leitura de dados acima do padrão de normalidade.

### A. *Tabela de Decisão Adaptativa*

Durante a execução de uma transição adaptativa, o AA sofre alguma mudança em sua configuração (eliminando, acrescentando, ou simplesmente modificando estados e transições). Isto faz com que uma nova máquina de estados apareça, no lugar da anterior, caracterizando a execução de um passo adicional [7].

Uma função adaptativa é formada por um conjunto de ações adaptativas, que podem ser executadas em sequência na ocasião da aplicação da função. Há três ações elementares, que podem ser usadas para compor as ações adaptativas que permitem: consultar (representado pelo "?"), eliminar (representado pelo "-") e adicionar (representado pelo "+") transições no autômato. Uma Tabela de Decisão Adaptativa (TDA) permite representar as regras de uma função adaptativa e consequentemente representar toda a lógica do AA de maneira intuitiva.

Considerando o AA, apresentado no diagrama da Figura 2, para o problema de determinação autônoma do intervalo de amostragem em uma RSSF, propõe-se a respectiva TDA apresentada na Tabela I.

### B. *Validação da TDA*

Visando verificar a TDA apresentada na Tabela I, foi utilizado o *software* AdapTools [11], que permite representar, implementar e simular o comportamento de um AA a partir da sua TDA, em função de uma entrada pré-determinada. O sistema inclui recursos de depuração, visualização de variáveis, animação gráfica, bem como mecanismo para controle de projetos e execução simultânea de múltiplos dispositivos.

As simulações executadas demonstraram o comportamento esperado do AA, portanto, a correta construção da TDA. A Figura 4 corresponde à representação da TDA no *software* AdapTools e a Figura 5 à simulação gráfica do AA em um determinado instante.

## VI. Conclusões

As RSSF têm apresentado potencial de aplicação em diversas áreas. Entretanto, um dos seus principais desafios é a economia de energia sem que isso signifique perda de eficiência da rede de sensores. Para aplicações que não exigem um monitoramento com alta frequência de coleta de dados, como no monitoramento ambiental e na coleta de dados em agricultura de precisão, pode-se considerar uma estratégia em que o intervalo entre as amostragens dos dados seja longo enquanto as informações obtidas estiverem dentro de um padrão esperado. Ao ocorrer fenômenos fora desse padrão, então a rede se ajusta, de maneira autônoma, de modo a monitorar o evento com mais acurácia. Essa estratégia representa uma importante economia de energia e significa um maior tempo de vida útil da RSSF.

TABELA I

Tabela de Decisão Adaptativa referente ao AA proposto para a determinação do intervalo de amostragem em uma RSSF

| | Label | Tag | Condições | | Ações | | | Funções Adaptativas | | Parâmetros | | | | Geradores | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Estado = | Entrada = | Estado ← | Consome | Aceita ← | R1 | R2 | P1 | P2 | P3 | P4 | G1 | G2 |
| 1 | | S | | | 1 | √ | | | | | | | | | |
| 2 | | R | 1 | = | 1 | √ | | | | | | | | | |
| 3 | | R | 1 | | | | x | | | | | | | | |
| 4 | | R | 1 | ε | 2 | | | | | | | | | | |
| 5 | | R | 1 | < | 1 | | | √ | | 1 | 1 | | | | |
| 6 | | R | 1 | > | 1 | | | | √ | | | 1 | 1 | | |
| 7 | | R | 2 | | | | √ | | | | | | | | |
| 8 | | E | | | | | | | | | | | | | |
| | | H | | | | | | √ | | √ | √ | | | √ | |
| | | - | P1 | < | P2 | | | √ | | P1 | P2 | | | | |
| | | + | P1 | < | G1 | √ | | | | | | | | | |
| | | + | G1 | > | P1 | √ | | | | | | | | | |
| | | + | G1 | = | G1 | √ | | | | | | | | | |
| | | + | G1 | < | G1 | | | √ | | G1 | G1 | | | | |
| | | + | G1 | ε | 2 | | | | | | | | | | |
| | | H | | | | | | | √ | | | √ | √ | | √ |
| | | - | P3 | > | P4 | | | | √ | | | P3 | P4 | | |
| | | + | P3 | > | G2 | √ | | | | | | | | | |
| | | + | G2 | < | P3 | √ | | | | | | | | | |
| | | + | G2 | = | G2 | √ | | | | | | | | | |
| | | + | G2 | > | G2 | | | | √ | | | G2 | G2 | | |
| | | + | G2 | ε | 2 | | | | | | | | | | |

Este trabalho apresentou como proposta um AA capaz de modificar, de maneira autônoma, a configuração do tempo de intervalo entre uma amostra e outra em um nó sensor em uma RSSF.

Em trabalhos futuros, a eficiência e a economia de energia promovida pelo AA devem ser verificadas em um ambiente de simulação de uma RSSF (como o NS2). Além disso, outros aspectos podem sofrer adaptabilidade, como o padrão de normalidade, que pode, por exemplo, oscilar durante um determinado período. Essas perspectivas demonstram que a tecnologia adaptativa apresenta potencial em ser aplicada à área de RSSF.

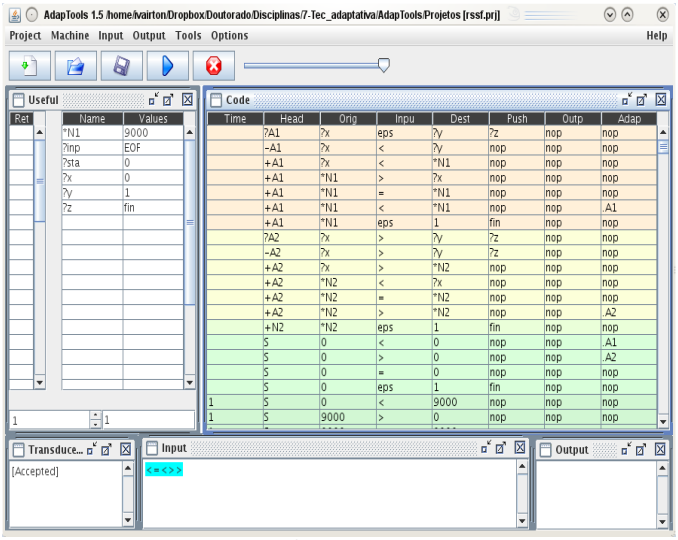



Fig. 4. Tela principal do *software* AdapTools com o AA proposto representado.

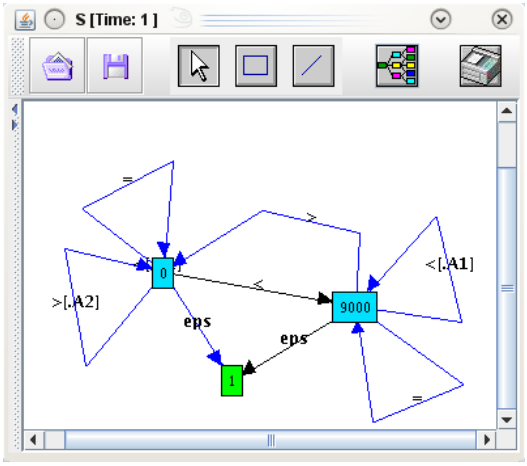



Fig. 5. Tela com a representação gráfica da simulação do AA apresentado.

## AGRADECIMENTOS

Os autores agradecem a Fundação de Amparo à Pesquisa do Estado de Mato Grosso – FAPEMAT – pelo apoio a este trabalho via projetos de pesquisa Nº 465450/2009 e Nº 278718/2010.

## REFERÊNCIAS

[1] I. F. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, E. CAYIRCI. "Wireless sensor networks: a survey", Computer networks. 38, p.393-422, 2002.

[2] M. TUBAISHAT, S. MADRIA. "Sensor networks: an overview". IEEE Potentials. 22 (2), 2003.

[3] P. GAJBHIYE, A. MAHAJAN. "A survey of architecture and node deployment in Wireless Sensor Network." In: Applications of Digital Information and Web Technologies, ICADIWT, p.426-430, 2008.

[4] D. ESTRIN, L. GIROD, G. POTTIE, M. SRIVASTAVA. "Instrumenting the world with wireless sensor networks." In: International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, USA, 2001.

[5] G. J. POTTIE, W. J. KAISER. "Wireless integrated network sensors." Communications of the ACM 43, p.51–58, 2000.

[6] D. ESTRIN, R. GOVINDAN, J. HEIDEMANN, S. KUMAR. "Next century challenges: Scalable coordination in sensor networks." In: Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCom'99), Seattle, Washington, USA, ACM Press, 1999.

[7] J. J. NETO. "Adaptive rule-driven devices – general formulation and case study" CIAA'01: Revised Papers from the 6th International Conference on Implementation and Application of Automata, London, UK: Springer-Verlag, 2002, pp. 234-250, ISBN 3-540-00400-9.

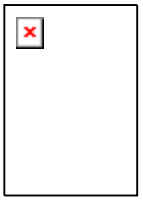
[8] J. P., MOLIN, "Tendências da agricultura de precisão no Brasil". In: Congresso Brasileiro de Agricultura de Precisão, 2004. Anais do Congresso Brasileiro de Agricultura de Precisão - ConBAP 2004. Piracicaba, p. 1-10, 2004.

[9] I. M. SANTOS, M. A. DOTA, C. E. CUGNASCA. "Visão Geral da Aplicabilidade dE Redes de Sensores Sem Fio no Monitoramento Agrícola no estado de Mato Grosso". Anais do Congresso Brasileiro de Agricultura de Precisão – ConBAP 2010. Ribeirão Preto/SP, Setembro de 2010.

[10] J. C. C. BENAVENTE, C. E. CUGNASCA, H. P. SANTOS. "Um Estudo da Variabilidade Microclimática em um Vinhedo Cultivado sob Cobertura Plástica Mediante o uso de uma Rede de Sensores Sem Fio". Anais do Congresso Brasileiro de Agricultura de Precisão- ConBAP 2010. Ribeirão Preto – SP. Setembro de 2010.

[11] L. D. JESUS, D. G. D. SANTOS, A. A. D. CASTRO, H. PISTORI. WTA 2007 - II.2 - "AdapTools 2.0: Implementation and Utilization Aspects." IEEE Latin Am. Trans. 5, 527-532 (2007).

**Ivairton Monteiro Santos** é graduado em Ciência da Computação pela Universidade Federal de Mato Grosso (2002) e mestre em Ciência da Computação pela Universidade Federal Fluminense (2005). Atualmente é aluno de doutorado da Escola Politécnica da USP, sob a orientação do Prof. Dr. Carlos Eduardo Cugnasca. É membro do Laboratório de Automação Agrícola da Escola Politécnica da USP e professor da Universidade Federal de Mato Grosso, no Campus Universitário do Araguaia. Tem experiência na área de Ciência da Computação e seus interesses em pesquisa concentram-se em Redes de Sensores Sem Fio e otimização combinatória.



**Mara Andrea Dota** é graduada em Ciência da Computação pela Universidade Estadual Paulista Júlio de Mesquita Filho (1998) e mestre em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo (2001). Atualmente é aluna de doutorado da Escola Politécnica da USP, sob a orientação do Prof. Dr. Carlos Eduardo Cugnasca. É membro do Laboratório de Automação Agrícola da Escola Politécnica da USP e professora da Universidade Federal de Mato Grosso no Campus Universitário de Rondonópolis/MT. Tem experiência na área de Ciência da Computação e seus interesses em pesquisa concentram-se em Redes de Sensores Sem Fio, avaliação de desempenho e Sistemas Distribuídos.

**Carlos Eduardo Cugnasca** é graduado em Engenharia de Eletricidade (1980), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Elétrica (1993). É livre-docente (2002) pela Escola Politécnica da Universidade de São Paulo (EPUSP). Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo, e pesquisador do LAA - Laboratório de Automação Agrícola do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Supervisão e Controle de Processos e Instrumentação, aplicadas a processos agrícolas e Agricultura de Precisão, atuando principalmente nos seguintes temas: instrumentação inteligente, sistemas embarcados em máquinas agrícolas, monitoração e controle de ambientes protegidos, redes de controle baseados nos padrões CAN, ISO11783 e LonWorks, Redes de Sensores Sem Fio e computação pervasiva. É editor da Revista Brasileira de Agroinformática (RBIAgro).

# Projeto e implementação de uma linguagem de programação para a execução de algoritmos adaptativos (18 Outubro 2010)

J. A. Sabaliauskas[8], R. A. Rocha[9]

***Resumo* - Este artigo descreve uma proposta de implementação de uma linguagem de programação de alto nível capaz de expressar adequadamente algoritmos adaptativos seguindo a proposta de linguagens adaptativas descritas em [1] e a proposição de um paradigma de programação adaptativa descrito em [2]. Ao invés de se especificar toda uma nova linguagem de programação, foi escolhida a linguagem de programação Oberon [5] e a ela adicionadas construções sintáticas para a expressão de algoritmos adaptativos junto à programação imperativa usual. Para a geração de código executável foi utilizada a LLVM [3] (Low Level Virtual Machine), uma biblioteca disponível para o estudo de compiladores e de algoritmos de otimização, capaz de gerar código executável para diversas arquiteturas existentes. Essa proposta foi realizada e exercitada, os resultados foram mostrados no anexo do artigo.**

***Palavras chave*— Programação, Compilação, Adaptatividade, Oberon, LLVM, recursão de cauda.**

## I. Nomeclatura

LLVM *Low Level Virtual Machine* [3]. Conjunto de componentes para a implementação de compiladores e máquinas virtuais

DAG *Direct Acyclic Graph*. Tipo de grafo que não contém ciclos.

## II. Introdução

A ADAPTATIVIDADE é a técnica na qual um programa altera a si próprio em tempo de execução para resolver determinado problema. Os algoritmos que utilizam essa técnica de modificação de código são chamados de algoritmos adaptativos.

Entre as décadas de 1950 e 1970 os programadores utilizavam essa técnica para minimizar a utilização de memória pelo programa pois as memórias possuíam baixa capacidade de armazenamento. As linguagens de programação utilizadas nessa época eram linguagens de baixo nível (linguagem de máquina ou linguagens de montagem) onde o código do programa era expresso em uma linguagem de fácil compreensão pela máquina e difícil compreensão pelo ser humano.

Com o aumento da complexidade dos sistemas de software essas técnicas de programação tornaram-se uma fonte de erros devido à dificuldade de leitura do código em linguagens de baixo nível por outros programadores, pouco poder de expressão da linguagem para expressar a adaptatividade e liberdade total à utilização de recursos do sistema computacional.

Para lidar com o aumento da complexidade dos sistemas, a engenharia de software passou a guiar o processo de desenvolvimento através de um conjunto de técnicas consideradas eficientes para a implementação de sistemas o que fez com que algumas técnicas caíssem em desuso, entre elas a alteração do próprio código durante a execução.

O surgimento das linguagens de alto nível contribuiu com o desuso das técnicas de modificação do próprio código durante execução pois não forneciam mecanismos de modificação de código em tempo de execução e as linguagens de baixo nível foram aos poucos abandonadas ficando com uso restrito à tarefas específicas.

Por um grande período a área de programas adaptativos ficou estagnada até o surgimento de novas aplicações nas áreas de proteção de programa, compressão de código, inteligência artificial, otimização de código, etc.

## III. Motivação

As linguagens de alto nível atuais não foram projetadas para uma correta expressão de algoritmos adaptativos, devido aos problemas que o uso não estruturado dessa técnica pode trazer.

Para se implementar a adaptatividade atualmente pode-se

[8] Jorge Augusto Sabaliauskas <jaugustosaba@gmail.com> é aluno do ultimo ano de graduação de engenharia de computação da Escola Politécnica da Universidade de São Paulo

[9] Ricardo Luís Azevedo da Rocha <ricardoluis.rocha@gmail.com> é docente do Laboratório de Tecnologia Adaptativa (LTA) da Escola Politécnica da Universidade de São Paulo (www.pcs.usp.br/~lta).

recorrer ao uso de linguagens de baixo nível, linguagens interpretadas ou a adição de um interpretador ao programa.

Os mesmos problemas que foram encontrados no passado serão obtidos ao se utilizar alguma linguagem de baixo nível para expressar um algoritmo adaptativo. Esses códigos são dependentes da arquitetura do computador em uso e determinarão a portabilidade do programa. A técnica de alteração do próprio código de máquina é considerado ilegal pelos sistemas operacionais modernos. Para se utilizar essa técnica será necessário utilizar-se de chamadas de sistema a fim de remover essa proteção.

