WTA 2008 – Segundo Workshop de Tecnologia Adaptativa

MEMÓRIAS DO WTA 2008

SEGUNDO 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
2008

MEMÓRIAS DO WTA 2008

SEGUNDO 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
2008

I. FICHA CATALOGRÁFICA

Workshop de Tecnologia Adaptativa (2. : 2008 : São Paulo)
Memórias do WTA'2008 : segundo Workshop de Tecnologia Adaptativa / organizadores, Ricardo Luis de Azevedo da Rocha, João José Neto. -- São Paulo, 2008.
60 p.

ISBN 978-85-86686-46-7

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. Rocha, Ricardo Luis de Azevedo da II. José Neto, João III. Universidade de São Paulo. Escola Politécnica. Laboratório de Linguagens e Técnicas Adaptativas IV. t.
CDD 004

WTA‘2008

A segunda edição do WTA realizou-se em São Paulo, BRASIL, nos dias 31 de janeiro e 01 de fevereiro de 2008, 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 2008: Segundo Workshop de Tecnologia Adaptativa

J. J. Neto

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 2008, o Segundo Workshop de Tecnologia Adaptativa, realizado em São Paulo no dia 31 de Janeiro e 01 de fevereiro de 2008. A exemplo da edição de 2007, este evento contou com uma presença maciça da comunidade de pesquisadores que se dedicam ao estudo e ao desenvolvimento de trabalhos ligados a esse tema em diversas instituições brasileiras, sendo o material aqui compilado representativo dos avanços alcançados nas mais recentes pesquisas e desenvolvimentos realizados.**

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

I. INTRODUÇÃO

É com grande satisfação que apresento a todos os interessados estas Memórias do WTA 2008 – Segundo Workshop de Tecnologia Adaptativa, evento realizado pelo LTA – Laboratório de Linguagens e Técnicas Adaptativas, nos dias 31 de janeiro e 01 de fevereiro de 2008, nas dependências da Escola Politécnica da USP.

Com muita felicidade, constatou-se novamente o sucesso dessa iniciativa, que no ano passado congregou, em uma jornada de trabalho, os interessados em Tecnologia Adaptativa que trabalharam em estreita relação com o LTA.

As memórias do WTA 2007, compiladas na forma de um interessante e informativo tutorial, se encontram publicadas no número especial da prestigiosa Revista IEEE Latin American Transactions, vol. 5, no. 7, de Novembro de 2007 (ISSN 1548-0992).

II. O SEGUNDO WORKSHOP DE TECNOLOGIA ADAPTATIVA

Nesta edição de 2008 do Workshop de Tecnologia Adaptativa, importantes evoluções em relação à do ano anterior foram observadas:

- **Duração**: o evento foi realizado em dois dias, o que ampliou as oportunidades de contatos e interação entre os participantes;
- **Chamada de trabalhos**: houve uma chamada aberta de trabalhos, facilitando o envolvimento de interessados e a aproximação de elementos externos ao LTA;
- **Origem dos participantes e dos trabalhos**: observou-se o afluxo de participantes, o recebimento e a aceitação de contribuições técnicas excelentes, oriundas de outras cidades, algumas fora do Estado de São Paulo;
- **Mini-curso**: tivemos o prazer de ouvir um ilustre convidado, o jovem e experiente prof. Dr. Ítalo Santiago Vega, da Pontifícia Universidade Católica de São Paulo, que apresentou um mini-curso muito interessante intitulado *Implementação OO para Autômatos Adaptativos: Impactos das Tomadas de decisão efetuadas durante o Design*, com duração de 4 horas, na forma de tutorial sobre técnicas do projeto e da implementação, orientada a objetos, de programas que implementam o comportamento adaptativo;
- **Artigos Completos IEEE**: destacaram-se, nas atividades do evento, artigos completos, que foram apresentados como palestras técnicas, baseadas em textos aceitos para publicação pela IEEE Latin American Transactions;
- **Resumos estendidos**: relatando trabalhos em estágio avançado de andamento, ou programas adaptativos desenvolvidos, o evento contou com a apresentação de *short papers* com significativo número de trabalhos de excelente nível, em um largo espectro de aplicações;
- **Mostra de software**: coordenada pelo Prof. Dr. Ricardo Luis de Azevedo da Rocha, essa mostra abriu espaço para a exibição de diversos interessantes programas, desenvolvidos com Tecnologia Adaptativa, os quais complementaram diversos *short papers* apresentados no evento;
- **Mesa Redonda**: o evento foi finalizado com muito êxito por meio de uma mesa redonda, composta de professores convidados, e presidida pelo Prof. Dr. André Riyuiti Hirakawa, em que foram apresentadas as visões, experiências e opiniões de cada um de seus participantes acerca de diversos assuntos de interesse da comunidade, destacando-se: tendências e aplicações da Tecnologia Adaptativa; aproximação entre grupos interessados; formação de novos grupos de interesse na área; experiências individuais e institucionais dos participantes, em Tecnologia Adaptativa; iniciativas, sugestões e recomendações dos participantes para o desenvolvimento da área.
- **Apoios**: esta edição do WTA contou com diversos prestigiosos apoios: da SBC (Sociedade Brasileira de Computação), da SPC (Sociedad Peruana de Computación), do IEEE (The Institute of Electrical and Electronics Engineering), da EPUSP (Escola Politécnica da USP) e do PCS (Departamento de Engenharia de Computação e Sistemas Digitais).;

João José Neto é professor associado do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP, e foi o coordenador geral do evento WTA 2008 – Segundo Workshop de Tecnologia Adaptativa. (joao.jose@poli.usp.br) fone: (11) 3091-5402

- **Formação de novos grupos**: observou-se, com satisfação, um movimento significativo dos participantes, em direção à criação de diversos grupos de interesse em Tecnologia Adaptativa, inclusive em âmbito internacional. Em particular, foram registrados importantes grupos emergentes na Universidade Federal de São Carlos, SP, na Pontifícia Universidad Católica del Perú, em Lima e na Universidad Católica San Pablo, em Arequipa.

I. Esta Publicação

Estas memórias espelham o conteúdo apresentado no WTA 2008. Cabe observar que os artigos técnicos completos, apresentados no evento, serão publicados separadamente, como artigos regulares da IEEE Latin American Transactions, e aqui, por esta razão, estão reproduzidos apenas seus resumos (abstracts), e não o texto integral.

A exemplo do que foi feito no ano anterior, todo o material referente aos trabalhos apresentados no evento estarão acessíveis no portal do WTA 2008, 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.

II. Conclusão

Foi compensador o esforço despendido na preparação e realização deste evento, e sumamente gratificante observar-se após um ano os bons frutos resultantes dos trabalhos do workshop anterior.

A repetição do sucesso daquele primeiro evento, e o nível de qualidade dos trabalhos apresentados atestam incontestavelmente a seriedade com que esse trabalho vem sendo realizado e a sua repercussão na comunidade.

Com cerca de cento e vinte inscritos, e com o comparecimento registrado de uma centena participantes, tivemos o prazer de constatar a permanência de cerca de quarenta participantes até o final do evento, às 18:00h de uma sexta-feira de carnaval, e isso demonstra indiscutivelmente o interesse despertado pela área e o comprometimento dessa pequena comunidade que em torno dela está se formando e crescendo a passos largos.

Para encerrar, gostaríamos de agradecer a todos aqueles que compareceram ao evento, e aos que, de alguma forma, colaboraram para o seu brilho, e aproveitamos esta ocasião para convidá-los, no início do próximo ano, para comparecerem à edição de 2009, para o qual estamos lançando à comunidade o desafio de publicarmos um número significativamente maior de trabalhos, na forma de artigos técnicos completos, *short papers* e softwares adaptativos, de modo que essa produção técnica reflita cada vez melhor os tantos avanços que estão sendo obtidos na América Latina, nesta importante linha de pesquisa e desenvolvimento da Tecnologia Adaptativa.

Agradecimento

Nosso agradecimento às instituições que apoiaram o WTA 2008, à comissão organizadora, ao pessoal de apoio e aos colaboradores voluntários, sem cujo auxílio este evento não lograria aquele sucesso que todos nós tivemos a satisfação de presenciar.

São Paulo, fevereiro de 2008

Prof. Dr. João José Neto

Coordenador Geral do WTA 2008
Laboratório de Linguagens e Técnicas Adaptativas
Departamento de Engenharia de Computação e Sist. Digitais
Escola Politécnica da Universidade de São Paulo

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 2008:

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

Gostaríamos de agradecer ao IEEE, através do Prof. Dr. José Antonio Jardini, pela parceria com o nosso evento, e pelo trabalho de seleção e revisão dos artigos enviados para o evento através da revista IEEE Latin American Transactions, realizado pelos revisores anônimos do seu corpo editorial.

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. Dr. Almir Rogério Camolesi, 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;
- 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;
- Prof. Dr. André Riyuiti Hirakawa, pela organização e condução dos trabalhos da mesa redonda, e seus convidados – nominalmente, Profa. Cristina Toshie Motohashi Matsusaki (FSA – Fundação Santo André), Prof. Aparecido Valdemir de Freitas (IMES – Universidade Municipal de São Caetano do Sul), Prof. Amaury Antônio de Castro Junior (UFMS – Universidade Federal de Mato Grosso do Sul, campus Coxim), Prof. Luciano Gonda (UCDB – Universidade Católica Dom Bosco, Campo Grande – MS).
- Prof. Dr. César Alberto Bravo Pariente, que se encarregou de diversos aspectos da organização do evento tais como: gestão junto ao IEEE para apoio ao evento; contatos com palestrantes; busca de apoios; editoração em papel das memórias do WTA 2007, publicadas eletronicamente pelo IEEE;
- Prof. Dr. Ítalo Santiago Vega, pela prontidão e competência com que aceitou nosso convite para proferir o mini-curso sobre implementação da Adaptatividade, vista sob o prisma da Engenharia de Software, com o qual tantos elogios logrou ganhar de todos os participantes;
- Prof. Dr. Ricardo Luis de Azevedo da Rocha, a quem devemos diversos agradecimentos: pela realização desta publicação das memórias do WTA 2008; pelo seu registro junto à Biblioteca Nacional do Rio de Janeiro; pela organização dos trabalhos de divulgação e de seleção dos *short papers*; pela promoção, divulgação e realização dos dois dias da mostra de software adaptativo que tivemos durante o evento;

Agradecemos finalmente a boa vontade da qual resultou o importante auxílio prestado ao WTA 2008 pelos seguintes profissionais (ordem alfabética):

- Profa. Ivone Penque Matsuno, pela decisiva colaboração que dela recebemos, garantindo que os cartazes de divulgação do evento ficassem prontos e disponíveis a tempo;
- Leia Sicília, que secretariou o evento, pela permanente alegria e boa vontade que demonstrou, antes e durante seus trabalhos na recepção do WTA 2008;
- Nilton Araújo do Carmo, pelos trabalhos de apoio técnico prestados durante o evento, inclusive pela gravação de todas as imagens dos trabalhos do WTA 2008;
- Arqt. Regina Gauer José, pela graça que deu ao evento através do seu trabalho de criação e execução artística dos cartazes utilizados na sua divulgação;
- Eng. Tomás dos Reis Nogueira, que colaborou não somente cedendo seu equipamento fotográfico, como também encarregando-se do trabalho técnico de cobertura fotográfica de todo o evento.

Resumo do Mini-curso: *Implementação OO para Autômatos Adaptativos: Impactos das Tomadas de decisão efetuadas durante o Design*

I. S. Vega

Resumo — **Este mini-curso, em duas sessões, discute aspectos da realização computacional de dispositivos adaptativos – em particular, dos Autômatos Adaptativos, dispositivos da classe dos autômatos, com topologia dinâmica. Essa discussão é feita do ponto de vista do projetista, e se vale de técnicas de projeto e de programação orientadas a objetos, analisando os efeitos da adoção de algumas das alternativas possíveis sobre a qualidade da implementação resultante. Na primeira sessão são discutidos os aspectos tradicionais da modelagem de software relevantes à segunda sessão, e esta dedica-se à modelagem de uma semântica operacional proposta para a representação de Autômatos Adaptativos, feita mediante a aplicação dos conceitos apresentados na primeira.**

***Palavras-chave* — Projeto e programação orientados a objetos, Autômatos adaptativos, Semântica Operacional, Tecnologia Adaptativa.**

I. INTRODUÇÃO

Este resumo condensa o mini-curso, realizado na forma de um tutorial proferido em duas partes, a primeira com duração de uma hora e meia, e a segunda, de duas horas e meia, no Segundo Workshop de Tecnologia Adaptativa – WTA 2008, nos dias 31 de janeiro e 01 de fevereiro de 2008 na Escola Politécnica da USP, em São Paulo.

II. SESSÃO 1: TÓPICOS DE MODELAGEM DE SOFTWARE

Na primeira parte do mini-curso, o assunto tratado foram os aspectos essenciais da modelagem de software, com ênfase nos pontos mais relevantes para a representação de fenômenos que possam ser descritos como dispositivos abstratos de comportamento auto-modificável. Para isso, cobriu-se o essencial das notações, dos construtos característicos do paradigma,

A. Notação UML

Uma categoria de artefatos produzidos durante o desenvolvimento corresponde aos modelos da própria aplicação de software. A notação UML pode ser utilizada para documentar tais modelos da aplicação.

B. Construtos do Paradigma de Objetos em UML

Fundamentais ao Paradigma de Objetos são os construtos classe, objeto, mensagem, método e herança (entre outros), os quais devem ser apropriadamente representados em UML visando a comunicação humana de intenções.

C. Sintomas de Modelos de Software

Modelos de software exibem propriedades internas e externas. Embora as externas sejam perceptíveis por interação com a aplicação, as internas produzem sintomas que devem ser identificados para que sejam estabelecidas no sentido de melhorar a qualidade do modelo. Em um processo iterativo e incremental tais propriedades podem ser ajustadas de acordo com o objetivo do desenvolvimento.

D. Integração de Modelos Distintos

Ao longo do desenvolvimento, diversos modelos da aplicação são produzidos, devendo se integrar, harmoniosamente, durante a implementação. Estratégias de integração devem ser aplicadas quando os modelos da aplicação se apóiam em visões com naturezas distintas.

BIBLIOGRAFIA RECOMENDADA PARA A SESSÃO 1

UML

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. ***The Unified Modeling Language User Guide***. Addison Wesley, 1999.

FOWLER, M. UML ***Distilled: A Brief Guide to the Standard Object Modeling Language***. Addison-Wesley, 3rd ed., 2003, ISBN 978-0321193681.

Técnicas de Modelagem

LAKOS, J. ***Large-Scale C++ Software Design***. Addison-Wesley, 1996, ISBN 0-201-63362-0.

BOOCH, G. ***Object-Oriented Analysis and Design with Applications***. Addison-Wesley, 2007, ISBN 978-0201895513.

EVANS, E. ***Domain-Driven Design: Tackling Complexity in the Heart of Software***. Addison-Wesley, 2003, ISBN 0-321-12521-5.

MARTIN, R. C. ***Agile Software Development, Principles, Patterns, and Practices.*** Prentice Hall, 2002, ISBN 978-0135974445.

Tecnologia de Objetos

TAYLOR, D. A. ***Object-Oriented Technology: A Manager's Guide***. Addison-Wesley, 1998.

Implementação

HUNT, A.; THOMAS, D. ***The Pragmatic Programmer: From Journeyman to Master***. Addison-Wesley, 1999, ISBN 0-0201-61622-X.

MCCONNEL, S. ***Code Complete.*** Microsoft Press, 2nd ed, 2004, ISBN 978-1556154843.

I. Sessão 2: Autômatos Adaptativos - Semântica Operacional

O particular dispositivo escolhido para ilustrar esse processo foi o Autômato Adaptativo, um autômato que apresenta um conjunto dinâmico de transições de estados, cuja variação de comportamento é determinada autonomamente, pelo próprio autômato.

Na segunda parte do mini-curso, recordam-se, resumidamente, os conceitos dos dispositivos adaptativos que mais são relevantes a este estudo, apresenta-se o conceito de onda de clones e destacam-se os demais conceitos que preenchem algumas lacunas de modelagem existentes nas publicações existentes sobre o autômato adaptativo.

A. Dispositivos Adaptativos

São aqueles cujo comportamento e topologia sofrem alterações quando estimulados por uma entrada, mesmo sem a participação de agentes externos. Uma característica marcante é que as alterações são dinamicamente estabelecidas durante a realização de transições adaptativas.

B. Transições Habilitadas

São aquelas cujas condições de disparo combinam com as condições correntes do autômato. Uma vez habilitada, o autômato estabelece o contexto no qual a transição adaptativa é disparada.

No modelo proposto neste trabalho, o conjunto de transições habilitadas origina uma "onda de clones" que caracteriza uma fase que antecede aos seus respectivos disparos.

C. Considerações

Destaque para a complementação da representação na forma de tabelas de decisão proposta por J. J. Neto, que ajuda no entendimento do mecanismo de adaptação, fundamental na Tecnologia Adaptativa.

II. Conclusão

Este mini-curso apresentou as técnicas essenciais de projeto e de implementação de programas orientados a objetos, destacando os principais percalços previstos em resposta às diversas possíveis decisões efetuadas durante o projeto.

Bibliografia Recomendada para a Sessão 2

Técnicas de Modelagem:

Wirfs-Brock, R.; McKean, A. ***Object Design: Roles, Responsibilities, and Collaborations.*** Addison-Wesley, 2002, ISBN 978-0201379433.

Larman, C. ***Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development***. Prentice Hall, 3rd ed, 2004, ISBN 978-0131489066.

Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides ***Design Patterns: Elements of Reusable Object-Oriented Software.*** Addison-Wesley, 1995, ISBN 978-0201633610.

SUMÁRIO

Apresentação

WTA'2008
João José Neto _____ vi

Mini-curso

Implementação OO para Autômatos Adaptativos: Impactos das Tomadas de decisão efetuadas durante o Design
Ítalo S. Vega _____ ix

Short-papers

SP-1 : Aplicação de Técnicas Adaptativas em Reconhecimento de Formas
Leoncio C. Barros Neto, André R. Hirakawa e Antonio M. A. Massola _____ 1

SP-2 : Tabela de Decisão Adaptativa: Simulação de um Autômato Adaptativo
A. H. Tchemra _____ 5

SP-3 : *SPA*^{3}: Um *framework* para privacidade
Cereda, P. R. M. e Zorzo, S. D. _____ 9

SP-4 : Aplicação de Autômatos Adaptativos na avaliação de autômatos gerados por algoritmos genéticos
Victor D. Lopes, Ricardo L. A. Rocha _____ 14

SP-5 : Implementação de Extensibilidade Sintática com Tecnologia Adaptativa utilizando o JavaCC
A. S. Mignon _____ 18

SP-6 : Biblioteca para autômatos adaptativos com regras de transição armazenadas em objetos feita em C++
N. L. Werneck _____ 22

SP-7 : Autômatos Adaptativos para Emparelhamento de Cadeias
E. S. C. Rodrigues, F. A. Rodrigues, R. L. A. Rocha _____ 27

SP-8 : Visualizador gráfico para Redes de Petri Adaptativa
W. C. M. Gomes e A. R. Camolesi _____ 31

SP-9 : Tradutor Texto-Voz baseado em autômatos adaptativos
D. P. Shibata, F. R. Koike _____ 35

SP-10 : Conversão de partituras para tablaturas usando algoritmo baseado em autômato adaptativo
Danilo de Jesus da Silva Bellini, Anna Catarina Batista Tavella _____ 39

Regular papers

RP-1 : Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão: Uma Abordagem Estratégica na Seleção de Fornecedores
A. H. Tchemra, R. Camargo 44

Comunicações

CM-1 : Correspondência internacional 1 52

CM-2 : Correspondência internacional 2 53

CM-3 : Mesa Redonda (01/02/2008) 54

Índice de autores 60

SHORT PAPERS

Aplicação de Técnicas Adaptativas em Reconhecimento de Formas

Leoncio C. Barros Neto, André R. Hirakawa e Antonio M. A. Massola

***Resumo*—Este artigo apresenta a proposta do desenvolvimento de um método para classificação de formas geométricas básicas por meio de Autômatos Adaptativos (AA). O método proposto utiliza a abordagem sintática de reconhecimento de padrões, apresentando algumas vantagens sobre práticas tradicionais permitindo realizar uma identificação das propriedades estruturais das formas contidas na imagem. A característica adaptativa aperfeiçoa o processo de reconhecimento ao considerar pequenas diferenças entre bordas, geradas por distorções residuais nas fases de segmentação e pré-processamento. O artigo apresenta ainda os resultados dos testes de reconhecimento de formas utilizando AA e implementado experimentalmente.**

***Palavras Chave*— Reconhecimento de Padrões, Autômatos Adaptativos, Experimentação, Modelagem.**

II. Introdução

No sentido de aumentar a capacidade de percepção do ambiente, elevando o desempenho dos sistemas modernos, sensores mais sofisticados tem sido integrados. Dentre esses estão aqueles que empregam imagens, por exemplo, aplicados em robótica para reconhecimento de formas.

No processo de reconhecimento e classificação de padrões, a primeira etapa é a do processamento de imagens e a última é a da classificação de formas, passando pelo processo de extração de características e pré-processamento. Nesse processo, é inegável que a qualidade da imagem e atributos extraídos [1] interferem no resultado. Por outro lado, a classificação continua sendo considerada um desafio para os pesquisadores, podendo comprometer o processo.

A abordagem sintática é considerada fundamental para o reconhecimento de formas devido a sua característica de incorporar informações estruturais [2]. Entretanto, conforme descrito no item a seguir, as técnicas sintáticas tradicionais apresentam restrições tal como a dificuldade em tratar distorções não solucionadas pelas fases preliminares que alimentam dados para a etapa de classificação.

Manuscrito recebido em 11 de Dezembro de 2007. Trabalho desenvolvido no Laboratório de Automação Agrícola (LAA), da Escola Politécnica da USP, São Paulo, Brasil.

L. C. Barros Neto; LAA (número telefônico: 55-11-36824400; e-mail: leoncio.barros@poli.usp.br).

A. R. Hirakawa; LAA (número telefônico: 55-11-30915104; e-mail: andre.hirakawa@poli.usp.br).

A. M. A. Massola; LAA (número telefônico: 55-11-30914289; e-mail: antonio.massola@poli.usp.br).

III. Definição Do Problema

A. Reconhecimento de Formas

O Reconhecimento de formas está inserido no contexto do reconhecimento de padrões (RP). RP pode ser definido como um processo de identificação de estruturas ou padrões nos dados ou medidas, por meio de comparações com estruturas conhecidas [3]. Trata-se de uma ciência complexa, pois a maioria das informações oriundas do mundo real está sujeita a transformações, deformações ou corrompida por ruído.

A Figura 1 apresenta as etapas típicas do processo de RP, envolvendo sensor, pré-processamento, extração de atributos e classificação.

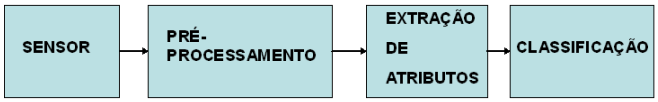


Fig. 1. Estrutura típica de reconhecimento de padrões.

Na etapa inicial do sensoriamento, dados analógicos do mundo físico são coletados por transdutores ou sensores e convertidos em conjuntos de dados digitais. Na fase de pré-processamento, esses dados são tratados convenientemente e encaminhados para a etapa de extração de atributos. A etapa de extração de atributos agrupa os dados recebidos em conjuntos de vetores ou primitivas. No caso de imagens, de modo geral, a extração de atributos agrupa os *pixels* em regiões com características semelhantes num processo denominado segmentação.

Após a segmentação da imagem, a classificação é realizada atuando nesses conjuntos fornecidos pela fase anterior. O algoritmo de classificação compara os atributos encontrados na segmentação dentro de certos limiares de aceitação pré-determinados identificando estruturas similares. As classes resultantes são então associadas às classes de interesse.

Caracterização detalhada dessas fases pode ser encontrada em trabalhos sobre processamento de imagens [4].

Este artigo se concentra na etapa final de classificação utilizando a abordagem sintática, comentada a seguir.

B. Abordagem Sintática

Na abordagem sintática, detalhes locais da imagem são descritos por intermédio de padrões elementares ou primitivas e os sinais de entrada para classificação através da cadeia concatenada dessas primitivas (*string*) [3]–[6]. A seleção de primitivas é especificada de acordo com o problema e visa facilitar a descrição estrutural dos padrões. A Figura 2

apresenta as primitivas utilizadas neste artigo, convenientes para reconhecimento de quadrados e retângulos. Para simplificar, as dimensões das primitivas são consideradas neste trabalho como uma unidade da dimensão unitária convencionada para a imagem. Portanto, as formas segmentadas são descritas por meio de *strings* relacionados aos contornos das regiões associadas.

Metodologias habituais de reconhecimento dessas seqüências de símbolos aplicam autômatos finitos representando uma gramática formal em que as primitivas são símbolos terminais do alfabeto da gramática. Um aspecto interessante é que poucas primitivas podem representar muitas classes, entretanto, o tópico abaixo enfatiza o requisito de uma máquina de Turing para o reconhecimento dos *strings* correspondentes.

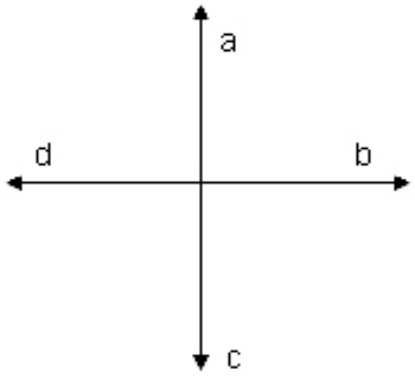


Fig. 2. Exemplo de primitivas.

C. Stringology e Autômatos Adaptativos (AA)

Stringology é o apelido de uma área relativamente nova em Ciência da Computação [5] e engloba a classe de problemas envolvendo cadeias de símbolos, ou passíveis de terem solução por essas cadeias que é o caso do reconhecimento em questão.

Ocorre que as formas a serem reconhecidas envolvem *strings* pertencentes a linguagens dependentes de contexto, requerendo uma máquina de Turing. Em certos casos particulares, um autômato finito pode efetuar o reconhecimento, limitado a formas com dimensões fixadas.

A equivalência dos AA com a máquina de Turing [7] possibilitou o reconhecimento de triângulos eqüiláteros [8]. A questão seguinte é como diferenciar entre formas com estruturas similares, independente das dimensões, tal como um quadrado de um retângulo. O problema do reconhecimento do *string* ganha em dificuldade ao tratar o caso de *strings* gerados nos casos práticos, ou seja, como padrões não ideais das formas e sujeitos a vários tipos de distorções que ocorrem devido aos erros residuais dos estágios preliminares de segmentação e pré-processamento.

D. Ruído e Distorção

Ruído e distorção causados pelo posicionamento relativo da câmera ou sensor são questões em qualquer sistema envolvendo imagens. Existe literatura técnica para a solução de cada tipo de distúrbio e para a redução de erros, minimizando a complexidade da etapa final de reconhecimento [9].

No caso de padrões rotacionados, Costa [8] comentou não existir possibilidade de classificá-los, pois as primitivas estarão deslocadas de um ângulo determinado. Uma solução relativamente simples desse efeito implica na rotação de todas as formas por um eixo comum da imagem antes de aplicar as primitivas. Com isto, todas as formas terão um eixo comum. O erro de paralaxe pode ser minimizado pela rotação da imagem em relação a algum eixo perpendicular ao plano da lente da câmera, preliminarmente à aplicação das primitivas.

Há situações caracterizadas por freqüentes alterações dos parâmetros indicativos da orientação dos sensores, tal como descrito em [10]. Nesse caso, soluções encontradas buscam integrar modelos já existentes formando módulos, levando em conta as necessidades dos usuários, dados reais disponíveis, particularidades dos sensores e os requisitos de precisão e tolerância a fim de obter uma solução ótima.

De modo geral, os algoritmos de classificação têm que reconhecer *strings* representando padrões não ideais, ou seja, formas com pequenas diferenças das formas corretas. Requisitos como esses geram as principais restrições para uso dos procedimentos sintáticos tradicionais em classificação.

IV. O MÉTODO

Foram identificadas três estruturas computacionais principais tendendo a se repetir em qualquer reconhecedor, denominadas pilha (*stack*), laço (*loop*) e ponteiro (*pointer*), que facilitam a modelagem do autômato. Considerando-se uma única forma, um modelo básico contempla as seguintes etapas:

1) De acordo com a forma, elaborar o modelo topológico do reconhecedor introduzindo as estruturas computacionais (laços, pilhas e ponteiros) conforme necessário.
2) Projetar a máquina não-adaptativa.
3) Dividir o projeto em unidades de acordo com as estruturas computacionais e projetar as funções adaptativas para acesso às estruturas.
4) Integrar as funções adaptativas a uma ou mais cópias da máquina não-adaptativa para atuação com cada estrutura.
5) Teste das unidades.
6) Integração geral das unidades e teste global.
7) Documentar o processo.

Sendo possível o projeto de um autômato para cada forma, torna-se viável a criação de banco de autômatos para reconhecer várias modalidades de formas.

As funções adaptativas que manipulam pilhas costumam ser simples e normalmente de duas espécies. Uma espécie é de incremento da pilha em que a função gera uma nova transição e um novo estado que passam a ser os últimos da seqüência. A segunda espécie é de redução da pilha, de modo que a última transição é removida e o penúltimo estado passa a ser o último. Quando as pilhas são construídas entre dois estados fixos, as funções adaptativas correspondentes são similares às descritas, com poucas variações.

Para reconhecer formas quadradas, sem considerar distorções, à medida que o autômato consome a seqüência dos *tokens* relativos ao primeiro lado, o AA deve gerar três grupos

de transições unidas, agrupadas em três pilhas, apresentando número igual de transições ao *token* lido (no caso *token a*). A forma estará correta se o autômato consumir essas três pilhas com sucesso, quando do recebimento dos símbolos dos demais lados.

A Figura 3 apresenta exemplos de pilhas construídas pelo autômato durante seu processo adaptativo. A primeira pilha é formada entre os estados 101 e 102 e duas pilhas interligadas serão construídas entre os estados 301 e 302 e entre os estados 401 e 402.

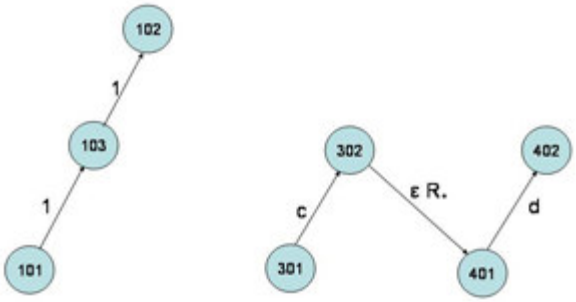


Fig. 3. Três pilhas construídas pelo AA.