As linguagens interpretadas fornecem uma alternativa para a implementação de adaptatividade. A maioria das linguagens de programação destinadas à execução através de um interpretador possuem uma função chamada *eval*. A função *eval* faz com que um fragmento de código seja executado em determinado contexto. A usual presença dessa função em linguagens interpretadas é devido a reaproveitamento de componentes do próprio interpretador para implementar essa função. Em uma linguagem interpretada qualquer pode-se simular a adaptatividade através do uso da função *eval*, gerando durante a execução do programa um trecho de código que será executado.

As linguagens interpretadas não podem ser utilizadas em todas as situações por possuírem um menor desempenho quando comparadas à linguagens compiladas. Essa perda de desempenho é acentuada ainda mais em programas adaptativos devido ao uso excessivo da função *eval* durante a execução.

Para tentar agilizar a execução do programa, ou caso a linguagem de programação não seja interpretada (não possui o comando *eval*), pode-se acrescentar ao programa um interpretador que ficará responsável exclusivamente por executar os algoritmos adaptativos. Pode-se construir um interpretador dedicado aos algoritmos adaptativos ou então adotar alguma linguagem de extensão disponível (a maioria das linguagens de programação interpretadas permitem que sejam usadas também como linguagens de extensão). A utilização do interpretador dedicado a execução de algoritmos adaptativos pode não ter um ganho de performance significativo caso o programa utilize intensamente algoritmos adaptativos.

Nas três formas de implementar adaptatividade o algoritmos adaptativo não fica expresso explícitamente, necessitando de documentação adicional (comentários ou documentos) para informar a presença de um algoritmo adaptativo. A responsabilidade de uma correta documentação fica sobre o programador. O programador também fica responsável pelo correto teste do programa, pois o compilador (ou interpretador) não poderá auxiliar na detecção de erros devido ao seu desconhecimento particular sobre algoritmos adaptativos.

Para expressar de maneira coerente um algoritmo adaptativo faz-se necessário a especificação da linguagem de programação e do compilador estarem cientes sobre a existência de algoritmos adaptativos a fim de auxiliar o programador no processo de desenvolvimento e padronização do código.

Através da proposta de uma linguagem de programação com características adaptativas em [1] e [2] tornou-se possível a implementação de algoritmos adaptativos sem que haja necessidade de alterar o código de máquina durante tempo de execução, servindo também como base para a proposição de compilador ao invés de um interpretador.

## IV. Definição da linguagem

A maioria dos programadores atuais estão acostumados ao uso de linguagens de programação que seguem o paradigma imperativo. Para facilitar a adoção da linguagem pelos programadores, a linguagem possuirá todos os comandos e declarações presentes em linguagens imperativas e a adaptatividade será expressa através de uma extensão que poderá ser utilizada opcionalmente pelo programador.

Porderia-se propor uma extensão a uma linguagem não necessariamente imperativa como por exemplo as linguagens que seguem paradigmas declarativos como o funcional e o lógico. Esses outros tipos de linguagem necessitam de técnicas de compilação mais complexas que uma linguagem imperativa pois as linguagens imperativas refletem a arquitetura dos computadores atuais, que são baseados na arquitetura de Von Neumann. Portanto adotou-se o uso do paradigma de programação imperativo por simplicidade.

Ao invés de definir uma linguagem de programação do zero, optou-se por utilizar a linguagem de programação Oberon [5], por possuir uma sintaxe simples, representar praticamente todas as construções presentes em linguagens imperativas e ao seu sistema de módulos.

## V. Extensão da linguagem

A adaptatividade será expressada na linguagem de programação através de blocos de código e conexões entre eles ao invés de se trabalhar com a alteração de código de máquina durante a execução. A alteração do código do programa durante a execução será feita através da modificação da ligação entre cada bloco de código, onde cada uma dessas ligações indicam qual é o próximo bloco de código a ser executado. Para o prosseguimento de um trecho do programa pode ser necessário com que as ligações entre os blocos sejam alteradas e, para determinado bloco, ao invés de simplesmente transferir o controle, é executada uma ação de rearranjo das ligações entre todos blocos.

A extensão adaptativa foi inserida na linguagem através de um tipo especial de procedimento que é chamado como se fosse um procedimento comum da linguagem Oberon (no momento da chamada é impossível diferenciar um procedimento adaptativo de um procedimento comum).

Dentro do procedimento adaptativo é possível declarar de um procedimento convencional, um procedimento adaptativo, declarar fragmentos de código e declarar ações. Em cada procedimento adaptativo é obrigatório declarar o estado inicial das conexões entre os fragmentos.

```
DeclarationSequence = ["CONST" {ConstDeclaration ";"}]
```

```
                    ["TYPE" {TypeDeclaration ";"}]
                                ["VAR"
{VariableDeclaration ";"}]
                                        {ProcedureDeclaration
";"}

{AdaptProcedureDeclaration ";"}.

AdaptProcedureDeclaration = AdaptProcedureHeading ";"
                            AdaptProcedureBody
                                                    "END"
ident.

AdaptProcedureHeading = "ADAPTIVE" identdef
                        [FormalParameters].

AdaptProcedureBody = DeclarationSequence
                                [AdaptFragment {";"
AdaptFragment}]
                                    ["ACTIONS"
                    [AdaptAction {";"AdaptAction}]]
                        "CONNECTIONS"
                    AdaptStmts.
```

O algoritmo adaptativo é executado através dos fragmentos e cuja lógica de controle é definida nos blocos de ação.

*A. Declaração de fragmentos*

Os fragmentos constituem os blocos de código imperativo atômicos. Dentro de cada fragmento podem conter qualquer comando da linguagem Oberon (comandos não adaptativos), referência a variáveis do procedimento, variáveis globais, procedimento locais, procedimentos globais e acesso aos parâmetros do procedimento.

Cada fragmento é referenciado a partir do seu nome, são visíveis apenas no escopo do procedimento adaptativo no qual foi declarado e cada nome só pode ser associado a um único fragmento dentro desse escopo.

```
AdaptFragment = "FRAGMENT" ident ";"
                                StatementSequence
                        "END" ident ";".
```

*B. Declaração de ações*

São blocos de definições das conexões entre os fragmentos dentro de um procedimento adaptativo.

```
AdaptAction = "ACTION" ident ";"
                AdaptStmts
                    "END" ident ";".
```

*C. Comandos adaptativos*

Os dois comandos adaptativos tem como objetivo selecionar qual ação será executada para um determinado fragmento de origem. Um deles permite que uma entre diversas ações de controle seja selecionada para a execução a partir de uma condição e o outro comando executa uma ação de controle apenas informando qual é o fragmento de origem.

```
AdaptStmts = [AdaptStmt {";" AdaptStmt}].
AdaptStmt = ConditionalAdaptStmt | InconditionalAdaptStmt.
```

*1) Seleção de comando adaptativo*

Seleciona uma ação de controle através de uma condição para determinado fragmento de origem. A ação de controle que será executada será a primeira em que sua condição estiver verdadeira e deverá ser informada a ação de controle que deverá ser executada caso nenhuma das condições seja verdadeira.

```
ContionalAdaptStmt = "AFTER" ident
                                    AdaptControl
"CASE" expression ","
                            {AdaptControl    "CASE"
expression ","}
                            AdaptControl
"OTHERWISE".
```

*2) Execução de comando adaptativo incondicional*

Executa uma ação de controle para determinado fragmento de origem.

```
InconditionalAdaptStmt = "AFTER" ident AdaptCommand.
```

*3) Comandos de ação de controle*

São comandos que irão indicar qual é o próximo fragmento de código que será executado, qual deve ser a ação usada para continuar a execução do altorimo adaptativo ou terminar a execução do algoritmo adaptativo e opcionalmente retornar um valor.

```
AdaptControl =          GotoControl
               |    PerformControl
               |    ReturnControl.
```

*a) Comando de salto para próximo fragmento*

Indica o próximo fragmento a ser executado. O próximo fragmento é identificado a partir de seu nome.

```
GotoControl = "GOTO" ident.
```

*b) Comando de redefinição de conexões*

É um comando que transfere a lógica de conexão entre os fragmentos para um outro bloco de ações. O controle da execução do algoritmo adaptativo passa para o bloco de ações identificado até que ocorra uma outra chamada de redefinição de conexões (o controle só volta para essa ação caso ocorra um outro comando de redefinição de conexões tendo como alvo a ação atual).

```
PerformControl = "PERFORM" ident.
```

*c) Comando de retorno de procedimento*

É o comando que encerra a execução do procedimento

adaptativo, pode ou não devolver um resultado. O resultado devolvido por esse comando deve ter o mesmo tipo de retorno definido na declaração do procedimento adaptativo.

ReturnControl = "RETURN" [expression].

## VI. FUNÇÃO FATORIAL

```
MODULE module;
    VAR fat10:INTEGER;

        (* procedimento adaptativo para calcular fatorial *)
        ADAPTIVE fat(n:INTEGER):INTEGER;
            VAR result:INTEGER;

    (* a inicialização do algoritmo
       começa pelo primeiro
       fragmento *)
            FRAGMENT init;
                result := 1
            END init;

            (* passo de cálculo *)
            FRAGMENT calc;
                result := result * n;
                n := n - 1
            END calc;
        ACTIONS
            (* conexões executadas caso n > 0 *)
            ACTION not0;
                AFTER calc RETURN result CASE n = 0,
                    GOTO calc OTHERWISE
            END not0;
        CONNECTIONS
            (* conexões iniciais *)
        AFTER init RETURN result CASE n = 0,
            PERFORM not0 OTHERWISE
        END fat;
BEGIN
    fat10 := fat(10)
END module.
```

## VII. COMPILAÇÃO

A compilação da linguagem de programação é feita com o auxílio da LLVM [3] para gerar o código de máquina. O compilador, ao invés de gerar o código de máquina diretamente, gera um texto que contém uma representação intermediária do programa e a LLVM fica responsável pela transformação da representação intermediária em código de máquina.

A representação intermediária da LLVM utiliza o formato de atribuição única e é muito parecida com linguagens de montagem (porém utiliza um sistema de verificação de tipos). A LLVM recebe como entrada um texto em notação intermediária, realiza as otimizações e gera um código chamado de *bytecode* que pode ser transformado em código de máquina para alguma arquitetura ou executado diretamente através de um interpretador.

Para cada comando da linguagem Oberon foi mapeado um conjunto de instruções na representação intermediária da LLVM. Nesse mapeamento foi utilizado o conceito de blocos básicos de instruções que também é utilizado pela LLVM.

O protótipo de um compilador ficou dividido em analisador léxico, analisador sintático, gerador de código intermediário, gerador de código LLVM e emissão de código LLVM.

### *A. Analisador léxico e sintático*

O analisador léxico e sintático é feito utilizando-se as técnicas tradicionais de compiladores. Em ambos os casos foram utilizadas ferramentas para a geração do analisador léxico e analisador sintático. O analisador sintático possui como saída a representação do programa em forma de árvore.

### *B. Gerador de código intermediário*

O gerador de código intermediário recebe como entrada a árvore do programa, a valida e gera o código intermediário. O código intermediário é representado através de um grafo direto acíclico (DAG - *Direct Acyclic Graph*). Cada comando do código fonte do programa é substituído por um comando mais simples como por exemplo os comandos *IF* e WHILE que são substituídos por um simples comando de salto condicional. Essa representação mais simples dos comandos visa facilitar a sua tradução para código de máquina nos passos posteriores.

### *C. Gerador de código LLVM*

O gerador de código LLVM recebe como entrada o grafo do programa e o transforma em uma estrutura de dados representando o programa em linguagem de montagem na LLVM. Cada instrução simples do programa é mapeada em uma ou mais instruções em linguagem de montagem constituindo um bloco básico no código LLVM.

### *D. Emissão de código LLVM*

A emissão de código LLVM é feita percorrendo a estrutura de dados contendo o código LLVM. É representado o código de montagem LLVM em estrutura de dados antes para facilitar a sua manipulação pelo gerador de código intermediário.

## VIII. COMPILAÇÃO DA EXTENSÃO ADAPTATIVA

A compilação da extensão adaptativa foi projetada para utilizar intensamente a otimização de chamadas de cauda realizada em chamada de funções (consule [4] para maiores

detalhes sobre chamadas de cauda).

A otimização de chamada de cauda é feita quando uma função é chamada em posição de cauda. Uma chamada de função é dita em posição de cauda quando a função que realiza a chamada não realiza nenhum processamento após a chamada, permitindo com que a nova chamada seja feita descartando o contexto da função atual. Ao não ter que armazenar o contexto da função permite que um número arbitrário de chamadas de função em posição de cauda sejam realizadas sem que ocorra estouro de pilha.

Tanto fragmentos quanto ações foram mapeadas em funções em código de montagem LLVM.

Os fragmentos recebem como parâmetro a função de ação que será executada depois do fragmento o que permite com que a função de ação seja chamada em posição de cauda.

As ações recebem como parâmetro a função de fragmento que acabou de executar a fim de identificar o fragmento de origem para selecionar a ação de controle que deverá ser executada. As duas únicas chamadas de função executadas dentro de um bloco de ação também permitem a otimização de chamada de cauda: a redefinição de conexões que é implementada através da chamada de uma segunda função ação e a transferência de controle para fragmento através da chamada de uma função que representa um fragmento. Em ambas as chamadas de função o bloco de ação terminou o seu processamento.

## IX. Conclusão

A extensão proposta para a implementação de algoritmos adaptativo possui como vantagem manipular de maneira organizada o próprio código fonte. A adaptatividade fica claramente exposta no código fonte de um programa através da existência de um procedimento dedicado para a implementação de algoritmos adaptativos ao contrário das técnicas de auto-modificação de código de máquina e uso da função *eval*.

A implementação em baixo nível do procedimento adaptativo é feito unicamente através de funções não necessitando armazenar as informações sobre as conexões em variáveis globais. Dessa maneira o algoritmo adaptativo pode ser executado no mesmo instante por múltiplas linhas de execução sem que haja a necessidade de sincronização.

Os procedimentos adaptativos não possuem memória, ou seja, o procedimento adaptativo não armazena as conexões de uma chamada anterior. Se esse for um efeito desejado, pode-se implementar esse efeito através da definição do uso de variáveis ou estruturas de dados para serem passados como parâmetro para o procedimento adaptativo, ficando ele responsável pelo armazenamento do estado atual no final de sua execução e leitura e a execução de ação de redefinição de conexões no início de sua execução.

## Referencias


[1] S. B. da Silva e J. J. Neto “Projeto de uma linguagem para programação adaptativa“, 2010.

[2] S. B. da Silva e J. J. Neto “Um método para programação adaptativa” XVI Congreso Argentino de Ciencias de la Computacion, 2010.

[3] Low Level Virtual Machine, llvm.org acessado em 18/10/10.

[4] Daniel P. Friedman e Mitchel Wand, “Essentials of Programming Languages”, MIT Press, 2008.

[5] Niklaus Wirth, “The Programming Language Oberon”, revisão 01/11/2008.

## ANEXO: REPRESENTAÇÃO INTERMEDIÁRIA PARA A FUNÇÃO FATORIAL

```
;********************
; apelidos para tipos            *
;********************
%l1 = type i32 (%l2, i32*, i32*)* ; função tipo conexão
%l2 = type i32 (%l1, i32*, i32*)* ; função tipo fragmento

;***********************
; VAR fat10:INTEGER;        *
;***********************
@fat10 = internal global i32 0

;******************
; FRAGMENT init; *
;******************
define i32 @init(%l2 %l4, i32* %l0, i32* %result)
{
l8:                                ;
  store i32 1, i32* %result  ;  result := 1
  br label %l6                     ;
l6:
  %l7 = tail call i32 (%l1, i32*, i32*)* %l4 (%l1 @init, i32* %l0,
i32* %result)
  ret i32 %l7
}

;******************
; FARGMENT calc; *
;******************
define i32 @calc(%l2 %l4, i32* %l0, i32* %result)
{
l17:
  %l9 = load i32* %result
  %l10 = load i32* %l0                    ;
  %l11 = mul i32 %l9, %l10          ;  result := result * n;
  store i32 %l11, i32* %result      ;
  br label %l16
l16:
  %l12 = load i32* %l0              ;
  %l13 = sub i32 %l12, 1            ;  n := n - 1
  store i32 %l13, i32* %l0          ;
  br label %l14
l14:
  %l15 = tail call i32 (%l1, i32*, i32*)* %l4 (%l1 @calc, i32* %l0,
i32* %result)
  ret i32 %l15
}

;********************
; FRAGMENT not0;            *
;********************
define i32 @not0(%l1 %l3, i32* %l0, i32* %result)
{
l26:
          ;
  %l27 = icmp eq %l1 %l3, @calc          ; AFTER calc
  br i1 %l27, label %l24, label %l25     ;
l24:
  %l18 = load i32* %l0                   ;
  %l19 = icmp eq i32 %l18, 0             ; CASE n = 0
  br i1 %l19, label %l21, label %l22
l21:
                                         ;
  %l20 = load i32* %result                    ; RETURN
result
  ret i32 %l20                                ;
l22:
  ;
  ; GOTO calc OTHERWISE
  ;
  %l23 = tail call %l1 @calc (%l2 @not0, i32* %l0, i32* %result)
  ret i32 %l23
l25:
   ; Aqui poderia ser informado um erro por falta de ação adaptativa
ret i32 0
}

;*****************************
; ações de conexão iniciais *
;*****************************
define i32 @l5(%l1 %l3, i32* %l0, i32* %result)
{
l36:                    ;
  %l37 = icmp eq %l1 %l3, @init      ; AFTER init
  br i1 %l37, label %l34, label %l35  ;
l34:
  %l28 = load i32* %l0                          ;
  %l29 = icmp eq i32 %l28, 0                    ; CASE n = 0
  br i1 %l29, label %l31, label %l32    ;
l31:
  %l30 = load i32* %result          ; RETURN result
  ret i32 %l30                                  ;
l32:
   ; PERFORM not0 OTHERWISE
   %l33 = tail call %l2 @not0 (%l1 %l3, i32* %l0, i32* %result)
  ret i32 %l33
l35:
  ; Aqui poderia ser informado um erro por falta de ação adaptativa
  ret i32 0
}

;************************************
; ADAPTIVE fat(n:INTEGER):INTEGER; *
;************************************
define i32 @fat(i32 %n)
{
l38:
  %l0 = alloca i32
  store i32 %n, i32* %l0
  %result = alloca i32
  %l39 = tail call %l1 @init (%l2 @l5, i32* %l0, i32* %result)
  ret i32 %l39
}

;***********************
; Inicialização do módulo *
;***********************
define void @module()
{
l42:
  %l40 = call i32 @fat (i32 10)          ;
  store i32 %l40, i32* @fat10            ; fat10 := fat(10)
  br label %l41                                  ;
l41:
  ret void
}
```

# Um Algoritmo Adaptativivo de Unificação de Redes Haplotípicas

R. H. G. Guiraldelli, R.L A. Rocha

***Resumo*— Para representação da realidade evolutiva das espécies, a rede haplotípica é uma das modelagens gráficas de referência em filogenética e filogeografia, condensando grande quantidade de informação mutacional simplesmente em vértices e arestas. Essa, no entanto, não costuma ser única para um mesmo conjunto de dados geradores de entrada e o resultado apresenta-se dependente do modelo de construção escolhido. Este artigo propõe uma representação unificada das diversas redes produzidas — fornecendo novos dados antes inexistentes — utilizando-se de tecnologia adaptativa para sua construção.**

*Palavras-chave*— **Rede haplotípica, grafos, adaptatividade, NCPA.**

## I. Introdução

O estudo da genética de populações, nomeado filogenética, assim como a filogeografia, que relaciona o estudo filogeográfico com as influências ambientais, processa grande quantidade de dados para a extração de informações relevantes sobre a evolução de determinada espécie. Para esse objetivo, no entanto, faz-se necessário o uso de ferramentas de suporte eficiente para análise desses dados; a experiência na pesquisa biológica, portanto, guiou a solução para a representação gráfica através de grafos.

Essa representação, chamada na biologia de rede haplotípica, é de amplo uso e parte fundamental do principal algoritmo utilizado na filogeografia, o *nested clade phylogeographic analysis*; este algoritmo, porém, admite apenas uma única rede haplotípica de entrada para análise quando, na verdade, várias hipóteses de redes existem para o mesmo conjunto de dados.

Para uma melhor representação da realidade evolutiva das espécies, independentemente do modelo proposto para formação da rede haplotípica, propõe-se o uso de adaptatividade para unificação das diversas instâncias de redes existentes para o mesmo conjunto de dados em uma única rede, adicionando, ainda, informações de probabilidade para melhores escolhas de caminhos evolutivos.

Assim, este artigo se organiza da seguinte forma: primeiramente, este seção introdutória; a seção 2 descreve a rede haplotípica e as técnicas de sintetização existentes; a próxima seção explica, brevemente, o conceito de adaptatividade; então, a seção explicativa sobre a técnica proposta para unificação das redes; por fim, na seção 5, as conclusões, propondo, ainda, trabalhos futuros.

## II. Redes Haplotípicas

No estudo da genética de populações, o sequenciamento do genoma de diversos indivíduos continua uma ferramenta básica para a realização de análises, mas não mais a única ferramenta. Para tanto, existe a necessidade de uma ferramenta de análise que relacione os indivíduos amostrados através de características comuns e que sintetize, em uma simples representação, grande densidade de informação.

Buscando satisfazer esses requisitos, através de notação por grafos, a rede haplotípica relaciona os haplótipos da população (de mesma espécie) amostrada através das diferenças mutacionais representadas pelas arestas, sendo os próprios haplótipos os vértices. É importante notar que um rede haplotípica faz sentido, apenas, com dados coletados de populações da mesma espécie, avaliando diferenças sutis no material genético sequenciado entre os indivíduos da espécie em estudo.

Contudo, há mais de um método para a construção das redes haplotípicas, com a particularidade de que cada um desses produz um modelo de grafo diferente do outro. Tais modelos, descritos brevemente em [1], são: (a) minimum spanning tree; (b) statistical parsimony e; (c) full median. A literatura destaca os métodos (a) e (b) com maior enfoque, especialmente o último por ser a metodologia de escolha do aplicativo TCS [2].

A **Erro! Fonte de referência não encontrada.** abaixo epresenta uma rede haplotípica extraída de [3] sintetizada utilizando-se o aplicativo supracitado.

Ricardo Henrique Gracini Guiraldelli é aluno de mestrado, tendo como o orientador o Prof. Dr. Ricardo Luis de Azevedo da Rocha, do Departamento de Sistemas Digitais da Escola Politécnica da Universidade de São Paulo. E-mails: rguira@usp.br; rlarocha@usp.br.

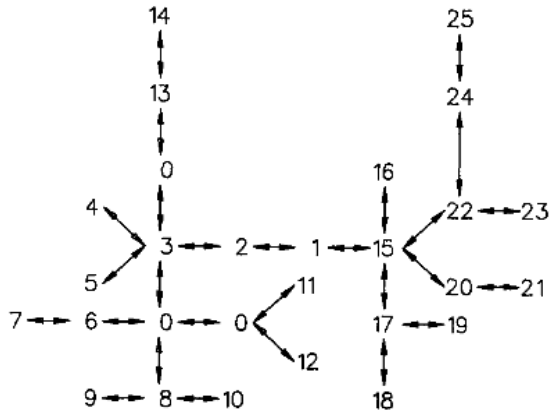



Fig. 1. Rede haplotípica de uma população de *Drosophila melanogaster* [3].

## III. Adaptatividade

Os dispositivos computacionais tradicionais, como os autômatos ou a máquina de Turing, são formalismos de grande utilidade para a representação e reprodução de algoritmos. Em particular, os autômatos são dispositivos muito simples e de fácil entendimento até mesmo ao leigo, porém têm poder de expressividade limitada — limitam-se à representatividade de linguagens livres-de-contexto [4]; a máquina de Turing, por outro lado, apresenta maior complexidade para a representação de algoritmos, porém com a capacidade de aceitar, até mesmo, as linguagens recursivamente enumeráveis.

Contudo, embora possuam grande capacidade de representação de algoritmos e decisão sobre linguagens, os dispositivos de computação tradicionais são inflexíveis quanto a mudanças em tempo de execução[10]. Esta limitação, embora aparentemente desprezível, mantém forte relação com aprendizado [5] e com o poder computacional do dispositivo [6].

Buscando preencher essa lacuna, o conceito de adaptatividade emergiu, sendo matematicamente formalizado para aplicação em dispositivos computacionais [7]. Neste, um dispositivo adaptativo $AD = (ND_0, AM)$ com:

- $ND_0$ um dispositivo computacional não-adaptativo;
- $AM$ um mecanismo adaptativo tal que $AM \subseteq BA \times NR \times AA$ e $NR$ seja o conjunto de regras de $ND_0$;
- $BA$ e $AA$ são conjuntos de funções adaptativas, de forma que ambos contenham a ação vazia ( $e \in BA \cup AA$).

Descritivamente, a adaptatividade é uma caminhada sobre um possivel espaço de (todos os) dispositivos de uma deteminada classe (e.g., autômatos) onde as funções adaptativas escolhem uma instância particular deste espaço para a execução do passo computacional não-adaptativo. É importante ressaltar que as funções adaptativas podem ser classificadas em *anterior* ( $F_B \in BA$) e *posterior* ( $F_A \in AA$), representando a ordem de execução destas em relação à transição não-adaptativa do dispostivo $ND_k$.

## IV. Proposta de Algoritmo Adaptativo

Atualmente o uso de redes haplotípicas, especialmente aqueles gerados por sistemas computacionais como o aplicativo TCS, é de grande utilidade para os estudos de filogenética e fiologeografia, automatizando o processo não-imediato de construção da rede utilizando técnicas como a "parsimônia estatística" [2].

No entanto, o uso dessas ferramentas produz como resultado uma única rede haplotípica traduzindo a representação de apenas um modelo (como visto na seção II) e, assim, a inferência dos processos de evolução e mutação naturais segundo um único modelo de análise dos dados. No entanto, o uso de informações enganosas na análise filogeográfica guia a pesquisa a conclusões falsas; esse efeito ocorre, ainda, quando essas informações provêm da rede haplotípica [1].

Buscando auxiliar os estudos biológicos de filogeografia, propõe-se a unificação dos modelos geradores da rede haplotípica com o objetivo de, independentemente da modelagem escolhida, retratar os eventos naturais da melhor maneira observada e mais coerentemente possível. No entanto, para a utilização do método nested clade phylogeographic analysis, um único grafo se faz necessário e, assim, tal unificação vai além daquela de conceitos e inclui a dos próprios grafos.

Para a ocorrência deste, o uso de tecnologia adaptativa se mostra extremamente eficaz e adere-se naturalmente à representação gráfica da rede haplotíca, uma vez que esta mantém morfismo com autômatos [8], dispostivos computacionais imediatos para aplicação de adaptatividade.

A estratégia se traduz na busca, em todos os elementos do conjunto de redes haplotípicas para determinada população, por vértices do tipo zero, i.e. vértices (ou haplótipos) não mapeados na população mas necessários para manter a conectividade do grafo, ou pelo vértice representativo do centro de massa do grafo; esses vértices representam aspectos centrais do grafo, contendo alta densidade de informação por serem elementos-chave para a formação dos diversos ramos do grafo.

Após encontrado os nós com as propriedades acima enunciadas, inicia-se a busca por semelhanças entre os diversos grafos baseados no nó em análise, formando um novo grafo (nomeado grafo unificador) compartilhando as propriedades entre os diversos grafos e adicionando probabilidades às arestas. Formalmente, para um vértice $v_i$, o conjunto dos vértices vizinhos de $v_i$ no grafo unificador será $\bigcup_k \{ v_{j_k} | v_{j_k}$ seja vizinha de $v_i$ no grafo $k \}$; o valor de contagem da quantidade de arestas unindo o vértice $v_i$ com um vértice $v_j$ qualquer é determinado pela função

[10] A máquina de Turing universal, no entanto, é uma exceção: por possuir em sua fita de entrada a definição da máquina de Turing que irá interpretar, pode, portanto, alterar as regras daquela.

$\text{Count}(v_i, v_j) = \sum_{k=0}^{N-1} \text{Id}(k)$, onde $\text{Id}(k)$ é a função característica do conjunto gerado pela união em relação a aresta e $N$ representa a quantidade de redes haplotípicas; se o grafo $k$ contém $v_j$.

Por construção, o grafo unificador $G_\cup = (V_\cup, A_\cup)$ inicialmente definem-se os conjuntos $V_\cup = A_\cup = \text{Æ}$. Durante a busca por arestas semelhantes nas $N$ redes haplotípicas, para cada vértice $v_i$ em estudo, caso encontrada um elemento $(v_i, v_j) \in A$ da relação, então uma função adaptativa prévia $F_B$ é disparada adicionando o vértice $v_j$ em $V_\cup$ e criando uma aresta $(v_i, v_j)$ no conjunto $A_\cup$ (relação simétrica).

Após a inserção do nó $v_j$, uma função adaptativa posterior $F_A$ remove a aresta $(v_i, v_j)$ e adiciona a aresta rotulada por $\ell$, representada da forma $((v_i, v_j), \ell)$, onde $\ell$ é o rótulo da transição atribuído pela função de contagem $\text{Count}(v_i, v_j)$; $G_k \mid 0 \leq k < N$ no qual exista a relação $(v_i, v_j)$, sendo assim representado por um número de maneira que $\ell \in \{0,1,2,\ldots\}$.

O grafo unificador, por fim, conterá todos os nós (ou haplótipos) dos $N$ grafos concorrentes gerados pelos modelos iniciais indicando, através dos rótulos, as conexões com maior probabilidade de ocorrência após a unificação dos modelos. Espera-se, dessa maneira, diagnosticar a rede haplotípica que mais se assemelhe com a realidade encontrada na natureza e que melhor contribua com as inferências do método nested clade phylogeographic analysis ou outros métodos de estudo filogenéticos e filogeográficos.

Exemplificando a aplicação da técnica acima descrita, utiliza-se como entrada os três grafos representados pelas **Erro! Fonte de referência não encontrada.**, **Erro! Fonte de ferência não encontrada.** e **Erro! Fonte de referência não encontrada.**, interpretados como três redes haplotípicas diferentes (de até quatro haplótipos) para um conjunto de dados filogenéticos hipotético. É imediata a observação da diferença entre essas redes, expressando relações (de mutação) não-equivalentes entre os haplótipos nas diferentes figuras.

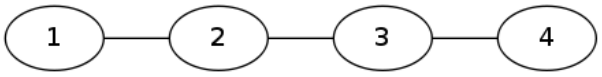



Fig. 2. Rede haplotípica de quatro haplótipos (ou grafo $G_1$), com mutações ocorrendo sequencialmente.

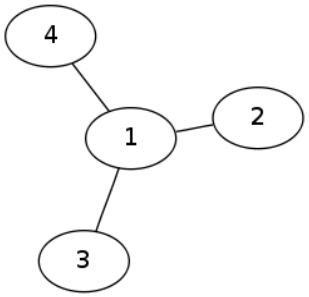



Fig. 3. Rede haplotípica de quatro haplótipos (ou grafo $G_2$), com todas as mutações diretamente derivadas do haplótipo **1**.

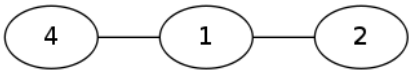



Fig. 4. Rede haplotípica de três haplótipos (ou grafo $G_3$), com mutações ocorrendo sequencialmente.

Das figuras, elege-se o *haplótipo 1* como o centro de massa e inicia-se, a partir deste, a formação do grafo unificador $G_\cup$, adicionando-se ao conjunto $V_\cup$ o *vértice 1*. Então, percorre-se os grafos $G_1$ a $G_3$ buscando todos os nós que mantém a relação simétrica $(1, k)$. E.g., de $G_1$ verifica-se a relação $(1,2)$, fazendo $V_\cup = \{1,2\}$ e $A_\cup = \{1,2\}$. A função adaptativa posterior $F_A$ é então executada, removendo $(1,2)$ de $A_\cup$ e adicionando, $((1,2),1)$, fazendo-no $A_\cup = \{((1,2),1)\}$; este último nada mais é que a aresta $(1,2)$ nomeada pelo rótulo **1**, simbolizando o número de vezes que a relação foi encontra nos conjuntos *A* dos grafos analizados até então.