Os laços são entendidos como um caso particular de pilha formando grupo de transições unidas que começam e terminam no mesmo estado, convenientes para armazenamento de dados ou instruções a serem acessadas ciclicamente. A Figura 4 mostra um laço em torno do estado 302. Os ponteiros são comumente usados pelo AA para acessar dados ou instruções existente nos laços e pilhas. A Figura 5 mostra exemplos de ponteiros.

De acordo com as formas a serem reconhecidas, pode-se iniciar o projeto do classificador a partir de uma descrição topológica do autômato, possibilitando estimativas de memória utilizada e dimensões do autômato. Esse procedimento é análogo com o seguido pela engenharia de *software* em que cada *design pattern* [11] apresenta uma solução para um determinado tipo de problema recorrente.

V. Implementação Efetuada

A título de exemplo, foi implementado um reconhecedor de quadrados e retângulos em que a característica adaptativa constrói dinamicamente uma função linear (concretizada através de laços) relacionando a quantidade de estados auxiliares gerados em pilhas às tolerâncias das dimensões das formas. Conhecimentos estruturais das formas estão implícitos nas funções adaptativas que constroem as pilhas.

Não considerando distorções, qualquer forma associada a um *string* do tipo $a^n b^n c^n d^n$ deve ser classificado como quadrado. O classificador tem que diferenciar essa linguagem dos quadrados da linguagem $a^n b^m c^n d^m$ $m \neq n$, referente aos retângulos.

Entretanto, levando em conta a existência de distorções, as suposições abaixo são pertinentes:

1) A situação prática operacional implica em formas apresentando lados maiores que um determinado limiar k (número inteiro) de unidades de primitivas, correspondendo à região de distorção e ruído das etapas de pré-processamento e de segmentação.
2) O processo de reconhecimento iniciando pelo menor lado da forma.

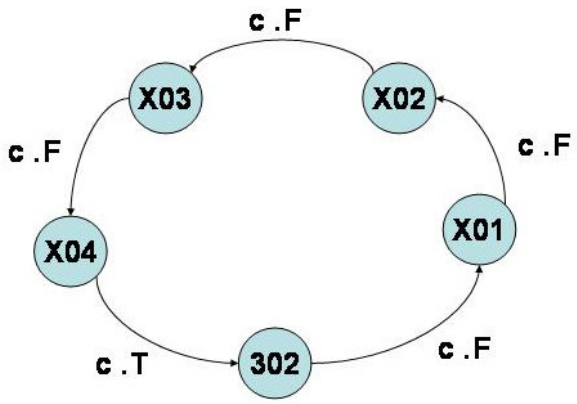


Fig. 4. Laço que inicia e termina no estado 302.

Assim, qualquer forma representada por linguagem do tipo $a^n b^m c^l d^u$ será aceita desde que satisfeitas as condições $n-k \leq l \leq n+k$ e $m-k \leq u \leq m+k$. Adicionalmente, a forma será classificada como quadrado quando atendida a condição seguinte $n-k \leq l,m,u \leq n+k$. Tendo o *string* atendido todas as condições, trata-se de um quadrado. Se apenas as duas primeiras condições forem satisfeitas, a forma é retangular. Este é apenas um exemplo em que retângulos e quadrados são identificados em um único processo adaptativo, passível de variações nos seus detalhes de acordo com as especificações.

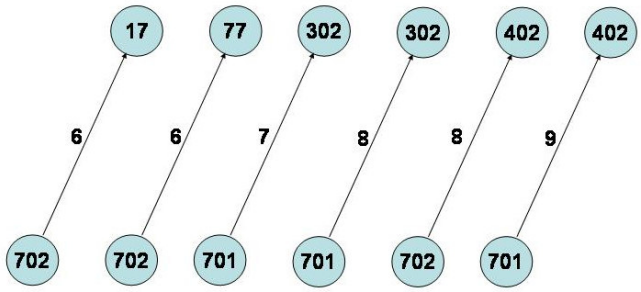


Fig. 5. Exemplo de seis ponteiros.

VI. Resumo dos Testes

A fim de validar o classificador, a tabela I indica alguns *strings* de teste, associados às respostas do autômato.

De acordo com a metodologia do experimento, os referidos *strings* correspondem a formas distorcidas, simulando os estágios de RP anteriores ao classificador. O teste procura varrer a faixa de aceitação do classificador variando as dimensões dos lados das formas conforme o expoente de cada símbolo dos *strings*. A coluna Resposta do AA mostra que o autômato reconhece os padrões adequadamente mesmo havendo aumento ou diminuição de qualquer lado da forma, até uma dimensão limite, a partir da qual as seqüências passam

a ser rejeitadas. Havendo aceitação do *string*, o estado final 7 indica o retângulo e o estado final 10, o quadrado. A rejeição de qualquer *string* corresponde aos casos em que a forma é desconhecida.

TABELA I
STRING DE ENTRADA E CORRESPONDENTE ESTADO FINAL E RESPOSTA DO AA

***STRING* DE ENTRADA**	**ESTADO FINAL**	**RESPOSTA DO AA**
$a^{10}b^{15}c^{13}d^{16}$	**7**	**ACEITO**
$a^{10}b^{15}c^{13}d^{18}$	**7**	**ACEITO**
$a^{10}b^{15}c^{13}d^{20}$		**REJEITADO**
$a^{10}b^{12}c^{12}d^{14}$	**10**	**ACEITO**
$a^{10}b^{12}c^{10}d^{14}$	**10**	**ACEITO**
$a^{10}b^{12}c^{9}d^{14}$	**10**	**ACEITO**
$a^{10}b^{12}c^{8}d^{14}$	**10**	**ACEITO**
$a^{10}b^{12}c^{7}d^{14}$		**REJEITADO**
$a^{10}b^{12}c^{9}d^{9}$		**REJEITADO**
$a^{10}b^{12}c^{9}d^{10}$	**10**	**ACEITO**

Sob o enfoque de ferramental de *software*, destaque-se a simplicidade para o operador dentre as principais funcionalidades desta implementação. Basta o usuário introduzir os quatro expoentes do *string* via teclado do microcomputador para obter na tela as respostas aceito ou rejeitado e estado final. Quanto a aspectos de simulação de condições variadas de ruído bem como de interface com o sistema de RP, o autômato deve incorporar o respectivo limiar de distorção e ruído, especificado em número de primitivas. Basicamente, tal atividade é efetuada alterando-se a quantidade de estados em estrutura *loop* da máquina não-adaptativa, mantendo-se as funções adaptativas protegidas de acesso e aptas a atuarem em novas condições conforme as máquinas não-adaptativas.

VII. CONCLUSÃO

Através dos resultados dos experimentos, constatou-se que o método proposto pode ser utilizado como referência para outros classificadores a serem desenvolvidos, auxiliando também na modelagem de problemas recorrentes em reconhecimento de formas.

Apesar de haver uma alteração de paradigma com os AAs, as funções adaptativas se mostraram sem grandes dificuldades, sendo a simplicidade e facilidade de implementação duas das principais vantagens constatadas com a proposta.

Os AAs preservam a característica positiva da abordagem sintática de favorecer o tratamento de padrões estruturais além de reconhecer as formas em qualquer escala. O uso de técnicas costumeiras exigiriam autômatos finitos grandes amarrados a uma única dimensão da forma.

Devido a ser um assunto relativamente novo, não se tem encontrado trabalhos em reconhecimento de padrões utilizando AA, em particular envolvendo *strings* sujeitos a ruído. Extensão do trabalho será voltada a potenciais aplicações do método e respectivas técnicas adaptativas.

REFERÊNCIAS

[1] Rodrigo de O. Plotze, Odemir M. Bruno. Análise de Formas e Reconhecimento de Padrões por Meio da Assinatura Fractal Multiescala. Journal of Computer Science, University of Lavras (Brazil),VOL.6, N.2, June, 2007.

[2] Wu , W-Y; Two-dimensional Object Recognition through String Matching. Image Science Journal vol. 49, September 2001.

[3] Fu, K. S. Syntactic Methods in Pattern Recognition. New York: Academic Press, 1974.

[4] Ritter, G. X.; Wilson, J. N. Handbook of computer vision and algorithms in image algebra. 2nd. ed. 2000.

[5] A. Apostólico; Z. Galil. Pattern Matching Algorithms. Oxford University Press, New York, 1997.

[6] Schalkoff, R. J. Pattern Recognition: statistical, structural and neural approaches. John Wiley & Sons, Inc., 1992.

[7] Rocha, R. L. de Azevedo; Neto, J. J. Autômato adaptativo, limites e complexidade em comparação com máquina de Turing. Proceedings of the second Congress of Logic Applied to Technology - LAPTEC'2000, São Paulo, Brasil, 2001.

[8] Costa, E. R.; Hirakawa, A. R.; Neto, J. J. An Adaptive Alternative for Syntactic Pattern Recognition. Proceeding of 3rd International Symposium on Robotics and Automation, Toluca, Mexico, 2002.

[9] Javidi, Bahram; Okano, Fumio. Three-Dimensional Television, Video, and Display Technologies, Springer-Verlag, 2002.

[10] Michael Breuer, Jörg. Geometric Correction of Airborne Whiskbroom Scanner Imagery Using Hybrid Auxiliary Data, 19th ISPRS congress, Amsterdam, 2000.

[11] Gamma, E., Helm, R., Johson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995.

Leoncio C. Barros Neto. Nascido em Campo Grande, MS, em 14 de setembro de 1954. Recebeu os títulos de Engenheiro Eletricista pela Escola de Engenharia Mauá, de São Caetano do Sul, SP, em 1979. Em 1994 recebeu o título de Mestre em Engenharia Elétrica da Universidade de São Paulo. Em 1981, após atuação em Telecomunicações e Sistemas de Controle de Processos em empresas como Siemens e Duratex, foi aprovado em concurso público de âmbito nacional, sendo integrado ao Corpo de Engenheiros Navais da Marinha do Brasil (MB). Na MB, atingiu o mais alto posto de Oficial Superior, exercendo atividades técnicas em Setores de Manutenção e Pesquisa. Especializou-se através de cursos na Alemanha e Inglaterra sobre sistemas eletrônicos aplicados em segurança e defesa.

Após transferência para a reserva, iniciou programa de doutorado no Departamento de Engenharia de Computação e Sistemas Digitais da USP em 2006, sob orientação do Prof. Dr. Massola, passando também a exercer atividades profissionais de consultoria. Suas áreas de interesse e pesquisa abrangem Automação, sensores inteligentes, eletrônica de potência, sistemas embarcados, Sistemas de Detecção, Guerra Eletrônica e suas aplicações.

André R. Hirakawa. Nascido em São Paulo, Brasil, em 15 de junho de 1965. Recebeu os títulos de Engenheiro Eletricista e correspondente Mestrado da Universidade de São Paulo (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 Assistente. 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 e orientador do trabalho de doutorado do primeiro autor.

Tabela de Decisão Adaptativa: Simulação de um Autômato Adaptativo

A. H. Tchemra

Resumo. **Este artigo tem como objetivo apresentar o software TDA baseado na Tabela de Decisão Adaptativa. O seu desenvolvimento e aplicação simulam um Autômato Adaptativo como um dispositivo reconhecedor de linguagens livres de contexto.**

Palavras-chave: **tecnologia adaptativa, tabela de decisão adaptativa, autômato adaptativo.**

I. INTRODUÇÃO

O software TDA foi um trabalho apresentado na disciplina PCS5004 - Fundamentos e Aplicações da Tecnologia Adaptativa, no ano de 2004. A proposta do trabalho consistia em criar um ambiente de desenvolvimento de aplicações baseadas em Tabelas de Decisão Adaptativas.

O projeto do software TDA teve como alicerce o formalismo do dispositivo dirigido por regras adaptativo definido em [1]. Para mostrar sua aplicabilidade, foi simulado um autômato adaptativo [2] para reconhecer cadeias de entrada.

II. TABELA DE DECISÃO ADAPTATIVA

Neto define um dispositivo dirigido por regras adaptativo por M = (ND, AM), onde ND é um dispositivo dirigido por regras (não-adaptativo) e AM é um mecanismo adaptativo [1].

O dispositivo não-adaptativo ND é definido por:

ND = (C, NR, S, c_0, A, NA) onde:

- C é o conjunto de todas as configurações possíveis do dispositivo;
- NR $\subseteq$ CxSxCxNA é o conjunto de regras;
- S é o conjunto finito de eventos possíveis para entradas válidas para ND, com ε (vazio) $\in$ S;
- $c_0 \in$ C é a configuração inicial do dispositivo;
- A $\subseteq$ C é o sub-conjunto de configurações aceitas pelo dispositivo;
- NA é o conjunto finito, com ε $\in$ NA, de todos os símbolos possíveis de saída de ND, como resposta à aplicação de uma das regras NR;
- cada regra r $\in$ NR é da forma r = (c_i, s, c_j, z) , isto é, em resposta a uma entrada s$\in$ S, r muda a configuração corrente c_i para c_j , consome o símbolo s e produz z$\in$ NA como saída;
- w = $w_1w_2...w_n$ é uma cadeia de entrada, onde $w_k \in$ S-{ε}, para $1 \leq k \leq n$.

Assim, uma cadeia de entrada w é aceita por ND quando parte de uma configuração inicial c_0 e, após a execução de uma seqüência de configurações que consome todos os símbolos da cadeia, atinge a configuração final c:

($c_0 \Rightarrow^w$ c, com c$\in$ A).

O mecanismo adaptativo AM é definido por:

AM $\subseteq$ BAxNRxAA, onde BA e AA são conjuntos de ações adaptativas a serem executadas, respectivamente, antes e depois das regras não-adaptativas; ambos contém a ação nula ε (ε$\in$ BA$\cap$AA).

Denota-se por AR_k o conjunto de regras adaptativas, dada pela relação: $AR_k \subseteq$BAxCxSxCxNAxAA.

Regras ar $\in$ AR_k são da forma: ar = (ba, c_i, s, c_j, z, aa), ou seja, em resposta a algum estimulo de entrada s$\in$ S, ar inicia a execução de uma ação adaptativa ba$\in$ BA, se houver, antes da regra não-adaptativa r = (c_i, s, c_j, z), ou executa a ação aa$\in$ AA depois da aplicação da regra r.

Este mecanismo adaptativo AM pode ser aplicado, numa configuração k do dispositivo, para cada regra em $NR_k \subseteq$ NR, e determina as regras adaptativas AR_k que devem ser executadas naquela regra. A aplicação de AR_k leva a um novo conjunto de regras adaptativas AR_{k+1}, pela eliminação e/ou inclusão de regras, modificando a estrutura não-adaptativa.

Como exemplo de dispositivo dirigido por regras adaptativo, [1] apresenta a Tabela de Decisão Adaptativa (TDA). Ela é composta por duas camadas, onde é preservada na primeira camada uma tabela de decisão convencional, que representa um dispositivo dirigido por regras não-adaptativo, e a segunda, denominada camada adaptativa, onde são definidas as ações adaptativas. Uma TDA é definida por:

T = (ND, AM), onde ND representa a tabela de decisão não-adaptativa e AM a camada adaptativa.

O formato da TDA apresentado em [1] mostra a camada subjacente na parte superior e as ações adaptativas na parte inferior da tabela. Colunas adicionais permitem que sejam declaradas as funções adaptativas. A figura 1 apresenta o formato da tabela.

Além dos elementos de uma tabela de decisão convencional, linhas de condições e ações, e colunas de regras, a figura 1 mostra a disposição da camada adaptativa que é descrita a seguir:

A. H. Tchemra – Universidade Presbiteriana Mackenzie (correspondência: R. Jorge Tibiriçá, 74 – ap.64 – São Paulo, SP, Brasil – cep: 04126-000; e-mail: angela.tchemra@poli.usp.br).

▪ uma linha de título (“Tag”) contém o tipo de cada coluna,

<table>
<tr><td colspan="2"></td><td colspan="2">número de cada regra</td></tr>
<tr><td colspan="2">Tag →</td><td colspan="2">tipo de cada coluna</td></tr>
<tr><td rowspan="3">tabela de decisão subjacente</td><td>linhas de condições</td><td rowspan="4">declaração das funções adaptativas</td><td rowspan="3">tabela de decisão não-adaptativa</td></tr>
<tr><td>linhas de ações</td></tr>
<tr><td>variáveis</td></tr>
<tr><td rowspan="2">funções adaptativas</td><td>nomes das funções</td><td rowspan="2">chamadas para as ações adaptativas</td></tr>
<tr><td>parâmetros, variáveis e geradores</td></tr>
</table>

Figura 1 – Tabela de Decisão Adaptativa

representado pelos caracteres: H ? + - S R E

▪ em cada coluna do tipo H:

H = cabeçalho da especificação da função adaptativa que deve conter:

- ▪ caracter A, para indicar se a função deve ser executada antes da regra não-adaptativa subjacente, ou o caracter B, se a função será executada depois da regra não-adaptativa subjacente;
- ▪ caracter P, para cada célula correspondente a um parâmetro formal;
- ▪ caracter V, para cada célula correspondente a uma variável;
- ▪ caracter G, para cada célula correspondente a um gerador (o gerador é global e único, e cria rótulos dinâmicamente);

▪ as colunas do tipo ? + - representam, respectivamente, as ações adaptativas elementares de consulta, inclusão e exclusão de regras na tabela de decisão subjacente; em cada coluna pode-se ter:

- ▪ células das condições preenchidas com valores das condições a serem testadas; células de ações a serem executadas devem ser marcadas; células das variáveis podem ser preenchidas com valores determinados;
- ▪ as chamadas para as ações adaptativas devem ser marcadas e os parâmetros correspondentes devem ser preenchidos, assim como as variáveis e os geradores da função correspondente à ação adaptativa;

▪ coluna do tipo S (“start”) indica a primeira regra a ser executada pelo autômato adaptativo, que deve conter:

- ▪ as condições e ações iniciais para que se inicie a operação do autômato;

▪ colunas do tipo R indicam colunas de regra normal, isto é, são todas as colunas de regras da tabela não-adaptativa, cujas células contêm:

- ▪ as condições, ações e valores das variáveis que devem ser testadas;
- ▪ células das funções adaptativas podem ser marcadas, indicando a chamada para as funções, com passagem dos parâmetros e geradores correspondentes;

▪ coluna do tipo E indica regra final e serve como delimitador do conjunto de regras.

A operação da TDA inicia na tabela de decisão não-adaptativa subjacente para buscar uma regra que satisfaça as condições de uma situação estabelecida. Se a regra encontrada for uma regra não-adaptativa, ela é executada. Caso a regra selecionada tenha na camada adaptativa uma chamada a uma função adaptativa, as ações adaptativas são executadas, de acordo com o tipo da função. As ações adaptativas são executadas antes da regra não-adaptativa ou depois da regra, quando a função for do tipo A (antes) ou B (depois).

Em seguida, é apresentado o software que foi desenvolvido para executar o mecanismo descrito da TDA.

III. SOFTWARE TDA

O software TDA foi desenvolvido em C++ Builder (versão 6.0). O ambiente computacional permite que o usuário implemente tabelas de decisão adaptativas para a execução de aplicações diversas. Cabe ressaltar que o ambiente foi desenvolvido com objetivos didáticos, podendo ser utilizado para apoiar no processo de ensino e aprendizagem dos conceitos sobre tabelas de decisão adaptativas.

A. Interface

A interface do software TDA é apresentada na figura 2. Sua interface permite ao usuário definir as especificações para a implementação de um projeto usando tabela de decisão adaptativa.

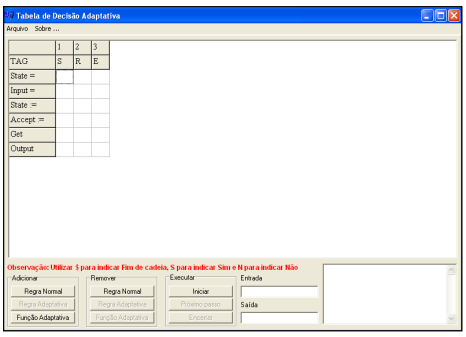


Figura 2 – Tela inicial do software TDA

B. Como adicionar ou remover regras e funções na TDA

Na parte inferior da tela principal do software encontra-se o quadro mostrado na figura 3.

Figura 3 - Quadro de botões Adicionar e Remover

No quadro **Adicionar** são encontrados os botões:

1) **Regra Normal** – adiciona uma regra do tipo R na TDA: ao clicar o botão, uma nova coluna é adicionada à tabela, permitindo ao usuário digitar os dados da regra (observar que a regra S, a primeira regra normal, deve ter apenas o estado inicial e o **get** do primeiro símbolo);

2) **Regra Adaptativa** – adiciona regra adaptativa de uma função adaptativa: ao clicar o botão, a janela mostrada na figura 4 é aberta:

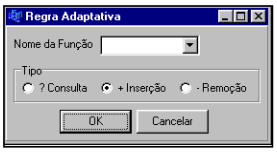


Figura 4 – Janela para definir uma nova Regra Adaptativa

▪ selecionar a função adaptativa, cuja regra será acrescentada à tabela;
▪ definir o tipo da regra (consulta, inserção ou remoção);
▪ em seguida, definir os dados da regra adaptativa na coluna criada na tabela.

3) **Função Adaptativa** – adiciona uma função adaptativa
▪ ao clicar o botão, uma janela como a apresentada na figura 5 é aberta, onde o usuário pode digitar:

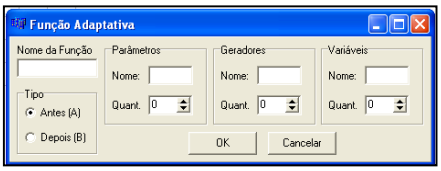


Figura 5 – Janela para definir uma Função Adaptativa

▪ o nome da função;
▪ escolher o tipo da função anterior ou posterior;
▪ o nome dos parâmetros – o usuário deve digitar apenas a letra inicial do nome do parâmetro;
▪ a quantidade de parâmetros – indicar quantos parâmetros a função adaptativa terá;
▪ o nome dos geradores – como no nome dos parâmetros, o usuário deve digitar apenas a letra inicial do nome do gerador;
▪ a quantidade de geradores – indicar quantos geradores a função terá;
▪ o nome das variáveis – deve-se digitar apenas a letra inicial da variável;
▪ a quantidade de variáveis – indicar a quantidade de variáveis da função;

Os parâmetros, geradores e variáveis serão criados de acordo com as quantidades indicadas e nomeados em seqüência, por exemplo: para 3 parâmetros, 2 geradores e 2 variáveis, e sendo **p** a letra inicial do nome do parâmetro, **g** a letra inicial do nome do gerador e **v** a letra inicial do nome da variável, serão criados os parâmetros **p1**, **p2** e **p3,** os geradores **g1*** e **g2*** e as variáveis **v1?** e **v2?**.

No quadro **Remover** são encontrados os botões:

4) **Regra Normal** – remove uma regra do tipo R da TDA, escolhida pelo seu número (figura 6):

Figura 6 – Janela para remover uma regra normal

5) **Regra Adaptativa** – remove uma regra adaptativa de uma função adaptativa, indicada pelo seu número (figura 7):

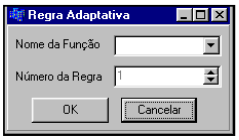


Figura 7 – Janela para remover uma regra adaptativa

6) **Função Adaptativa** – remove a função adaptativa selecionada (figura 8):

Figura 8 – Janela para remover uma função adaptativa

C. *Como salvar o projeto*

Após preencher o conjunto de regras normais e as funções adaptativas, o usuário pode salvar a TDA em um arquivo de extensão **.tda**, para uso posterior.

D. *Gerenciamento de arquivos e projetos*

Na tela principal do software há a barra de menu, mostrada na figura 9:

Figura 9 – Barra de menu

As opções do menu representam:

1) opção **Arquivo –** permite o gerenciamento dos arquivos e do projeto através das opções:
▪ **Nova TDA** – inicia uma nova tabela para a definição de um novo projeto
▪ **Abrir TDA ...** – carrega uma TDA já definida, isto é, um projeto específico para ser executado
▪ **Salvar TDA como ...** – salva a TDA de um novo projeto ou uma TDA executada
▪ **Abrir Histórico ...** – carrega um arquivo com o histórico da execução de uma TDA, permitindo ao usuário analisar todos os passos executados
▪ **Salvar Histórico como ...** – salva todos os passos executados na TDA
▪ **Sair** – encerra a execução do software

2) opção **Sobre** – informa versão e autores do software.

Os arquivos do tipo Tabelas de Decisão Adaptativas (TDA) são identificados com a extensão **.tda** e os arquivos que contém os históricos, pela extensão **.txt**.

Deve-se observar que é possível salvar a TDA após sua execução, porém deve-se tomar o cuidado para não sobrepor o arquivo original, com o prejuízo de não se ter acesso novamente a ele.

Para executar dois testes em uma TDA, executar o primeiro teste e, em seguida, abrir novamente o arquivo para eliminar as modificações feitas pelo primeiro teste, e executar o segundo teste. É possível, porém, executar dois testes consecutivos, onde o segundo utiliza a TDA já modificada pelo primeiro teste.

E. *Como executar o software*

Após carregar a TDA, o usuário deve digitar a cadeia de entrada na caixa posicionada na parte inferior da tela principal.

Observar que para finalizar a entrada da cadeia, o usuário deve digitar o símbolo "$". Por exemplo: aabbcc$

Na parte inferior da tela principal do software encontra-se o quadro que contém os botões:

1) Botão **Iniciar** – permite o início da execução da TDA

2) Botão **Próximo passo** – permite o acompanhamento passo a passo da operação do software e visualização dos dados e modificação da estrutura da tabela

3) Botão **Encerrar** – permite que a execução da TDA seja finalizada

F. Algoritmo

Segue a descrição geral do algoritmo implementado:

1) estando na configuração inicial da tabela, inicia na regra S;

2) **enquanto** houver símbolo a ser lido na cadeia de entrada **faça**:

2.1) procura regra aplicável:

se não existir nenhuma regra aplicável, encerra a execução e a cadeia é rejeitada;

2.2) se existir função adaptativa anterior na regra aplicável:

2.2.1) executa a função adaptativa

2.2.2) se a regra aplicável continua sendo a mesma, no caso de não ter sido removida, segue para o passo 2.3, senão volta para o passo 2.1;

2.3) aplica a regra;

2.4) executa a ação semântica;

2.5) executa a função adaptativa posterior, se existir;

2.6) verifica Aceitação/Rejeição;

2.7) **fim enquanto**

G. Observações importantes

Durante a execução do software deve-se observar:

- a aplicação de regras não é feita em paralelo, apenas a primeira regra aplicável é executada;
- na mesma regra pode-se ter as duas funções adaptativas anterior e posterior;
- as regras eliminadas continuam aparecendo na tabela, mas com um "X" marcando as colunas; elas não são consideradas na hora de procurar uma regra aplicável;
- as colunas de cabeçalho (H) ficam travadas para que o usuário não possa alterá-las (são apenas para leitura);
- #i# significa o símbolo de entrada corrente;
- #a# significa o símbolo de entrada anterior.

H. Aplicações

Algumas aplicações estão disponíveis para o usuário realizar, como os exemplos:

1) **Letra seguida de letra ou dígito**

Aplicação para reconhecer uma cadeia de símbolos de entrada de uma linguagem natural. O teste é baseado no exemplo apresentado em [1]. O arquivo ExemploAula1.tda contém a tabela de decisão adaptativa inicial e o teste pode ser realizado para reconhecer a cadeia "ll$".

2) **Aplicação para reconhecer cadeias $a^n b^n c^n$**

Abrir o arquivo AnBnCn.tda que traz a configuração inicial da TDA, podendo ser feito o teste: "aaabbbccc$".

3) **Aplicação para reconhecer anagramas**

Arquivo: Anagrama_abc.tda – para reconhecer todos os anagramas de abc, ou seja, abc, cba, acb, bca, cab e bac.

4) **Aplicação para reconhecer merge de cadeias**

Arquivo: Merge.tda – para reconhecer as cadeias "abcd" e "efgh" intercaladas, por exemplo: "aebfcgdh".

V. CONCLUSÃO

O software TDA foi desenvolvido de tal maneira que suas especificações atendessem requisitos elementares como estar em concordância com a teoria e os mecanismos de funcionamento das tabelas de decisão adaptativas. A notação e o formato utilizados no software procuraram seguir os padrões encontrados na literatura, tornando-o amigável e de fácil utilização por usuários familiarizados com o tema.

Devido à simplicidade da ferramenta, pode ser utilizado tanto por um usuário final de tecnologia adaptativa, não necessariamente especialista em programação, como por aquele que está iniciando os estudos na área. A arquitetura do software é aberta e por ser de caráter genérico, possibilitam a implementação de outros projetos com tabelas de decisão adaptativas.

VI. CONSIDERAÇÕES FINAIS

O software TDA desenvolvido neste trabalho, como foi visto, oferece amplas possibilidades de aplicações baseadas em tabelas de decisão adaptativas. Por sua simplicidade, pode ser útil como ferramenta educacional no processo de ensino-aprendizagem de dispositivos dirigidos por regras adaptativos.

Para dar continuidade a este trabalho e aperfeiçoar o projeto, algumas sugestões para trabalhos futuros são apresentadas:

- inserir novas ações semânticas, alterando o código fonte;
- criar um mecanismo para inserção de novas condições, da mesma forma como foi feito para a inserção de novas ações semânticas, para tornar viáveis implementações de projetos que possuem número variável de entradas (input);
- aplicar regras em paralelo;
- introduzir uma pilha no mecanismo (necessária em alguns casos).

REFERÊNCIAS BIBLIOGRÁFICAS

[1] 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.

[2] NETO, João José. *Introdução à Compilação*. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 1987.

Angela Hum Tchemra é bacharel e licenciada em Matemática pela Universidade Presbiteriana Mackenzie e mestre em Administração de Empresas pela mesma universidade. Atualmente é professora de cursos de graduação na Universidade Presbiteriana Mackenzie, Faculdade de Engenharia São Paulo e Fundação Armando Álvares Penteado. É Doutoranda do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

SPA^3: Um *framework* para privacidade

Cereda, P. R. M. e Zorzo, S. D.

***Resumo*—Este artigo apresenta um *framework* para privacidade, chamado *SPA*^{3}, que contribui para a construção simplificada de sistemas, com reforços nos aspectos de privacidade. O *SPA*^{3} utiliza comandos de privacidade, que são reutilizáveis, para a especificação de situações de violação de privacidade. O *framework* apresentado é uma solução viável para incorporar reforços de privacidade nos sistemas.**

***Palavras-chave*—Controle de acesso, Privacidade, Autômato Adaptativo.**

I. INTRODUÇÃO

A privacidade está seriamente comprometida na sociedade de informação global, pois o tráfego gerado em uma rede global transcende os limites territoriais, e não é gerenciado de modo centralizado. Com isso, surgem os riscos de invasão de privacidade, uma vez que dados dos usuários podem ser interceptados ou monitorados. Assim, buscou-se uma padronização de ações consideradas intrusivas que são distintas a uma determinada região ou país para formulação de leis e estabelecendo penas às situações de violação.

Fernandes [1] afirma que "para ter privacidade, uma pessoa precisa ter controle sobre as informações existentes sobre si mesma e exercer este controle de forma consistente com seus interesses e valores pessoais". Assim, durante a interação de um usuário em um determinado sistema, suas informações devem ser divulgadas apenas com o consentimento do mesmo.

Mecanismos de segurança são utilizados para representar, de modo objetivo e simplificado, os requisitos de segurança de um determinado sistema. Entretanto, ao mesmo tempo que exercem um controle sobre os dados de usuários que trafegam no sistema, podem invadir a privacidade de tais usuários, conforme uma pesquisa feita por [2] [3].

Este artigo apresenta um *framework* para privacidade chamado *SPA*^{3} (Sistema de Privacidade Auditável com Autômato Adaptativo), que objetiva simplificar a especificação dos aspectos de privacidade de um determinado sistema. Os comandos de privacidade fornecidos pelo *framework* verificam se uma determinada situação caracteriza invasão de privacidade, de acordo com um conjunto de leis específicas.

Na seção II é apresentada a motivação para o desenvolvimento do *framework* a partir do Sistema de Privacidade Auditável e do Modelo de Controle de Acesso Adaptativo. Na seção III, o *framework* é apresentado, com enfoque em suas características. As considerações finais são apresentadas na seção IV.

Paulo Roberto Massa Cereda e Sérgio Donizetti Zorzo.

Este trabalho foi financiado pela Capes.

Os autores são do Departamento de Computação, Universidade Federal de São Carlos, Rod. Washington Luís, Km 235, Caixa Postal 676, 13565-905, (e-mails: paulo_cereda@dc.ufscar.br e zorzo@dc.ufscar.br).

II. MOTIVAÇÃO

Nesta seção, os tópicos que motivaram a criação do *framework* são apresentados, como Sistema de Privacidade Auditável, que gerencia informações entre domínios, e o Modelo de Controle de Acesso Adaptativo, que utiliza um autômato adaptativo para prover o controle de acesso.

A. Sistema de Privacidade Auditável

Um *Sistema de Privacidade Auditável* [4] é definido como um sistema que permite o gerenciamento de objetos entre domínios e a realização de auditorias em tempo de execução e *post-hoc*.

O gerenciamento de objetos entre domínios é suportado pelos seguintes eventos: transferência, ação, criação, estabelecimento de direitos, notificação e *logging*. O evento de transferência consiste em transferir um objeto, copiando-o e passando a cópia ao domínio do receptor. Na ação, usa-se um objeto privado para um determinado propósito, enquanto que no evento de criação permite-se realizar a adição de novos objetos ou indivíduos ao sistema e à matriz de acesso. No estabelecimento de direitos, um indivíduo ou dono de um dado objeto pode atribuir ou revogar permissões. O evento de notificação informa o indivíduo acerca de ações realizadas por outros indivíduos no sistema, e no *logging* o sistema monitora todos os eventos – os que foram bem-sucedidos e os que falharam (é usado para verificação e auditoria).

Um Sistema de Privacidade Auditável necessita de um mecanismo de controle de acesso para atender seus requisitos. O modelo HRU [5] , proposto por Harrison, Ruzzo e Ullman, é um modelo de controle de acesso que foi utilizado no Sistema de Privacidade Auditável proposto por May, Gunter e Lee [4], através de seis operações primitivas presentes no modelo, com a adição de duas novas operações que realizam a notificação (**inform** *s* **of** *t*) e *logging* (**log** *t*). Alterações no modelo HRU foram necessárias para que este implementasse totalmente as características de um Sistema de Privacidade Auditável [4].

A utilização do modelo HRU estendido em um Sistema de Privacidade Auditável justifica-se por sua simplicidade de uso. Entretanto, outros modelos de controle de acesso poderiam ser utilizados [4], como o Modelo de Controle de Acesso Adaptativo, apresentado a seguir.

B. Modelo de Controle de Acesso Adaptativo

Para atender as necessidades do Modelo de Controle de

Acesso Adaptativo, definiu-se uma extensão do autômato adaptativo [6] [7] [8], chamado de *autômato adaptativo de controle de acesso*, apresentado a seguir.

O *Autômato Adaptativo de Controle de Acesso* é uma extensão do Autômato Adaptativo, que permite que um símbolo da cadeia de entrada seja passado como parâmetro a uma dada função adaptativa.

Um *autômato adaptativo de controle de acesso* M é definido por $M = (Q, S, \Sigma, \Gamma, P, q_0, Z_0, F)$, tal que Q é o conjunto finito de estados, $Q \subset Q^A$, Q^A é o conjunto de todos os estados possíveis, Q^A é enumerável, S é o conjunto de sub-máquinas, Σ é o alfabeto finito de entrada, $\Sigma \subset \Sigma^A$, Σ^A é o alfabeto enumerável de todos os símbolos possíveis, Γ é o alfabeto finito da pilha, $\Gamma \subset \Gamma^A$, $\Gamma = Q \cup \{Z_0\}$, Γ^A é o alfabeto enumerável de todos os símbolos possíveis da pilha, $\Gamma^A = Q^A \cup \{Z_0\}$, P é uma função $P: Q^A \times \Sigma^A \times (\Sigma^A \cup \{\epsilon\}) \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$, H^0 definido a seguir, $q_0 \in Q$ é o estado inicial, $Z_0 \in \Gamma$ é o símbolo inicial da pilha, $F \subset Q$ é o conjunto de estados finais.

H^0 é definido como $H^0 = \{f \mid f : E \rightarrow E\}$. Foi utilizado $g: X \rightarrow Y$ para uma relação g tal que $g \subseteq X \times Y$ e se $g(x, y)$ e $g(x, z)$, $x \in X$, $y, z \in Y$, então $y = z$. Como abreviação, usa-se $g(x) = y$ para $g(x, y)$.

Define-se $E = \{N : N$ é um autômato adaptativo de controle de acesso $N = (Q', \Sigma', \Gamma', P', q_0, Z_0, F)$, onde $Q' \subset Q^A$, $\Sigma' \subset \Sigma^A$, $\Gamma' \subset \Gamma^A$, $P': Q^A \times \Sigma^A \times (\Sigma^A \cup \{\epsilon\}) \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$. Observe que q_0, Z_0 e F são os mesmos em qualquer $N \in E$.

O conjunto de todas as sub-máquinas do autômato adaptativo de controle de acesso é representado por S. Cada *sub-máquina* i é definida como $s_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$, onde $Q_i \subseteq Q$ é o conjunto de estados da sub-máquina i, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada da sub-máquina i, $P_i \subseteq P$ é o mapeamento da sub-máquina i, $q_{i0} \in Q_i$ é o estado de entrada da sub-máquina i, e $F_i \subseteq Q_i$ é o conjunto de estados finais da sub-máquina i.

Uma transição pode ser da forma $(q, a, \beta) \vdash (q', a', \beta', A, B)$ para $P(q, a, \beta) = (q', a', \beta', A, B)$, ou $(q, ab, \beta) \vdash (q', a', \beta', A, B(b))$ para $P(q, ab, \beta) = (q', a', \beta', A, B(b))$, A e B são funções adaptativas, $A \in H^0$, $B \in H^0$. Se $\tilde{q}$, $\tilde{a}$, $\tilde{b}$ ou $\tilde{\beta}$ não pertencem à máquina atual, então $P(\tilde{q}, \tilde{a}, \tilde{\beta}) = (\tilde{q}, \tilde{a}, \tilde{\beta}, I, I)$ ou $P(\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}) = (\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}, I, I)$, onde I é a função identidade em E.

A linguagem aceita por um autômato adaptativo de controle de acesso M é dada por $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \epsilon, Z_0)$, onde $q_f \in F\}$.

A especificação do *Modelo de Controle de Acesso Adaptativo* é descrita por um autômato adaptativo de controle de acesso M. O alfabeto do autômato adaptativo de controle de acesso é representado por Σ, $\Sigma \subset \Sigma^A$, $\Sigma^A = \{w \mid w$ é um nome de indivíduo válido $\} \cup \{x \mid x$ é um nome de objeto válido $\} \cup \{\lambda_x \mid x$ é um nome de objeto válido $\} \cup \{y \mid y$ é um direito válido $\} \cup \{\lambda_{NS}, \lambda_{NO}, \lambda_{RS}, \lambda_{RO}, \lambda_{RAO}, G, T\}$. Assim, o modelo de controle de acesso contém um conjunto finito de indivíduos atuais $S = \{s_1 \ldots s_{n_1}\}$, $S \subset \Sigma$; um conjunto finito de objetos atuais $O = \{o_1 \ldots o_{n_2}\}$, $O \subset \Sigma$; um conjunto finito de direitos $R = \{r_1 \ldots r_{n_3}\}$, $R \subset \Sigma$; um conjunto λ de símbolos que representam as ações sobre os objetos, $\lambda = \{\lambda_{o_1} \ldots \lambda_{o_{n_2}}\}$, $\lambda \subset \Sigma$; e um conjunto de símbolos que representam as alterações a serem realizadas na configuração do autômato, $\{\lambda_{NS}, \lambda_{NO}, \lambda_{RS}, \lambda_{RO}, \lambda_{RAO}, G, T\}$.

O Modelo de Controle de Acesso Adaptativo possui um conjunto C de *comandos de privacidade*, $C = \{c_1 \ldots c_i\}$, $i \in \mathbb{N}$. Esses comandos são codificações de regras-alvo obtidas a partir de uma legislação ou política de privacidade escrita em linguagem natural, e verificam se uma determinada situação pode ser caracterizada como violação de privacidade. A gramática da linguagem de um comando de privacidade é descrita a seguir.

```
Program ::= Valor Id "(" ParamList ")" "início"
            StatementList "fim"
Valor ::= "true" | "false"
ParamList ::= Id | Id "," ParamList
StatementList ::= | Statement StatementList
Statement ::= CondDecl | AtribDecl | ExecDecl |
              returnStat
CondDecl ::= "se" CondExp "então" "{"
             StatementList "\}" | "se" CondExp
             "então" "{" StatementList "}" "senão"
             "{" StatementList "}"
CondExp ::= CondCom | CondCom "e" CondExp
CondCom ::= "eval" "(" Arg "," Id ")" | "equiv"
            "(" Arg "," Id ")"
Arg ::= Id | "<" Id ">" | Id Arg | "<" Id ">" Arg
AtribDecl ::= Var "=" Var
Var ::= Id
ExecDecl ::= Var "=" "exec" "(" Arg "," Id ")"
```

```
returnStat ::= "retornar" Valor
```

Um comando de privacidade $c_i \in C$ é uma função que recebe como parâmetro um conjunto não-vazio de argumentos e retorna *"true"* se as condições presentes no corpo da função foram satisfeitas e se os comandos foram executados com sucesso, ou *"false"* caso contrário.

É definida uma variável de escopo global "e", que representa o autômato adaptativo de controle de acesso do modelo. Todos os comandos de privacidade podem referenciar-se ao autômato através da variável `"e".

A função $eval(u,e)$ recebe uma cadeia u e um autômato e, e retorna *"true"* se essa cadeia pertencer à linguagem descrita pelo autômato, $u \in L(e)$, ou *"false"* caso contrário.

A função $exec(u,e)$ recebe uma cadeia u e um autômato e, e retorna um autômato adaptativo. A cadeia u deve possuir algum dos símbolos que representam as alterações a serem realizadas na configuração do autômato. Ao executar a função $exec(u,e)$, espera-se que a cadeia altere a configuração do autômato; assim, o retorno da função será o autômato alterado, em caso de sucesso, ou o autômato original, caso a cadeia não pertença à linguagem descrita por e, $u \notin L(e)$, ou que u seja apenas uma cadeia de consulta.

A função $equiv(e,f)$ toma como argumentos os autômatos adaptativos e e f e verifica se são equivalentes pela inspeção de isomorfismo. Se e e f são isomorfos, a função $equiv(e,f)$ retorna*"true"*, ou *"false"* caso contrário.

As cadeias submetidas às funções são definidas através da concatenação de símbolos pré-definidos pertencentes ao alfabeto Σ e os argumentos do comando de privacidade $c_i, i \in \mathbb{N}$. De acordo com a gramática da linguagem do comando de privacidade, cada símbolo pré-definido $v_j \in \Sigma$ é representado na forma $< v_i >$, e os argumentos são representados através de seus nomes.

Através do Controle de Acesso Adaptativo, uma determinada legislação sobre privacidade pode ser formalizada em um conjunto de comandos de privacidade. A especificação formal de uma dada legislação possibilita o tratamento não ambíguo e obtenção de resultados que a teoria de autômatos adaptativos nos propicia, como tempo e custo de processamento, entre outros.

III. Framework

Este artigo descreve um *framework* para privacidade, chamado *SPA*[3] (Sistema de Privacidade Auditável com Autômato Adaptativo). O *SPA*[3] tem como objetivo simplificar a especificação dos aspectos de privacidade de um determinado sistema. O *framework* pode ser utilizado em vários domínios, através da definição da representação do Controle de Acesso pelo Sistema de Privacidade Auditável.

O autômato adaptativo de controle de acesso é representado no *framework* utilizando a linguagem de marcação *XML*[1]. O arquivo *XML* possui a configuração inicial do autômato adaptativo de controle de acesso a ser utilizado. As *tags* do arquivo *XML* representam as transições do autômato, que podem ser na forma $P(q,a,\beta) = (q',a',\beta',A,B)$ ou $P(q,ab,\beta) = (q',a',\beta',A,B(b))$. Um exemplo de arquivo *XML* pode ser visto a seguir.

```
<autômato>
...
    <transicao>
        <origem> q </origem}>
        <simbolo> a </simbolo>
        <topo> β </topo>
        <destino> q' </destino>
        <novosimbolo> a' </novosimbolo>
        <novotopo> β' </novotopo>
        <funcao1> A(...) </funcao1>
        <funcao2> B(...) </funcao2>
    </transicao>
...
</automato>
```

O acesso ao *framework* é feito através de chamadas aos comandos de privacidade do conjunto C. Esses comandos são escritos de acordo com a sintaxe apresentada na seção II. Ao executar um comando de privacidade, o *framework* retorna um valor lógico, *true* se o comando foi satisfeito, ou *false* caso contrário.

A construção do *framework* requer o arquivo *XML*, contendo o autômato adaptativo, e os comandos de privacidade, de acordo com a Figura 1.

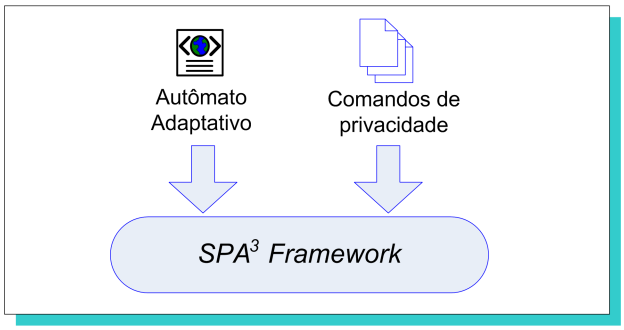


FIG. 1 Construção do *framework*.

O *SPA*[3] atua entre o usuário e o sistema, verificando se um determinado comando do sistema pode ser executado de acordo com o conjunto de comandos de privacidade definidos para o *framework*, conforme é ilustrado na Figura 2.

[1] *eXtensible Markup Language*

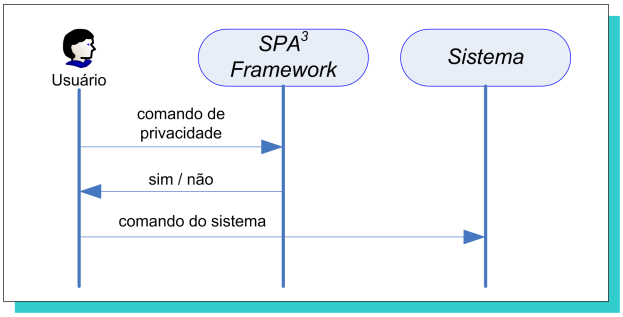


FIG. 2 Usuário, *framework* e sistema.

O *framework* fornece ao usuário um conjunto de comandos de privacidade (Figura 3). Os comandos verificam se uma determinada situação do sistema pode ser caracterizada como invasão de privacidade, de acordo com essa legislação.

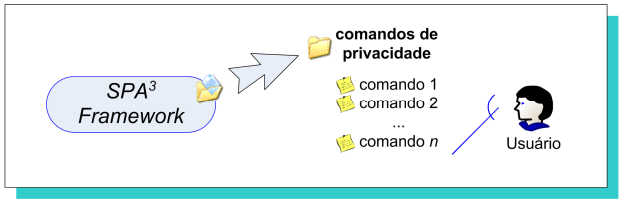


FIG. 3 Conjunto de comandos de privacidade disponíveis ao usuário.

Como exemplo, considere uma situação onde, em um determinado sistema, um usuário só poderá copiar um determinado arquivo para outro usuário, se este primeiro for o dono do arquivo a ser copiado, e o arquivo copiado deverá ser somente leitura (*read-only*). O comando de privacidade referente a essa situação pode ser codificado como a seguir (*e* é o autômato adaptativo).

CopiarArquivo(*a, // usuário dono do arquivo*
b, // usuário que receberá a cópia
c) // arquivo a ser copiado
início
// verifica se o usuário a é dono do arquivo c
// "o" representa "owner"
se *eval({a}{c} "o", e)* **então** {
// cria-se o arquivo c associado ao usuário c
tmp = exec({b} λ_{NO} *{c}, e)*
// se os autômatos são iguais, não foi possível
// criar o arquivo c para o usuário b
se *equals(tmp, e)* **então retornar** *false*
tmp = exec({b} $\lambda_{\{c\}}$ *"Gr", e)*
// se os autômatos são iguais, não foi possível
// inserir o direito de leitura r no arquivo c
// do usuário b
se *equals(tmp, e)* **então retornar** *false*
// o comando foi executado com sucesso
retornar *true*
}
senão {
// o usuário a não é dono do arquivo c
retornar *false*
}
fim

O comando **CopiarArquivo** é descrito utilizando-se a sintaxe apresentada, e o *framework* realizará a interpretação do mesmo, disponibilizando-o em seguida.

Ao tentar realizar uma operação de cópia de arquivo, o usuário solicita a permissão de execução da operação ao *SPA*^{3}, que analisará a situação e retornará um valor lógico. Se o retorno for *true*, o sistema realizará a operação de cópia de arquivo (Figura 4), caso contrário o sistema informará ao usuário que não foi possível realizar a operação de cópia porque a situação não está em conformidade com as regras definidas no comando de privacidade (Figura 5).

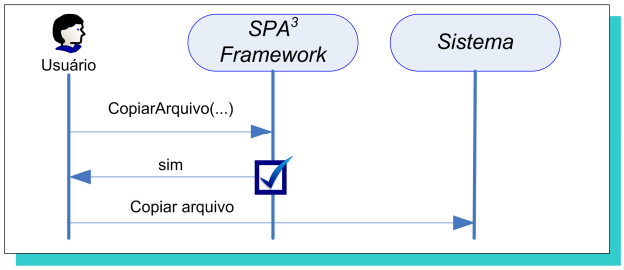


FIG. 4 Operação permitida pelo comando de privacidade.

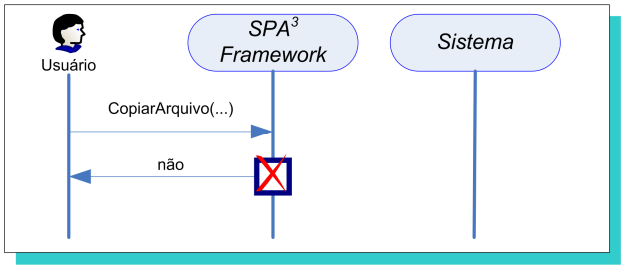


FIG. 5 Operação proibida pelo comando de privacidade.

O *SPA*^{3} permite que os comandos de privacidade sejam executados e avaliados em um sistema, aumentando o controle sobre as informações que trafegam e minimizando situações que caracterizem violação de privacidade.

IV. CONSIDERAÇÕES FINAIS

A utilização de um *framework* é justificada pela facilidade proporcionada no desenvolvimento de sistemas. Um *framework* para privacidade contribui para a construção simplificada de sistemas com reforço em privacidade de modo flexível e extensível.

O *framework SPA*^{3}, apresentado neste trabalho, provê uma solução que reforça os aspectos de privacidade de um determinado sistema. A utilização do formato *XML* para representação do autômato adaptativo proporciona um padrão bem definido, simples e formal para ser utilizado como entrada do *framework*. A modelagem do autômato adaptativo

através de *XML* é clara e concisa, além de permitir uma especificação completa do mesmo. Os comandos de privacidade possuem uma sintaxe de alto nível e de fácil representação, contribuindo para a especificação de situações de violação de privacidade em um sistema.

O trabalho apresentado contribui com a área de Tecnologias Adaptativas, conferindo um novo domínio de aplicação, e a utilização de um formalismo adaptativo provê ao modelo redução de tempo e espaço computacional.

O *SPA*^{3} é uma solução viável para incorporar reforços de privacidade nos sistemas. Os comandos de privacidade são reutilizáveis e podem ser executados a qualquer momento em um sistema, conferindo ao *framework* um nível de abstração para ser utilizado em diversas classes de problemas, sem sofrer modificações.

AGRADECIMENTOS

Os autores agradecem aos revisores do WTA 2008 pelas recomendações feitas para a elaboração da versão final do artigo.

REFERÊNCIAS

[1] C. H. Fernandes, A Privacidade na Sociedade da Informação, 2003..

[2] S. Fischer-Hübner, L. Yngström, and J. Holvast, "Addressing vulnerability and privacy problems generated by use of IT-security mechanisms," in *Proceedings of the IFIP 12th World Computer Congress on Education and Society – Information Processing '92 – Volume 2*. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co., 1992, pp. 314-321.

[3] S. Fischer-Hübner, *IT-security and privacy: design and use of privacy-enhancing security mechanisms.* New York, NY, USA: Springer-Verlag New York, Inc., 2001.

[4] M. J. May, C. A. Gunter, and I. Lee, "Privacy APIs: Access control techniques to analyze and verify legal privacy policies," in *CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations*. Washington, DC, USA: IEEE Computer Society 2006, pp. 85-97.

[5] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM,* vol. 19, no. 8, pp. 461-471, 1976.

[6] J. J. Neto, *Contribuições à Metodologia de Construção de Compiladores*, Thesis (Livre Docência) – Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

[7] _____, "Adaptive automata for context-dependent languages," *ACM SIGPLAN Notices*, vol. 29, no. 9, 1994..

[8] P. R. M. Cereda and S. D. Zorzo, "Formalismo com autômato adaptativo em mecanismo de privacidade e personalização," in *XXXIII Conferencia LatinoAmericana en Informatica - CLEI*. 2007, San Jose, Costa Rica, 2007.

Aplicação de Autômatos Adaptativos na avaliação de autômatos gerados por algoritmos genéticos

Victor D. Lopes, Ricardo L. A. Rocha

Abstract— **The language inference is a field of great importance in computer science. This work studies the application of genetic algorithms in the inference of regular languages, through the evolution of a population of finite automata that accept the strings belonged to it.**
In the automata evaluation obtained in each population, it is proposed the utilization of an adaptive automaton as base for this evaluation.

***Palavras chaves*—Autômato adaptativo, árvore de prefixo, algoritmos genéticos, inferência de autômato.**

I. Introdução

A inferência de linguagens é um campo de grande importância dentro da ciência da computação. No presente trabalho, será estudada a aplicação de algoritmos genéticos na inferência de linguagens regulares, por meio da evolução de uma população de autômatos finitos que aceitam as cadeias pertencentes a ela.
Na avaliação dos autômatos obtidos em cada população, propõe-se a utilização de um autômato adaptativo como base para essa avaliação.

II. Tecnologia Adaptativa

Como visto em [3], um dispositivo adaptativo possui estrutura interna que se modifica dinamicamente durante sua execução, em função de seus estímulos de entrada, sem a intervenção de agentes externos.
Esses dispositivos são divididos em duas partes; a camada subjacente, que são dispositivos usuais como autômatos, gramáticas ou tabelas de decisão. Esses dispositivos são tipicamente não adaptativos, determinísticos e definidos por um conjunto finito de regras. Considera-se como segunda parte o mecanismo adaptativo, definido como responsável pela formalização e execução das auto-modificações, que ocorrerão no conjunto de regras do dispositivo subjacente.
A execução do mecanismo adaptativo, como definido em [4], é feita através de ações adaptativas. Para cada passo na execução de um dispositivo adaptativo pode haver uma ação adaptativa anterior e uma posterior a esse passo. Essas ações são normalmente descritas como funções adaptativas, parametrizadas ou não, constituídas basicamente de uma lista de ações adaptativas elementares.
Há três tipos de ações adaptativas elementares: ação de consulta, que possibilita a busca de padrões na estrutura definida pelas regras da camada subjacente; ação de eliminação e ação de inclusão, que permitem a alteração da estrutura da camada subjacente.
Essas funções podem conter três tipos de variáveis: uma lista de parâmetros que são passados no momento de sua execução; uma lista de geradores, que são variáveis especiais automaticamente preenchidas a cada execução da função com valores não utilizados pelo dispositivo subjacente até a ocasião. Por último, uma lista de variáveis preenchidas através das ações adaptativas elementares de consulta, cujo valor, depois de determinado, permanece constante até o final da execução da função.

III. Autômato Adaptativo

Autômato adaptativo é uma instância de dispositivo adaptativo que tem como camada subjacente um autômato de estados finitos ou um autômato de pilha estruturado. Para este estudo será utilizado um autômato adaptativo de estados finitos simples.
Baseando-se em [4], um autômato adaptativo M é definido por uma máquina de estados inicial E_0 a qual será submetida uma cadeia W de comprimento n. E_n é a máquina de estados final, depois de reconhecer a cadeia W inteira. E_i é uma máquina de estados intermediária que será submetida a uma cadeia W_i, que é uma sub-cadeia de W, sem os i primeiros elementos.
Definindo W como sendo constituída pelos elementos $a_0a_1a_2...a_n$, pode-se dizer que a trajetória de reconhecimento do autômato adaptativo pela cadeia W é (E_0, a_0) -> (E_1, a_1) -> (E_2,a_2) ->...->(E_n,a_n).
Para um autômato de estados finitos simples, em um determinado momento da trajetória de reconhecimento, as transições do autômato são representadas pela seguinte notação ***(e, s) : A, -> e', B***, sendo que ***e*** é o estado de origem da transição, ***s*** é o elemento recebido pelo autômato, ***A*** é a ação adaptativa anterior da transição, ***e'*** é o estado destino e ***B*** é a ação adaptativa posterior.

V. D. Lopes é pesquisador do Laboratório de Linguagens e Técnicas Adaptativas, Escola Politécnica da USP, São Paulo/SP, Brasil; fone: 55 11-3091-5583; e-mail: victor.lopes@ poli.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.

IV. Autômato de Árvore de Prefixo

Com base em [5], um autômato de árvore de prefixo é um autômato em forma de árvore que aceita, de forma determinística, uma linguagem finita representando alguns exemplos positivos de uma linguagem mais ampla.

Implementando esse autômato através de um autômato adaptativo, este inicialmente possuirá apenas um estado, o inicial que também será o final. Esse estado apresentará apenas uma transição para ele mesmo com uma ação adaptativa associada a essa transição.

Essa ação adaptativa fará com que o autômato cresça guiado pelos elementos que recebe da cadeia de entrada e realize a memorização dessa cadeia, através de alterações no conjunto de regras que o definem.

V. Algoritmos Genéticos

Baseando-se em [7], algoritmos genéticos são algoritmos de busca ou classificação que se baseiam no processo de evolução natural de Darwin. Eles trabalham com populações de indivíduos, onde cada indivíduo é uma solução em potencial para o problema em estudo.

Cada indivíduo é representado por um cromossomo, normalmente uma cadeia de 0's e 1's. Durante o processo de recombinação genética, esse cromossomo sofre operações inspiradas nas que ocorrem nos núcleos celulares, como *crossover* e mutação.

Todos os indivíduos são submetidos a um ambiente, que representa o problema a ser resolvido, e recebem uma nota de acordo com seu desempenho. Assim os mais aptos, os que melhor resolveram o problema, recebem notas mais altas.

Depois de avaliados, são escolhidos alguns indivíduos de forma aleatória, para gerar a próxima população, sendo que os mais aptos têm probabilidade maior de serem escolhidos, podendo ser escolhidos até mais de uma vez.

Os indivíduos escolhido são agrupados em pares e submetidos ao processo de reprodução, seus descendentes são os indivíduos que irão constituir a próxima geração.

VI. Representação do cromossomo

Para o presente trabalho, a representação do indivíduo por meio do cromossomo se baseou em [2], primeiramente foi limitado o número máximo de estados e elementos distintos na cadeia de entrada, os autômatos têm no máximo sete estados e quatro elementos distintos na cadeia de entrada.

O formato do cromossomo é bastante simples, ele é dividido em sete partes, uma para cada estado, cada parte possui treze bits, a identificação dos estados é posicional, ou seja, o primeiro bloco de treze bits representa o estado 0, o segundo representa o estado 1 e assim por diante. O primeiro bit da representação do estado indica se o estado é de aceitação ou não, os doze bits restantes são divididos em quatro parte, uma para cada possível elemento de entrada, esses elementos serão identificados hipoteticamente como **a**,**b**,**c** e **d**, de forma que a primeira parte dos doze bits indica o estado destino (em binário) da transição recebendo o elemento **a**, a segunda parte indica o estado destino recebendo o elemento **b** e assim por diante, sendo que o número sete (111) indica que a transição não está ativa.

VII. Implementação do algoritmo genético

Para a implementação do algoritmo foi usado como mecanismo de seleção o *roulette wheel* padrão, sem utilização de *elitist strategy* para garantir a permanência dos indivíduos mais aptos. Como operadores genéticos, foram utilizados o *crossover* de ponto único com probabilidade de 90% e a mutação com probabilidade de 1%.

A população é de vinte indivíduos criados aleatoriamente ou algum padrão de cromossomo é forçado na primeira população. Esses indivíduos são submetidos a um determinado número de cadeias aleatórias de uma linguagem regular.

Ao final do processamento de cada uma das cadeias, os indivíduos têm seus desempenhos avaliados e armazenados. No final será utilizada a avaliação média de cada um deles na seleção.

O critério de parada foi definido empiricamente de forma a parar após executar mil gerações.