Esse procedimento algorítmico é repetido até o momento em que $V_\cup = \bigcup_{k=1}^{3} V_k$, $A_\cup = \bigcup_{k=1}^{3} A_k$ e todos os as arestas estejam rotuladas com o valor da frequência que aparecem. Nesta configuração, portanto, temos o grafo $G_\cup$ conforme indicado na **Erro! Fonte de referência não encontrada.**.

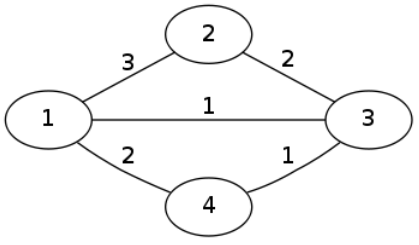



Fig. 5. Grafo unificador $G_\cup$.

Como é possível notar, $G_\cup$ mantém as relações das três

redes haplotípicas de entrada com a informação adicional de frequência em que as relações (mutacionais) ocorrem. Desta nova representação, é possível uma análise mais abrangente da evolução de populações, mesmo para processos como o nested clade phylogeographic analysis.

A figura a seguir (**Erro! Fonte de referência não ncontrada.**) destaca, em $G_{\cup}$, as relações com maior frequência, definindo um subgrafo com as mutações mais prováveis segundo os dados de entrada, podendo servir como entrada, por exemplo, para um processamento onde apenas uma rede haplotípica é admitida.

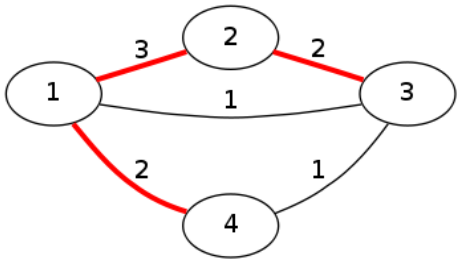



Fig. 6. Grafo unificador $G_{\cup}$ com as conexões mais prováveis em destaque (proximidade entre os vértices e a coloração das arestas).

## V. Conclusão

Embora a notação gráfica da rede haplotípica seja uma ferramenta que agregue grande diferencial à pesquisa filogenética e filogeográfica, a não uniformidade nas representações para um mesmo conjunto de dados (devido aos diferentes modelos de construção das redes) torna-na objeto entrópico nas análises das populações [1]; a seleção de um único modelo, também, leva a perda de informações como a frequência com o qual algumas mutações ocorrem nas diversas modelagens dos dados.

A geração de um grafo unificador, por sua vez, unifica todos os benefícios fornecidos por cada um dos métodos de sintetização de redes haplotípicas além de adicionar novas informações, possibilitando ao utilizador extrair novas relações desta nova representação, como o exemplo do subgrafo da **Erro! Fonte de referência não encontrada.** e, inda, se ater ao antigo modelo de sua preferência.

Por fim, a adaptatividade trás benefícios na construção do grafo unificador, possibilitando não apenas a construção do mesmo em tempo de execução, mas também como a sua modificação para a inserção dos rótulos numéricos nas arestas representando a função de contagem das relações entre vértices.

### A. Trabalhos Futuros

Tem-se em vista a complementação teórica do método proposto neste artigo, projetando-se uma modelagem de inferência indutiva para a extração de um conjunto de subgrafos ótimos segundo a ótica bio-evolutiva; neste, ainda, é possível realizar a extração do grafo unificador diretamente dos dados de entrada, independentemente dos métodos de síntese de rede haplotípica existentes.

Sugere-se, ainda, modificações no aplicativo TCS [2] para que gere como saída uma rede haplotípica baseado no grafo unificador para as múltiplas possíveis redes concebidas pela aplicação, ao invés de única rede através do método *statistical parsimony*.

Finalmente, também há a possibilidade de alterar o aplicativo GeoDis [9] para que suporte um grafo unificador como entrada de rede haplotípica e, então, realizar múltiplos processamentos paralelos do nested clade phylogeographic analysis.

## Referências

[1] S. Joly, M. Stevens, and B. J. van Vuuren, "Haplotype networks can be misleading in the presence of missing data," *Systematic Biology*, vol. 56, no. 5, pp. 857–862, 2007.

[2] M. Clement, D. Posada, and K. A. Crandall, "Tcs: a computer program to estimate gene genealogies," *Molecular Ecology*, vol. 9, no. 10, pp. 1657–1660, 2000.

[3] A. R. Templeton, E. Boerwinkle, and C. F. Sing, "A cladistic analysis of phenotypic associations with haplotypes inferred from restriction endonuclease mapping. i. basic theory and an analysis of alcohol dehydrogenase activity in drosophila," *Genetics*, vol. 117, pp. 343–351, October 1987.

[4] C. Papadimitriou and H. Lewis, *Elements of Theory of Computation*, Prentice-Hall, Ed. Prentice-Hall, 1998.

[5] H. Pistori and J. J. Neto, "Decision tree induction using adaptive fsa," *CLEI Eletronic Journal*, vol. 6, 2003.

[6] R. L. A. Rocha and J. J. Neto, "Autômato adaptativo, limites e complexidade em comparação com máquina de Turing," in *Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000*. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia, 2000, pp. 33–48.

[7] J. J. Neto, *Lecture Notes on Computer Science*. Springer-Verlag, 2001, ch. Adaptive Rule-Driven Devices - General Formulation and Case Study, pp. 234–250.

[8] B. C. Pierce, *Basic Category Theory for Computer Scientists*. MIT Press, 1991.

[9] D. Posada, K. A. Crandall, and A. R. Templeton, "Geodis: A program for the cladistic nested analysis of the geographical distribution of genetic haplotypes," *Molecular Ecology*, vol. 9, no. 4, pp. 487–488, 2000.

**Ricardo Henrique Gracini Guiraldelli** nasceu em Sorocaba, Brasil, em 10 de março de 1986 e gradou-se em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo. Desde 2009 é mestrando na EPUSP em Engenharia de Computação e membro da ACM (Association for Computing Machinery), atuando na área de Bioinformática e Fundamentos de Computação.

**Ricardo Luis A. Rocha** é natural do Rio de Janeiro-RJ e nasceu em 29/05/1960. Graduou-se em Engenharia Elétrica modalidade Eletrônica na

PUC-RJ, em 1982. É Mestre e Doutor em Engenharia de Computação pela EPUSP (1995 e 2000, respectivamente). Suas áreas de atuação incluem Tecnologias Adaptativas, Fundamentos de Computação e Modelos Computacionais.

Dr. Rocha é membro da ACM (Association for Computing Machinery) e da SBC (Sociedade Brasileira de Computação).

# Um Modelo Adaptativo Aplicado no Aprimoramento da Representação de Linhas Retas em Geometria Digital

Leoncio C. Barros Neto, André H. Hirakawa e Antonio M. A. Massola

***Resumo*—Apesar de representações digitais fundamentadas em linhas retas e arcos serem ferramentas de muitos estudos em ciência da computação, o seu uso não é tão popular em cenários que requerem estruturas flexíveis capazes de responder às imprecisões dos modelos ou às condições variáveis do ambiente. O objetivo deste trabalho é investigar uma proposta baseada na adaptatividade por meio do dispositivo denominado segmento digitalizado adaptativo (SLRDA), que incorpore o poder expressivo de representar segmentos em linha reta digitalizada (SLRD). Assim, partindo-se de primitivas, são concebidas as funções adaptativas correspondentes relacionadas a arcos digitais pelos quais *strings* estimulam, em apenas uma passada, a reação de determinado SLRDA conforme, por exemplo, as tolerâncias dos parâmetros envolvidos. Consequentemente, o método proposto torna-se uma alternativa versátil, relativamente simples e intuitiva às abordagens existentes, apresentando capacidade de aprendizagem, além de ser computacionalmente poderoso.**

***Palavras Chaves*— Computação reconfigurável, Geometria e modelagem computacional, Reconhecimento de padrões, Teoria dos autômatos, Erro (Recuperação)**

## I. INTRODUÇÃO

PARTINDO de estudos na área de construção de compiladores por [1], surgiram aplicações da adaptatividade nas mais diversas áreas que resultaram em todo um acervo de recursos teóricos, conceituais e de ferramental em aplicações variadas que foram sendo acumuladas, descritas nos levantamentos de [2] e [3]. No entanto, a despeito da relevância da representação computacional de linhas retas e arcos, inclusive sendo uma área ativa de pesquisas há quase meio século, citado por [4], esse assunto ainda não foi estudado sob o enfoque adaptativo com o potencial do mencionado acervo, até o momento.

Assim como o conceito de linha reta é importante na geometria Euclidiana, é dos objetos considerados fundamentais em computação, com vários pesquisadores tendo se interessando por esse assunto, pois, apesar da aparente simplicidade das linhas digitais, estas incorporam todas as dessemelhanças e disparidades entre o discreto e o contínuo.

No processo de digitalização, é inevitável que determinado segmento de reta contínuo no espaço Euclidiano seja afetado por transformações, deformações ou corrupção por ruído gerando um uma *string* com imperfeições (não seguindo o modelo ideal, afetada por erros).

"Diz-se que um dispositivo é adaptativo sempre que seu comportamento se altera dinamicamente, em resposta a estímulos de entrada, sem interferência de outros agentes externos, inclusive de operadores ou usuários, executando as correspondentes automodificações" [5]. Os dispositivos adaptativos apresentam naturalmente a flexibilidade requerida para atuarem em cenários dinâmicos, por serem capazes de responder às variações nas condições ou situação momentânea do ambiente.

A hipótese inicial deste trabalho é quanto à viabilidade em modelar propriedades das retas digitais em imagens digitalizadas por um conjunto de regras a fim de aplicar a adaptatividade. Adicionalmente, caso as interferências espúrias possam também ser modeladas, mesmo que indiretamente por intermédio de tolerâncias permitidas, passa a ser viável que um dispositivo adaptativo represente as diferentes instâncias possíveis das estruturas.

Face ao exposto, esta pesquisa tem como principal objetivo investigar a representação de segmentos de linhas retas digitalizadas (SLRD[11]) por meio da adaptatividade, cotejando com outras formas de representação ao mostrar aspectos positivos e negativos comparativamente.

O conceito de "linhas retas digitais adaptativas[12]" desta proposta utiliza o Autômato Finito Adaptativo (AFA), abrangendo incorporar a capacidade de representar parâmetros tais como as tolerâncias, a escalabilidade, os desvios em ângulo ou os desvios em comprimento dos mencionados segmentos em linha reta por intermédio de segmentos digitalizados adaptativos (SLRDA). Pela linearidade adaptativa, determinada *string*, relacionada a um arco digital qualquer, pode corresponder a um SLRD, mesmo que aproximado, após avaliação de parâmetros do arco em diversas escalas. Por outro lado, arcos que não corresponderem a retas, são representáveis pela concatenação de SLRD [7].

### *A. Fundamentos sobre Segmentos Digitalizados*

Este tópico apresenta conceitos básicos sobre SLRD.

Entendendo-se pontos digitais como pontos cujas

[11] As siglas deste trabalho indicam tanto o singular como o plural do que se referem, com cada significado a ser interpretado em adequação com o contexto.

[12] [6] observa que a diferença entre linha reta digitalizada e linha reta digital é sutil. Entende-se que a primeira é o resultado da digitalização de uma reta Euclidiana específica. O foco principal desta pesquisa é quanto a retas digitalizadas, mas se aplica também ao caso de retas digitais.

coordenadas são inteiras resultantes de um esquema de digitalização; um arco digital qualquer S é um conjunto de pontos digitais interligados, componentes de uma imagem digital. A característica do conjunto de pontos pertencentes ao arco S é o seu posicionamento relativamente a uma grade, tal que cada ponto desse conjunto tem exatamente apenas dois vizinhos; exceto dois pontos, denominados extremos, que possuem apenas um vizinho em S [9].

Em [8] o chain code foi apresentado como um descritor de contornos, de apenas um pixel de espessura, e um modelo definidor de retas digitais foi também conjecturado por Freeman. Nesse modelo, dado um ponto digital, as vizinhanças principais e imediatas desse ponto estão mostradas na Fig. 1, que mostra o relacionamento dos símbolos do chain code com a vizinhança-4 e vizinhança-8.

Após seleção da vizinhança-4 ou vizinhança-8, o chain code é definido da seguinte maneira:

Definição 1: O chain code, ou o código da cadeia, é uma sequência de elementos com finalidade de representar um arco digital, onde cada elemento é um símbolo na vizinhança selecionada, que representa o vetor unindo dois pontos digitais vizinhos do arco em questão.

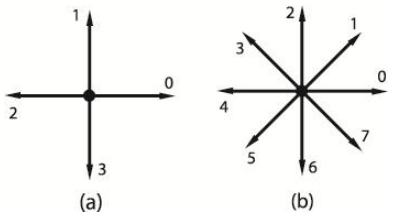



Fig. 1 À esquerda estão os símbolos de 0 a 3 do chain code para vizinhança-4. À direita, os símbolos de 0 a 7 do chain code para vizinhança-8.

Um arco digital é reto se for o resultado da digitalização de um segmento em linha reta. No modelo de Freeman, as *strings* que representam linhas retas obedecem a três propriedades, em uma codificação utilizando números de 0 até 7, em vizinhança-8:

--Prop1: No máximo dois tipos de símbolos, representando direções distintas no código do chain, podem estar presentes, e estes são números consecutivos correspondentes do chain, módulo oito;

--Prop2: Um dos dois símbolos sempre ocorre isoladamente, solitário;

--Prop3: As ocorrências sucessivas do símbolo isolado são tão uniformemente espaçadas quanto possível entre códigos do outro valor, que ocorre em grupos ou corridas (runs).

O significado de Prop1 a Prop3 é representar a linha por uma sequência de vetores com inclinações múltiplas de 45º e cujos comprimentos são unitários (se horizontal ou vertical) ou $\sqrt{2}$ (se diagonal).

Uma das questões fundamentais desta pesquisa é o poder computacional requerido para a análise sintática de SLRD.

### *B. Análise Sintática de Segmentos Digitalizados*

Um dos procedimentos do modelo sintático inicia pela definição de uma gramática associada a algum tipo de dispositivo de reconhecimento [10]. Esse reconhecedor é denominado parser por ser responsável pela análise sintática, decidindo (parsing) se uma dada *string* observada pertence ou não à classe representada pela gramática.

Porém, ruído e distorções complicam o processo computacional da análise sintática: além das distorções, primitivas espúrias são geradas, bem como primitivas reais podem não ser detectadas. Além disso, a própria natureza variável dos períodos dos símbolos dos SLRD, associado a comprimentos variáveis da escala das retas é um desafio para efetuar a análise sintática de SLRD. Dependências de contexto e alterações no ângulo de orientação em segmentos de comprimento arbitrário afetam a estrutura dos códigos das retas digitais forçando o parser a rever a sua análise [11]. O entendimento do problema, sob o ponto de vista sintático, envolve os conceitos de linguagem, gramática e tipos de gramáticas.

Segundo a hierarquia de Noam Chomsky datada de 1956, descrita em [12], as linguagens são classificadas em quatro classes diferentes: Linguagens Enumeráveis Recursivamente (ou Tipo 0), Linguagens Sensíveis ao Contexto (ou Tipo 1), Linguagens Livres de Contexto (ou Tipo 2) e Linguagens Regulares (ou Tipo 3). Associa-se um grau de complexidade entre as classes mencionadas, em que a classe Tipo 3 é um subconjunto da classe Tipo 2, a classe Tipo 2 é um subconjunto da classe Tipo 1, e a classe Tipo 1 é um subconjunto da classe Tipo 0.

Entre as abordagens de pesquisa existentes, os métodos sintáticos são normalmente considerados pela literatura como não sendo convenientes para tarefas envolvendo SLRD. Isso porque a análise sintática de SLRD requer gramáticas poderosas, sensíveis ao contexto, inviabilizando a aplicação de formalismos simples, tais como AF [13] [4]. Relembre-se que uma Linguagem Regular é especificada por uma Gramática Regular. Os conceitos de Linguagem Regular e Autômato Finito (AF) são equivalentes no sentido de que para cada Linguagem Regular existe, pelo menos, um AF que a reconhece e vice-versa (ver [12] para o formalismo de gramáticas, linguagens e autômatos).