VIII. Procedimento de avaliação dos indivíduos de uma população

A avaliação será feita através da comparação da nota do autômato com a nota de um autômato adaptativo usado como referência implementando uma árvore de prefixo. Esse método foi baseado no método proposto em [1], com várias diferenças importantes, desde a representação do cromossomo dos indivíduos até a função de avaliação.

No lugar da função que calcula o comprimento mínimo de mensagem (MML) para cada autômato, proposta originalmente, foi usada uma função mais simples que não leva em consideração não determinismos, pois como é possível verificar, a estrutura proposta do cromossomo não permite a criação de autômatos não determinísticos. Além do mais, a nova função leva em consideração a eficiência das transições do autômato, pois considera a relação entre o número de transições utilizadas no processamento das cadeias de entrada e o número total de transições presentes nele. Considera também a quantidade de estados e a quantidade de estados finais.

A função é:

$$fitness = a + b * (c + d + e * f)$$

sendo que:

a é a relação entre a quantidade de elementos que o autômato conseguiu computar com a quantidade de elementos da cadeia multiplicada por um determinado peso.

b é 1 se autômato conseguiu computar todos os elementos da cadeia e 0 no caso contrário.

c é a relação entre a quantidade de estados do autômato com a quantidade máxima de estado do autômato multiplicada por um determinado peso. [2]

d é a relação entre a quantidade de transições utilizadas pelo autômato com a quantidade total de transições no autômato multiplicada por um determinado peso.

e é 1 se o autômato processou a cadeia toda e parou em um estado de aceitação e 0 no caso contrário.

f é a relação entre a quantidade de estados de aceitação com a quantidade máxima de estados multiplicada por um determinado peso.[3]

Finalmente, a função que realiza a comparação com o autômato de referência e retorna a avaliação dos indivíduos é a mesma exponencial proposta em [1].

IX. EXPERIMENTO

Foram realizados alguns experimentos com o algoritmo e como base de comparação entre os autômatos obtidos foi utilizada suas relações de tamanho. O cálculo da relação de tamanho é feito através da fórmula:

r = 1 – (T/Max), sendo T a quantidade de estados do autômato e Max a quantidade máxima de estados que o autômato pode ter, nesse caso é sete, devido às restrições na representação dos cromossomos dos indivíduos da população.

Foram utilizadas 12 linguagens regulares escolhidas de [1] para a execução do algoritmo. Os resultados obtidos são a média de 10 execuções para cada linguagem, segue a tabela 1 com eles.

Na coluna "Alvo" estão os dados obtidos de [1] dos autômatos esperados para as linguagens. Na coluna "GA20" estão as relações obtidas dos autômatos gerados pelo algoritmo genético com as configurações descritas anteriormente utilizando 20 cadeias na avaliação da população. Por último, na coluna "GA5", estão as relações obtidas dos autômatos gerados com apenas 5 cadeias usadas na avaliação da população.

Nos dois casos, fixou-se um padrão de cromossomo que faz com que os indivíduos da primeira população tenham todas as transições desativadas e todos os estados sejam de não aceitação. Iniciando a primeira população aleatoriamente obtém-se resultados bem abaixo do esperado, com relações iguais a 0 em praticamente todos os casos.

	Alvo	**GA20**	**σ**	**GA5**	**σ**
a*	0,86	0,53	0,16	0,43	0,21
(ab)*	0,71	0,57	0,19	0,49	0,25
s/ mais de 2 a's seg.	0,57	0,39	0,19	0,53	0,16
N# par de a's e b's	0,43	0,39	0,16	0,4	0,28
a*b*a*b*	0,43	0,28	0,2	0,33	0,13
a*b	0,71	0,44	0,22	0,4	0,26
(a* + c*)b	0,43	0,39	0,23	0,26	0,15
(aa)*(bbb)*	0,29	0,27	0,25	0,29	0,19
a(aa)*b	0,57	0,3	0,23	0,33	0,16
N# par de a's	0,71	0,64	0,09	0,64	0,11
(aa)*ba*	0,57	0,31	0,19	0,24	0,18
bc*b+ac*a	0,43	0,36	0,27	0,33	0,18
Médias	0,56	0,41	0,20	0,39	0,19

Tabela 1: Média e desvio padrão das relações de tamanho dos autômatos gerados pelo algoritmo.

X. RESULTADOS OBTIDOS

Os resultados obtidos, comparados com os autômatos alvos, não foram satisfatórios. Primeiramente, analisando a execução do algoritmo com a primeira população iniciada aleatoriamente, a baixa qualidade dos resultados indica que esse método não é bom em aperfeiçoar soluções já existentes, pois não conseguiu diminuir a quantidade de estados dos autômatos.

Para as outras duas configurações GA20 e GA5, há dois exemplos que deixam as deficiências do algoritmo bastante evidentes. O primeiro é o caso da linguagem com a quantidade par de ***a***'s, analisando mais detalhadamente os resultados é possível observar que o algoritmo inferiu autômatos com uma quantidade de estados menor que o autômato alvo. Isso aconteceu porque além das cadeias com quantidade par de ***a***'s eles aceitam cadeias com quantidade impar. Isso é devido a não utilização de exemplos negativos na avaliação da população, não houve pressão evolutiva alguma garantindo que os autômatos reconhecedores apenas das cadeias com quantidade par de ***a***'s predominassem no ambiente.

O segundo exemplo é o da linguagem ***a*b***, o autômato alvo para essa linguagem é bastante trivial e possui apenas dois estados, analisando os resultados é possível verificar que o algoritmo nunca conseguiu gerar um autômato com dois estados apenas. Isso ocorre pelo mesmo motivo que fez com que a configuração do algoritmo com a primeira população iniciada aleatoriamente tivesse um desempenho tão baixo, o algoritmo não consegue aperfeiçoar a solução. Ele encontrou uma solução com três estados, mas ficou preso nesse ótimo local.

Por último, foi possível observar que os resultados obtidos na configuração GA5 tiveram um desvio padrão maior que os obtidos com a configuração GA20. Isso ocorre porque a variação da quantidade de cadeias na avaliação dos indivíduos altera a avaliação do autômato adaptativo de referência. Utilizando mais cadeias este autômato perde eficiência na utilização das transições, quantidade de estados e estados de aceitação. Como a avaliação dos indivíduos é feita em relação à do autômato de referência, com a configuração GA5, que utiliza uma quantidade menor de cadeias, a avaliação fica mais rigorosa. Uma avaliação mais rigorosa varia menos, isto

[2] Nesse caso, a relação é inversa, pois quanto menor mais eficiente é o autômato, por tanto, foi utilizado o seguinte cálculo: 1- v/T, sendo v o valor que pode variar na relação (numero de estado e o número de estados finais) e T é o valor total, que é a base da relação (no caso o número máximo de estados).

[3] Idem nota 1.

dificulta ainda mais que o algoritmo escape de ótimos locais, pois evita a convergência das medidas em um único ponto, aumentando a variação dos resultados obtidos.

XI. CONCLUSÃO

A utilização de algoritmos genéticos na inferência de autômatos finitos não é trivial, isso porque, no espaço de soluções de autômatos existe uma quantidade considerável de ótimos locais, onde o algoritmo acaba preso. Além disso, ficou evidente que é muito importante a utilização de exemplos negativos na avaliação da população.

Para obter melhores resultados será necessário melhorar o mecanismo de seleção dos indivíduos, utilizar a *rolette whell* simples não pareceu ser suficiente, principalmente quanto a questão dos ótimos locais. Também a utilização de *crossover* de dois pontos pode melhorar a recombinação dos cromossomos. Por último, é interessante melhorar os métodos de avaliação dos indivíduos, por exemplo, no lugar de usar simplesmente uma Arvore de prefixo, pode ser mais proveitoso utilizar o autômato gerado pelo método proposto em [5], que é um autômato mais eficiente e próximo do ótimo.

REFERÊNCIAS

[1] P. Hingston, “A Genetic Algorithm for Regular Inference”, 2001 Genetic and Evolutionary Computation Conference. GECCO 2001, pp. 1299-1306. Morgan Kaufmann, July 2001.

[2] V. Fabera, “Automata Construct with genetic algorithm”, 2006 9th EUROMICRO Conference on Digital System Design (DSD'06) pp. 460-463

[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

[6] J. H. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, Ann Arbor, 1975

[7] J. Kennedy e R. Eberhart, “Swarm Intelligence”, San Francisco, California – USA: Morgan Kaufmann Publishers, 2001, 512p.

Implementação de Extensibilidade Sintática com Tecnologia Adaptativa utilizando o JavaCC

A. S. Mignon

***Resumo*— Este trabalho objetiva relatar uma implementação de um analisador léxico e sintático de uma linguagem de programação procedimental simples, originalmente não extensível, associada a um mecanismo de extensão, que confere a tal linguagem a possibilidade de extensibilidade sintática. A linguagem de programação e o mecanismo de extensão foram originalmente propostos em [1]. A extensibilidade sintática da linguagem é implementada utilizando a tecnologia adaptativa. Para a geração do analisador léxico e sintático foi utilizada a ferramenta JavaCC.**

***Palavras chave*— JavaCC, extensibilidade, linguagem de programação, tecnologia adaptativa.**

I. Introdução

EM [1] os autores propõe um mecanismo de extensão de linguagens de programação imperativas originalmente não extensíveis. A associação deste mecanismo de extensão ao tipo de linguagem de programação citada permite a criação de novas construções sintáticas a partir de construções sintáticas pré-existentes na linguagem. Este tipo de mecanismo permite criar e utilizar construções sintáticas mais próximas ao domínio do problema em que se está trabalhando.

Este documento objetiva relatar uma implementação de um analisador léxico e sintático de uma linguagem de programação imperativa simples associada ao mecanismo de extensão citado. O objetivo deste mecanismo é permitir que a linguagem de programação apresente características de extensibilidade sintática. A tecnologia adaptativa [2] é utilizada para possibilitar que o reconhecedor original da linguagem seja capaz de incorporar o reconhecimento de novas construções sintáticas. Para a implementação do analisador léxico e sintático é utilizada a ferramenta JavaCC[4]. Esta ferramenta tem por objetivo gerar analisadores léxicos e sintáticos de linguagens de programação.

A Seção II apresenta sucintamente o trabalho na qual implementação foi baseada e como a tecnologia adaptativa é utilizada para implementar a extensibilidade sintática da linguagem. A Seção III apresenta a ferramenta JavaCC utilizada na implementação do analisador léxico e sintático. A Seção IV apresenta o relato da implementação realizada. Por fim, a Seção V apresenta conclusões e trabalhos futuros.

[4] Disponível em: <https://javacc.dev.java.net/>. Acesso em: 10 de dezembro de 2007.

II. Mecanismo de Extensibilidade Sintática

Esta seção baseia-se integralmente em [1]. Neste trabalho os autores propõem um mecanismo de extensão que, associado a linguagens de programação imperativas, originalmente não extensíveis, proporciona a linguagem características de extensibilidade sintática.

Uma linguagem de programação imperativa com características de extensibilidade sintática permite adicionar novas construções sintáticas a linguagem com o objetivo de representar abstrações não contidas em sua sintaxe. As novas construções sintáticas são especificadas a partir de termos existentes na linguagem. Estes termos podem tanto estar contidos na linguagem núcleo, quanto em alguma camada de extensão da linguagem já existente. A extensibilidade sintática permite que novas abstrações sejam adicionadas a linguagem, sem a necessidade de nenhuma alteração em seu compilador.

Um modo possível para permitir a inclusão de novas construções sintáticas em uma linguagem de programação imperativa, é oferecer alguma característica meta-lingüística a esta linguagem. A notação de Wirth [3] foi escolhida pelos autores para oferecer tais características. Este modo é utilizado pelo mecanismo de extensão para permitir a extensibilidade sintática da linguagem.

Permitir a especificação de novas construções sintáticas não é tudo que se precisa para a extensão de uma linguagem. É necessário também, expressar as extensões utilizando construções sintáticas já existentes na linguagem. A solução adotada para tal caso é tipificar a extensão e utilizar a semântica operacional clássica para interpretar cada nova construção sintática em relação a uma construção sintática existente: cada nova extensão é declarada como uma gramática livre de contexto, e o seu significado é declarado como um texto expresso como um programa usando a sintaxe básica da linguagem núcleo ou construções sintáticas previamente declaradas. Desta forma, o compilador é informado da nova construção a ser aceita, e também como exatamente deve traduzi-la em um nível de abstração mais baixo.

A. *O Mecanismo de Extensão*

Os autores apresentam uma proposta simplificada de um mecanismo de extensão, para ser associado a linguagens

de programação imperativas e originalmente não extensíveis. A proposta é apresentada através de uma gramática livre de contexto especificada em notação de Wirth modificada [4]. Esta gramática é dividida em duas partes: a primeira parte especifica uma linguagem de programação núcleo não extensível; a segunda parte especifica a proposta do mecanismo de extensão. As duas partes da gramática são apresentadas a seguir.

```
//* Linguagem Núcleo *//
PROG = "BEGIN" ( DECL \ ";" ) "START" ( COM \ ";" ) "END" .
DECL = "VAR" ( id \ "," ) ":" "INTEGER" | PROCEDURE | EXTENSION .
COM = LABEL ":" PROG | id := EXPARIT | "GOTO" LABEL |
      "IF" EXPARIT ( ">" | "=" | "<" | "<>" ) EXPARIT
         "THEN" PROG ( "ELSE" PROG | ε ) | PREVIOUSNTERM .
EXPARIT = (( id | int | CALL ) \ ( "+" | "-" | "*" | "/" )) .
CALL = id "(" ( id | int | CALL \ "," ) ")" .
PROCEDURE = "FUNCTION" id "(" ( id ":" "INTEGER" \ ";" )")"
                       ":" "INTEGER" ";"
                       "START" ( COM \ ";" ) "END" ";" .
LABEL = id .
```

```
//* Mecanismo de Extensão Proposto *//
PREVIOUSNTERM = ∅ .
EXTENSION = "DEFINE" NEWNTERM ":" "NEW" NTERM "AS" WIRTHMOD
                    "MEANING" PREVIOUSWIRTHMOD "ENDDEFINE" .
NTERM = "PROG" | "DECL" | "COM" | "EXPARIT" | "EXTENSION"
      | "LABEL" | "NTERM" | "CALL" | "PROCEDURE" | "NEWNTERM"
      | "WIRTHMOD" | "PREVIOUSWIRTHMOD" | PREVIOUSNTERM .
NEWNTERM = id .
WIRTHMOD = ((( TERM | NTERM | NEWNTERM | "ε"
          | "(" WIRTHMOD ( "\" WIRTHMOD | ε ) ")" )
             ( "#" int | ε ) \ ( "|" | ε ) ) .
PREVIOUSWIRTHMOD = ((( TERM | NTERM | "ε"
        | "(" PREVIOUSWIRTHMOD ( "\" PREVIOUSWIRTHMOD | ε ) ")" )
           ( "#" int | ε ) \ ( "|" | ε ) ) .
```

B. *O Uso da Tecnologia Adaptativa*

A simples associação da linguagem de programação com o mecanismo de extensão permite o reconhecimento apenas das construções sintáticas da linguagem núcleo e das construções sintáticas relacionadas ao mecanismo de extensão. Entretanto, o reconhecedor deve ser capaz também de reconhecer as novas construções sintáticas e incorporá-las a linguagem. Para que o reconhecedor comporte-se da maneira desejada, a tecnologia adaptativa é utilizada.

Ao reconhecedor existente, ações adaptativas são criadas, de modo que, quando o reconhecedor identificar uma nova construção sintática, a ação adaptativa crie um reconhecedor para a nova construção sintática e acrescente este novo reconhecedor ao reconhecedor da linguagem já existente. Além da alteração na estrutura do reconhecedor, a ação adaptativa insere a nova extensão da linguagem em um conjunto de extensões. Inicialmente, este conjunto de extensões é vazio.

C. *Um Exemplo de Extensão da Linguagem Núcleo*

Como exemplo de extensão da linguagem núcleo, propõe-se adicionar à linguagem a declaração de pré-condições [5]. Uma sintaxe sugerida é a construção de uma palavra "PRE" seguida por uma condição (no caso da linguagem utilizada, as condições podem ser definidas pela relação de duas expressões aritméticas). Para atingir o objetivo proposto no exemplo, o programa pode declarar a nova sintaxe desejada conforme apresentada seguir.

```
DEFINE PRECONDITION: NEW COM AS
   "PRE" EXPARIT # 1 ( "=" | "<" | ">" | "<>" )
   EXPARIT # 2
MEANING
   "IF NOT (" EXPARIT # 1 ( "=" | "<" | ">" | "<>" )
   EXPARIT # 2 ") THEN ERROR() ELSE"
ENDDEFINE;
...

FUNCTION divide (n: INTEGER; d: INTEGER): INTEGER;
   START
   PRECONDITION d <> 0;
   divide:=n/d
   END;
...
```

No trecho de programa apresentado, o denominador da divisão será checado automaticamente para verificar se o seu valor é diferente de zero toda vez que a função divide for chamada. Caso o denominador esteja associado com o valor zero um erro é reportado.

Em termos do pré-processador da linguagem, o comportamento deste pode ser identificado na geração de um código expandido conforme apresentado a seguir.

```
FUNCTION divide (n: INTEGER; d: INTEGER): INTEGER;
   START
   IF NOT ( d <> 0 ) THEN ERROR () ELSE divide:=n/d
   END;
```

III. A FERRAMENTA JAVACC

O JavaCC é um gerador de analisadores léxicos e sintáticos utilizado para a geração de compiladores de linguagens de programação do tipo LL{k}. Ele permite a especificação da gramática da linguagem de programação em uma notação próxima a EBNF. As definições léxicas e sintáticas da linguagem são descritas em um único arquivo.

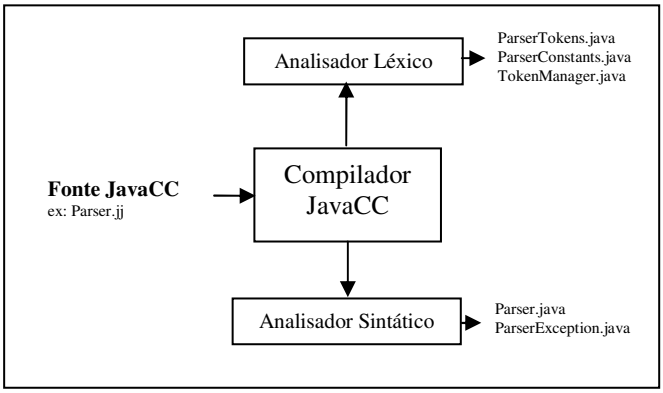


Fig. 1 Estrutura Básica do JavaCC.

A Fig. 1 apresenta a estrutura básica do JavaCC. Esta figura foi baseada em [6]. A especificação da gramática da linguagem de programação é descrita em um arquivo de extensão *.jj* (*Parser.jj*). Este arquivo é então submetido ao compilador do JavaCC. O compilador é responsável por gerar o analisador léxico e sintático da linguagem descrita. Os analisadores gerados são definidos em termos de classes Java. Para o analisador léxico são geradas três classes: *ParserTokens.java*, *ParserConstants.java* e *ParserTokenManager.java*. Para o analisador sintático são geradas duas classes: *Parser.java* e *ParserException.java*.

A. *Análise Léxica com o JavaCC*

O analisador léxico gerado pelo JavaCC é denominado *TokenManager*. O *TokenManager* é uma máquina de estados que move entre diferentes estados léxicos para a classificação dos *tokens*. Ele é utilizado para agrupar caracteres de um fluxo (*stream*) de entrada em *Tokens* de acordo com as regras léxicas especificadas. Cada regra léxica especificada no *TokenManager* é associada a expressões do tipo:

- SKIP: Descarta a expressão correspondida.
- MORE: Continua obtendo a próxima expressão para construir uma expressão maior.
- TOKEN: Cria um *token* utilizando a expressão correspondida e o envia ao analisador sintático.
- SPECIAL_TOKEN: Cria um *token* com a expressão correspondida e opcionalmente o envia ao analisador sintático.

O código a seguir apresenta um exemplo de uma especificação de *Tokens* em JavaCC. O tipo *SKIP* define que os espaços em branco e os caracteres de final de linha devem ser descartados. O tipo *TOKEN* define dois *tokens* da linguagem: o primeiro é o sinal de + e o segundo um número. Segundo a definição apresentada, um número pode conter um ou mais dígitos.

```
SKIP : { " " | "\n" | "\r" | "\r\n" }
TOKEN : {
      < PLUS : "+" >
  |   < NUMBER : (["0"-"9"])+ >
}
```

B. *Análise Sintática com o JavaCC*

O analisador sintático gerado pelo JavaCC é um analisador recursivo-descendente do tipo LL{k}. Este tipo de analisador utiliza um número *k* de *tokens lookahead* para gerar um conjunto de produções mutuamente exclusivas. Por padrão, o analisador sintático do JavaCC utiliza *k* igual a 1, entretanto, os desenvolvedores podem especificar o número do *lookahead* necessário para combinar corretamente as regras produção da linguagem [6].

O código a seguir apresenta um exemplo, em JavaCC, da especificação de uma regra de produção de uma gramática de uma linguagem de programação. Uma regra de produção é inicialmente definida por um nome, um tipo de retorno e, opcionalmente, parâmetros a serem passados para a regra. Após a definição do nome da regra, há um bloco delimitado por chaves que permite a inserção de código Java. Geralmente este bloco é utilizado para declaração de variáveis e chamada de métodos. Posteriormente, declaram-se regras de produção, contidas na gramática da linguagem, que estão associadas à regra de produção que está sendo definida. É possível associar código Java a cada uma das regras de produção declaradas.

```
void PROG() : {Token t;} {
  t=<NUMERO>
  {System.out.println(n.image.toString());}
      (<MAIS> <NUMERO>)*
      <EOF>
}
```

IV. UMA PROPOSTA DE IMPLEMENTAÇÃO

Esta seção relata a implementação da linguagem de programação livre de contexto e do seu mecanismo de extensão, apresentado na Seção II. Considera-se na implementação da linguagem apenas o seu analisador léxico e sintático. A ferramenta JavaCC, apresentada na Seção III, foi utilizada para gerar o analisador léxico e sintático proposto. A particular utilização desta ferramenta deve-se a familiaridade do autor com a mesma. Outro fator para a escolha foi considerar a implementação proposta em uma ferramenta não aderente a tecnologia adaptativa. O processo para a implementação da linguagem de programação e do mecanismo de extensão é descrito a seguir.

Inicialmente, a linguagem de programação núcleo foi especificada na ferramenta sem que o mecanismo de extensão fosse considerado. A única particularidade foi a inserção de um código escrito em linguagem Java para a geração de um *arquivo* de saída. O arquivo gerado não será gravado em disco, permanecendo, portanto, na memória enquanto o analisador sintático é executado. Este arquivo é responsável por armazenar o código-fonte do programa e a substituição das extensões encontradas. Posteriormente, foi especificada a parte da linguagem referente ao mecanismo de extensão proposto. Novamente, códigos em linguagem Java foram inseridos para auxiliar na geração do arquivo de saída.

À regra de produção denominada **EXTENSION** adiciona-se uma ação adaptativa posterior a análise sintática da extensão, escrita em linguagem Java, com o objetivo de colocar a definição da extensão em uma tabela. A implementação desta tabela foi realizada utilizando-se uma classe da linguagem Java denominada *HashMap*. Uma classe auxiliar, denominada *ExpansorExtensao*, foi criada e adicionada ao analisador sintático. Esta classe tem por objetivo, neste momento, armazenar a definição da extensão. A tabela armazena, portanto, o nome do novo termo criado e uma instância da classe *ExpansorExtensao*. A seguir é apresentado o código da regra de produção **EXTENSION** conforme especificado na ferramenta JavaCC. Nota-se que esta regra não grava nada no arquivo de saída que está sendo gerado pelo analisador sintático.

```
void EXTENSION(): {String n; String s, s1;}{
    <DEFINE> n=NEWNTERM() ":" <NEW> NTERM()
      <AS> s=WIRTHMOD()
    <MEANING> s1=PREVIOUSWIRTHMOD() <ENDDEFINE>
    {ExpansorExtensao ee =
      new ExpansorExtensao(n, s, s1);
     map.put(n, ee);}
}
```

O mecanismo de expansão de extensão, isto é, a substituição da nova construção sintática pelo seu significado, expresso por construções existentes na linguagem, foi implementado na regra de produção **PREVIOUSITEM**. Esta regra é acionada quando o analisador sintático encontra um identificador de acordo com a especificação do *Token* **<INDENT>**. A esta regra é associada uma ação adaptativa, descrita em linguagem Java, que verifica se o valor associado a este *Token* consta na tabela de extensões. Caso o identificador encontrado não conste na tabela de extensões, o

analisador sintático grava o *Token* consumido no arquivo de saída e continua o processamento do arquivo de entrada. Caso o valor conste na tabela, é obtida a instância da classe *ExpansorExtensao* associada ao identificador. Para efetuar a substituição do novo termo encontrado, o método *substituir* da classe *ExpansorExtensao* é acionado. Este método retorna uma *String* que contém o significado do termo encontrado. Esta *String* é então gravada no arquivo de saída que está sendo gerado pelo analisador sintático. Neste momento, o analisador sintático lê o restante do arquivo de entrada, grava os dados lidos no arquivo de saída, sem processá-los, e seu processamento é propositalmente interrompido. A necessidade da interrupção proposital do processamento do analisador sintático deve-se a limitações da ferramenta utilizada.

```
void PREVIOUSITEM() : {Token n;} {
      n=<IDENT>
      {if (map.containsKey(n.image.toString()))
{
         ExpansorExtensao ee =

      map.get(n.image.toString());
      saida.append(ee.substituir());
```

Após a interrupção do processamento do analisador sintático, este é novamente acionado, agora utilizando como arquivo de entrada o arquivo gerado no processamento anterior. Este processo continua até que o arquivo de entrada contenha somente notações sintáticas da linguagem núcleo e o analisador sintático aceite ou rejeite o arquivo que está sendo processado.

V. Conclusão

Este trabalho apresentou uma proposta de implementação de um analisador léxico e sintático de uma linguagem de programação imperativa simples, originalmente não extensível, associada a um mecanismo de extensão. O objetivo deste mecanismo é permitir que a linguagem de programação apresente características de extensibilidade sintática. A tecnologia adaptativa foi utilizada para possibilitar que o reconhecedor original da linguagem fosse capaz de incorporar o reconhecimento de novas construções sintáticas. Ações adaptativas foram utilizadas para alterar estruturalmente o reconhecedor da linguagem a cada nova construção sintática especificada.

A ferramenta JavaCC foi utilizada para a geração do analisador léxico e sintático da linguagem e do mecanismo de extensão. Entretanto, como a ferramenta não é aderente a tecnologia adaptativa, não se pode explorar ao máximo os recursos oferecidos por tal tecnologia. Para efeito de comparação, é recomendado que a implementação proposta fosse realizada em uma ferramenta de geração de analisadores léxicos e sintáticos aderente à tecnologia adaptativa.

Em trabalhos futuros pretende-se: criar um compilador, utilizando a tecnologia adaptativa, para a linguagem de programação e o mecanismo de extensão apresentados; efetuar uma comparação da implementação proposta em relação ao compilador citado; utilizar a tecnologia adaptativa em outros tipos de linguagens, como por exemplo, linguagens de transformação de modelos (ver ATL [7]).

Referências

[1] P. S. M. Silva and J. J. Neto, “An Adaptive Framework for the Design of Software Specification Languages,” in Proceedings of International Conference on Adaptive and Natural Computing Algorithms – ICANNGA 2005, Coimbra, March 21-23 2005.

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

[3] N. Wirth, “What can we do about the unnecessary diversity of notation for syntactic definitions?” Commun. ACM, vol. 20, no. 11, p. 822–823, 1977.

[4] J. J. Neto, “Contribuições à Metodologia de Construção de Compiladores,” Tese de Livre Docência, EPUSP, São Paulo, 1993.

[5] B. Meyer, Object-Oriented Software Construction, 2nd ed. New Jersey, USA: Prentice-Hall, 1997.

[6] G. Succi and R. W. Wong, “The Application of JAVACC to Develop a C/C++ Preprocessor,” SIGAPP Appl. Comput. Rev., vol. 7, no. 3, pp. 11–18, 1999.

[7] F. Jouault and I. Kurtev, “Transforming Models with ATL,” in Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005, 2005.

Biblioteca para autômatos adaptativos com regras de transição armazenadas em objetos feita em C++

N. L. Werneck

***Abstract*—We created a library with the C++ language to simulate adaptive automata. Some implementations use a table to store the transition rules and adaptive actions to be performed. In our system, the informations relative to each state reside in separate objects in the program. The execution does not happen in a loop that analyzes the same transitions table repeatedly, but rather trough nested calls to methods of the different objects. We used the library to create an interpreter for a language that describes adaptive finite automata, and executed some examples.**

***Index Terms*—Autômatos finitos adaptativos (adaptive finite automata), programação orientada a objetos (object oriented programming)**

I. Introdução

NESTE artigo apresentaremos uma biblioteca desenvolvida na linguagem C++ capaz de simular o funcionamento de autômatos finitos adaptativos. Buscamos fazer uma modelagem orientada a objetos adequada das entidades envolvidas. Uma importante característica da arquitetura implementada é que optamos por uma forma um pouco menos usual de simulação de autômato finito, utilizando chamadas de subrotinas ao invés de uma tabela de transição.

Criamos ainda um simulador de autômatos finitos com esta biblioteca. Ele funciona a partir da descrição de uma máquina realizada com a linguagem SDMBA, definida por nós. Esta linguagem reflete as peculiaridades e nossa arquitetura, e acreditamos ser adequada para autômatos mais simples.

II. Teoria

A. Arquiteturas para simulação de autômatos

A aplicação da adaptatividade [1] a autômatos finitos (AF) dá origem aos autômatos finitos adaptativos (AFA) [2], ou autômatos finitos auto-modificáveis [3]. Estes autômatos possuem um poder computacional maior do que o dos AFs, porém preservam a mesma estrutura. Isto é atraente devido à adequação dos AFs a determinados problemas.

Existem duas formas básicas de se implementar AFs em programas de computador. Uma forma bastante popular é utilizar uma tabela que armazena todas as regras de transição. Um registrador guarda o estado em que a máquina se encontra. O programa possui um laço que modifica o registrador de acordo com seu valor, o símbolo na entrada, e as regras especificadas na tabela. Ao tornarmos adaptativo um sistema deste tipo, o controlador torna-se capaz de modificar também a tabela de transições de acordo com o estado da máquina a cada instante, seguindo as funções adaptativas especificadas. A Figura 1 ilustra este sistema.