A maneira que o AFA aceita linguagens tipo 1 e 0 é apresentado por [14] pelo recurso do AFA alterar seu próprio conjunto de estados e de regras de transição [14].

Em princípio, a linguagem tipo 0 não faria parte do escopo desta pesquisa. Entretanto, visando estar de acordo com os comprimentos dos SLRD, os quais podem estar em várias escalas, teoricamente até infinito, implica em linguagens tipo 0 neste trabalho.

### *C. Codificação*

Nesta pesquisa, os SLRD são codificados por meio de *strings* de caracteres, e não de números, bastando considerar $a = 1$ e $b = 0$ no segmento exemplificado pela Fig. 2 a fim de atender Prop1. Uma *string* é uma seqüência de zero ou mais símbolos pertencentes ao alfabeto Σ. Esses símbolos são também denominados como primitivas, tokens, elementos do chain code ou simplesmente estímulos. O conjunto de todas as *strings* pertencentes a Σ é denotado por $\Sigma^*$. O comprimento de uma *string* qualquer *S* é denotado por $|S|$. A *string* vazia, de

comprimento zero, é representada por ε. O *i*-ésimo símbolo de uma *string* $S = s_1 \dots s_n$ é representado por $s_i$.

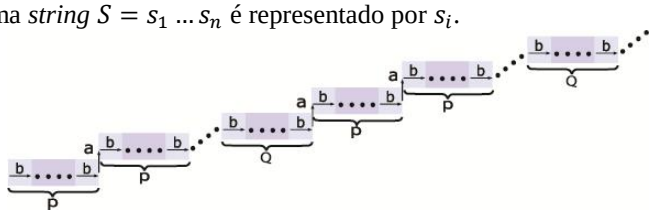



Fig. 2 Modelo de SLRD genérico no primeiro quadrante, com corridas de P e Q símbolos *b*, tão uniformemente espaçados quanto possível entre tokens *a*, que ocorrem isolados.

Caso nada em contrário seja especificado, sem qualquer redução em generalidade, neste trabalho é utilizada a vizinhança-4 como padrão, tal que os símbolos da propriedade Prop1 devem ser consecutivos, módulo quatro. Mais precisamente, os símbolos que compõem as *strings* pertencem a $\Sigma = \{a, b, c, d\}$ tal que, para atender Prop1, basta considerar módulo 4, juntamente com *a=1*, *b=0*, *c=3*, *d=2*, relativamente à vizinhança-4 da Fig. 1.

Comparando-se a vizinhança-4 com outras, por exemplo, vizinhança-8, a primeira apresenta vantagens como simplicidade nas implementações, além dos algoritmos de processamento digital considerarem normalmente apenas dois eixos *x* e *y*, porém tem a desvantagem de maior perda de informação. A escolha envolve também o aspecto da notação mais conveniente, pois autores utilizam uma e outra, por exemplo [15] e [16]. Uma *string* qualquer $\boldsymbol{S = s_1 \dots s_n}$ pode também ser representada no formato da Expressão 1.

$$S: s_i; i = 1,2, \dots, n. \quad (1)$$

Caso todos os elementos de S sejam idênticos, $\boldsymbol{S = s_1 = s_2 = \dots = s_{n-1} = s_n} = s$, uma representação compacta é $S = s^n$.

As estruturas adaptativas deste texto são do primeiro quadrante. Caso seja necessário incorporar segmentos de outros quadrantes a um determinado SLRDA representativo de apenas um quadrante, basta manter a topologia inicial, alterando os símbolos convenientemente, obtendo o SLRDA homólogo do primeiro para o outro quadrante.

### *D. Organização do Texto*

A seção II apresenta fundamentos sobre AFA. A seção III apresenta os trabalhos principais que embasam conceitualmente esta pesquisa. Na seção IV, avaliam-se os SLRD partindo-se de suas propriedades de ângulo de orientação e comprimento de segmentos. A seção V apresenta as estruturas que estão sendo propostas por esta pesquisa. A seção VI apresenta uma síntese dos conceitos principais envolvidos na proposta, bem como posiciona este trabalho com relação ao estado-da-arte ao mostrar suas vantagens mais relevantes. A seção VII apresenta as considerações finais deste relato, contribuições e sugestões de trabalhos futuros.

## II. AUTÔMATO FINITO ADAPTATIVO (AFA)

A Definição 2 facilita as descrições dos autômatos desta pesquisa.

Definição 2: Uma sequência conexa é um conjunto de estados em sequência concatenada e interligados por transições que consomem o mesmo símbolo em toda a sequência. Apenas dois estados, denominados extremos, se conectam a um único estado do conjunto; enquanto os demais estados se conectam a dois estados. Na terminologia sintática esse conjunto é denominado como do tipo "head-to-tail concatenation", de acordo com a página 3 de [17].

A Definição 3 apresenta o AFA.

Definição 3: O Autômato Finito Adaptativo (AFA) é um dispositivo auto-modificável da forma AFA = (ND0, AM) onde ND0 é o mecanismo subjacente, e AM é a camada adaptativa associada, formalizada nos mesmos moldes que ND0. O AFA é caracterizado por ter o autômato finito (AF) como mecanismo subjacente.

A camada adaptativa da Definição 3 integra o conjunto de operações de auto modificação possíveis, denominadas ações adaptativas, entendidas como atividades que são executadas em resposta aos estímulos, alterando a configuração do dispostivo.

Definição 4: Ações adaptativas são chamadas paramétricas de funções adaptativas (FAD), que podem ser considerada como coleções de ações adaptativas elementares a serem aplicadas para definir as transições do autômato.

Um AFA é representado pela 7-tupla da Expressão 2.

$$\text{AFA} = (Q; AR_0; \Sigma; q_o; F; \mathscr{B}; \mathscr{A}) \quad (2)$$

tal que:

Q é um conjunto não vazio e possivelmente infinito, de estados do autômato;

$AR_0$ é um conjunto de regras adaptativas:

$AR_0 \subset \mathscr{B}$ X (Q X (Σ ∪ {ε})) X $\mathscr{A}$, sendo que

(Q X (Σ ∪ {ε})) corresponde ao conjunto de regras δ do dispositivo subjacente;

Σ é o alfabeto de entrada do autômato. É um conjunto não vazio contendo número finito de símbolos;

$q_o$ é o estado inicial do autômato;

F é o conjunto de estados finais;

$\mathscr{B}$ e $\mathscr{A}$ são conjuntos de ações adaptativas anteriores e posteriores, respectivamente. O conjunto $\mathscr{B}$ é denominado anterior por ser executado antes da transição entre estados; enquanto o conjunto posterior $\mathscr{A}$ é executado após a transição.

Por sua vez, as ações adaptativas componentes dos conjuntos $\mathscr{B}$ e $\mathscr{A}$ são formadas por conjuntos de ações adaptativas elementares ou primitivas a serem aplicadas para definir o processo adaptativo do autômato. Tais ações adaptativas elementares podem ser de três modalidades, de acordo com o prefixo representado na Tabela I, em que o padrão especifica uma transição em relação à qual as transições em uso devem ser testadas:

1) Ações de busca ou consulta, denotadas pelo prefixo "?": efetuam um busca por regras que casem com o padrão específicado, sem alterar o conjunto de regras;
2) Ações de remoção, denotadas pelo prefixo "-": removem do conjunto as regras que casam com o padrão especificado;
3) Ações de inserção, denotadas pelo prefixo "+": inserem o padrão específicado no conjunto de regras.

A Tabela I mostra as três modalidades de ações adaptativas

elementares ou primitivas com o padrão especificado entre colchetes. A maneira de especificar esse padrão é apresentada no tópico seguinte que mostra o formato das ações adaptativas elementares.

TABELA I
SIMBOLOGIA PARA AS AÇÕES ADAPTATIVAS PRIMITIVAS

| PREFIXO | SIGNIFICADO | SIMBOLOGIA |
|---|---|---|
| ? | consulta | ?[padrão] |
| - | remoção | -[padrão] |
| + | inserção | +[padrão] |

O padrão corresponde à regra a ser especificada

*A. Formato das Ações Adaptativas Elementares*

Na 7-tupla da Expressão 2, o conjunto de ações adaptativas $AR_0$ leva conta o formalismo do dispositivo subjacente por meio de δ.

Complementando a Definição 4, conjuntos de ações adaptativas elementares são abstraídas em FAD, as quais interconectam o dispositivo subjacente à contraparte adaptativa, em conformidade com a Fig. 3 pelas FAD R e S. Essa figura mostra uma representação gráfica estática de uma transição genérica do AFA do tipo: *(x, i) : R→y : S* onde *x* é o estado atual do autômato antes da transição entre estados; *y* é o estado de destino do autômato após a transição; *i* é o estímulo de entrada; R e S são as FAD executadas respectivamente "antes" e "depois" da transição do estado *x* para o estado *y*.

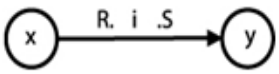



Fig. 3 Uma transição adaptativa genérica, onde R e S são FAD opcionais.

Consequentemente, o mecanismo adaptativo, composto de regras contidas em AM, tal que AM ⊆ $\mathscr{B}$ X $\delta_k$ X $\mathscr{A}$. Essas regras são aplicadas em cada passo operacional k em que $\delta_k$ é a função de transição do dispositivo subjacente no passo k.

Observe-se que, havendo mais de uma ação adaptativa elementar a ser executada, independentemente da ordem em que foram declaradas, têm precedência as consultas. Em seguida são efetuadas as remoções, e posteriormente as inserções. Possíveis transições em vazio são sempre executadas por último.

Ações elementares que referenciam elementos indefinidos nãosão executadas.

O formato das três modalidades de ações adaptativas elementares da Tabela I é indicado na Tabela II, com o padrão da Tabela I especificado entre colchetes.

TABELA II
FORMATO DAS AÇÕES ADAPTATIVAS PRIMITIVAS

| MODALIDADE | SIMBOLOGIA |
|---|---|
| consulta | ?[ *x, i) : R→y :S*] |
| remoção | -[*(x, i) : R→y :S*] |
| inserção | +[*(x, i) : R→y S*] |

Detalhes sobre o formato das FAD são descritos a seguir.

*B. Formato das Funções Adaptativas (FAD)*

Quanto ao formato das FAD, pela Definição 4, as ações adaptativas são indicadas pelo par ordenado (F;P), correspondendo a chamadas paramétricas de FAD, abrangendo os seguintes dados:

*1) A identificacão:* F é o nome da FAD;

*2) Possíveis argumentos:* P é uma sequência de argumentos ($p_1$, $p_2$, .... ) da FAD F, todos opcionais.

Esses argumentos ($p_1$, $p_2$, .... ) tem a finalidade de designar valores a serem substituidos aos correspondentes parâmetros formais de F.. Tais parâmetros comporão as ações adaptativas primitivas de inserção, consulta ou eliminação do autômato. Apesar de opcionais, se parâmetros forem especificados, terão de ser fornecidos para ativar a correspondente FAD F.

Em outras palavras, os parâmetros formais de FAD são variáveis especiais, utilizados para que valores reais (argumentos) sejam atribuídos, sempre que determinada FAD seja ativada. Assim, aos parâmetros da FAD são atribuidos os valores recebidos como argumentos antes de ativar a FAD, permanecendo inalterados durante toda a sua execução.

No caso geral, além dos parâmetros, uma FAD pode ser especificada por variáveis e geradores, também opcionais, com os significados seguintes:

Variáveis: identificadores que recebem um valor de acordo com o resultado de ações adaptativas elementares de consulta ou exclusão. As variáveis são utilizadas para representar entidades existentes em transições do AFA tais como estados ou estímulos.

Geradores: identificadores que recebem um valor único (isto é, nunca antes utilizado) no início da ação adaptativa e permanecem com este valor até o final da ação. São uma espécie de variável que contém a identificação do próximo estado a ser criado.

Os geradores são indicados com um expoente * apenas na declaração de nomes, diferenciando das variáveis. A expecificação de uma FAD F qualquer, com n parâmetros q consiste no seguinte:

*3) Um cabeçalho:* F($q_1$;$q_2$; :::;$q_n$)

*4) Um corpo, com o seguinte formato*

{declaração de nomes (opcional):
declaração de ações (opcional)}.

Sobre as variáveis, tanto as ações adaptativas elementares de consulta quanto as de exclusão efetuam uma busca no conjunto corrente de transições para encontrar qualquer transição que corresponda ao padrão dado. Quando tal transição é encontrada, às variáveis são atribuídos os valores reais correspondentes à transição encontrada, pois as mesmas são usadas no lugar de qualquer um dos componentes de ação adaptativa elementar.

Considerando $v_i; i = 1,2, \ldots, m$ como variáveis e $g_j; j = 1,2, \ldots, n$ como geradores, a declaração de nomes apresenta o aspecto seguinte: $v_1$, $v_2$, ....$v_m$, $g_1^*$, $g_2^*$,...$g_n^*$.

A título de esclarecimento, veja abaixo o formato de uma FAD hipotética η, e a correspondente especificação de um gerador ger1, uma variável var1, dois parâmetros $\alpha$ e $\beta$ e três ações adaptativas elementares, para um alfabeto $\boldsymbol{\Sigma} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}\}$.

O par ordenado (F,P) é indicado por (η, $\alpha$, $\beta$) que pode ser exemplificado por uma chamada de transição adaptativa qualquer tal como [(1, a) : η (2, 6) →2]. Nesse caso, a ação adaptativa que ativa a FAD η ocorre antes que o AFA mude do estado 1 para o estado 2, desde que um token *a* seja recebido.

```
η ( α, β) {ger1*,var1:
?[(r_{i-1},b)→ β]
+[(ger1,a)→ α]
-[(var1,ε) : A r_{i+1}]}
```

Concluindo, da mesma forma como a FAD η é ativada nesse exemplo, pela maneira que as FAD são conectadas às transições do AFA define-se o comportamento do autômato resultando em aceitar ou rejeitar a *string* de entrada.

Gráficamente, uma determinada FAD R é representada por R**.** (R seguido de ponto**)**, caso seja do tipo antes. Similarmente, uma FAD S é representada por **.**S (ponto seguido de S**)**, caso seja do tipo depois.

## III. O ESTADO-DA-ARTE

Este tópico apresenta uma breve revisão do estado-da-arte sobre SLRD.

Tendo em vista que a Prop3 não é muito clara, esta foi posteriormente melhor formalizada pela propriedade da corda, descrita a seguir.

### *A. Linha Reta*

A noção de linha reta está associada à propriedade da corda.

Para a vizinhança-8, [9] demonstrou que a condição necessária e suficiente para que um arco digital qualquer seja um SLRD ideal é atender à propriedade da corda.

Definição 5: A Propriedade da Corda: Diz-se que um arco digital C, representando "objetos sólidos" delgados em uma imagem digitalizada, apresenta a propriedade da corda se, para cada dois pontos digitais c e d pertencentes a C, e para cada ponto p = (x, y) em $\overline{\text{cd}}$, existe um ponto e = (h, k) pertencente a C tal que $\max\{|x-h|, |y-h|\} < 1$, onde $\overline{\text{cd}}$ é o segmento de reta entre c e d [9].

A Definição 5 implicou na demonstração da existência de uma estrutura hierárquica composta de números consecutivos correspondentes às corridas dos símbolos especificados por Prop1 e Prop2. Essa estrutura de números consecutivos é expressa por uma propriedade adicional Prop4: Quanto à direção referente ao símbolo que ocorre em grupos (não isolado), as corridas correspondentes podem ocorrer com apenas dois valores, os quais diferem de uma unidade (por exemplo, P e P+1).

Pela propriedade da corda, define-se um SLRD ideal como a seguir.

Definição 6: Linha Reta Digitalizada Ideal (SLRD ideal): Uma linha reta digitalizada ideal é aquela que atende à propriedade da corda, na vizinhança respectiva. Note que a propriedade da corda foi originariamente definida para a vizinhança-8, existindo especializações para a vizinhança-4 [18].

Também o conceito de pré-imagem é importante para o entendimento da perda de informação no processo de representar linhas e segmentos.

Definição 7: Pré-imagem: Dado um segmento digital em linha reta (SLRD), pré-imagem é o conjunto de linhas retas Euclidianas, cuja codificação para o formato de *string* resulta no mesmo SLRD dado. Demonstra-se que a pré-imagem corresponde a um polígono convexo em um espaço de parâmetros [6].

Além do comentado pela Definição 7, o processo de digitalização de uma linha reta Euclidiana específica introduz distorções e ruído, causando que o correspondente SLRD não atenda à propriedade da corda, resultando em muitos segmentos curtos na vizinhança de um pixel, de acordo com a mencionada propriedade. Portanto, é necessário algum tipo de medida, denominada métrica, apropriada para avaliar se dois SLRD pertencem a uma única estrutura linear [19]. Essa métrica pode definir uma função de vizinhança escolhida de modo a expressar como agrupar SLRD aproximados.

Contudo, "aproximado" é entendido no tocante a linhas retas visualmente corretas dentro de uma tolerância, não necessariamente no tocante à propriedade da corda. Portanto, de agora em diante, a menos que de outra maneira seja especificado, a nomenclatura deste texto não faz nenhuma distinção entre um SLRD ideal e as linhas "quase" retas nas proximidades da primeira, considerando que esse conjunto (de SLRD ideais e SLRD aproximados) é reconhecido pelo dispositivo adaptativo proposto.

Por outro lado, trabalhos como [20] mostraram exemplos de SLRD que violam a regularidade implícita na propriedade da corda, comentando que, na prática, Prop4 e Prop3 e são inviáveis na análise de arcos digitais. É mais razoável esperar uma ligeira variação nas corridas, dentro de um nível de tolerância, mas sempre mantendo a inclinação geral, delineando assim SLRD aproximados. O critério usado por [20] se concentrou em *strings* que satisfaçam as duas primeiras propriedades da conjectura, denominados códigos monotônicos, pois representam arcos digitais que são ascendente ou descendente com referência aos eixos de coordenadas *x* e *y*.

Adicionalmente, três trabalhos principais norteiam esta proposta. No primeiro, [21] segue um procedimento algorítmico similar ao de Freeman o qual define linhas discretas como digitalização de linhas Euclidianas. O segundo, [22] utiliza a noção de "segmentos borrados" fundamentada na "geometria discreta aritmética" introduzida por [23]. O terceiro, [24] fundamenta-se na noção de "linha discreta aritmética" fundamentada na "geometria discreta aritmética" introduzida por [23]. O trabalho [24] se propôs a ser uma solução para a rigidez no modelo de Freeman, mesmo após certa flexibilida ter sido introduzida por [21]; bem como para a perda de conexão com a aritmética de [22].

Enfatize-se que o relacionamento da geometria discreta aritmética com a geometria Euclidiana ocorre no limite tendendo a infinito, da mesma maneira que uma grade discreta sendo observada de um ponto suficientemente distante aparenta ser contínua [25].

Esta pesquisa leva em conta que a representação adaptativa possibilita alteração das escalas de segmentos, de tal maneira que arcos irregulares numa determinada escala possam ser

reconhecidos como SLRD alterando-se as escalas relativas. Portanto, arcos irregulares podem se revelar, na verdade, como retas ao serem analisados numa escala compatível, utilizando métricas. A adaptatividade surge, portanto, como uma alternativa para incorporar os princípios da geometria discreta aritmética ao modelo de Freeman.

## IV. ANÁLISE ESTRUTURAL

Outro método de representação, aplicado neste tópico, é a representação das linhas digitais com base em frações contínuas, estudado por [10], que resultou num algoritmo válido apenas para retas com inclinações de números racionais (a inclinação é o ângulo do segmento com referência ao eixo *x*, expressado, por exemplo, pela tangente desse ângulo). O estudo da modelagem de retas por frações contínuas tem sido continuamente pesquisado, com vários desdobramentos após o trabalho de [10], descritos em [4].

Similarmente à aproximação de pontos por polinômios de determinada ordem, o enfoque matemático da representação por frações contínuas propicia modelos, facilitando avaliação de erros e aproximações no processo de digitalização de retas partindo das correspondentes inclinações.

Sobre as correspondências entre inclinações e periodicidade, o chain code de retas com inclinações representadas por números racionais são periódicas, enquanto para inclinações representadas por irracionais não ocorre periodicidade, comentado na página 312 de [26].

Entretanto, mesmo sem considerar distorções e ruído, os modelos digitais têm que atender às condições da grade de amostragem corresponder a números inteiros num reticulado, o que resulta em modelos aproximados, independentemente de inclinações racionais ou irracionais.

A seguir, apresenta-se uma breve análise estrutural dos SLRD por meio dos formatos das respectivas *strings*, apresentadas como exemplos. Sem perda de generalidade, considerase o chain code representando uma linha contínua de ângulo de orientação φ com o eixo positivo *x* tal que $\varphi \in \left[0, \frac{\pi}{4}\right]$ (indica que φ pertence ao intervalo fechado entre 0 até $\frac{\pi}{4}$ radianos) [27]. Nesse intervalo de φ, as *strings* correspondentes aos SLRD apresentam o símbolo *a* ocorrendo isolado enquanto o símbolo *b* ocorre agrupado.

### A. Ordem dos Modelos

Conforme [28], dependendo do grau de precisão requerido, derivam-se fórmulas para o ângulo de orientação dos SLRD bem como modelos matemáticos gerais ao se aproximar um SLRD a um segmento Euclidiano. Os modelos são paramétricos, calculados diretamente em função do ângulo de orientação φ do segmento Euclidiano. Para tais modelos, o conceito de unidade de segmento digital em linha reta (USLR) é importante:

Definição 8: Unidade de Segmento Digital em Linha Reta (abrevido USLR) é o menor segmento possível em que um SLRD pode ser subdividido a fim de manter o correspondente ângulo de orientação.

As premissas de [28] para especificação dos modelos foram: *i)* O SLRD deve atender à propriedade da corda; *ii)* O SLRD não pode ser subdividido idefinidamente, de acordo com a Definição 8; *iii)* A estrutura do arranjo de pontos digitais do SLRD depende exclusivamente da correspondente inclinação.

A inclinação, definida pela tangente do segmento Euclidiano no primeiro octante, pode ser expressa pela fração contínua da Expressão 3 aproximada a um modelo nos moldes da Definição 9, a seguir.

$$SL = \frac{B}{A} = \frac{1}{P \pm \frac{1}{M \pm \frac{1}{K \pm \cdots}}} \quad (3)$$

em que os sinais positivos e negativos são para acelerar a convergência da fração e reduzir o número de termos da mesma, A, B e P são inteiros (B < A), enquanto M, K .....são inteiros não negativos.

Definição 9: Ordem de Modelo de Reta Digital (abreviado modelo de reta): A ordem de um modelo de reta digital corresponde ao número de termos da Expressão 3.

Assim, a Expressão 3 pode se apresentar de várias formas; tais como $SL = \frac{B}{A} = \frac{1}{P}$, $SL = \frac{B}{A} = \frac{1}{P \pm \frac{1}{M}}$, $SL = \frac{B}{A} = \frac{1}{P \pm \frac{1}{M \pm \frac{1}{K}}}$ , denominadas frações contínuas de primeira ordem, de segunda ordem e terceira ordem, respectivamente. A essas três, seguem-se ordens superiores, dependendo dos valores de A, B, ou dos requisitos de precisão.

Os modelos de primeira e segunda ordem são considerados principais para descrever o arranjo no ajuste dos padrões de pontos de SLRD na grade. Note que os ângulos das USLR relativamente ao eico *x*, denominados $\theta_U$ podem variar no SLRD. A Definição 10, abaixo, comenta sobre o ângulo de orientação principal do SLRD, que é o ângulo que se destaca dentre a distribuição de $\theta_U$ ao longo do segmento.

Definição 10: Ângulo de Orientação Principal: Entende-se como ângulo de orientação principal do SLRD, denominado $\theta_S$, como aquele ângulo que se destaca, por algum critério, dentre a distribuição de ângulos das USLR individuais do segmento. Por exemplo, [20] denomina a direção (o ângulo de inclinação) correspondente ao símbolo que ocorre isoladamente de direção de transição, e a direção correspondente ao outro símbolo de direção principal.

### B. Modelo de Primeira Ordem

O primeiro modelo é denominado de primeira ordem porque a inclinação é uma fração contínua de primeira ordem. Nesse modelo, o SLRD da Fig. 4 atravessa B = 1 linhas e P colunas, resultando para a inclinação SL o valor de $SL = \frac{B}{A} = \frac{1}{P}$, onde P é um número inteiro denominado fator de inclinação de primeira ordem. P como número inteiro significa que os pontos digitais do segmento de reta são uniformemente distribuídos em linhas e colunas da grade, tal que cada linha é um USLR que contém P pontos consecutivos, com o arranjo dos padrões do SLRD do tipo PPP...PP....P.

Para o modelo de primeira ordem, as *strings* $S_U$ de cada USLR seguem o formato da Expressão 4 em que m é o mesmopara toda USLR do SLRD.

$$S_U: b^m a; m \geq 1 \quad (4)$$

Com $SL = \frac{B}{A} = \frac{1}{P}$, é evidente que, no modelo de primeira ordem, o ângulo de orientação principal do SLRD $\theta_S$ é idêntico ao ângulo de cada USLR $\theta_U$, ou seja:
$\theta_S = \theta_U = \arctan(1/P)$.

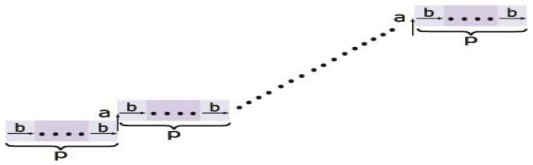



Fig. 4. Modelo de primeira ordem, caracterizado por P constante

### C. Modelo de Segunda Ordem

No modelo de segunda ordem, P é o número inteiro mais próximo de A / B, ou seja, $P = \lfloor A / B \rfloor$. Nesse modelo, o SLRD tende a ajustar os pontos digitais em B linhas, tão uniformemente quanto possível, a fim de manter a inclinação global. No modelo de segunda ordem da Fig. 2, o arranjo dos padrões do SLRD é do tipo PPP.....QPP.....PQ em que o ajuste na grade é efetuado por intermédio de Q, o qual pode ter apenas duas possibilidades: Q = (P+1) ou Q = (P-1). Ou seja, o ajuste é efetuado diminuindo-se P de uma unidade, ou aumentando-se P de uma unidade. As *strings* $S_U$ das USLR serão de dois tipos, em que o primeiro tipo é $b^m a$ e o segundo vai depender do padrão de números consecutivos (m+1) ou (m-1): $S_U: \begin{cases} b^m a. \\ b^{m+1} a; m \geq 1 \; ou \; b^{m-1} a; m \geq 2. \end{cases}$

Nesse modelo, o ângulo de orientação principal, aquele que se destaca, é $\theta_S$ = arctan(1/P) com inclinação 1/P, entretanto ocorrem USLR com inclinação 1/Q tão espaçadas quanto possível. Desta forma, o ângulo de orientação φ com o eixo positivo *x* da linha contínua que deu origem à codificação estará na faixa $\varphi \in [\arctan\left(\frac{1}{P+1}\right), \arctan\left(\frac{1}{P}\right)]$ (supondo Q = P+1) [27]. Esse resultado de $\varphi$ confirma o comentado na seção III sobre o trabalho de [9] quanto ao padrão de números inteiros consecutivos das corridas.

## V. ESTRUTURAS ADAPTATIVAS PROPOSTAS

[29] comenta que "semelhança" é um termo fuzzy, enfatizando que, quando os erros inerentes a determinados cenários não seguem um comportamento conhecido, é mais viável utilizar modelos apresentando a melhor "semelhança" com o modelo ideal, exemplificando com métodos fuzzy aplicados à teoria clássica das linguagens e autômatos. Posteriormente, descreve um autômato fuzzy capaz de avaliar a similaridade entre duas *strings*, a *string* observada de entrada e a *string* correspondente ao modelo ideal.

Analogamente, com o propósito de apresentar os fundamentos da solução proposta por este trabalho, seja um AFA capaz de avaliar a similaridade (dentro de uma certa tolerância) entre duas *strings*, o SLRD observado e a *string* reta, ou seja a *string* definida pelo modelo de SLRD conforme a propriedade da corda. A configuração inicial do AFA é um AF $ND_0$ que aceita a *string* reta. A contraparte adaptativa AM é formada de "ações adaptativas", que levam em conta todas as possíveis instâncias da *string* observada devido aos possíveis erros. Tais ações adaptativas têm a finalidade de alterar a configuração inicial para uma nova configuração (representada por um AF diferente de $ND_0$), consequentemente levando o AFA a representar as diferentes instâncias do modelo ideal, considerando os erros envolvidos.

Essa questão de erros envolvidos em análise sintática foi estudada principalmente na recuperaçao de erros em compiladores, que se enquadram no enfoque desta pesquisa, comentado a seguir.

### A. Preliminares

## 1. Recuperação de Erros

Vários autores têm relatado os métodos de recuperação de erros, em particular [30] na área de compiladores. Supondo uma *string* submetida a um processo de análise sintática por um reconhecedor, denomina-se recuperação de erros ao método de tratamento de erros que consiste em termos gerais, nas seguintes etapas: *i)* O analisador se mantém no processo de consumo de símbolos (ou caracteres) de entrada, até que um símbolo errado seja identificado; *ii)* Isolado o erro, ativa-se algum mecanismo de recuperação de erros; *iii)* O mecanismo de recuperação normalmente modifica ou adapta o reconhecedor para que este se mantenha consumindo um ou mais símbolos, de modo que a análise sintática especificada seja retomada em estágio posterior da sequência de entrada. Restrições existem em alguns métodos, exigindo que o usuário forneça informações adicionais sobre o processo de reparo.

É evidente a analogia entre a recuperação de erros na análise sintáticas de SLRD com a recuperação de erros em outros processos sintáticos gerais. Entretanto, é possível concluir que não ocorre uma igualdade total ou completa ao se retomar o trabalho de [29], referenciado no início desta seção.

[29] abstraiu todos os erros no processo de digitalização, independentemente da sua origem, nos denominados erros de edição: substituição, eliminação e inserção de símbolos. Todavia, o mesmo procedimento não é viável para o caso de SLRD. A primeira razão é que existe previamente um grau de incerteza representado pelo domínio da pré-imagem. A segunda, determinados erros de edição podem ser aceitos numa faixa, enquanto outros tipos de erros, se aceitos irrestritamente, iriam contrariar a estrutura das linhas retas, expressadas pelas propriedades Prop1 a Prop4.

O propósito desta pesquisa em recuperação de erros compreendeu estudos introdutórios sobre como modelar, por meio do AFA, linguagens do tipo $L = \{a^{m \pm \alpha} \; b^{n \pm \beta} \; c^{m \pm \delta} \; d^{n \pm \gamma} : m \geq 1; \; n \geq 1\}$; em que α, β, γ e δ são pequenos erros correspondentes às influências espúrias. Esses estudos preliminares foram cumpridos a fim de incorporar a incerteza nos modelos das linguagens.

### B. Técnica Utilizada em Recuperação de Erros

A fim de exibir características comuns ao que está sendo aludido, relembre-se que, frequentemente, é conveniente representar os números reais em uma determinada circuferência C, e não em uma linha reta, como habitual. Em especial, a partir da circunferência C de comprimento unitário, ao definir-se um ponto origem arbitrário, representa-se um ponto quaquer P pela sua distância medida em volta da circunferência, no sentido anti-horário (esse sentido é por

definição). Desse modo, todos os números inteiros serão representados pelo seu deslocamento com relação ao ponto origem, enquanto os números que difiram por um número inteiro terão o mesmo ponto representativo do círculo. A divisão do círculo pode ser a partir da série de Farey, na forma de *spyrographs* descritos na página 326 de [26].

As técnicas de recuperação de erros de SLRD desta abordagem empregam um dipositivo similar aos *spyrographs* na forma de *loops* adaptativos da Fig. 5, tal que, nesses *loops*, a circuferência mencionada é representada por $to$ estados (de $L_1$ a $L_{to}$) do AFA, os quais são percorridos cíclicamente pelo autômato.

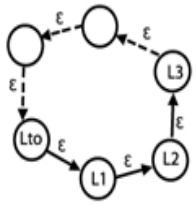



Fig. 5 Um grafo na forma de *loop* adaptativo genérico, interpretado como caso particular de sequência conexa da Definição 2 cujos estados inicial e final são idênticos**.**

Por questão didática, as técnicas delineadas a seguir estão segregadas por dois tipos de erros característicos dos SLRD: erros em ângulo e erros em comprimento.