Outra forma de se implementar um AF é escrever um programa onde existe uma efetiva passagem do controle da execução do programa entre diferentes subrotinas, que representam os estados da máquina. O primeiro tipo de implementação é geralmente considerado como mais próximo do paradigma de programação estruturada. Este segundo método, entretanto, possui algumas características que desejamos estudar. Criamos portanto um modelo orientado a objetos para AFAs inspirado neste modelo. Nossa implementação não utiliza uma tabela geral de transições acessada dentro de um laço, mas sim objetos que representam os estados do autômato, e que se alternam seqüencialmente no controle da execução do programa.

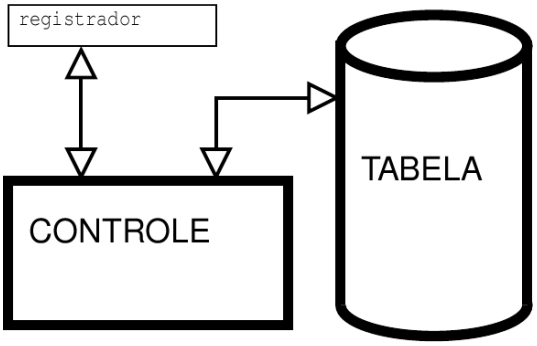


Fig. 1. Simulação de um AFA com registrador e tabela de transições.

B. Justificativas

Apesar de arquitetura com tabela funcionar, e possuir valor didático, ela se afasta da estrutura dos AFs. Máquinas de estados são modulares, com nodos e vértices. Não podemos ver isto bem ao misturar todas as transições numa só entidade.

Mecanismos com estruturas conexionistas mais poderosos do que AFs já foram abordados por diversos pesquisadores, incluíndo Shannon, e o próprio Turing [4]. O conexionismo tornou-se um tema de pesquisa especialmente popular nos anos 1980 [4,5,6], mas permanece um pouco sub-

Este trabalho foi financiado pela CAPES.

N. L. Werneck é aluno de doutorado em Engenharia Elétrica na EPUSP, onde integra o Laboratório de Técnicas Inteligentes (LTI) do PCS (telefone: 55-11-3091-5397; e-mail: nwerneck@gmail.com).

utilizado fora do meio acadêmico, sendo ainda algo mais comum como tema de pesquisa do que como ferramenta.

Queremos evidenciar a natureza conexionista dos autômatos adaptativos, buscando uma implementação descentralizada onde os nodos possuem contato apenas com seus vizinhos. Pretendemos assim explorar a possibilidade dos AFAs poderem às vezes atuar como uma "ponte" entre mecanismos conexionistas e monolíticos, devido justamente a estas duas formas alternativas de implementação desta entidade em programas de computador.

Além do conexionismo, queremos aproximar nossa implementação de outra das inspirações dos AFAs: os programas auto-modificáveis.

Programas auto-modificáveis são aqueles que modificam a própria área de memória em que se encontram, explorando a memória compartilhada da arquitetura de von Neumann [8]. Esta técnica de programação tornou-se um pouco marginalizada após o desenvolvimento de computadores melhores e de linguagens de programação mais sofisticadas, além da preocupação com o problema prático da segurança.

Por poderem às vezes ser difíceis de analisar, programas auto-modificáveis conflitam com alguns conceitos da ciência da computação, como a programação estruturada. Este paradigma é defendido na famosa carta de Dijkstra [9], que fala sobre as limitações cognitivas de humanos para compreender programas. Estas limitações justificariam o uso de linguagens com estruturas bem-definidas, o que permitiria uma melhor compreensão dos programas criados com elas.

Algumas pessoas definem a programação estruturada apenas como sendo a eliminação da instrução `go to` de programas e linguagens. Mas esta é uma visão muito superficial do problema, que não beneficia a discussão elaborada por Dijkstra, e ignora a existência de algumas interessantes estruturas que são bem descritas utilizando-se o `go to`. Nosso desejo é investigar as estruturas oferecidas pelo formalismo de AFAs, e determinar em que condições elas podem beneficiar a descrição de programas, e qual seu impacto na eficiência. Knuth menciona em um artigo [10] programas simples onde o `go to` é utilizado de uma forma benéfica e insubstituível. Queremos averiguar se não existem programas auto-modificáveis igualmente interessantes, utilizando AFAs como ponto de partida.

Por outro lado, temos também interesse no poder de expressão das AFAs, ou seja, em conhecer que sistemas são expressos de forma simples por AFAs, por mais que estas estruturas simples possam ser difíceis de se compreender.

Ao questionar políticas simplórias de programação, como a eliminação do `go to`, e ao procurar estudar estruturas de computação sem concepções rígidas sobre o que é fácil ou não de se compreender, nosso trabalho ainda se alinha ao desejo de outros programadores que compõem uma minoria [11] que busca questionar certos dogmas sendo adotados por alguns programadores de forma muito irrestrita, sem atenção ao contexto. Acreditamos que estas práticas podem estar não apenas trazendo benefícios a certas práticas de programação, mas também malefícios, principalmente ao restringir desnecessariamente a visão de pesquisadores.

Na pesquisa em Sistemas Operacionais o estudo de algoritmos de baixo-nível auto-modificáveis e adaptativos têm apresentado resultados interessantes. Existem trabalhos que demonstram a possibilidade de se obter grandes melhorias de eficiência nos sistemas utilizando programas de baixo nível sofisticados mais flexíveis e com adaptatividade [12,13,14].

III. A ARQUITETURA

A Figura 1 acima ilustra a arquitetura utilizada em algumas implementações de AFAs [15]. Um registrador armazena o estado atual da máquina, e o processo continua até atingir algum critério de parada, como o término da cadeia de entrada.

A arquitetura que implementamos é ilustrada pela Figura 2. Cada estado da máquina é representado por um objeto da classe `Estado`. Esta classe possui um método principal que deve ser executado no instante da entrada da máquina em cada estado. Cada objeto possui sua própria tabela de transições, implementada utilizando um *template* `map` da STL [16], que mapeia símbolos de entrada a ponteiros para objetos do tipo `Estado`, e do tipo `Funcao`. O método é responsável pela leitura da entrada, execução de funções adaptativas, e por executar o método do objeto que representa o próximo estado.

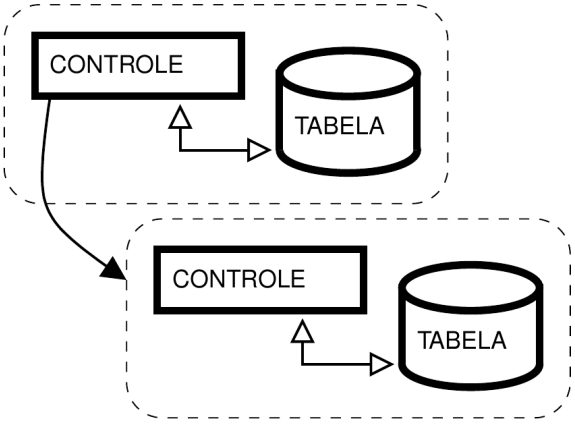


Fig. 2. Arquitetura implementada. Estados são objetos que executam os seguintes recursivamente.

Os ponteiros para os objetos criados são todos armazenados em uma lista implementada a partir do *template* padrão `list`. A entrada da máquina é feita pela entrada-padrão do sistema operacional, acessada por um objeto do tipo `istream`.

O programa resultante possui portanto diversos objetos do tipo `Estado`, cada um relativo a um estado da máquina, e estes podem criar e destruir livremente outros estados, além de modificar suas regras de transição. Ao invés de utilizarmos um laço que realiza as transições iterativamente, o funcionamento da máquina foi representado por chamadas recursivas do método principal da classe `Estado`.

Para realizar uma implementação adequada desta arquitetura é preciso utilizar um compilador que suporte chamadas de funções "irmãs", ou *sibling calls* ou ainda *tail calls* [17]. Neste tipo de chamada a sub-rotina retorna diretamente à função superior da que realizou a chamada.

Caso isto não corra, o programa resultante pode ficar lento, e até mesmo estourar a área de memória reservada ao programa, devido aos sucessivos empilhamentos de endereços de retorno. Em nosso sistema utilizamos a versão 4.2.1 do compilador GNU GCC [18], que utiliza este tipo de chamada como uma técnica de otimização do código.

A. *Funções adaptativas*

Cada transição pode possuir uma função adaptativa, modelada pela classe `Funcao`. Uma função em nosso sistema é uma lista de ações adaptativas que são executadas seqüencialmente. Existem dois tipos de ação: a criação de um novo estado, e a determinação de uma transição. Não implementamos ações explícitas de consulta ou de remoção.

Ao determinar uma transição, uma ação adaptativa só é capaz de acessar o último estado criado, o estado atual, e outros estados alcançáveis a partir destes. Isto é feito fornecendo-se uma cadeia de caracteres que deve ser seguida a partir do estado de partida (o último estado criado, ou o atual). Na versão atual do sistema, não é possível portanto fazer referência a estados de maneira global, ou fazer consultas em toda a máquina, mas apenas percorrer a sua estrutura momentânea como em um autômato finito, seguindo as cadeias de caracteres definidas dentro da função. Apesar de não podermos remover transições, podemos modificar livremente o destino das transições existentes. Na versão atual do sistema também é impossível apagar estados.

Estas restrições impostas às ações adaptativas que podemos realizar facilitam muito a tarefa de implementação de nossa arquitetura, e provém naturalmente de suas características. É preciso ainda estudar o impacto destas restrições no funcionamento das máquinas e na complexidade de suas descrições. Note que é possível superar algumas das dificuldades surgidas com táticas como criar transições com símbolos reservados para acessar estados específicos a partir de qualquer estado, ou ainda criar sempre pares de transições de sentidos opostos, permitindo navegar nos dois sentidos.

B. *Algumas considerações*

Nosso projeto foi desenvolvido a partir da idéia de possuir diferentes subrotinas executando umas às outras com regras variantes no tempo. Entretanto, o sistema compilado ao final, na presente forma, não possui exatamente este formato. A execução de um método de um objeto em C++ nada mais faz do que executar uma mesma função na memória que recebe o endereço do novo objeto como parâmetro [16].

O que realizamos ao final foi uma troca do laço utilizado na outra arquitetura pela execução de uma função recursiva que modifica iterativamente os parâmetros da chamada. Estes parâmetros são determinados a partir das regras de transição, que ao invés de serem lidos em uma grande lista são agora obtidos de uma estrutura semelhante a uma árvore encadeada.

Conceitualmente, estamos realizando uma troca do controle de execução do programa entre métodos de diferentes objetos, o que se assemelha um pouco com o que acontece em programas auto-modificáveis. Mas para obtermos isto no baixo nível, com diferentes rotinas, seria necessário definir diferentes classes para os estados.

Uma possibilidade para se criar esta alternância de chamadas de rotinas variante no tempo seria a ativação colateral de funções dentro de cada estado. Isto poderia ser obtido com o uso de funções do tipo *callback* e de sinais [19]. Por um lado isto não seria muito diferente de executar uma grande estrutura `case` para realizar algo dado cada estado, mas isto seria similar à substituição de nosso armazenamento de dados em árvore encadeada por uma lista, o que estraga a característica mais importante de nosso sistema: a concentração dos dados de cada estado em objetos separados.

Note ainda que estamos sempre realizando apenas modificações nos valores de variáveis do tipo ponteiro. Para realmente criarmos um programa auto-modificável seria necessário efetivamente escrever sobre o código, e não simplesmente alterar endereços de instruções de desvio e chamada de sub-rotina. A técnica de programas auto-modificáveis foi apenas uma inspiração para o desenvolvimento de nossa arquitetura. Mas ainda assim ela sai um pouco do paradigma de programação estruturada.

IV. SIMULADOR PROGRAMÁVEL E LINGUAGEM DESCRITIVA

Utilizando nossa biblioteca, criamos um programa capaz de simular AFAs projetados pelo usuário. Este programa lê a descrição da máquina a partir de um arquivo, lê a cadeia de entrada a partir da entrada-padrão, e ao final da cadeia ele avisa se esta foi aceita ou não, de acordo com a característica do estado no momento.

Decidimos definir uma nova linguagem para a descrição dos AFAs para deixar mais explícito as características de nossa arquitetura. Ela foi batizada de SDMBA—Sintaxe para Descrição de Máquinas da Biblioteca Adaptóide.

A linguagem permite nomear estados e funções, porém estes nomes não são utilizados posteriormente no funcionamento da máquina, quando apenas ponteiros são utilizados. Nomes não precisam ser declarados antes de ser utilizados.

```
programa = ( regra )*
regra = ( função | transição
        | ( "FINAL" | "INIT ") identificador ";" )
transição = nome símbolo nome (nome)? ";"
função = "FUN" ("POS" | "PRE") nome ":"
          (ação)* "END" ";"
estado_fun = ("T"|"*")(simbolo)*
ação = ( estado_fun símbolo estado_fun (nome)? ";"
       | "GEN" ";")
```

Nossa linguagem possui a seguinte gramática:

Um programa é uma seqüência de regras, que podem ser uma definição de estado final ou do inicial, uma transição da máquina inicial, ou uma declaração de função adaptativa. Transições podem ou não possuir funções associadas.

Na declaração das funções é que se define se ela é do tipo “pré” ou “pós”, executada antes ou depois da transição. O corpo da declaração possui uma lista de ações. O comando `GEN` cria um novo estado, que pode ser referenciado pelo símbolo `*` nas definições das ações.

Cada ação é executada imediatamente quando acionada. Isto significa que um estado referenciado por `T100`, por exemplo, pode referenciar um estado no começo da lista mas outro ao final, quando uma das transições no caminho for alterada.

V. TESTES

Utilizamos nosso simulador para implementar um AFA que reconhece números triangulares, ou seja, da série

O alfabeto utilizado possui apenas um símbolo, e os números foram representados em base unária. Este autômato está ilustrado na Figura 3. Inicialmente a máquina possui apenas um estado, que é o estado final, e uma transição para si próprio que coleta um símbolo e executa a função adaptativa posterior *H*. Esta função apenas cria um novo estado intermediário para o qual é apontado o estado de origem. Este novo estado possui uma nova transição para o estado final, executando a função *H*. O resultado é que o autômato cresce com uma estrutura de anel, ganhando mais um estado a cada volta.

Utilizando nossa linguagem, descrevemos a máquina da seguinte maneira:

```
FUN POS H:
  GEN;
  * 1 T1 H;
  T 1 *; END;
q1 1 q1 H;
FINAL q1; INIT q1;
```

Este programa pode ser modificado para realizar o reconhecimento de números quadrados, bastando modificar a função para que se acresça dois novos estados ao invés de um. Também é possível reconhecer potências de dois: basta executar a função *H* em ambas as transições criadas.

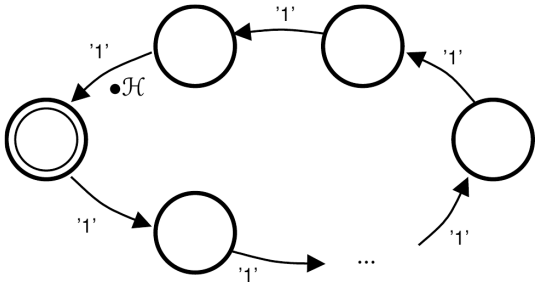


Fig. 3. Autômato finito adaptativo reconhecedor de números triangulares.

Um outro exemplo que implementamos com sucesso foi uma máquina capaz de reconhecer a linguagem `A^nB^nC^n`. Uma possibilidade de implementação pode ser descrita em nossa linguagem da seguinte maneira:

```
FUN POS F1: T a T F2;
  GEN; T b * G;
  * c T o; * a Tc; * b Tc; END;
FUN POS F2: T a T F3;
  GEN; Tb c * o;
  * b Tb; * a Tc; * c T x;
  T b *; END;
FUN POS F3:
  GEN; Tb c * x;
  * b Tb; * a Tc; * c T x;
  T b *; END;
FUN PRE x: T c Ta; END;
FUN POS o: Tc c Tcc o; END;

FINAL q1; INIT q1;
q1 a q1 F1; q1 b err; q1 c err;
err a err; err b err; err c err;
```

O estado `err` recebe transições referentes a palavras não-reconhecidas. Caracteres `a` no estado inicial geram uma cadeia de *n* estados com transições de ida para o símbolo `b`, e de volta para o símbolo `c`. As funções *x* e *o* atuam prevenindo que a quantidade de `b`s seja menor do que a de `a`s.

Os códigos-fonte da biblioteca e do simulador se encontram em nossa página, localizada em:

VI. CONCLUSÃO

`http://www.lti.pcs.usp.br/~nwerneck/adaptoide`

A arquitetura proposta foi utilizada com sucesso para implementar AFAs em programas em C++. Também pudemos criar um simulador de AFAs descritas por uma linguagem definida por nós. Demonstramos assim a aplicabilidade do modelo, e ainda da linguagem C++ em sua implementação.

Nossa arquitetura foi concebida de uma forma diferente da técnica popular de se implementar AFs, com tabelas de transição e um laço que realiza as leituras e transições. Entretanto, o programa que geramos ao final não ficou muito longe disto. No nível mais baixo, nosso sistema se diferencia apenas por ter trocado o laço de controle por uma função recursiva, e a estrutura de dados com as regras de transição passou a ser uma árvore encadeada. Porém, estas características estão implícitas em nosso programa, que foi construído em um nível de abstração mais alto, com objetos representando os estados da máquina, e passando o controle da execução do programa de um para outro.

Acreditamos que nossa arquitetura se compara à alternativa da mesma forma que listas encadeadas se comparam a vetores, e comprometimentos similares aos do uso destas estruturas de memória devem ocorrer na operação com ambas as propostas. Em especial, acreditamos que há uma maior dificuldade em nossa arquitetura de se realizar modificações envolvendo estados arbitrários ou distantes do atual, o que não é problema com uma tabela centralizada de transições. Tabelas entretanto possuem problemas, como menor flexibilidade.

Procuramos fazer o melhor uso possível de bibliotecas-padrão em nossos programas, esperando facilitar a integração com outros programas e obter mais eficiência. Um desejo nosso era ainda utilizar a programação orientada a objetos sem perder de vista a forma final do sistema no nível mais baixo, e acreditamos ter cumprido esta meta.

Esperamos no futuro introduzir diversas melhorias ao sistema, como empurrar símbolos na entrada utilizando métodos da biblioteca-padrão, implementar as funções adaptativas utilizando sinais de *callback*, engatilhar chamadas de função externas em cada estado, e testar a inicialização de múltiplas *threads* para simular indeterminismo.

REFERENCES

[1] J. J. Neto, *Adaptive rule-driven devices - general formulation and case study*, in CIAA, ser. Lecture Notes in Computer Science, B. W. Watson and D. Wood, Eds., vol. 2494. Springer, 2001, pp. 234-250.

[2] ----, *Adaptive automata for context-dependent languages*, SIGPLAN Notices, vol. 29, no. 9, pp. 115124, 1994.

[3] R. S. Rubinstein and J. N. Shutt, *Self-modifying finite automata: An introduction*, Inf. Process. Lett., vol. 56, no. 4, 1995, pp. 185190.
[4] C. Teuscher, Turing's Connectionism. An Investigation of Neural Network Architectures. London: Springer-Verlag, 2002.
[5] M. L. Minsky, S. A. Papert, *Perceptrons*. MIT Press, 1969.
[6] Wikipedia, C*onnection Machine*, Wikipedia, 15-Janeiro-2007.
[7] Wikipedia, *Inmos Transputer*, Wikipedia, 17-Dezembro-2007.
[8] Wikipedia, *von Neumann Architecture*, Wikipedia, 17-Dezembro-2007.
[9] E. W. Dijkstra, *Letters to the editor: go to statement considered harmful*, Commun. ACM, vol. 11, no. 3, pp. 147148, 1968.
[10] D. E. Knuth *Structured programming with go to statements*. ACM Computing Surveys 6(4), 1974, pp. 261301.
[11] U. Boccioni, P. Haeberli, B. Karsh, R. Fischer, P. Broadwell, and T. Wicinski, *The manifesto of the futurist programmers*, Internet, June 1991. Disponível em: `www.graficaobscura.com/future/futman.html`
[12] B. N. Bershad, C. Chambers, S. J. Eggers, C. Maeda, D. McNamee, P. Pardyak, S. Savage, E. G. Sirer. *Spin - an extensible microkernel for application-specific operating system services*. Operating Systems Review, vol. 29(1), 1995, pp. 74-77.
[13] H. Massalin, *Synthesis: an efficient implementation of fundamental operating system services*. PhD thesis, Columbia University, New York, NY, USA, 1992.
[14] H. Massalin, C. Pu, *Reimplementing the synthesis kernel*. In: *USENIX Workshop on Microkernels and Other Kernel Architectures*, USENIX, 1992, pp. 177186
[15] D. G. Santos, L. de Jesus, *Aspectos de projeto e implementação do não-determinismo no adaptools e seus impactos no aperfeicoamento da ferramenta*. Tech. rep., Universidade Católica Dom Bosco, 2006.
[16] B. Stroustrup, *The C++ Programming Language*, Second Edition. Addison-Wesley, 1991.
[17] Wikipedia, *Tail recursion*, Wikipedia, 18-Dezembro-2007.
[18] GCC manual, "GCC command options." [Online]. Disponível em: `gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`
[19] A. Pereira and M. Cumming, *Libsigc++*, 2004, Disponível em: `libsigc.sourceforge.net/libsigc2/docs/manual/html`

Autômatos Adaptativos para Emparelhamento de Cadeias

E. S. C. Rodrigues, F. A. Rodrigues, R. L. A. Rocha

***Resumo* — Este artigo apresenta um modelo baseado em Autômatos Adaptativos para o problema de emparelhamento de cadeias. Este problema surge quando é necessário encontrar todas as ocorrências de uma determinada palavra em um texto. As soluções para o emparelhamento de cadeias também podem ser usadas para encontrar, por exemplo, padrões específicos em seqüências de DNA. Os autômatos adaptativos foram usados para criar um conjunto de submáquinas responsáveis pela construção de um autômato que reconhece apenas o padrão que se deseja encontrar. O autômato construído é utilizado posteriormente para encontrar suas ocorrências em um dado texto. O uso de autômatos finitos para encontrar padrões em t :tos tem vantagem sobre outras soluções, pois examinam cada caractere do texto somente uma vez. No entanto, nenhuma comparação foi feita entre o uso de autômatos finitos e de autômatos adaptativos.**

***Palavras chave*— Autômatos Adaptativos (*Adaptive Automata*), Emparelhamento de Cadeias (*Pattern Matching*).**

I. Introdução

O problema de emparelhamento de cadeias sempre surge quando uma seqüência de caracteres é procurada em um texto. Para entender o problema, suponha que P e T são duas cadeias de caracteres sobre o mesmo alfabeto finito Σ. Suponha também que m é o tamanho de P e que n é o tamanho de T, com $m \leq n$. O padrão P é encontrado a partir da posição $s + 1$ no texto T, se $0 \leq s \leq n–m$ e $T[s+1..s+m] = P[1..m]$. De forma equivalente, diz-se que o padrão P ocorre com deslocamento s em T. Neste caso, s é dito um deslocamento válido. Caso contrário, s é dito um deslocamento inválido. Quando uma seqüência de caracteres é buscada em um texto, deseja-se encontrar todos os deslocamentos válidos desta seqüência no texto. A Figura 1 mostra um exemplo de um deslocamento válido de um padrão em um texto, ambos sobre o alfabeto $\Sigma = \{a, b, c\}$.

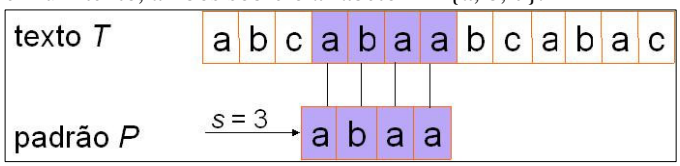


Fig. 1. Exemplo de um deslocamento válido do padrão abaa no texto T.

Vários algoritmos de emparelhamento de cadeias podem ser encontrados na literatura e todos são baseados nos conceitos apresentados acima[1]. O objetivo deste artigo é apresentar uma solução para o emparelhamento de cadeias, baseada em Autômatos Adaptativos.

Os autômatos adaptativos são usados quando é desejável que a estrutura do modelo seja modificada durante sua execução. Em [2] foi mostrado através de vários exemplos, como os autômatos adaptativos podem ser empregados como ferramentas com grande potencial para representar soluções elegantes, eficientes, compactas e práticas para muitos problemas complexos.

Autômatos adaptativos são dispositivos que podem reagir a seus estímulos de entrada, modificando sua própria estrutura. Eles utilizam os autômatos de pilha estruturados [3] como seu formalismo não adaptativo subjacente.

Em [4] pode ser encontrada uma formulação geral para dispositivos adaptativos baseados em regras e que usam dispositivos não adaptativos subjacentes. Na mesma referência é mostrado um estudo de caso para a simulação de um autômato finito em uma Tabela de Decisão Adaptativa.

Os autômatos adaptativos são dispositivos computacionalmente equivalentes às Máquinas de Turing [5]. Portanto, eles podem reconhecer linguagens representadas por outros formalismos, além dos autômatos finitos. Esta característica motiva o seu uso, pois além da generalidade do modelo, ele possui a capacidade de aprendizagem através de sua auto-modificação.

Em [6] é descrita uma proposta de algumas restrições para o uso de autômatos adaptativos. Isto foi feito com o objetivo de facilitar a criação e a interpretação dos dispositivos sem perda de poder computacional.

A seção 2 apresenta a notação utilizada para representar as sub-máquinas e funções que compõem o modelo adaptativo. A seção 3 descreve o modelo para o problema de emparelhamento de cadeias utilizando autômatos adaptativos. Em seguida, na seção 4 são sugeridos alguns trabalhos futuros e na seção 5 são apresentadas as conclusões.

II. Notação de Autômatos Adaptativos

Antes de apresentar o modelo desenvolvido, é necessário descrever a notação utilizada para a representação dos autômatos adaptativos.

As transições das sub-máquinas que compõem o modelo adaptativo são representadas por regras de produção com a seguinte forma: $(\gamma g, s, \sigma\alpha), \mathcal{A} :\rightarrow (\gamma g', s', \sigma'\alpha), \mathcal{B}$, em que:

- γ são meta-símbolos que representam o conteúdo da pilha não considerado pelo autômato (não influencia na aplicação da produção);
- g e g' representam o estado armazenado no topo da

E. S. C. Rodrigues e F. A. Rodrigues são alunos de Doutorado da Escola Politécnica da Universidade de São Paulo (e-mail: elisangela.rodrigues@poli.usp.br; fabricio.rodrigues@poli.usp.br).

R. L. A. Rocha é professor Doutor da Universidade de São Paulo (e-mail: luis.rocha@poli.usp.br).

pilha antes e depois da aplicação da transição, respectivamente;

- s e *s'* representam o estado corrente da transição e o próximo estado, respectivamente;
- σ é o símbolo do alfabeto que será consumido da cadeia e σ' é o símbolo do alfabeto que será empilhado na cadeia (ambos podem ser iguais a cadeia vazia, representada pelo símbolo ε);
- α são meta-símbolos que representam o restante da cadeia não considerada pelo autômato (não influencia na aplicação da produção);
- $\mathcal{A}$ e $\mathcal{B}$ são chamadas a funções adaptativas a serem executadas antes e depois da mudança de estados determinada pela produção. As funções adaptativas podem ter *n* parâmetros e assumem a seguinte forma $\mathcal{A}(a_1, a_2, ...,a_n)$, em que a_i são argumentos que assumirão valores a serem usados no corpo da função. As funções adaptativas são declaradas conforme descrito em [3].

III. Modelo Proposto

No problema de emparelhamento de cadeias usando autômatos, para cada padrão a ser procurado no texto, um autômato que reconheça somente aquele padrão deve ser construído. A busca por um padrão em um texto é feita em duas etapas: a primeira etapa é a construção do autômato (pré-processamento); a segunda etapa é o uso do autômato construído para realizar a busca e encontrar todas as ocorrências do padrão no texto. A seguir, será descrito um modelo que utiliza autômatos adaptativos na fase de construção do autômato usado na busca. O modelo também possui uma função adaptativa que será usada somente durante a busca do padrão no texto.

Para facilitar a compreensão do modelo, o alfabeto foi limitado a apenas dois elementos. Suponha que o alfabeto seja $\Sigma = \{0, 1\}$ e que o padrão a ser buscado no texto seja $P = 01\,\vdash$. O traço de asserção ($\vdash$) é usado para delimitar o final do padrão.

A idéia geral do modelo é que a cada símbolo lido no padrão, um novo estado seja criado e novas transições sejam inseridas para todos os símbolos do alfabeto. Para criar estas transições é necessário saber qual o estado destino para cada símbolo, pois é possível que haja sobreposição de símbolos de ocorrências sucessivas do padrão no texto. Por exemplo, se o padrão é 010 e o texto é 01010, então o último zero da primeira ocorrência do padrão é o primeiro zero da segunda ocorrência do padrão no texto.

Com o objetivo de criar um autômato determinístico para ser usado na busca, é necessário saber, para qualquer estado, qual o próximo estado para cada símbolo do alfabeto. Considerando-se as sobreposições possíveis, é preciso determinar se a função cria um novo estado ou cria uma transição para um estado já existente com cada símbolo do alfabeto.

Para solucionar este problema, é necessário encontrar o maior prefixo de *P* que é um sufixo da subcadeia que já é reconhecida pelo autômato, concatenado com o símbolo para o qual será criada a transição. Uma cadeia *w* é um prefixo de uma cadeia *x*, se $x = wy$ para alguma cadeia $w \in \Sigma^*$ e alguma cadeia $y \in \Sigma^*$. Da mesma forma, uma cadeia *w* é um sufixo de uma cadeia *x*, se $x = yw$ para alguma cadeia $w \in \Sigma^*$ e alguma cadeia $y \in \Sigma^*$.

É necessário que a máquina conheça o comprimento do prefixo mais longo de *P*, que é um sufixo da subcadeia lida até o momento [1]. Este procedimento inicia-se com a passagem do símbolo lido no padrão para uma função adaptativa responsável por criar um novo estado em uma sub-máquina, que irá reconhecer o padrão lido até o momento. Em seguida, uma sub-máquina empilha na cadeia os símbolos anteriormente lidos em ordem invertida, que representam o prefixo de *P*, para fins de comparação com a sub-cadeia lida até o momento. Os símbolos são empilhados em ordem invertida porque é preciso verificar se este prefixo é um sufixo da cadeia lida até então.

Para separar o prefixo de *P* dos símbolos que ainda não foram lidos do padrão, é usado o símbolo especial #. A sub-máquina que reconhece o padrão lido até o momento é executada consumindo os símbolos empilhados na cadeia até encontrar o símbolo #. À medida que os símbolos são consumidos e a sub-máquina muda de estado, um estado equivalente no autômato final aponta para o estado de destino de acordo com o símbolo que se deseja inserir a transição. Portanto, cada vez que o sufixo é encontrado, uma máquina com um ponteiro para o estado correspondente indica qual estado deve receber a transição.

O autômato adaptativo para o problema de emparelhamento de cadeias será composto por cinco sub-máquinas. A sub-máquina principal responsável pelo reconhecimento do padrão no texto é Z. Nesta sub-máquina, Z0 é o estado inicial e ZF é o estado final. A definição de Z é mostrada abaixo.

(ε, Z0, ε) :→ (ε, ZI, ε)
{(ε, ZI, σα) :→ (ε, ZJ, α), $\mathcal{B}$(σ) ∀σ ∈ Σ}
(ε, ZJ, ε) :→ (ε, ZK, ε)
(γ, ZK, ε) :→ (γZQ, PK0, ε)
(ε, ZQ, ε) :→ (ε, ZX, ε), $\mathcal{I}$(PK1, PC1)
(ε, ZX, ε) :→ (ε, ZU, ε), $\mathcal{J}$( )
(γ, ZU, ε) :→ (γZT, PQ0, ε)
(ε, ZT, ε) :→ (ε, ZW, ε)
(ε, ZW, ⊢α) :→ (ε, ZV, α), $\mathcal{C}$( )
(ε, ZW, ε) :→ (ε, ZI, ε)
(ε, ZV, ε) :→ (ε, ZF, ε)
(ε, PZ, ε) :→ (ε, PZ, ε)
(ε, PO, ε) :→ (ε, PO, ε)

As sub-máquinas que representam, respectivamente, a sub-cadeia de *P* que já foi lida até o momento e que será usada para determinar o próximo estado de cada transição inserida, o prefixo de *P* que será inserido novamente na cadeia em ordem inversa, o contador de ocorrências do padrão no texto e o maior prefixo de *P* que é um sufixo da sub-cadeia lida, são PQ, PK, A e PC, conforme definidas abaixo. Os estados iniciais de PQ, PK, A e PC são PQ0, PK0, A0 e PC0, respectivamente. Da mesma forma, os estados finais são PQ1, PK2, A1 e PC2.

PQ: (ε, PQ0, ε) :→ (ε, PQ1, ε)
(ε, PQ0, #α) :→ (ε, PQ1, α)
(γπ, PQ1, ε) :→ (γ, π, ε)
PK: (ε, PK0, α) :→ (ε, PK1, #α)
(ε, PK1, ε) :→ (ε, PK2, ε)
(γπ, PK2, ε) :→ (γ, π, ε)
A: (ε, A0, ε) :→ (ε, A1, ε)
(γπ, A1, ε) :→ (γ, π, ε)
PC: (ε, PC0, α) :→ (ε, PC1, #α)
(ε, PC1, ε) :→ (ε, PCF, ε)
(γπ, PC2, ε) :→ (γ, π, ε)

As funções adaptativas $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$ e $\mathcal{H}$, $\mathcal{I}$ e $\mathcal{J}$ são declaradas da seguinte forma:

$\mathcal{B}(\sigma)$ = {g1*, x, y, z:
– [(ε, x, ε) :→ (ε, ZI, ε)]
+ [(ε, x, σα) :→ (ε, g1, α)]
+ [(ε, g1, ε) :→ (ε, ZI, ε)]
– [(ε, PZ, ε) :→ (ε, y, ε)]
– [(ε, PO, ε) :→ (ε, z, ε)]
+ [(ε, PZ, ε) :→ (ε, x, ε)]
+ [(ε, PO, ε) :→ (ε, x, ε)]
– [(ε, ZJ, ε) :→ (ε, ZK, ε)]
{– [(ε, ZJ, ε) :→ (ε, ZK, ε), $\mathcal{D}(\theta)$] ∀θ ∈ Σ}
{+ [(ε, ZJ, ε) :→ (ε, ZK, ε), $\mathcal{D}(\rho)$] ∀ρ ∈ Σ – {σ}}
– [(ε, ZT, ε) :→ (ε, ZW, ε)]
{– [(ε, ZT, ε) :→ (ε, ZW, ε), $\mathcal{E}(\theta)$] ∀θ ∈ Σ}
{+ [(ε, ZT, ε) :→ (ε, ZW, ε), $\mathcal{E}(\rho)$] ∀ρ ∈ Σ – {σ}}}

$\mathcal{D}(\sigma)$ = {g1*, g2*, g3*, x:
– [(ε, PQ0, ε) :→ (ε, x, ε)]
+ [(ε, PQ0, ε) :→ (ε, g1, ε)]
+ [(ε, g1, σα) :→ (ε, x, α)]
+ [(ε, g1, #α) :→ (ε, PQ1, α)]
{+ [(ε, g1, ρα) :→ (ε, g2, α), $\mathcal{F}(\,)$] ∀ρ ∈ Σ – {σ}}
{+[(ε, g2, βα) :→ (ε, g2, α)] ∀β ∈ Σ
+[(ε, g2, #α) :→ (ε, g3, α)]
+ [(γ, g3, ε) :→ (γPQ0, PC0, ε)]}

$\mathcal{E}(\sigma)$ = {g1*, g2*, x, y, z, w, t, u:
– [(ε, PQ0, ε) :→ (ε, x, ε)]
? [(ε, x, σα) :→ (ε, y, α)]
? [(ε, PZ, ε) :→ (ε, z, ε)]
? [(ε, PO, ε) :→ (ε, w, ε)]
+ [(ε, w, σα) :→ (ε, z, α)]
– [(ε, t, ε) :→ (ε, PK2, ε)]
{+ [(ε, t, α) :→ (ε, g1, ρα)] ∀ρ ∈ Σ – {σ}}
+ [(ε, g1, ε) :→ (ε, PK2, ε)]
+ [(ε, PQ0, ε) :→ (ε, g2, ε)]
{+ [(ε, g2, ρα) :→ (ε, y, α)] ∀ρ ∈ Σ – {σ}}
+ [(ε, g2, #α) :→ (ε, PQ1, α)]
?[(ε, x, ρα) :→ (ε, u, α), $\mathcal{F}(\,)$] ∀ρ ∈ Σ – {σ}}
+ [(ε, g2, σα) :→ (ε, u, α), $\mathcal{F}(\,)$]}

$\mathcal{F}(\,)$ = {x, y, z, t:
– [(ε, PZ, ε) :→ (ε, x, ε)]
{? [(ε, y, σα) :→ (ε, x, α)] ∀σ ∈ Σ}
+ [(ε, PZ, ε) :→ (ε, y, ε)]
? [(ε, z, ε) :→ (ε, PC2, ε)]
{– [(ε, t, ε) :→ (ε, z, β)] ∀β ∈ Σ}
+[(ε, t, ε) :→ (ε, PC2, ε))]}

$\mathcal{C}(\,)$ = {g1*, g2*, g21*, g22*, g3*, g4*, g5*, g6*, g61*, g62*, g7*, g8*, x, y, z, w, t:
– [(ε, PZ, ε) :→ (ε, x, ε)]
– [(ε, PO, ε) :→ (ε, y, ε)]
– [(ε, z, ε) :→ (ε, ZI, ε)]
+ [(ε, z, ε) :→ (ε, ZF, ε), $\mathcal{G}(\,)$]
+ [(ε, PZ, ε) :→ (ε, z, ε)]
+ [(ε, PO, ε) :→ (ε, z, ε)]
– [(ε, ZV, ε) :→ (ε, ZF, ε)]
+ [(ε, ZV, ε) :→ (ε, g1, ε), $\mathcal{D}(0)$]
+ [(γ, g1, ε) :→ (γg2, PK0, ε)]
+ (ε, g2, ε) :→ (ε, g21, ε), $\mathcal{I}$(PK1, PC1)
+ (ε, g21, ε) :→ (ε, g22, ε), $\mathcal{J}(\,)$
+ [(γ, g22, ε) :→ (γg3, PQ0, ε)]
+ [(ε, g3, ε) :→ (ε, g4, ε), $\mathcal{H}(0)$]
+ [(ε, g4, ε) :→ (ε, g5, ε), $\mathcal{D}(1)$]
+ [(γ, g5, ε) :→ (γg6, PK0, ε)]
+ (ε, g6, ε) :→ (ε, g61, ε), $\mathcal{I}$(PK1, PC1)
+ (ε, g61, ε) :→ (ε, g62, ε), $\mathcal{J}(\,)$
+ [(γ, g62, ε) :→ (γg7, PQ0, ε)]
+ [(ε, g7, ε) :→ (ε, g8, ε), $\mathcal{H}(1)$]
+ [(ε, g8, ε) :→ (ε, ZF, ε)]
? [(ε, z, 0α) :→ (ε, w, α)
+ [(ε, ZF, 0α) :→ (ε, w, α)]
? [(ε, z, 1α) :→ (ε, t, α)
+ [(ε, ZF, 1α) :→ (ε, t, α)]}

$\mathcal{H}(\sigma)$ = {g1*, g2*, x, y, z, w:
– [(ε, PQ0, ε) :→ (ε, x, ε)]
? [(ε, x, σα) :→ (ε, y, α)]
+ [(ε, PQ0, ε) :→ (ε, y, ε)]
– [(ε, PZ, ε) :→ (ε, z, ε)]
? [(ε, PO, ε) :→ (ε, w, ε)]
+ [(ε, w, σα) :→ (ε, z, α)]
+ [(ε, PZ, ε) :→ (ε, w, ε)]}

$\mathcal{G}(\,)$ = {g1*, x:
– [(ε, x, ε) :→ (ε, A1, ε)]
+ [(ε, x, +) :→ (ε, g1, ε)]
+ [(ε, g1, +) :→ (ε, A1, ε)]
}

$\mathcal{I}(\sigma 1, \sigma 2)$ = {g1*, x, y, z:
– [(ε, σ2, α) :→ (ε, y, ϕα)] ∀ϕ ∈ Σ
– [(ε, σ2, ε) :→ (ε, y, ε)]
+ [(ε, σ2, ε) :→ (ε, PCF, ε)]
{? [(ε, σ1, ε) :→ (ε, x, β)] ∀β ∈ Σ}
+ [(ε, σ2, ε) :→ (ε, g1, β)]
? [(ε, z, ε) :→ (ε, g1, β)]
– [(ε, z, ε) :→ (ε, PCF, ε)]
+ [(ε, g1, ε) :→ (ε, PCF, ε)]
$\mathcal{I}$(x, g1)}

$\mathcal{J}(\,)$ = {x, y:
? [(ε, x, ε) :→ (ε, PCF, ε)]
{? [(ε, y, ε) :→ (ε, x, β)] ∀β ∈ Σ ∪ {#}}
– [(ε, x, ε) :→ (ε, PCF, ε)]
+ [(ε, x, ε) :→ (ε, PC2, ε)]}

Esta solução foi criada para um alfabeto de apenas dois

símbolos. No entanto, é possível ampliar esta solução para qualquer alfabeto. Para cada símbolo do alfabeto seria necessário incluir uma função adaptativa para incluir uma transição para os demais símbolos. Portanto, quanto maior o tamanho do alfabeto, mais funções adaptativas são necessárias.

Após todo o padrão ter sido varrido e o autômato ter sido criado, o símbolo ├ é encontrado. Neste caso, uma função adaptativa é executada para remover as transições com a função que criava os estados e acrescentava as transições. Além disso, a função insere uma ação adaptativa na transição que chega ao estado final. Esta ação inserida será executada na etapa da busca para contar o número de ocorrências do padrão procurado no texto.

Uma vez que o autômato foi construído, começa a etapa de busca no texto. A Figura 2 ilustra o uso do autômato construído para efetuar a busca em um determinadno texto T. A busca inicia no primeiro símbolo do texto e vai consumindo cada símbolo até atingir o final. O autômato muda de estado de acordo com cada símbolo lido no texto. Cada vez que o autômato alcança o estado final significa que o padrão foi encontrado no texto.

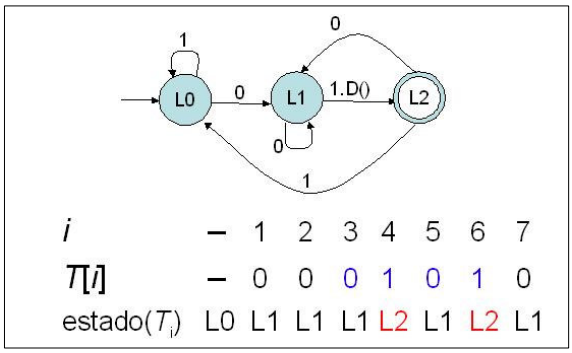


Fig. 2. Exemplo de busca em texto.

Para saber o deslocamento do padrão no texto, basta subtrair o valor da posição do texto em que o padrão terminou pelo tamanho do padrão. A função adaptativa que é executada cada vez que um padrão é encontrado no texto, cria um autômato que representa a quantidade de vezes que o padrão ocorreu no texto.

IV. Trabalhos Futuros

Apesar do uso de autômatos adaptativos para o problema de emparelhamento de cadeias ter sido proposto neste artigo, esta solução ainda não foi implementada completamente. A versão implementada atualmente já conta com duas funções adaptativas: uma para a criação da sub-máquina principal e outra para a criação da máquina que representa a quantidade de ocorrências do padrão no texto.

Desta forma, pretende-se implementar completamente a solução para o problema de emparelhamento de cadeias proposta neste artigo.

Além do modelo apresentado, é possível criar outras funções adaptativas para a fase de busca. Esta é uma sugestão para trabalhos futuros.

Uma outra sugestão, é a criação de outras funções adaptativas que reaproveitem os estados e transições de um autômato para a busca de novos padrões.

Também faz-se necessário um estudo sobre a complexidade de tempo e espaço de um modelo baseado em autômatos adaptativos para o problema de emparelhamento de cadeias, bem como uma comparação com outras técnicas existentes.

V. Conclusões

A tecnologia adaptativa pode ser muito útil para a representação de diversos problemas. Isto porque, a capacidade de auto-modificação em sua estrutura, permite agregar conceitos desejáveis em um modelo, como por exemplo, a aprendizagem.

O modelo apresentado neste artigo é mais uma solução para o problema de emparelhamento de cadeias. Este trabalho contribuiu com mais uma aplicação dos autômatos adaptativos para a solução de problemas computacionais.

Referências

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Algoritmos: Teoria e Prática*, Tradução da 2ª edição americana, Editora Campus, 2002.

[2] J. J. Neto, *Solving Problems Efficently with Adaptive Automata*, CIAA 2000 – Fifth International Conference on Implementation and Application of Automata, July 2000 – London, Ontario, Canada.

[3] J. J. Neto, Adaptive automata for context-dependent languages, ACM SIGPLAN Notices, Vol 29, n. 9, pp 15 – 24, 1994.

[4] 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, 2002, pp. 234-250.

[5] R. L. A. Rocha e 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, pp. 33-48, 2001.

[6] J. J. Neto e C. A. B. Pariente, *Adaptive Automata – a reduced complexity proposal*, In: Seventh Intenational Conference on Implementation and Application of Automata, CIAA 2002, Tours, Pre-proceedings Seventh Intenational Conference on Implementation and Application of Automata, CIAA 2002, Tours – France: University of Tours, 2002, pp. 161 – 170.

Elisângela Rodrigues nasceu em Pelotas/RS, Brasil, em 18 de abril de 1978. Graduou-se em Informáica pela Universidade Federal de Pelotas em 2000 e obteve o título de Mestre em Informática pela Universidade Federal de Campina Grande em 2002. Iniciou suas atividades docentes em 2002, na Faculdade de Tecnologia e Ciências de Feira de Santana/BA, onde atuou até 2005. Neste mesmo ano, transferiu-se para a Faculdade ÁREA1 em Salvador/BA, onde atuou até 2007. Em 2006, participou como pesquisadora bolsista (com bolsa da Petrobrás) do Projeto Sistema de Gerenciamento de Poços Automatizados (SGPA), na Universidade Federal da Bahia. Atualmente, é aluna regular do Programa de Pós-graduação em Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo, em regime integral.

Fabrício Rodrigues nasceu em Mossoró/RN, Brasil, em 09 de agosto de 1973. Graduou-se em Sistemas de Informação pela Universidade Potiguar em 2000 e obteve o título de Mestre em Informática pela Universidade Federal de Campina Grande em 2002. Iniciou suas atividades docentes em 2002, na Faculdade de Tecnologia e Ciências de Feira de Santana/BA, onde atuou até 2005. Neste mesmo ano, transferiu-se para a Faculdade ÁREA1 em Salvador/BA, onde atuou até 2007, além de atuar como Chefe do Núcleo de Computação. Atualmente, é aluno regular do Programa de Pós-graduação em Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo, em regime integral.

Visualizador gráfico para Redes de Petri Adaptativa

W. C. M. Gomes e A. R. Camolesi

***Resumo* — Este trabalho tem por objetivo apresentar os conceitos teóricos dos dispositivos Redes de Petri e Redes de Petri Adaptativa. Neste contexto é apresentado um visualizador gráfico para facilitar a visualização dos elementos de especificação das aplicações que usam tais dispositivos como linguagem de modelagem.**

***Palavras clave* — Tecnologia Adaptativa, Rede de Petri, Rede de Petri Adaptativa, Dispositivos Adaptativos**

I. INTRODUÇÃO

Camolesi nos diz em [1] que o projeto de aplicações adaptativas com suporte a multiformalismos exige, em geral, soluções relativamente complexas, nas quais é imprescindível o uso de um método bem definido, que dê suporte à especificação de aplicações e, posteriormente, à análise e à implementação das especificações produzidas. Fundamentado nos conceitos apresentados em [2] é proposto um método para o projeto de aplicações adaptativas constituído de três fases: especificação, análise e implementação.

Na fase de especificação, um projetista utiliza uma linguagem de especificação e gera modelos da aplicação desejada. Com base nos modelos obtidos, é realizada a fase de análise, que tem por objetivo verificar inconsistências na aplicação projetada. Nessa fase, o projetista submete estímulos ao modelo e obtém respostas que lhe permitem validar o modelo definido. Também nessa fase podem ser realizadas simulações do modelo, cujas respostas geradas auxiliarão o projetista na validação e análise da aplicação projetada. Ao serem detectadas inconsistências ou falhas, a especificação deve ser editada e, na seqüência, novamente analisada. Finalmente, depois de validada a aplicação, o modelo gerado poderá ser traduzido para uma linguagem de representação física.

Para auxiliar o projeto de aplicações que use tecnologia adaptativa, faz-se necessária a utilização de um ambiente que integra um conjunto de ferramentas que dê suporte aos projetistas na realização de seu trabalho. A Figura 1 apresentada em [1] ilustra a arquitetura geral de tal ambiente, que é organizado em 3 (três) grupos de ferramentas: Ferramentas de Edição, Ferramentas de Análise e Ferramenta de Implementação.

Valendo-se das Ferramentas de Edição, um projetista de aplicações utilizando-se de um dispositivo adaptativo específico poderá realizar a especificação de sua aplicação com o auxilio de um editor de texto qualquer ou de um editor gráfico.

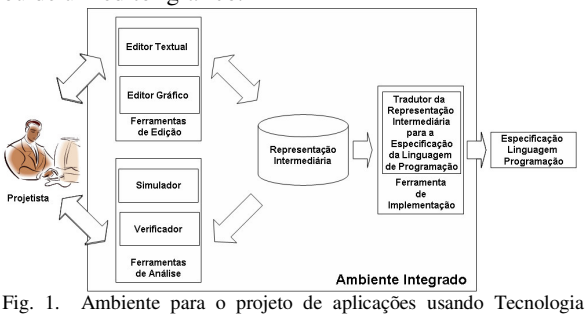


Fig. 1. Ambiente para o projeto de aplicações usando Tecnologia Adaptativa.

Para possibilitar o intercâmbio das especificações produzidas, os editores devem gerar objetos no formato da Representação Intermediária definida para suportar os elementos conceituais de dispositivos adaptativos. Caso a especificação seja produzida em um editor textual, esta deverá ser compilada para transformar a codificação realizada no formato definido para a representação intermediária.

Depois de realizada a especificação, o projetista de aplicações poderá utilizar as Ferramentas de Análise. Tais ferramentas utilizam a codificação da especificação (no formato da representação intermediária) como base e permitem a realização de uma análise do comportamento da aplicação adaptativa em desenvolvimento. Por fim, depois de especificada e analisada a representação de uma aplicação, o projetista pode se utilizar das Ferramentas de Produção de Representações Físicas e gerar uma representação de uma aplicação em um determinado padrão de linguagem de representação física e obter a aplicação desejada.

Motivado pela definição de dispositivos e a construção de ferramentas é apresentado neste trabalho o formalismo Redes de Petri Adaptativa [3] e a estrutura de uma ferramenta para visualização dos objetos gráficos para tal dispositivo. O visualizador construído utilizará como base a biblioteca OpenJgraph [4], construída com base na linguagem Java e que permite a manipulação de grafos.

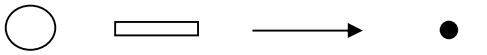
Este trabalho está organizado da seguinte forma: na seção II serão descritos os conceitos de Redes de Petri, na seqüência, é definido, na seção III, as Redes de Petri Adaptativas. Na seção IV é realizada a descrição da implementação do visualizador gráfico para Redes de Petri Adaptativa e, por fim, na seção V serão apresentadas as conclusões e sugeridos alguns trabalhos futuros.

W. C. M. Gomes (wilsoncmartins@gmail.com)
A. R. Camolesi (camolesi@femanet.com.br)

II. REDES DE PETRI

Redes de Petri é uma representação matemática e gráfica utilizada para modelagem de sistemas comunicantes e concorrentes [5].

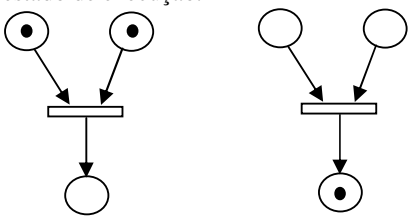
Uma Rede de Petri é uma quádrupla (*P, T, I, O*), onde *P* é o conjunto do lugar ou estado que representa a situação corrente da rede; T é conjunto de transições ou conexões, responsáveis por receber os sinais enviados pelos estados e os direcionar; *I* representa o conjunto de regras de entrada para ocorrência das transições da rede e o conjunto *O* representa as saídas, respostas de uma transição ocorrida na rede. Tais elementos podem ser representados graficamente conforme ilustrado na Figura 2.



a) Local b) Transição c) Arco d) Ficha

Fig. 2 Elementos gráficos Redes de Petri

Nas Redes de Petri uma ficha é responsável por marcar os estados em execução. Na Figura 3a é ilustrada uma Rede de Petri, na qual os estados *p1* e *p2* estão em execução aguardando o disparo da transição *t1*. Na Figura 3b observa-se a rede após o disparo da transição *t1*, onde *p1* e *p2* passaram para um estado de espera e *p3* passou para um estado de execução.



a) antes disparo b) após disparo

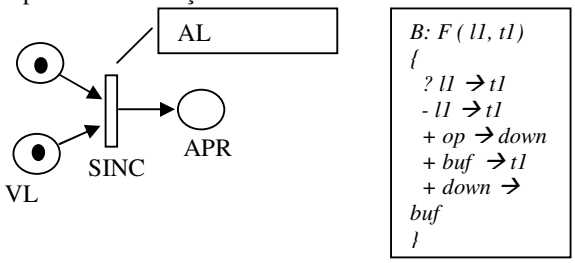
Fig. 3 – Exemplo Rede de Petri.

III. REDES DE PETRI ADAPTATIVA

Um dispositivo adaptativo é aquele que é capaz de se auto modificar em tempo de execução e sem influência externa [6, 7]. Para a transformação de uma Rede de Petri não adaptativa em uma Rede de Petri adaptativa, é acrescentada a esse formalismo uma camada adaptativa como apresentado na Figura 4, na qual após a execução das funções e ações adaptativas anteriores acontecem mudanças no comportamento do dispositivo, e após a execução das funções e ações adaptativas posteriores acontecem novas mudanças de comportamento.

Fig 4. Mecanismo Adaptativo de Rede de Petri.

Ao envolver uma Rede de Petri com o mecanismo adaptativo temos as funções e ações adaptativas associadas as transições. A seguir, é apresentado um exemplo de uma Rede de Petri Adaptativa que descreve a modelagem de uma aplicação de apresentação de áudio e vídeo sincronizados. Tal modelagem possui uma função adaptativa que permite que novos áudios sejam adicionados conforme as características de entrada da aplicação. Na Figura 5a os estados seriam representados por *AL* (Áudio Local), *VL* (Vídeo Local) e *APR* (Apresentação) enquanto que *SINC* (Sincronização) representa a transição do modelo.



a) Modelo aplicação b)Função Adaptativa *B*.

Fig. 5 Exemplo Rede de Petri Adaptativa Áudio e Vídeo Sincronizados

No exemplo apresentado, na ocorrência de uma ficha em *AL* e outra em *VL* ocorre a transição *SINC* que permite a apresentação das mídias de forma sincronizada na ocorrência de uma ficha no local *APR*. Na Figura 5b pode ser observado que a transição *SINC* possui a função adaptativa *B(..)* associada a mesma.

Para descrever uma função adaptativa devem ser elaboradas suas ações adaptativas que podem ser de três tipos: Consulta (?), Eliminação (-) e Adição (+). Uma ação de consulta verifica a existência de uma determinada regra que compõe o conjunto de regras de um modelo. Já a ação de eliminação é responsável por eliminar do modelo certa regra, enquanto, uma ação de adição serve para que novas regras sejam adicionadas ao modelo. Com base nos conceitos apresentados estrutura-se a função adaptativa *B* que ao ser executada recebe por exemplo, nos argumentos *l1* o valor *"AL"* e para o argumento *t1* o valor *"SINC"*. Inicialmente, é executada a ação adaptativa de consulta que verifica a existência de uma regra *AL* → *SINC*. Na seqüencia, caso a regra exista, ocorre a eliminação da mesma por meio de uma ação adaptativa de remoção e, por fim, executam-se as ações adaptativas de adição que adiciona novas regras ao modelo. Tais ações permitem que novas funcionalidades para obtenção de novos arquivos de áudio sejam adicionados e que estes áudios sejam disponibilizados ao sistema, para posterior apresentação. Desta forma, após a execução da função adaptativa *B* o novo comportamento da aplicação Áudio eVídeo, fica como ilustrado na Figura 6.

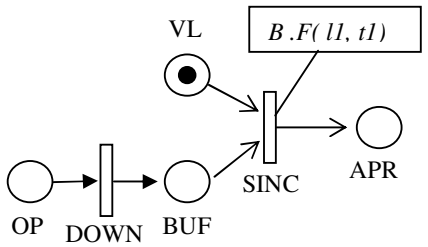


Fig 6. Aplicação Áudio e Vídeo após execução Função Adaptativa B

IV. IMLEMENTAÇÃO DO VISUALIZADOR GRÁFICO PARA REDE DE PETRI ADAPTATIVA

Para realizar a implementação do visualizador gráfico para Redes de Petri Adaptativa foi utilizada a biblioteca

OpenJgraph [4] que foi codificada usando a linguagem Java.

O primeiro passo para a realização do visualizador foi editar o modelo do vértice para cada um dos tipos de vértices necessários para a representação gráfica de Redes de Petri - o padrão do OpenJgraph são todos os vértices de forma gráfica quadrada -. A representação de cada forma de cada vértice da modelagem será de acordo com o valor que fica armazenado no modelo lógico representado pelo banco de dados da ferramenta. Para isso é necessário guardar o código do vértice e passá-lo por parâmetro no momento da instanciação do objeto *Vertex* e preparar o OpenJgraph para receber e retornar tal parâmetro. Essa etapa foi dividida nos seguintes passos descritos a seguir.

Inicialmente, na classe *VertexImpl.java* armazenada em *salvo/jesus/graph/VertexImpl.java* foi adicionado um novo parâmetro, do tipo inteiro, para armazenar o código de cada componente que é passado para o método. Na chamada do método *setLabel* passa-se a variável *cod_comp* por parâmetro. O *setLabel* está implementado na classe *LabeledGraphComponent Impl.java* e será descrito posteriormente. O código gerado é apresentado na Fgura 7.

```
public void setLabel(String label,int cod_comp)
   {
      //if (m_label != null) {
      //    throw new IllegalStateException("label cannot be
reassigned");
      //}
      m_cod_comp=cod_comp;
      m_label = label;
   }
```

Fig 7. Método *setLabel*.

Depois de armazenadas as informações referentes ao código do componente no OpenJgraph, o próximo passo é o desenvolvimento de métodos para retornar estes valores. E, para tal, criou-se na classe *VertexImpl.java* um outro método que foi chamado *getCodigo* e descrito na Figura 8.

```
public int getCodigo(){
    return this.labelDelegator.getCodigo();
  }
```

Fig 8. Método *VertexImpl.Java.*

Na classe *VertexImpl.Java* foi declarado o atributo *getCodigo* para ser, posteriormente, utilizado na classe *LabeledGraphComponentImpl.java.*

Na classe *LabeledGraphComponentImpl.java* que se encontra no pacote *salve/jesus/graph/LabeledGraph ComponentImpl.java*, foi criado um novo atributo para armazenar o código do componente (*m_cod_comp),* do tipo inteiro privado. No método *public void setLabel*, acrescentou-se um parâmetro do tipo inteiro *cod_comp* para receber a passagem de parâmetro do *m_cod_comp*, conforme descrito na Figura 9.

```
public void setLabel(String label,int cod_comp)
   {
      m_cod_comp=cod_comp;
      m_label = label;
   }
```

Fig 9. Método setLabel.

Por fim realizou-se o método *getCodigo()* conforme representado na Figura 10. Tal método permite o retorno do código de um componente.

```
public int getCodigo()
   {
      return m_cod_comp;
   }
```

Fig 10. Implementação getCodigo().

Na classe *AbstractVisualGraphComponent.java* pertencente ao pacote *salve/jesus/graph/visual/Abstract VisualGraphComponent.java*, definiu-se o método *getCodigo()* responsável por retornar o código do componente, como ilustrado na Figura 11.

```
public int getCodigo(){
    return this.component.getCodigo();
    }
```

Fig 11. Implementação método getCódigo().