### *C. Erros em Ângulo*

Supondo inicialmente um SLRD S como uma *string* de comprimento n primitivas, $|S| = n$, tal *string* é composta da concatenação de λ *substrings*, onde $n > \lambda$ e $1 \leq i \leq \lambda$ sendo que cada *substring* é uma USLR U de S da seguinte forma

$$S: U_i \; ; i = 1,2, \dots, \lambda. \quad (5)$$

Entretanto, na Expressão 5 a SLRD nula (λ = 0) não é definida, nem os casos em que o primeiro e o último USLR ($U_1$ e $U_\lambda$ ) estão truncadas, ou seja, fora da estrutura do ângulo $\theta_S$ de orientação do SLRD global. Essa situação é considerada no item C.1desta seção.

Assim, cada USLR é também um SLRD elementar que apresenta um ângulo e um comprimento "locais". Portanto, para cada USLR, definida pelo código isolado de Prop2, basta o SLRDA correspondente "entrar" em um loop adaptativo da Fig. 5, cuja quantidade de estados é variável adaptativamente em função das corridas do símbolo que ocorre em grupos.

### *D. Implementação de Segmento Digitalizado Adaptativo*

1. Unidades de Segmento Inicial e Final

Os extremos de um SLRD hipotético podem estar truncados ou completamente fora do modelo estrutural. No primeiro caso, o SLRD deve ser incorporado ao modelo adaptativo, e rejeitado no segundo. A Fig. 6 mostra um AFA que modela a $U_1$ pertencendo ao conjunto $\{a^n b: n = 3,4,5\}$. Na figura, o parâmetro $r_4$ é o último estado de uma sequência conexa iniciando no estado r. A partir dessa sequência conexa, o AFA pode remover até quatro transições em vazio por intermédio da FAD RA aprsentada na Tabela III, ou seja, cada vez que RA é ativada pelo token *a*, uma das transições em vazio da sequência conexa é removida. Além disso, qualquer token *b* conduz o AFA para o estado final; desde que não sejam recebidos mais do que quatro tokens *a*. Uma estrutura para a última USLR seria bastante semelhante.

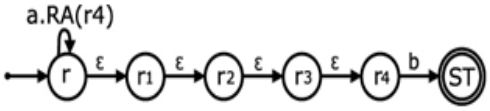



Fig. 6. SLRDA para modelar a primeira USLR de um SLRD.

TABELA III
FAD PARAMÉTRICA RA, DO AFA DA FIG 6

| |
|---|
| RA(xi){vr1, vr2, vr3, vr4: |
| -[(xi-1,ε) → xi] |
| -[(xi, vr1) : vr2 → vr3 : vr4] |
| +[(xi-1, vr1) : vr2 → vr3 : vr4] |
| -[(r,a) → a : RA(xi)] |
| -[(r,a) → a : RA(xi-1)] } |

2. O Segmento Digitalizado Adaptativo

A representação das diferentes instâncias do modelo ideal afetado pelos erros em ângulo requer que o SLRDA atue numa faixa de ângulos. Este tópico ilustra a modelagem de SLRDA, exemplificando com USLR $U_i$ pertencendo ao conjunto $\{a^n b: n = 3,4,5\}$ do primeiro quadrante. Nesse conjunto, o ângulo local $\theta_U$ relativo ao eixo *x* correspondente a cada $U_i$ de um SLRD S pode variar na faixa seguinte:

$\arctan(3) \leq \theta_U \leq \arctan(5)$ .

A Fig. 7 mostra a configuração inicial do autômato para modelar a $U_1$, mesmo sendo truncada (a estrutura para o caso de $U_\lambda$ ser truncada é similar e não está representada). As FAD do AFA estão descritas na Tabela IV.

A seguir, para cada USLR, o AFA percorre um ciclo de estados que inicia e termina no estado u (u, $u_1$, $u_2$, $u_3$, $u_4$, $u_5$, u) formando uma variante de *loop* adaptativo com quantidade de estados contante, porém as transições consumidas variam pela ação da FAD RA, seguindo o descrito no item D1. A FAD RB garante a existência das transições entre os estados $u_4$ e $u_5$ após cada USLR $U_i$ processada, tendo em vista que a FAD RA remove as transições para esses estados em cada volta no ciclo. Este processo é repetido até que o fluxo de entrada se esgote aceitando USLR do tipo $\{a^n b: n = 3,4,5\}$.

A transição que consome o *token c* é incluída apenas para indicar o final do segmento, com o autômato atingindo o estado final se o processo for bem-sucedido. Por outro lado, existindo mais do que 5 tokens *a*, a ação adaptativa elementar - [(xi, vr1): vr2 → vr3:vr4] de RA remove a transição correspondente ao token *c*, rejeitando o segmento.

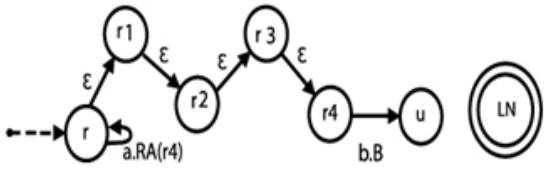



Fig. 7. Configuração inicial de SLRDA, considerando erros em ângulo

TABELA IV
FAD DO AFA DA FIG. 7 E FIG. 8. A FAD RA ESTÁ NA TABELA III.

| | |
|---|---|
| RB{:<br>-[($u_4$,ε)→$u_5$]<br>-[($u_3$,ε)→$u_4$]<br><br>+[($u_4$,ε)→$u_5$]<br>+[($u_3$,ε)→$u_4$] } | B{:<br>-[(r,a) → r : RA($r_i$)]<br>-[(r,ε) → $r_1$]<br>-[($r_1$,ε) →$r_2$]<br>-[($r_2$,ε) →$r_3$]<br>-[($r_3$,ε) → $r_4$]<br>+[(u,a) →$u_1$]<br>+[($u_1$,a) →$u_2$]<br>+[($u_2$,a) →$u_3$]<br>+[($u_3$,a) →$u_3$ : RA ($u_5$)]<br>+[($u_5$, ε) →$u_4$]<br>+[($u_4$, ε) →$u_3$]<br>+[($u_5$,b) →u : RB]<br>+[($u_3$,c) →LN] } |

Na Fig. 7, com o primeiro token *b* consumido, a FAD B é ativada, a qual remove as transições da configuração inicial, alterando a topologia do autômato para a da Fig. 8.

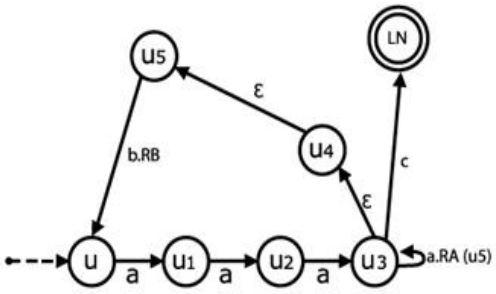



Fig. 8. Configuração do AFA da Fig. 7 após ativação da FAD B.

### *E. Erros em Comprimento*

Resumindo sobre as dificuldades existentes na modelagem de SLRD sob distorção e ruído, sobressaem-se: *i)* Representar e aplicar sintaticamente tolerâncias aos SLRD em qualquer escala; *ii)* Comparar segmentos de reta dentro da tolerância, para qualquer escala; *iii)* Estimadores de comprimento requerem normalmente parâmetros que compensem pelo comprimento, superestimado, obtido da codificação; *iv)* As soluções requerem métodos para identificar valores aproximados, definidos por inequações.

Uma maneira de representar ou aplicar tolerâncias é por um *loop* da Fig. 5, em que o seu número de estados $t_o$ (ou seja, sua dimensão) é alterado adaptativamente em função, por exemplo, do ângulo $\theta_S$ relativo ao eixo *x* da direção principal do SLRD S.

Note que o símbolo das transições entre os estados do *loop* é irrelevante, pois, uma vez alterado $to$, o *loop* torna-se apenas de consulta pelo autômato.

Considerando um parâmetro ψ como uma taxa de erro na estimação de comprimentos de um SLRD S, com $|S| = n$, com $0 < \psi < 1$, significando uma pequena percentagem de *n*. Também supondo o caso mais simples em que o comprimento de S é estimado pela quantidade de simbolos na vizinhança-4, corrige-se *n* para um comprimento estimado denominado $l_E$: $l_E \quad = \psi \times n$.

A idéia é que a cada símbolo do SLRD S, o autômato acesse o grafo, varrendo uma posição adiante no *loop*, circulando pelo grafo tantas vezes quanto seja *n* a fim de calcular o comprimento estimado $l_E$.

Para a implementação desse processo de correção no SLRDA, é necessário levar em conta que *n* é variável; bastando alterar, por meio de uma FAD, a quantidade de estados $t_o$ do loop da Fig. 5, de acordo com a Expressão 6 em função de $\theta_S$ (o ângulo principal do SLRD).

$$t_0 \cong \left\lfloor {}^{1}/_{(1-\psi)} \right\rfloor \quad (6)$$

Para um dado valor de *n*, a cada volta no grafo o SLRDA "bombeia" uma primitiva (por exemplo, uma transição qualquer com um simbolo não pertencente a Σ), desse modo obtendo uma medida sintática do parâmetro $(1-\psi)$ aplicado ao SLRD específico.

Resulta que $\lfloor {}^{n}/_{to} \rfloor$símbolos terão sido bombeados com o último estímulo do SLRD. Portanto $\lfloor {}^{n}/_{to} \rfloor$ simbolos são excluídos do total *n* pelo SLRDA, a fim de corrigir o comprimento estimado de *n* para $l_E$.

Também é possível incorporar ao SLRDA um processo de comparação de comprimentos relativos por meio de inequações conforme a Expressão7, que permite avaliar dois segmentos, $|S_1| = w$ e $|S_2| = l$, numa faixa de valores.

$$w - \lfloor {}^{w}/_{to} \rfloor \leq S_2 = l \leq w + \lfloor {}^{w}/_{to} \rfloor \quad (7)$$

### *F. Expressividade do Modelo Proposto*

Dois parâmetros de SLRD considerados importante estruturalmente são o comprimento e o ângulo principal do SLRD $\theta_S$. Localmente, uma USLR apresenta um determinado comprimento e ângulo de orientação $\theta_U$.

Comparando com as linguagens humanas, as quais se desenvolvem como um organismo vivo partindo de sementes, que se estruturam em raízes, posteriormente tais raízes criam famílias de palavras; da mesma maneira o SLRDA representa, na sua configuração inicial, uma semente correspondente aos ângulos e tolerâncias passível de se desenvolverem de acordo com os estímulos, em segmentos diversos pelas combinações de $\theta_S$ e $\theta_U$, cujos comprimentos podem variar teoricamente até infinito. A efetivação de um segmento específico é efetuada por condições de contorno, dentro da faixa de ângulos e comprimentos, atendendo à vizinhança adaptativa descrita no item a seguir.

## VI. SÍNTESE DA PROPOSTA

A proposta desta pesquisa pode ser sintetizada nos seguintes pontos: *i)* Aplicar a adaptatividade para representar os erros, as tolerâncis e os parâmetros envolvidos em segmentos digitalizados, conforme descrito no tópico V; *ii)* Utilizar uma vizinhança adaptativa, expressa pela propriedade da corda modificada para modelos de ordens superiores (ordem n), a fim de incorporar dinamicamente ao modelo as tolerâncias em parâmetros, tais como ângulo e comprimento; *iii)* Considerar a facilidade de escalas adaptáveis. Este tópico introduz a vizinhança adaptativa e comenta sobre a escala adaptável.

A característica da vizinhança adaptativa de um SLRD é de apresentar largura relativamente grande (proporcional ao comprimento medido) na direção da estrutura linear global [19]. Por essa característica, a Definição 5 é alterada para uma

vizinhança $\max\{|x-h|, |y-h|\} < n$; onde $n$ é a ordem do modelo, dependente da situação momentânea e dos estímulos.

Relembrando que o modelo de primeira ordem da Fig. 4 se caracteriza por P constante, um esboço dos modelos de ordens superioriores pode ser observado na mesma Fig. 4, desde que habilitando P a ser um número inteiro variável teoricamente até infinito: $P(a, b^m): 0 \leq m < \infty$.

### *A. Implementação e Teste da Vizinhança Adaptativa*

A vizinhança adaptativa associa um conjunto de arcos a um SLRDA correspondente, que reconhece esse conjunto. Este tópico apresenta a implementação e teste de modelos de SLRDA atendendo à proposta de vizinhança adaptativa, as quais estão associadas a regiões delimitadas por arcos responsáveis pelos seus contornos. Há dois tipos principais desses arcos delimitadores: arcos côncavos e arcos convexos. O item a seguir mostra apenas o caso de vizinhança delimitada por arco côncavo, pois o caso convexo apresenta estruturas aproximadas ao côncavo.

#### 1. Vizinhança delimitada por arco côncavo

Na seção V. o loop adaptativo da Fig. 5 foi a estrutura usada para obter os comprimentos corrigidos $l_E$. Neste tópico a mesma estrutura é aplicada, tendo o valor $to$ o significado de comprimento de referência para o autômato alterar a vizinhança adaptativamente.

O SLRDA da Fig. 9 reconhece *strings* do tipo $W = \Psi S$ em que S é o SLRD a ter sua vizinhança alterada adaptativamente em função de seu próprio comprimento dinâmico, relativamente à quantidade de estados $to$ do loop. $\Psi$ é uma *string* empregada para infomar o valor de $to$ ao SLRDA. Um símbolo adicional $y$ foi utilizado nas implementacões, apenas para informar os novos valores de $to$ ao autômato, ou seja, $\Psi$ é a *string* $\Psi = y^{to}$.

O SLRDA mencionado constrói o loop adaptativo de quantidade de estados $to$ de acordo com o número de símbolos $y$, pela FAD RO. A transição $[(r_1, a) \rightarrow r_1]$ é uma simplificação para modelar os possíveis símbolos $a$ da USLR inicial.

Com o primeiro símbolo $b$, o AFA passa a consumir as USLR subsequentes. Cada USLR de S ativa as FAD IB e RB,

Para uma *string* de entrada $S: U_i\,; i = 1,2, \dots, \lambda$, a configuração de SLRDA mostrada à esquerda da Fig. 9 é mantida enquanto $1 \leq i < to$ aceitando USLR do tipo $\{a^l b: l = 2\}$, sendo que quando $i = to$a FAD IB insere uma nova transição em vazio na sequência conexas de *s.* Para $t_0 \leq i < 2to$, as USLR aceitas passam a ser na faixa $\{a^l b: l = 2,3\}$. Quando $i = 2to$, a FAD IB insere nova transição em vazio na sequência conexas de *s* alterando a configuração para a Fig. 9, à direita. A USLR aceitas passam a ser na faixa $\{a^l b: l = 2,3,4\}$. E assim por diante, a cada $i$ multiplo de $t_0$ a FAD IB insere uma transição na mencionada sequência conexa alterando a vizinhança.

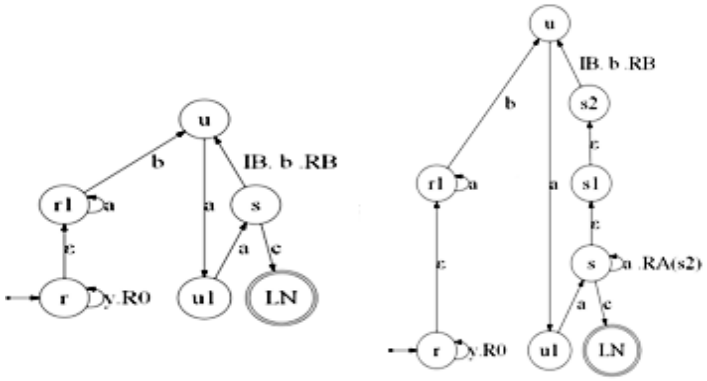



Fig. 9 Á esquerda está a configuração inicial de SLRDA que ajusta a correspondente vizinhança em função de $t_o$, do loop adaptativo, pela quantidade de estados da sequência conexa de s. Á direira está a configuração no início da terceira volta, pela inclusão do estado $s_2$.