Na classe *VisualVertex.java* localizada no pacote *salve/jesus/graph/visual/VisualVertex.java*, definiu-se um novo atributo para armazenar o tipo de forma gráfica do vértice. Tal atributo é do tipo de dados inteiro *(public int tipo;)*. Nesse caso, conforme o código do tipo de componente eu foi armazenado no banco de dados, tem que criar uma conexão com o banco de dados e fazer uma busca comparando o código do banco com o campo *getCodigo* que armazena o valor do código do componente, no OpenjGraph. Encontrando o componente desejado, armazena-se o seu código de tipo na variável *tipo* e as coordenadas que esse vértice deve ser apresentado na interface do visualizador (tais coordenadas também estão armazenadas no banco de dados). Ainda nessa classe foi acrescentado um comando condicional para verificar qual o tipo do vértice. Caso o tipo seja igual a quatro *(tipo==4)*, sua cor de fundo será branca (desta mesma forma pode-se alterar a cor de fundo dos vértices). Com essas alterações o método ficou conforme apresentado na Figura 12.

```
public VisualVertex( Vertex vertex, VisualGraph vGraph ){
   this.component = vertex;
   this.painter =
vGraph.getVisualVertexPainterFactory().getPainter( this );
   this.visualGraph = vGraph;
   this.setFont( new Font( "Lucida Sans", Font.PLAIN, 10 ));
   Componentes.listar("graph_x");
   Componentes.primeiro();
   do {

if(Componentes.getCampoInt("codigo_cpt")==vertex.getCodigo()){
        x=Componentes.getCampoInt("graph_x");
        y=Componentes.getCampoInt("graph_y");
        tipo=Componentes.getCampoInt("cod_tipo_cpt");
        Componentes.ultimo();
        }
     }while (Componentes.proximo());
   this.initLocation();
   // Force adjustment of width and height.
   this.rescale();
   this.setOutlinecolor( Color.black );
   if(tipo==4)this.setFillcolor( new Color( 255, 255, 255 ));
   else this.setFillcolor( new Color( 0, 225, 255 ));
  }
```

Fig 12. Método Visual Vertex

Finalizando, foi acrescentado no método *private void initLocation*" um comando de decisão *(switch-case)* para verificar o valor da variável tipo, para realizar a escolha da forma de representação gráfica do vértice, nesse exemplo o tipo do vértice pode variar de 0 à 17 que permite representar diversas formas geométricas. Nas

últimas linhas deste método são passadas as coordenadas de visualização do vértice na interface do visualizador, capturadas nos métodos definidos acima e armazenadas nos atributos *x* e *y*. O referido método é apresentado na Figura 13.

```
private void initLocation(){
   StringTokenizer   sttokenizer;
   int               height = 0, width, maxwidth = 0;
   int               lineheight;
   switch(tipo){
      case 0: drawpath = new GeneralPath( (Shape) new
Rectangle.Double( 2, 2, 2, 2 ));break;
      case 1: drawpath = new GeneralPath( (Shape) new
java.awt.geom.Ellipse2D.Double( 5, 5, 10, 10 )); break;
      case 2: drawpath = new GeneralPath( (Shape) new
java.awt.geom.Ellipse2D.Double( 5, 5, 10, 10 )); break;
      case 3:  drawpath = new GeneralPath( (Shape) new
java.awt.geom.Ellipse2D.Double( 5, 5, 10, 10 )); break;
      case 4: drawpath = new GeneralPath( (Shape) new
Rectangle.Double(10,10,10,10)); break;
      case 5: drawpath = new GeneralPath( (Shape) new
Rectangle.Double( 5, 5, 10, 10 )); break;
      case 6: drawpath = new GeneralPath( (Shape) new
Rectangle.Double( 5, 5, 10, 10 )); break;
      case 7: drawpath = new GeneralPath( (Shape) new
java.awt.geom.Arc2D.Double(16.0, 10.0,
30.0,30.0,90.0,180.0,Arc2D.CHORD));break;
      case 8: drawpath = new GeneralPath( (Shape) new
java.awt.geom.Ellipse2D.Double( 5, 5, 10, 10 )); break;
      case 16: drawpath = new GeneralPath( (Shape) new
java.awt.geom.Ellipse2D.Double( 5, 5, 10, 10 )); break;
      case 17: drawpath = new GeneralPath( (Shape) new
java.awt.geom.Ellipse2D.Double( 5, 5, 10, 10 )); break;
      default: drawpath = new GeneralPath( (Shape) new
Rectangle.Double( 2, 2, 2, 2 ));break;
    }
   if(x==0){
    this.setLocation( rand.nextInt( 500 ), rand.nextInt( 400 ) );
   }
   else{this.setLocation( x, y ); }
  }
```

Fig 13. Método initiLocation().

Depois de realizadas as alterações na biblioteca OpenJGraph é possível visualizar os vértices em sua forma gráfica. Tal fato comprova-se na Figura 14.

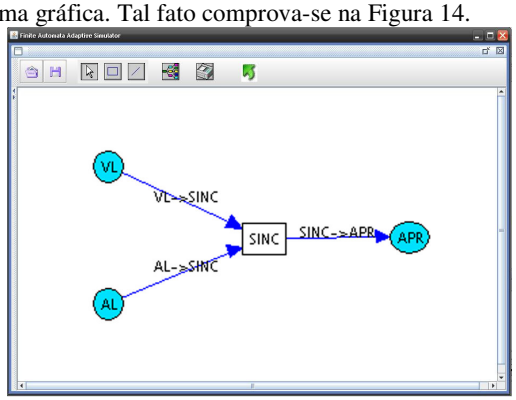


Fig 14. Interface do Visualizador gráfico para Redes de Petri Adaptativa.

V – CONCLUSÕES

Neste trabalho foram apresentados os conceitos de uma Rede de Petri Adaptativa e da arquitetura e construção de um Visualizador Gráfico para tal dispositivo.

Pode-se observar que a biblioteca OpenJGraph utilizada para a representação gráficas dos elementos conceituais do dispositivo estudado foi adequada e que a mesma permite a adição de novos elementos e estruturas para suportar outros dispositivos.

Este trabalho também permitiu comprovar que o modelo lógico definido para representação de dispositivos adaptativos representa de forma adequada tantos os elementos conceituais e de representação gráfica dos dispositivos nele definidos.

O visualizador implementado e integrado com a ferramenta de especificação de aplicações usando o modelo lógico (AdapSim) facilita o trabalho de simulação e o entendimento das falhas conceituais de especificação de uma aplicação.

Como trabalho futuro é sugerido a aplicação da arquitetura do visualizador para a representação de outros dispositivos adaptativos, bem como a proposta e o desenvolvimento de uma ferramenta gráfica para auxiliar os projetistas na realização do projeto de suas aplicações.

REFERÊNCIAS

[1] A.R. Camolesi, J. J. Neto "Modelagem Adaptativa de Aplicações Complexas," in 2004 *Anais XXX Conferencia Latinoamericana de Informática* Sep.

[2] W. L. Souza et al., "Design de Aplicações Multimídia Distribuídas (DAMD)," in *Anais do II Seminário Franco-Brasileiro em Sistemas Distribuídos Conf*, Fortaleza – CE, Novembro 1997

[3] A. R. Camolesi, J.J. Neto, "Representação Intermediária para Dispositivos Adaptativos Dirigidos por Regras," in *Proc 2004 3rd International Information and Telecommunication Technologies Symposium,* UFSCar, São Carlos, Brasil.

[4] Java Graph and Graph Drawing Project – OpenjGraph, avaible in http://openjgraph.sourceforge.net/

[5] C. A. Petri, "Kommunikation mit Automaten," Ph. D thesis, Institut für Instrumentelle Mathematik,Schriften, University of Bonn, Bonn, German, 1962.

[6] J. J. Neto, "Contribuições à metodologia de construção de compiladores," Tese de Livre Docência, Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 1993.

[7] J. J. Neto "Adaptive Rule-Driven Devices - General Formulation and Case Study," in *Proc. 2001 Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conf.,* Springer-Verlag, Vol.2494, pp. 234-250.

[8] H. Pistori, "Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações," Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 2003.

Wilson C. M. Gomes nasceu em Assis, São Paulo. Faz curso de graduação de Bacharelado em Ciência da Computação na Fundação Educacional do Município de Assis e participa do Programa de Iniciação Científica na mesma instituição.

Almir R. Camolesi nasceu em Cândido Mota, São Paulo, Brasil, em janeiro de 1972. Graduou-se em Tecnologia em Processamento de Dados na Fundação Educacional do Município de Assis, em 1992. Recebeu o título de mestre em Ciência da Computação pela Universidade Federal de São Carlos, em 2000, e doutorou-se em Engenharia de Computação e Sistemas Digitais pela Universidade de São Paulo, em 2007. Atualmente, é professor titular da Fundação Educacional do Município de Assis. Tem experiência na área de Ciência da Computação, com ênfase em Tecnologias Adaptativas, atuando principalmente nos seguintes temas: modelagem abstrata, metaferramentas, dispositivos adaptativos, ensino a distância, modelagem e desenvolvimento de aplicações distribuídas.

Tradutor Texto-Voz baseado em autômatos adaptativos

D. P. Shibata, F. R. Koike

***Resumo* — Este trabalho apresenta um tradutor texto-voz desenvolvido para a língua portuguesa baseado na tecnologia de autômatos adaptativos [1]. O tradutor é constituído por dois módulos de software desenvolvidos em paralelo e que integrados realizam o processo de síntese de voz a partir de um texto escrito na língua portuguesa. A primeira parte traduz o texto para uma seqüência fonética utilizando o Alfabeto Fonético Internacional desenvolvido pela Associação Fonética Internacional [2], enquanto a segunda parte recebe essa seqüência fonética e sintetiza a voz a partir de um processo de síntese por concatenação. Devido às diversas variações encontradas para a língua portuguesa, a língua falada na cidade de São Paulo foi escolhida como base para a saída gerada pelo tradutor.**

***Palavras chave* — Alfabeto Fonético Internacional, Autômatos Adaptativos, Tradutor Grafema-Fonema, Tradutor Texto-Voz.**

I. Introdução

ESTE trabalho apresenta um software desenvolvido para realizar a tradução texto-voz de textos escritos na língua portuguesa, baseado em autômatos adaptativos. O software apresentado se encontra dividido em dois módulos desenvolvidos separadamente. O primeiro módulo é responsável pela tradução grafema-fonema dos textos, recebendo como entrada um texto escrito na língua portuguesa e gerando como saída uma seqüência de fonemas que represente o texto de entrada. O segundo módulo sintetiza a voz a partir do método de síntese por concatenação.

Há dois fatores principais que motivam o desenvolvimento deste trabalho, sendo o primeiro o de aplicar os autômatos adaptativos para resolver alguns problemas relacionados à sensibilidade a contexto inerentes às linguagens naturais, utilizando assim todo o seu poder computacional, e o segundo, o caráter social do trabalho que pode ser utilizado para facilitar a comunicação de pessoas com deficiências que afetem o processo de comunicação por meio da fala ou até mesmo pessoas interessadas em utilizar o software para realizar a leitura de arquivos de texto enquanto realizam outras tarefas.

O trabalho inicia-se pela descrição de características de implementação do software, seguida das descrições do tradutor grafema-fonema baseado em Autômatos Adaptativos e do sintetizador de voz, apresentando de forma simplificada as técnicas utilizadas nesses processos. A seguir, encontra-se a discussão sobre os resultados obtidos pelo tradutor e pelo sintetizador e alguns dos problemas encontrados no processo de desenvolvimento dos dois softwares. Por fim, há a conclusão com discussão de trabalhos futuros e solução para possíveis problemas.

II. Características de Implementação

Tanto o tradutor grafema-fonema como o sintetizador de voz foram desenvolvidos utilizando a linguagem Java. Os softwares foram criados como APIs, e até o momento não há interface gráfica associada às bibliotecas.

No entanto, o tradutor pode ser executado por meio de linhas de comando, passando como parâmetro o arquivo com o caminho para o arquivo em que se encontra o texto a ser traduzido, o tipo de saída indicando se deve executar o som ou se deve gerar um arquivo de áudio no padrão WAV e, nos casos em que é gerado um arquivo de áudio, um caminho para esse arquivo.

O software ainda se encontra em versões iniciais de teste sem data definida para início de distribuição. Por essa razão, ainda não foi definida uma licença para distribuição do mesmo.

III. Tradutor Grafema-Fonema

Esta é a parte do software responsável pela recepção dos arquivos de entrada, escritos na língua portuguesa, e que gera como saída seqüências de símbolos escritas no Alfabeto Fonético Internacional, utilizada posteriormente pelo sintetizador de voz.

O software de tradução grafema-fonema foi baseado no modelo de autômato adaptativo apresentado em [3]. Este modelo foi utilizado como base para criar um tradutor grafema-fonema para palavras da língua portuguesa, e esse tradutor de palavras foi utilizado como base para traduzir textos da língua portuguesa.

Esse tradutor de palavras foi associado a um bloco de controle, responsável pela separação do texto em seqüências de palavras (incluindo seqüências de caracteres como números, datas, etc) e que fornece cada uma das palavras ao autômato para que possa ser executada gerando a saída fonética referente à palavra. A Figura 1 ilustra o tradutor texto-voz, enfatizando a separação dos blocos internos ao tradutor grafema-fonema.

D. P. Shibata (danilo.shibata@gmail.com).
F. R. Koike (fabio.koike@gmail.com).

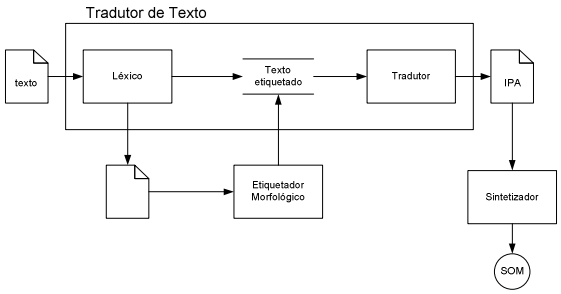


Fig. 1 – Ilustração do Tradutor Texto-Voz focando a tradução grafema-fonema.

Como pode ser visto pela Figura 1, o tradutor grafema-fonema possui um bloco inicial que funciona como analisador léxico e etiquetador, descobrindo as partes mínimas do texto que devem ser analisadas e transformadas em seqüências fonéticas e etiquetando cada uma dessas palavras.

Esse analisador léxico pode utilizar o auxílio de um etiquetador morfológico externo, gerando etiquetas que determinam as características morfológicas das palavras que são posteriormente utilizadas para definir qual o som mais provável para uma palavra considerando essa característica. O uso dessa funcionalidade não é obrigatório e o tradutor grafema-fonema é capaz de gerar uma saída mesmo sem o etiquetador morfológico ou em caso de falhas do mesmo. Nesse trabalho foi utilizado o etiquetador morfológico desenvolvido em [4].

As palavras são então submetidas ao autômato, e se o autômato obtiver mais de uma seqüência fonética para representar aquela palavra, uma delas será escolhida em função da etiqueta morfológica (se houver) ou qualquer uma delas será escolhida se não houver etiquetas.

A. *Tradutor de Palavras*

O tradutor grafema-fonema de palavras utiliza-se do modelo de autômato adaptativo definido em [3]. Esse módulo recebe como entrada uma palavra da língua portuguesa e gera como saída um conjunto de seqüências fonéticas que representa essa palavra.

Inicialmente, a palavra é submetida ao analisador léxico que gera uma seqüência de átomos semelhantes às sílabas definidas pelas regras de separação da língua portuguesa e coloca um símbolo indicando o final da cadeia após a seqüência. Essa seqüência de átomos é utilizada como entrada para o autômato que a traduz para o conjunto adequado de saídas.

O autômato adaptativo é composto por duas sub-máquinas chamadas de *Reconhecedor* e *Tradutor*. O *Reconhecedor* lê os átomos que representam sons e realiza chamadas de função adaptativa que alteram o *Tradutor*, e ao consumir o átomo que indica o final da cadeia executa uma chamada para o *Tradutor*. O *Tradutor* executa com a cadeia vazia e a partir de escrita e leitura de símbolos realiza a transdução da seqüência de átomos para as seqüências fonéticas que a representam.

As seqüências de chamada realizadas pelo *Reconhecedor* criam, no *Tradutor*, blocos de transições com a informação contextual necessária para definir o som adequado para um determinado átomo. Desta forma, toda a informação contextual sobre os átomos são escritas na cadeia e consumidas no devido momento, gerando a saída correta em função do contexto definido.

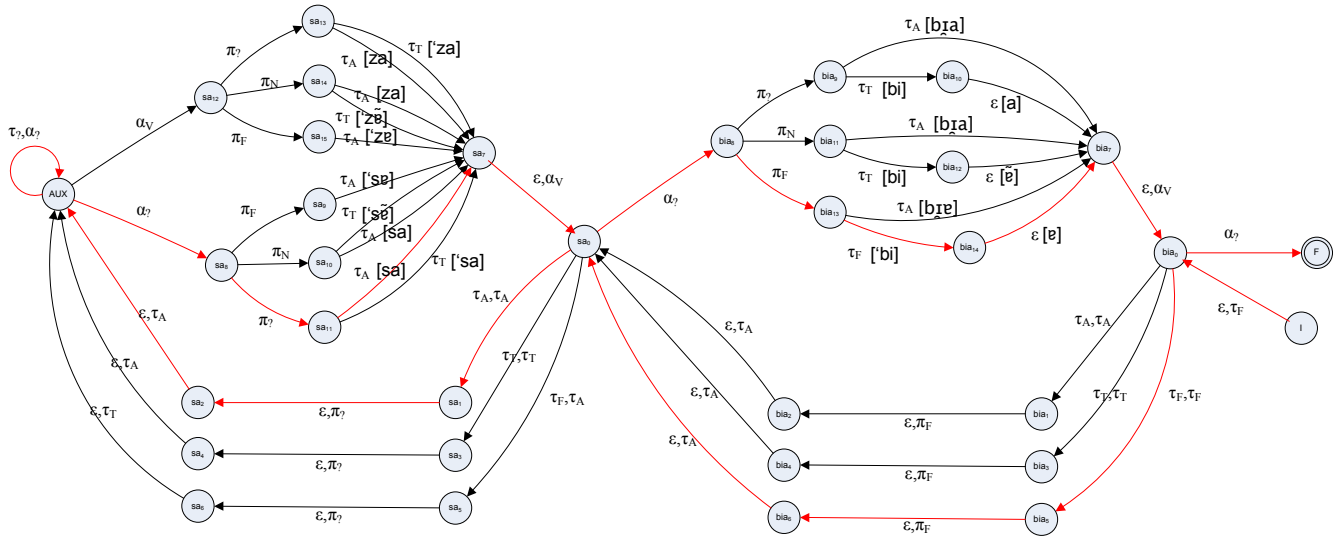


Fig. 2 – Transdutor alterado para representar a palavra "sabia".

A Figura 2 apresenta um exemplo de configuração do *Trasdutor*, quando alterado para representar a palavra "sabia". A palavra é dividida em dois átomos "sa" e "bia". O primeiro átomo é traduzido para o som /sa/, o som é átono pois a tônica se encontra no outro átomo, a letra 's' recebe o som /s/ por iniciar a palavra e o som a não sobre alterações por influências posteriores. O segundo átomo é separado em dois sons /'bi/ e /ɐ/ por ser final de uma palavra não acentuada. A letra 'a' recebe o som desvozeado por ser final, a letra 'b' não é alterada por influência anterior e o som de "bi" é tônico por ser anterior a uma letra 'a' final.

O processo é disparado pela transição que sai do estado "I" escrevendo o símbolo τ_F, que indica que o átomo que o lê é final em uma palavra não acentuada. Daí para frente, segue-se o processo de execução pelas transições marcadas em vermelho na figura. A partir da leitura de um símbolo define-se a ação a ser tomada (escrever novos símbolos ou escrever uma seqüência fonética na saída). As regras contextuais de cada átomo, que definem as ações realizadas a partir da leitura de um símbolo, são definidas pelas chamadas de funções executadas no *Reconhecedor*. O significado dos símbolos α, π e τ são definidos em [3].

IV. Sintetizador de Voz

O sintetizador de voz se encontra dividido em três partes, o Analisador Léxico, o Gerenciador e a Camada de Síntese de Voz. O Analisador Léxico e a Camada de Síntese realizam as tarefas necessárias ao processo de síntese, enquanto o módulo Gerenciador é responsável pela interface entre os dois outros módulos.

No Analisador Léxico identificam-se todos os *tokens* representados pelas sílabas. Adotou-se a sílaba como unidade de concatenação, por proporcionar efeito de coarticulação menores que a concatenação de fonemas [6]. Cada sílaba foi mapeada para sua respectiva pronúncia através da pré-gravação do áudio com a fala natural e cada arquivo de áudio teve que ser normalizado para possuir mesmas características sonoras com freqüência de 44.100 Hz, 16 bits, estéreo e com amplitude máxima de 1 db. Cada Token reconhecido é então armazenado numa estrutura de fila que é retornada para o gerenciador. No Analisador Léxico também são reconhecidas a tonicidade e as pontuações gramaticais, que orientam o processo de definição de entonação da voz sintetizada.

O gerenciador, de posse das unidades fonéticas reconhecidas, repassa-as para a camada de síntese de voz. A camada de síntese fica responsável pela recuperação dos respectivos arquivos de áudio de cada unidade, decodificando os mesmos de MP3 para WAV e extraindo os seus streams de áudio. Caso a unidade fonética seja tônica é aplicado um filtro de amplificação no stream para se conseguir o efeito desejado, esse processo se repete até finalizar a concatenação de todas as unidades fonéticas de uma sentença. Ao final de cada sentença verifica-se se existe alguma pontuação gramatical como interrogação, por exemplo, e aplica-se um novo filtro, agora em todos os streams já concatenados para se obter o efeito de entonação.

Como resultado deste processo, dependendo do comando determinado pelo módulo gerenciador, obtém-se um arquivo de áudio com a fala sintetizada para o texto de entrada ou reproduz-se o áudio. A Fig. 2 ilustra o processo de tradução texto-voz enfocando o processo de síntese a partir da representação fonética.

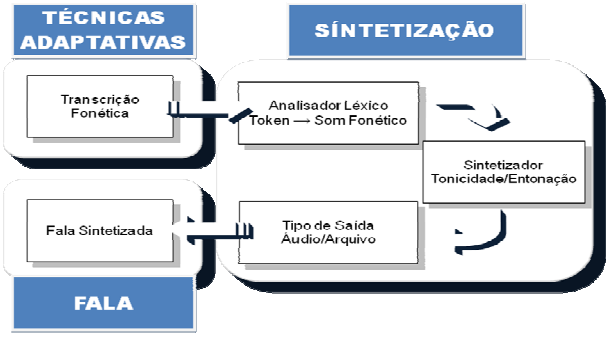


Fig. 3 – Ilustração do Tradutor Texto-Voz focando o processo de síntese.

V. Resultados Obtidos

Apesar de utilizar métodos simples para a tradução grafema-fonema e para o processo de síntese de voz a partir da linguagem fonética, o tradutor texto-voz apresentado obteve resultados satisfatórios e inteligíveis para pessoas com bom conhecimento da língua portuguesa.

Os maiores problemas em relação à tradução de texto para voz se encontram na definição do som da letra x que pode ter quatro sons diferentes (máximo, lixo, exato e fluxo), e das letras 'e' e 'o', quando estas duas últimas se encontram em uma sílaba tônica, sendo que alguns dos problemas com as vogais citadas foram resolvidas a partir do auxílio do etiquetador morfológico.

Eventualmente, a letra 'a' de uma sílaba átona que antecede uma letra nasal (o primeiro 'a' de "caminha" pode ter o som aberto ou fechado) ou uma junção de vogais anterior à sílaba tônica ("ia" forma ditongo em "financiamento", mas não em "adiamento") podem gerar problemas, mas são menos freqüentes e mais difíceis de resolver pois a escolha da forma a ser utilizada é opcional.

Em testes iniciais, foi obtida uma amostra de textos etiquetados com aproximadamente 9000 elementos (formados pelo par palavra e etiqueta). Desses 9000 elementos, aproximadamente 7000 tiveram tradução correta, 1600 geraram dúvidas por existência de 'e', 'o' ou 'x' e 400 falhas relacionadas à existência de 'a' pré nasal ou ditongo pré-tônico.

Para os casos em que houve dúvida, a escolha da saída mais provável gerou uma porcentagem de acerto um pouco superior a 50%, enquanto o uso do etiquetador levou a uma taxa de acertos próxima de 70%. Somando os resultados, o tradutor grafema-fonema obteve aproximadamente 87% de acerto sem o etiquetador e 90% com ajuda desse último.

Das traduções falhas, apenas 8 tiveram uma alteração grave que dificulta significativamente o entendimento (por exemplo, a palavra "porque" que é tônica final ou "anti" que não é, ao contrário do que definem as regras da língua porguesa), sendo

todas as outras falhas que não prejudicam significativamente o entendimento do usuário.

Os resultados obtidos apresentam uma significativa melhora em relação aos que foram obtidos anteriormente no trabalho de [5], que não separa os processos de tradução grafema-fonema, utiliza um autômato mais simples e não realiza processos de tratamento na saída gerada.

Apesar dessas melhoras, acredita-se que seja possível melhorar ainda mais os resultados a partir de um estudo mais aprofundado em outras técnicas de síntese e de melhorias pontuais no autômato existente.

VI. CONCLUSÕES

Este trabalho descreve um software para tradução texto-voz desenvolvido com base no modelo de autômatos apresentado em [2]. O software é dividido em duas partes, um tradutor grafema-fonema baseado no mesmo modelo e um sintetizador de voz baseado no processo de síntese por concatenação.

O software se encontra em estado inicial no processo de desenvolvimento e pode ser melhorado, com maior integração entre os módulos e com melhorias pontuais nos módulos de tradução grafema-fonema e de síntese de voz.

O software comprova a viabilidade do uso de autômatos adaptativos como base para a tradução texto-voz de textos escritos na língua portuguesa, e pode ser um incentivo para a utilização dessa mesma tecnologia em outras aplicações relacionadas ao processamento de linguagens naturais.

REFERÊNCIAS

[1] J. José Neto, “Adaptive Automata for Context-Sensitive Languages”, SIGPLAN NOTICES, Vol. 29, n. 9, pp. 115-124, September, 1994.

[2] International Phonetics Association. http://www.arts.glas.ac.uk/IPA/

[3] D. P. Shibata, R. L. A. Rocha, “An Adaptive Automata based method to improve the output of text-to-speech translators”, The Sixth Congress of Logic Applied to Technology, 2007.

[4] F. N. Kepler, “Um etiquetador morfo-sintático baseado em Cadeias de Markov de tamanho variável”, dissertação de mestrado, orientada por Marcelo Finger, Instituto de Matemática e Estatística, Univesidade de São Paulo,. 2005.

[5] D. A. Alfenas, A. J. B. Castro, D. P. Shibata, H. T. Soejima, “Sintetizador Texto-Voz com Autômatos Adaptativos”, trabalho de conclusão de curso, orientado por R. L. A. Rocha, Escola Politécnica, Universidade de São Paulo, 2004.

[6] F. R. Koike, L. R. E. Oliveira, R. H. S. Santos, T. M. Rizzo, “Síntese de Fala com apoio de Técnicas Adaptativas”, trabalho de conclusão de curso, orientado por R. L. A. Rocha, Faculdade Engenheiro Celso Daniel, Centro Universitário Fundação Santo André. 2007.

Danilo Picagli Shibata nasceu em São Paulo, Brasil em 22 de dezembro de 1981. Formou-se Engenheiro de Computação pela Escola Politécnica da Universidade de São Paulo em 2004 e cursa Mestrado em Engenharia de Computação pela mesma.

Exerceu profissionalmente pela Fundação para o Desenvolvimento Tecnológico da Engenharia (FDTE), na área de análise de confiabilidade e segurança de sistemas metro-ferroviários. Tem como áreas de interesse, Computação Formal e Processamento de Linguagens Naturais.

Fábio Robson Koike nasceu na cidade de Lins interior do estado de São Paulo, Brasil em 24 de novembro de 1972. Formou-se em Técnico em Processamento de dados em 1990 pelo Colégio Objetivo, Tecnólogo em Tecnologia da Computação em 2005 e Engenheiro da Computação em 2007 ambos no Centro Universitário Fundação Santo André - Faculdade Eng° Celso Daniel.

Atua na área de TI desde 1990 ocupando várias responsabilidades desde Estagiário em Processamento de Dados, passando por Programador Jr., Analista Sênior até Coordenador de Projetos e Arquiteto de Soluções em TI na atual ocupação. Tem experiência com projetos de Automação Industrial, Logística, Computação Embarcada, Aplicações Corporativas entre outros. Além de experiência profissional em TI trabalhou no exterior, especificamente no Japão onde domina o idioma, durante muitos anos exercendo a profissão de: Operador/Programador de Robô, Líder de produção em circuitos impressos e Gerente de Recursos Humanos.

Tem como áreas de interesse: Pesquisa e Desenvolvimento de: Processamento de Linguagens Naturais, Metodologia e Processos de Engenharia de Software.

Conversão de partituras para tablaturas usando algoritmo baseado em autômato adaptativo

Danilo de Jesus da Silva Bellini, Anna Catarina Batista Tavella

***Resumo*—Orfeu é um *software* que foi desenvolvido para auxiliar músicos amadores e profissionais na edição de músicas. Em suas funções, está inclusa a conversão de partitura para tablatura, realizada através de um algoritmo baseado em autômato adaptativo. O autômato é utilizado para trechos sem simultaneidade sonora enquanto que acordes e bicordes são tratados através de uma rotina específica.**

I. INTRODUÇÃO

O software Orfeu foi desenvolvido com o objetivo de auxiliar músicos amadores e profissionais na edição de músicas. Possui duas interfaces de edição: a de partitura e a de tablatura. A notação em partitura é, em geral, mais completa, por incluir informações sobre ritmos, dinâmicas, articulações e outras características comumente omissas em tablaturas segundo Junior (2007), além de ser uma notação musical historicamente mais importante (BENNETT, 1996), e a mais utilizada até os dias de hoje pelos músicos. A figura-1 a seguir apresenta as duas notações.

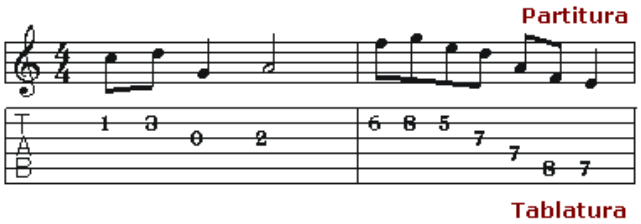


Figura-1: representação em partitura e em tablatura

Toda música no Orfeu é dividida em trilhas, cada uma com propriedades particulares, representando um instrumento musical pertencente à música. A interface visual utiliza apenas uma trilha por vez, exibindo seu conteúdo como partitura ou como tablatura.

A notação em tablatura é bastante utilizada por guitarristas e violonistas, pela maior proximidade com seu instrumento. Na tablatura, a única informação que é sempre apresentada é a altura das notas. Cada número representa uma altura, indicando uma posição ao longo do braço do instrumento (AZEVEDO, 1978), chamado de "casa". Também é comum o uso do termo "traste", principalmente em inglês (*fret*), por ser o nome técnico das barras metálicas que dividem o braço do instrumento em casas. O número zero representa que a corda deve ser tocada solta, ou seja, sem prendê-la a nenhum traste; outro número N indica que a corda deve ser tocada prendendo a ao N-ésimo traste, contando da mão do instrumento ao corpo do mesmo.