Resumindo o processo, cada USLR de S ativa as FAD IB e RB, tal que: i) As transições da sequência conexa $s_1$, $s_2$, $s_3$... $s_v$ que existirem no autômato são removidas pela FAD RB (isso porque a USLR anterior pode não ter a quantidade máxima de tokens a previstos na vizinhança afeta à configuração do AFA); ii) A FAD IB insere novamente no autômato as mesmas transições $s_1$, $s_2$, $s_3$... $s_v$ removidas pela FAD RB, a fim de possibilitar o consumo da USLR seguinte; iii). A FAD IB movimenta um ponteiro[13] uma posição adiante no loop adaptativo; iv) A cada volta completa no loop adaptativo efetuada pelo ponteiro, a FAD IB executa também o seguinte: a) Altera a vizinhança do SLRDA pela inclusão de uma transição $[(s_v, \varepsilon) \rightarrow s_{v+1}]$ incrementando a sequência conexa $s_1$, $s_2$, $s_3$... $s_v$ do estado $s_{v+1}$; b) Altera a FAD RB para que remova, no passo *i)* acima, também as transições relacionadas ao estado $s_{v+1}$, incluído no passo anterior; (c) Remove a transição $[(s, a) \rightarrow s.RA(s_v)]$ e insere a transição $[(s, a) \rightarrow s.RA(s_{v+1})]$.

A vizinhança é delimitada por arco côncavo devido ao ângulo máximo de cada USLR que aumenta gradativamente seguindo os ângulos arctan(2), arctang(3), arctang(4)....arctan($\lfloor \lambda/to \rfloor$) relacionados à sequência conexa do estado *s*.

### *B. Comparação com outros modelos*

Este tópico posiciona esta pesquisa dentro do contexto do estado-da-arte apontando as semelhanças e diferenças entre os diversos modelos, indicando vantagens e desvantagens.

Confrontando esta proposta com os modelos tradicionais, sem adaptatividade, um primeiro aspecto é quanto à questão da flexibilidade dos modelos.

#### 1. Flexibildade

O modelo tradicional, sem adaptatividade, tem que atender à propriedade da corda, rejeitando os arcos da Fig. 10 caracterizados por USLR U na faixa $U(b, a^m): 2 \leq m < 5$, pois, de acordo com a propriedade da corda, deve existir um

[13] Entende-se como ponteiro uma transição elementar, podendo ser em vazio, denominado pelo seu estado que se mantém fixo, enquanto o outro estado varia.

padrão de números consecutivos nas corridas, sem apresentar irregularidades. Na Fig. 10, à esquerda, o arco é codificado por $(a^3\ b)^6(a^4\ b)^5(a^5\ b)^5(a^6\ b)^2a^4b\ (a^3\ b)^6$. O arco à direita é codificado por $(a^3\ b)^6(a^4\ b)^5(a^5\ b)^5(a^6\ b)^5$.

Fig. 10. Exemplos de SLRD reconhecidos pelo SLRDA da Fig. 9 com $to = 5$; porém não reconhecidos pelo método tradicional, sem adaptatividade.

Dentre os modelos tradicionais, [21] obteve algoritmos com relativamente maior flexibilidade ao relaxar a propriedade da corda, numa faixa pré-definida de valores das corridas por meio de um valor denominado *d*, mantendo a característica de código estático e funcionalidade constante. Um exemplo foi $d = \left\lfloor {(p+1)}/{2} \right\rfloor$ tal que os arcos aceitos tenham USLR U apenas na faixa $U(b, a^m)$: $p \leq m \leq d$. Quanto aos resultados obtidos por [21] destacam-se a redução em espaço de memória afeto às estruturas de dados e a complexidade em tempo dos algoritimos torna-se linear.

De modo geral, ao apresentarem comportamento invariável, os métodos tradicionais tendem a segmentar os arcos digitais em grande quantidade de segmentos curtos. Em especial, os métodos com valores pré-definidos numa faixa, exemplicados por [21], reduzem o número de segmentos curtos, porém perdem a característica global dos arcos, rejeitando os SLRD da Fig. 10, que são seccionados em diversos SLRD.

As alterações dinâmicas de vizinhanças desta proposta, que alteram a funcionalidade dos algoritmos em função dos estímulos, pemitem uma melhor caracterização global dos arcos, reconhecendo, por exemplo, os SLRD da Fig. 10.

2. Complexidade computacional

Devido às limitações de comportamento invariável ou predefinido numa faixa, os métodos tradicionais tendem a seccionar os arcos digitais em um conjunto composto por muitos segmentos curtos. Ao reduzir sensivelmente os elementos componentes desse conjunto, a proposta adaptativa minimiza as estruturas de dados requeridas para representação e armazenamento de dados.

Além disso, a introdução de tolerâncias adaptativas simplifica os algoritimos comparativamente aos métodos que apresentam comportamento invariável, reduzindo a complexidade em tempo dos algoritimos adaptativos, além de torná-la linear.

3. A vantagem da escala adaptável

Por estarem limitados a linguagens regulares em 1979, [7] foi obrigado a introduzir um factor de normalização de escalas, que poderia ser o comprimento total de um contorno padrão. Entretanto, a finalidade principal da transformação de escala desta proposta passa a ser conforme os comprimentos relativos de partes de um arco com relação ao arco como um todo, e não eliminar o problema enfrentado por [7] dos comprimentos variáveis de arcos.

A escala adaptável reutiliza, num novo contexto, uma técnica conhecida desde a década de 70 do século passado. Nesse novo contexto, arcos identificados como linhas retas apenas pela geometria discreta aritmética de [23], podem ser avaliados corretamente pelo modelo proposto, numa escala compatível.

4. Posicionamento quanto ao estado-da-arte

O trabalho de [31] relaciona as seguintes tendências dos dispositivos ou algoritmos: guiado por regras (não-adaptativo) cujo conjunto de regras é invariável exemplificado pelo uso apenas da propriedade da corda; guiado por regras (com adaptatividade básica) cujo conjunto de regras é variável, exemplificado pelos trabalhos de [21] e [24]; e guiado por regras (com adaptatividade hierárquica multi-nível) cujo conjunto de regras é variável apresentando funções adaptativas modificáveis, na qual se insere esta proposta.

Resulta, portanto, em todo um potencial evolutivo desta proposta partindo da reutilização e generalizações de formalismos que têm sido tradicionalmente utilizados.

## VII. CONSIDERAÇÕES FINAIS

Esta pesquisa introduz a abordagem adaptativa no assunto da representação de linhas e arcos digitais, considerando pequenos erros. Comparando com outras representações, duas das principais vantagens verificadas na proposta foram simplicidade de modelagem e relativa facilidade de implementação, associadas a alto poder computacional. O método proposto é intuitivo, os seus modelos são fáceis de entender, relativamente simples de programar e flexíveis para aceitar mudanças em seu comportamento.

Existem aplicações de SLRD, aproximando curvas por conjuntos de segmentos. Conforme [27], um arco qualquer pode ser aproximado por linhas retas dentro de uma vizinhança (os autores apresentam inclusive critérios para seleção da densidade da grade de digitalização). Portanto, estimadores de comprimento de SLRD tendem a apresentar bom desempenho com curvas gerais, prevendo que um dispositivo adaptativo represente arcos digitais em diferentes escalas. Em outra aplicação, [33] propõe um *polyautomaton* que reconhece linhas retas visando computação paralela. A idéia era que cada célula da grade fosse representada por um AF, que pode ser substituído por um dispositivo adaptativo, conforme esta proposta.

Os modelos e as técnicas de tratamento de erros e de tolerâncias da proposta, certamente auxiliarão em problemas sintáticos recorrentes, independentemente da área aplicada, e vice-versa. Por exemplo, ao se interpretar um contorno digital complexo, composto por arcos orientados em diversas direções, como um percurso a ser percorrido pelo autômato, foi relatada por esta pesquisa uma arquitetura para reconhecimento do contorno, que se baseia no mapeamento de

ambientes desconhecidos por robôs móveis, nos moldes de [34], tendo por base técnicas de inferência descritas em [35]. Concluindo, destaque-se a versatilidade da proposta, no sentido de que as técnicas adaptativas apresentadas podem ser aplicadas sempre que ocorrerem abstrações envolvendo modelos de retas e arcos, permitindo reutilização, generalizações ou revitalização de formalismos que têm sido tradicionalmente utilizados [32].

## REFERENCES


[1] João José Neto, Contribuições à metodologia de construção de compiladores. Tese de livre docência. Escola Politécnica da Universidade de São Paulo, 1993.

[2] João José Neto, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa." Revista IEEE América Latina. Volume 5. Número 7, pag. 496–505, Novembro. 2007.

[3] Hemerson Pistori, Tecnologia Adaptativa em Engenharia da Computação: Estado da Arte e Aplicações, Ph. D. thesis, Escola Politécnica da Universidade de São Paulo. 2003.

[4] R. A. Klette and A. B. Rosenfeld, "Digital straightness - a review," Discrete Applied Mathematics, vol. 139, no. 1-3, pp. 197–230, 2004.

[5] João José Neto, "Adaptive rule-driven devices - general formulation and case study," in Proceedings of the..., Springer-Verlag, Ed., Pretoria, South Africa, July 2001, IMPLEMENTATION AND APPLICATION OF AUTOMATA 6TH INTERNATIONAL CONFERENCE, vol. 2494, pp. 234–250, CIAA 2001.

[6] L. Dorst and A. W. M. Smeulder, "Length estimators for digitized contours," Computer Vision, Graphics, & Image Processing, vol. 40, no. 3,pp. 311–333, 1987.

[7] Kai Ching You and King-Sun Fu, "A syntactic approach to shape recognition using attributed grammars," IEEE transactions os systems, man, and cybernetics, vol. 9, no. 6, pp. 334–345, June 1979.

[8] H. Freeman, "Boundary encoding and processing," Picture Proceedings and Psychopictorics, pp. 241–266, 1970.

[9] Azriel Rosenfeld, "Digital straight line segments," IEEE Transactions on Computers, vol. C-23, no. 12, pp. 1264–1269, 1974.

[10] R. Brons, "Linguistic methods for the description of a straight line on a grid," Computer Graphics and Image Processing, vol. 3, no. 1, pp. 48 – 62, 1974.

[11] S. Shlien, "Segmentation of digital curves using linguistic techniques" Computer Vision, Graphics and Image Processing, vol. 22, no. 2, pag. 277– 286, 1983.

[12] H.R. Lewis and C.H. Papadimitriou, Elements of the Theory of Computation, Prentice-Hall, 1981.

[13] J. Feder, "Languages of encoded line patterns," Information and Control, vol. 13, no. 3, pp. 230–244, 1968.

[14] R. L. A. Rocha and J. J. Neto, "Autômato adaptativo, limites e complexidade em comparação com máquina de Turing." in Proceedings of the..., Faculdade SENAC, Ed. PROCEEDINGS OF THE SECOND CONGRESS OF LOGIC APPLIED TO TECHNOLOGY, 2001, p. 33 a 48.

[15] Forchhammer S. Aghito, S.M., "Context-based coding of bilevel images enhanced by digital straight line analysis," IEEE Transactions on Image Processing, vol. 15, no. 8, pp. 2120–2130, 2006.

[16] D. Proffitt and D. Rosen, "Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges," Computer Graphics and Image Processing, vol. 10, no. 4, pp. 318–332, 1979.

[17] K. S. Fu, Syntactic Methods in Pattern Recognition, Academic Press, 1974.

[18] Christian Ronse, "A strong chord property for 4-connected convex digital sets," Computer Vision, Graphics, and Image Processing, vol. 35, no. 2, pag. 259 – 269, 1986.

[19] Peter F.M. Nacken, "Metric for line segments," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 12, pp. 1312–1318, 1993, cited By (since 1996) 18.

[20] H. C. Lee and K. S. Fu, "Using the fft to determine digital straight line chain codes," Computer Graphics and Image Processing, vol. 18, no. 4, pp. 359–368, 1982.

[21] Partha Bhowmick and Bhargab B. Bhattacharya, "Fast polygonal approximation of digital curves using relaxed straightness properties," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, pp. 1590–1602, 2007.

[22] I. Debled-Rennesson, F. Feschet, and J. Rouyer-Degli, "Optimal blurred segments decomposition of noisy shapes in linear time," Computers & Graphics, vol. 30, no. 1, pp. 30 – 36, 2006.

[23] J. P. Reveillès, Géométrie discrète, calcul en nombres entiers et algorithmique, Ph.D. thesis, Université Louis Pasteur, Strasbourg, 1991.

[24] F. Feschet, "The lattice width and quasi-straightness in digital spaces," in Proceedings of the..., Tampa, FL, 2008, INT. CONF. PATTERN RECOGN.

[25] C. Fiorio, D. Jamet, and J. L. Toutant, "Discrete circles: an arithmetical approach with non constant thickness," in Proceedings of the..., Electronic Imaging, Ed., San Jose (CA), 2006, SPIE VISION GEOMETRY XIV, vol. Volume 6066.

[26] Reinhard Klette and Azriel Rosenfeld, Digital geometry: geometric methods for digital picture analysis, Morgan Kaufmann, 2004.

[27] Nahum Kiryati and Olaf Kübler, "Chain code probabilities and optimal length estimators for digitized three-dimensional curves," Pattern Recognition, vol. 28, no. 3, pp. 361–372, 1995.

[28] Shu-Xiang Li and Murray H. Loew, "Analysis and modeling of digitized straight-line segments," in Proceedings of the..., Rome, Italy, 1988, PROCEEDINGS OF INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, pp. 294–296, Publ by IEEE, Piscataway, NJ, United States.

[29] J. R. Garitagoitia, J. R. G. de Mendívil, J. Echanobe, J. J. Astrain, and F. Fariña, "Deformed fuzzy automata for correcting imperfect *strings* of fuzzy symbols," IEEE Transactions on Fuzzy Systems, vol. 11, no. 3, pp. 299–310, 2003.

[30] Roland C. Backhouse, Syntax of programming languages: theory and practice, Prentice-Hall International, 1979.

[31] João José Neto, "Adaptatividade: Generalização conceitual," in 3º Workshop de Tecnologia Adaptativa. Escola Politécnica da Universidade de São Paulo, 2009.

[32] Gonzalo Bailador and Gracián Triviño, "Pattern recognition using temporal fuzzy automata," Fuzzy Sets and Systems, vol. 161, no. 1, pag. 37 – 55, 2010, Special section: New Trends on Pattern Recognition with Fuzzy Models.

[33] Jerome Rothstein and Carl Weiman, "Parallel and sequential specification of a context sensitive language for straight lines on grids," Computer Graphics Image Processing, vol. 5, no. 1, pp. 106–124, Mar. 1976.

[34] Miguel Angelo de Abreu de Sousa, "Mapeamento de ambientes desconhecidos por robôs móveis utilizando autômatos adaptativos," M.S. thesis, Escola Politécnica da Universidade de São Paulo, 2006.

**[35]** P. M. Ivone, "Um estudo dos processos de inferência de gramáticas regulares e livres de contexto baseados em modelos adaptativos." Dissertação, Escola Politécnica da Universidade de São Paulo, 2006.



**Leoncio C. Barros Neto**. Formado em Engenharia Elétrica pela Escola de Engenharia Mauá, São Caetano do Sul, SP, em 1979. Concluiu o mestrado pela USP em 1994. Em 1981 foi aprovado em concurso público de âmbito nacional, integrando o Corpo de Engenheiros Navais da Marinha do Brasil (MB). Na MB exerceu atividades técnicas em Setores de Manutenção e Pesquisas, iniciando programa de doutorado na USP em 2006, após transferência para a reserva. Suas áreas de interesse e pesquisa abrangem automação, sensores inteligentes, eletrônica de potência, sistemas embarcados, guerra eletrônica e suas aplicações.

**André R. Hirakawa.** Recebeu os títulos de Engenheiro Eletricista e correspondente mestrado pela USP em 1990 e 1992 respectivamente. Em 1992, como pesquisador, foi integrado no programa de doutorado da Yokohama National Universidade, em Yokiohoma, Japan, tendo recebido o título de PhD em 1997. Em 1998, iniciou atividades no Departamento de Engenharia de Computação e Sistemas Digitais da USP, sendo atualmente Professor Associado. Suas áreas de interesse e pesquisa abrangem automação e robótica, sensores inteligentes, eletrônica de potência, planejamento de rotas, AVG's, e sistemas sem fio, tanto aplicados à automação agrícola quanto em outros ambientes.

**Antonio M. A. Massola.** Professor Titular da USP.