Uma funcionalidade do software Orfeu é a conversão entre essas duas formas de representação. Toda nota é sempre representada na partitura e na tablatura simultaneamente, embora suas tablaturas possam ser inválidas, por incluir posições inexistentes no braço do instrumento a ser utilizado. Ao inserir uma nota em alguma das interfaces de edição, é realizada uma conversão dessa informação, porém ignorando o contexto musical presente e, inclusive, a possibilidade de execução. Para contornar esse problema, há no software um recurso adicional de conversão de partitura para tablatura, utilizando-se de adaptatividade (NETO, 1994), em que o contexto musical e a dificuldade de execução são levadas em consideração. O algoritmo utilizado para fornecer esse recurso considera que a trilha a ser convertida já esteja completa, ou seja, que todo o contexto musical já pertença à mesma.

II. POSSIBILIDADES DE CONVERSÃO DE PARTITURA EM TABLATURA

A estrutura de dados da partitura e da tablatura é basicamente a mesma, mas para este algoritmo, a informação de como os dados são estruturados é irrelevante, importando apenas a maneira de representação das notas e sua ordenação. As notas em partitura são identificadas por um conjunto de três atributos (FAGUNDES, 2004): nome, acidente e oitava; enquanto as notas em tablatura são identificadas por dois atributos: corda e posição (LEAVITT, 2007). Porém, apesar de, aparentemente, as tablaturas serem mais simples de representar, elas exigem o conhecimento prévio do número de cordas do instrumento e da afinação de cada corda, dentro da pauta musical em análise.

Ao fixar uma corda, pode-se facilmente converter cada nota da partitura em uma equivalente na tablatura. Na tablatura, cada nota tem uma posição, um número inteiro que denota a diferença de altura, em semitons, entre a nota representada e a nota da afinação da corda fixada. O processo de obter intervalos em semitons entre duas notas é bastante comum e pode ser encontrado facilmente em livros que contenham os fundamentos da teoria musical (MED, 2001). Quando a nota representada é mais grave que a nota da afinação, e apenas nessa situação, o sinal da posição é negativo. Toda nota possui, portanto, uma posição na corda fixada, porém esse número pode ser negativo ou excessivamente grande. A faixa de valores permitidos é um parâmetro do algoritmo e seu valor usual é [0;24], por ser a faixa completa de valores possíveis na maior parte das guitarras produzidas no mercado. Conforme a afinação das cordas, uma mesma nota pode ser tocada de mais de uma forma.

Embora seja dito que a conversão realizada é de partitura para tablatura, pode-se dizer que esse algoritmo também converte uma tablatura em outra, pois apenas os intervalos, em semitons, são utilizados, e eles estão presentes em ambas as notações.

É dado o nome de "digitação" ao conjunto de posições determinadas pelo algoritmo. O algoritmo não é capaz de converter, de uma forma ideal, qualquer partitura em tablatura. O único critério utilizado foi: em uma melodia, as notas próximas entre si devem possuir as menores diferenças de posição possíveis.

A partir desse critério é possível criar diversos algoritmos de conversão, porém o uso de um algoritmo baseado em autômatos adaptativos mostrou-se adequado para efetuar tal conversão.

III. Divisão da Trilha

A trilha é dividida em partes para que seja possível a conversão, pois o autômato trabalha apenas com trechos puramente melódicos e contínuos, ou seja, sem pausas e sem simultaneidade de notas; ocorrências como essas servem de separadores entre as partes da trilha, criando três tipos de partes: trechos melódicos, pausa única e bloco com um único agrupamento de notas simultâneas.

As pausas não necessitam de tratamento, visto que não há nota alguma para converter. Elas não são tratadas junto aos trechos melódicos pois, apesar de serem um caso particular de cadeia de entrada nula, o autômato não está preparado para isso.

Os acordes, agrupamento de três ou mais notas simultâneas, e os bicordes, par de notas simultâneas, não são tratados por autômatos adaptativos. Há apenas uma lógica que tenta dispor as notas em cordas que façam a posição das mesmas ficar dentro da faixa de valores permitidos (corda permitida), fornecida como parâmetro. Essa lógica consiste em ordenar as cordas do instrumento da mais aguda até a mais grave e fazer o mesmo com as notas a serem colocadas na tablatura. Cada nota, verificando da mais grave à mais aguda, é colocada na corda permitida de afinação mais grave.

Cada trecho melódico é tratado de forma isolada, utilizando o autômato adaptativo na conversão. O trecho melódico é a entrada do autômato. É designado o nome de "melodia" para esses trechos melódicos.

IV. Cadeia de Entrada do Autômato Através de n Fitas

Trabalhando com cada melodia, pode-se obter, para cada corda, uma seqüência de números que indicam como a melodia seria tocada no instrumento utilizando apenas essa corda, por exemplo, para um baixo elétrico de quatro cordas com afinação padrão (GDAE), pode-se ter a melodia representada conforme mostrada na tabela-1 (as notas da melodia de exemplo são o início de Greensleaves).

Cada uma das linhas da tabela a seguir é considerada uma fita utilizada pelo autômato, apontadas por um cursor sobre uma única nota, ou seja, sobre uma coluna. O conjunto de todas as fitas do autômato forma a cadeia de entrada do mesmo, ou seja, cada símbolo de entrada é a n-upla formada por cada coluna da tabela de fitas.

Tabela-1: exemplo de representação de melodia

Afinação	Melodia	A3	C4	D4	E4	F4	E4	D4
G3		2	5	7	9	10	9	7
D3		7	10	12	14	15	14	12
A2		12	15	17	19	20	19	17
E2		17	20	22	24	25	24	22

Dessa forma o alfabeto de entrada fica definido como todas as n-uplas possíveis para o dado número de cordas. Dentro de cada n-upla pode-se ter como valores qualquer número da faixa de posições permitidas para a tablatura, ou um símbolo que indica que a nota é inválida na corda em questão, o símbolo "E", também utilizado quando não há nota, ou seja, quando o cursor passa a estar fora da fita (antes ou depois da melodia). Nesse último caso descrito, a n-upla contém apenas valores iguais a "E".

Todos os dados de entrada são mantidos imutáveis, e o cursor tem plena liberdade de mover-se tanto para a direita como para a esquerda.

V. Estados e Transições

O autômato possui um estado para cada corda (indicado por números), um estado de aceitação, um estado de rejeição e um estado de transição (T), com transições conforme constam abaixo no exemplo com três cordas. Cada transição pode ter até três valores. O primeiro valor indica o que será consumido, o segundo valor indica a ação adaptativa que será realizada após efetuar a transição e o terceiro valor indica para que lado o cursor das fitas deve mover-se. Apenas o primeiro valor não pode ser omitido.

O operador unário "~" é usado para indicar um conjunto complementar, ou seja, ~K é o conjunto complementar a K, e K U ~K é o alfabeto de entrada, para qualquer conjunto K pertencente a esse alfabeto. As transições habilitadas apenas por (E,..,E) têm prioridade sobre as demais, por indicar que as fitas terminaram, resolvendo um problema de não-determinismo que ocorre em quase todos os estados.

Quando a cadeia de entrada possuir um elemento que faça parte do conjunto de símbolos de entrada Xc(i), a transição indicada por esse conjunto estará habilitada. Os conjuntos Xc(i) são alterados dinamicamente através das ações adaptativas Bx e Ux, isto é, essas ações inserem e removem transições ao colocar (ou retirar) um símbolo de Xc. Por exemplo, se o símbolo (0,...,0) estiver em Xc(0) então há uma transição do estado 0 para si mesmo, a partir dessa entrada; e ao retirar o símbolo do conjunto Xc(0) essa transição deixa de existir e passa a existir uma do estado 0 para o estado T, com a mesma entrada, caracterizando a adaptatividade (NETO, 1994).

L e R representam o movimento do cursor das fitas para a esquerda e para a direita, respectivamente, baseando-se no fato de que a primeira nota da parte analisada da música está sempre na extremidade esquerda da fita e é sempre a primeira a ser apontada. Transições sem essas indicações não movem o cursor. A figura-2 a seguir mostra um esquema do autômato utilizado pelo algoritmo.

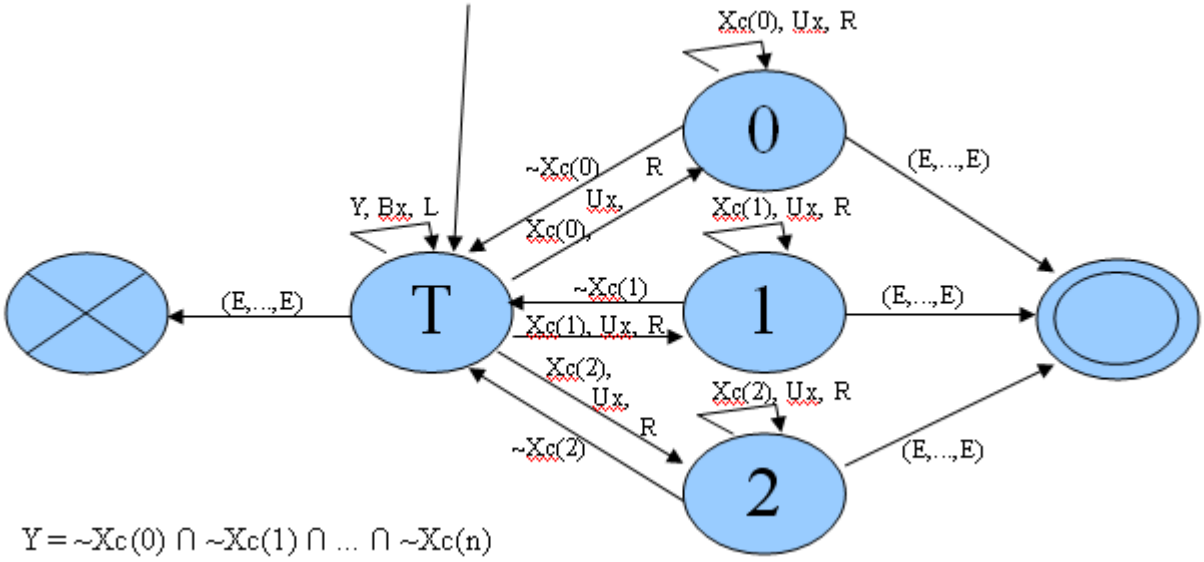


Figura-2: Esquema do autômato

VI. Lógica do Autômato

O autômato procura manter-se em uma mesma corda pelo maior número possível de notas, realizando uma busca em profundidade na árvore implícita nas possibilidades de combinação de cordas. Um exemplo de execução do autômato pode ser imaginado, a partir dos estados, como sendo:

T→0→0→T→1→1→T→4→T→T→5→T→2→2→Aceitação

Esse processo indica que o autômato iniciou no estado T (sempre utilizado como estado inicial) e já partiu para o estado 0, deixando a nota inicial sobre a corda 0, deixou a nota seguinte na mesma corda e verificou que a nota seguinte não podia estar nessa corda. A partir disso foi ao estado de troca e percebeu que poderia utilizar a corda 1 para essa nota que não podia estar na corda 0, e logo a utilizou, indo ao estado 1. A nota seguinte não apresentou problemas em estar nessa mesma corda, porém seguiu-se uma nota que não podia estar na corda 1, mas podia estar na corda 4 e 5. A escolha inicial foi a corda 4 e o autômato descobre que essa corda é inválida (a transição de T para T é a única que volta o cursor). Escolhendo a corda 5 o autômato continuou a busca normalmente até a aceitação.

Logo Xc(i) representa o conjunto de símbolos que são válidos para o autômato continuar a busca, se aprofundando na i-ésima corda. Para simplificar a visualização, não é necessário visualizar sempre cada n-upla (ou símbolo) de Xc(i) por completo, apenas o i-ésimo elemento da n-upla, ou seja, o número que representa a posição da nota na i-ésima corda. O conjunto Xc(i) conterá esse símbolo se, e somente se, esse número estiver na faixa [`MinX`;`MaxX`]. Dessa forma com apenas dois números inteiros é possível definir todos os conjuntos Xc(i).

Seja a função X(i) que retorna um valor booleano para a i-ésima corda. O significado de X(i) é: se Xc(i) contém a entrada apontada pelo cursor, então retorna verdadeiro, caso contrário retorna falso. Dessa forma X(i) serve como teste para verificar qual é o próximo estado de maneira mais ágil que a verificação individual com cada símbolo de Xc(i). Para realizar isso, X(i) apenas verifica se o valor da posição na i-ésima corda está na faixa [`MinX`;`MaxX`].

Os valores iniciais de `MinX` e `MaxX` a cada execução do autômato devem ser, obviamente, o limite inferior e superior da faixa de posições permitida na tablatura, respectivamente.

Há uma indeterminação no estado T, pois X(i) pode ser válido para mais de um valor de i. Nesse caso a varredura é feita do menor valor de i até o maior valor de i, conforme exemplificado. Isso justifica que sempre o segundo estado é o 0.

As ações adaptativas Ux (*Update* X) e Bx (*Backup* X) servem basicamente para atualizar os valores de MinX e `MaxX`. Dessa forma os conjuntos Xc(i) serão alterados, e conseqüentemente a função X(i). A maneira como `MinX` e `MaxX` são alterados é descrita adiante.

Cada transição com R atribui o número da corda como saída, mas cada transição com L indica que essa saída era incorreta, "retirando" esse número da saída. Isso pode ser feito imaginando-se uma fita adicional "de saída", com números que indicam a fita escolhida para aquela nota. A tabela-2 abaixo apresenta os números na fita de saída.

Tabela-2: Números da fita de saída

Afinação / Melodia	A3	C4	D4	E4	F4	E4	D4
G3 (Fita 0)	2	5	7	9	10	9	7
D3 (Fita 1)	7	10	12	14	15	14	12
A2 (Fita 2)	12	15	17	19	20	19	17
E2 (Fita 3)	17	20	22	24	25	24	22
Saída	2	1	1	0	0	0	1

VII. Algoritmo

Todo o núcleo do funcionamento desse algoritmo está nas ações adaptativas que atualizam o `MinX` e o `MaxX`. Essas ações

visam a obedecer ao critério previamente estabelecido de que as distâncias entre as posições devem ser as menores possíveis.

Diferenças de três posições na tablatura em um trecho de poucas notas serão consideradas sempre possíveis de serem executadas, pois supondo que o músico tenha quatro dedos, todas as notas são possíveis de serem mapeadas em um dedo a partir do número da posição, e, portanto, esse é um bom número inicial para o algoritmo.

O algoritmo basicamente repete a execução do autômato iterativamente para cada valor de faixa de distâncias (ou diferença de posições). O valor é incrementado a cada iteração. O algoritmo, visto em alto nível, em uma pseudo-linguagem:

```
Repetir para cada melodia
 Faixa de distâncias := 3
  Enquanto (Executa autômato) não retorna aceitação
   Faixa de distâncias := Faixa de distâncias + 1
  Impõe fita saída: cordas utilizadas para a melodia
 Até acabar a música
```

Essa faixa de distâncias é importante para o autômato, pois é o menor valor que `MaxX – MinX` pode assumir, ou seja, é o menor tamanho do intervalo `[MinX;MaxX]`.

VIII. AÇÕES ADAPTATIVAS

Nem sempre é necessário fazer a melodia inteira ficar na faixa de distâncias descrita. Por exemplo, se há uma melodia com vinte notas, a primeira nota não precisa estar na mesma região que a vigésima, ou seja, não é interessante forçar que a melodia inteira esteja sempre dentro da faixa de distâncias.

Seja W a janela que contém o conjunto de valores numéricos das últimas posições utilizadas como saída (e.g. no exemplo da Greensleaves seria 12, 10, 12, 9, 10, 9, 12). O tamanho dessa janela indica quantos números serão utilizados. Para isso é necessário uma pilha que armazene todos os valores já utilizados. Seja `fretStack` essa pilha. A janela sempre se resume a indicar os elementos do topo dessa pilha.

O tamanho da janela não pode ser demasiado pequeno, pois tornaria irrelevante a importância da faixa de distâncias. Se o tamanho da janela for menor do que 4, praticamente qualquer entrada tornar-se-á válida, em particular a escala cromática ficaria toda na mesma corda (0,1,2,3,4,5,6,...). Se o tamanho da janela for quatro, esse valor continua pequeno demais para garantir que a digitação será boa caso haja repetição de números, como a seqüência (0,1,1,2,3,3,4,5,5). No caso em que há uma repetição de todas notas, comum na presença do chamado "tremolo picking"(TURNER and WHITE, 1988), o primeiro valor a garantir a separação para outra corda é 7, pois (0,0,1,1,2,2,3,3) tem oito elementos (portanto o primeiro zero é descartado) e o elemento seguinte não poderá ser 4 caso a faixa de distância seja a inicial, 3. Valores maiores são um exagero na maior parte dos casos, pois essa janela de tamanho N obriga que qualquer conjunto de N notas adjacentes esteja dentro da faixa de distâncias. Sete notas é o suficiente para haver uma mudança de contexto e um pequeno deslocamento na região da melodia. Dentro da janela há um valor máximo e mínimo, sejam esses `MaxW` e `MinW`, respectivamente.

A ação Ux inicialmente empilha o novo valor de posição em `fretStack` para garantir que esse valor não será perdido e esteja na janela quando necessário. A ação Bx faz o inverso, descartando o valor do topo dessa pilha.

As ações Ux e Bx atualizam as variáveis `MinX` e `MaxX` e, conseqüentemente, a função X(i), indicando nova a faixa de valores permitidos para a posição da nota. Essa faixa pode ser facilmente calculada tendo todos os valores da janela de posições usadas, bastando possuir o valor máximo e mínimo dentro da janela. O valor de `MinX` e `MaxX` passam a ser os limites da janela, acrescidos de quanto falta para completar a faixa de distância em ambos os sentidos (diminuir `MinW` e aumentar `MaxW`), de forma que garanta que o próximo valor, estando nessa faixa, é incapaz de tornar a nova amplitude da janela (`MaxW - MinW`) maior que a faixa de distâncias:

```
MinX = MinW – FaixaDistâncias + (MaxW – MinW)
MaxX = MaxW + FaixaDistâncias – (MaxW – MinW)
```

Obviamente para uma janela vazia, `MinX` e `MaxX` serão iguais aos limites da faixa de valores permitidos para as posições, e após a aplicação das fórmula acima deve-se sempre saturar os valores de tais variáveis nessa faixa de valores permitidos.

IX. CONCLUSÃO

O algoritmo funciona bem para melodias, conjunto de notas sucessivas e totalmente sem simultaneidade sonora, sem pausas muito curtas (pausas são consideradas curtas quando têm duração inferior a uma colcheia). Entretanto, se houver grandes discrepâncias de altura (três oitavas, por exemplo) o algoritmo é lento. É interessante limitá-lo a poucos trastes quando isso for possível.

Embora não haja ganhos consideráveis de *performance* ou até mesmo de resultado utilizando o algoritmo aqui descrito na versão inicial em relação aos algoritmos convencionais, outros pontos podem ser explorados para melhorá-lo, como por exemplo o tratamento de notas múltiplas (bicordes e acordes) ou o tratamento de ornamentos existentes. Com tais melhorias, esperamos obter resultados melhores e mais abrangentes.

REFERÊNCIAS

[1] J. Junior, *O básico de guitarra*, Guitar X. Acesso em 17/12/2007. Disponível em: http://www.guitarx.com.br/index.asp?url=library_html/basico_guitarra/basico_da_guitarra%206_10.htm.

[2] R. Bennett, *Uma breve história da música*, Ed. Jorge Zahar Editor Ltda, 1996, 80p.

[3] F. Azevedo, *Método para Guitarra*, Ed. Bruno Quaino Editores, 1978, 16p.

[4] J. J. Neto, *Adaptive Automata for Context -Sensitive Language*. SIGPLAN NOTICES, Vol. 29, n. 9, pp. 115-124, September, 1994.

[5] M. D. Fagundes, *Teoria da Música*, Ed. Keyboard, 2004, 183p.

[6] W. Leavitt, *Método Moderno para Guitarra*, 1a Ed., Vol. 1, Ed. Irmãos Vitale, 2007, 128p.

[7] B. Med, *Teoria da Música*, 4a ed., Ed. Musimed, 2001, 420p.

[8] G. Turner, B. White, *Progressive Lead Guitar*, USA, Ed. Koala Publications, 1988, 94p.

[9] H. Pistori. *Tecnologia Adaptativa em Engenharia de Computacão: Estad. da Arte e Aplicações*. Tese (Doutorado) — Universidade de São Paulo.

REGULAR PAPERS

Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão: Uma Abordagem Estratégica na Seleção de Fornecedores

A. H. Tchemra, R. Camargo

Resumo. **Este artigo tem como objetivo apresentar como a Tecnologia Adaptativa pode ser aplicada, como forma alternativa de solução, em sistemas de apoio à decisão. A técnica utilizada é a da Tabela de Decisão Adaptativa, que tem como proposta um algoritmo resultante da combinação dos métodos multicritério, do uso de funções de utilidade e das técnicas adaptativas. Sua aplicabilidade é mostrada em um estudo sobre a homologação de fornecedores, como parte de um processo estratégico de compras da área de suprimentos de uma empresa. A tomada de decisão nesta área, geralmente, envolve múltiplos critérios com dados qualitativos e/ou quantitativos, nem sempre precisos ou completos.**

Palavras-chave: **tecnologia adaptativa, tabela de decisão adaptativa, tomada de decisão, múltiplos critérios, suprimentos.**

I. Introdução

A dinâmica e, principalmente, a complexidade dos processos de decisão exigiram estudos e pesquisas, tanto na área da teoria da decisão, quanto nas técnicas de desenvolvimento de softwares. Metodologias e métodos de auxílio à decisão para problemas que envolvam, entre outros, múltiplos critérios são pesquisados, e aliados à evolução da tecnologia da informação, permitiram a geração de programas computacionais que apresentam aos decisores, em alguns casos, a melhor alternativa de solução, e em outros, um conjunto de soluções possíveis para os problemas de decisão.

Os sistemas computacionais que servem de apoio aos processos de tomada de decisão, geralmente, utilizam bancos de dados com informações coletadas previamente. Esses sistemas, além de apresentar processos de realimentação de dados, permitem aos decisores criar vários cenários sobre um determinado problema de decisão [1]. Das análises dos cenários e da decisão tomada, outras informações são obtidas e armazenadas para servirem de entrada para processos semelhantes.

De acordo com [2], pesquisas em inteligência artificial permitem o desenvolvimento de ferramentas computacionais inteligentes que auxiliam os usuários na execução de tarefas, entre elas, em processos de tomada de decisão. Estas ferramentas são associadas à inteligência humana, como por exemplo, à capacidade de aprender, raciocinar e solucionar problemas.

Segundo [3], a Tecnologia Adaptativa é uma área da computação cujos estudos e pesquisas sobre técnicas adaptativas possibilitam aos dispositivos adaptativos apresentar como característica principal a capacidade de se automodificarem, 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.

O objetivo deste artigo é apresentar o modelo de um sistema adaptativo de tomada de decisão baseado em Tabelas de Decisão Adaptativas. Como parte dos estudos de pesquisa, é proposto um algoritmo para o modelo, que combina técnicas adaptativas e métodos multicritério de decisão.

Também faz parte da pesquisa, a aplicação da tabela de decisão adaptativa na área de gestão de negócios. Optou-se pelo estudo de caso no processo de compras da área de suprimentos, onde, de acordo com [4], decisões de compras deixaram de ser isoladas, e estão cada vez mais integradas com decisões de outras áreas, tais como engenharia, qualidade, finanças, entre outras, e que exercem um papel igualmente estratégico dentro de uma organização.

II. Tecnologia Adaptativa

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

Nos dispositivos adaptativos desenvolvidos, foram empregados formalismos conhecidos e tradicionais, tais como autômatos de pilha estruturados, *statecharts*, redes de Markov, gramáticas, árvores de decisão, tabelas de decisão, entre outros

A. H. Tchemra – Universidade Presbiteriana Mackenzie (correspondência: R. Jorge Tibiriçá, 74 – ap.64 – São Paulo, SP, Brasil – cep: 04126-000; e-mail: angela.tchemra@poli.usp.br).
R. Camargo – Universidade Presbiteriana Mackenzie (correspondência: R. Oscar Freire, 235 – ap.31- São Paulo, SP, Brasil – cep: 01426-001; e-mail: rubens.camargo@poli.usp.br).

[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.

O formalismo geral do dispositivo adaptativo definido por [3], também denominado dispositivo dirigido por regras adaptativo, caracteriza-se por possuir um dispositivo subjacente cujas operações são definidas por um conjunto finito de regras, e que se automodificam em decorrência de ações adaptativas.

As ações adaptativas são recursos implementados na camada adaptativa através de funções adaptativas 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.

Um exemplo de dispositivo dirigido por regras adaptativo é a Árvore de Decisão Não-Determinística Adaptativa definida em [5]. O dispositivo adaptativo simula uma árvore de decisão, que é percorrida a partir da sua raiz (regra inicial) e chega a uma folha (caso determinístico) ou encontra vários ramos a serem percorridos (não-determinístico), após uma seqüência de testes em cada nó (regra). Quando não é possível atingir uma folha, ações adaptativas podem ser executadas na sub-árvore, realizando ações de consultas ou de remoção da sub-árvore ou de inclusão de uma nova sub-árvore à estrutura não-adaptativa.

O dispositivo dirigido por regras adaptativo definido em [3], tem como núcleo uma tabela de decisão para simular um autômato adaptativo. E, o fato de que as árvores de decisão poderem ser representadas na forma de tabelas de decisão, justificam a proposta de estudar a Tabela de Decisão Adaptativa aplicada à área de tomada de decisão.

III. TABELA DE DECISÃO ADAPTATIVA

A tabela de decisão convencional apresenta características que facilitam a sua utilização e compreensão de seus dados. Auxilia na visualização e comunicação de informações entre profissionais e pode ser aplicada em sistemas baseados em conhecimento, como por exemplo, em processos que envolvam decisões [6] [7].

Uma tabela de decisão é construída de maneira que as condições (ou critérios) e as ações (ou alternativas) sejam facilmente identificadas pelo usuário [6] [7], assim como as regras do problema de decisão.

Utilizando o formalismo de uma tabela de decisão e a definição do dispositivo dirigido por regras adaptativo, [8] aplica a Tabela de Decisão Adaptativa para simular um autômato adaptativo para reconhecer sentenças de linguagens dependentes de contexto. À tabela convencional, Neto acrescenta as funções adaptativas, cujas ações adaptativas, quando executadas, modificam o conjunto de regras da tabela, através de inclusões e/ou exclusões de regras, de acordo com as entradas recebidas, de forma autônoma e dinâmica.

A estrutura geral de uma Tabela de Decisão Adaptativa (TDA) é encontrada em [9], e reproduzida aqui. O formato adotado é baseado tanto nas tabelas convencionais [6] [7], como no dispositivo adaptativo apresentado em [3].

A organização das informações na TDA é mostrada na figura 1.

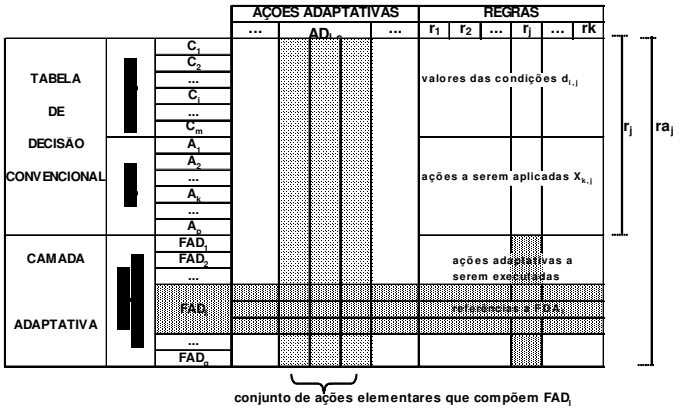


Figura 1 – Estrutura geral da TDA

A TDA pode ser definida pela dupla TDA = (TDN, CA), onde TDN representa a tabela de decisão convencional (não-adaptativa) e CA o mecanismo adaptativo.

A tabela de decisão não-adaptativa é composta pelos elementos TDN = (CT, t_0, R, C, A) onde:

- CT é o conjunto de todas as configurações possíveis da tabela de decisão;
- $t_0 \in$ CT é a configuração inicial da tabela de decisão;
- R é o conjunto de regras de decisão da tabela: $R = \{ r_j , 1 \le j \le n \}$
- C é o conjunto finito das condições (ou critérios) do problema: $C = \{ C_i , 1 \le i \le m \}$
- A é o conjunto finito de ações possíveis (ou alternativas) do problema: $A = \{ A_k , 1 \le k \le p \}$

Cada regra $r_j \in R$ é formada por $r_j = (d_{i,j}, X_{k,j})$, onde:

- $d_{i,j}$: representa o conjunto de valores das condições C_i na regra r_j;
- $X_{k,j}$: é o conjunto de valores assinalados para as ações A_k a serem executadas na regra r_j.

O mecanismo adaptativo CA para a TDA é definido pelo conjunto de funções adaptativas:

- $FDA = \{ FAD_l , 1 \le l \le q \}$

onde cada função FAD_l é composta por um conjunto de ações adaptativas:

$AD = \{ AD_{l,c} , 1 \le l \le q \text{ e } 1 \le c \le o \}$

As ações adaptativas $AD_{l,c}$ são definidas para cada função adaptativa FAD_l e podem ser executadas antes ou depois das regras não-adaptativas da tabela, de acordo com os parâmetros de entrada das funções. Cada ação adaptativa é definida por, pelo menos, uma das ações elementares: consulta às regras da tabela, inclusão de novas regras e exclusão de regras existentes; pode existir mais de uma ação elementar para cada função adaptativa.

Além das funções adaptativas, a camada CA possui o conjunto de regras RA composto das regras adaptativas:

$RA = \{ ra_j , 1 \le j \le n \}$.

Cada regra adaptativa ra_j é definida por (r_j, ba, aa), onde r_j é a regra não-adaptativa e:

- ba ∈ $AD_{l,c}$: se houver, indica ação adaptativa a ser executada antes da regra r_j
- aa ∈ $AD_{l,c}$: se houver, indica ação adaptativa que será executada depois da regra r_j

IV. TOMADA DE DECISÃO

Tomar decisão é um processo em que o decisor deve optar por uma alternativa dentre duas ou mais, para que se atinjam os objetivos de um problema de decisão [10].

Dependendo da complexidade, da natureza do problema e dos objetivos a serem alcançados, uma decisão pode ser tomada individualmente ou pode requerer a participação de um grupo de pessoas [1]. Além disso, em um processo de decisão deve-se levar em consideração tanto os elementos objetivos, próprios do problema e das ações, como os elementos subjetivos, que dependem dos valores e julgamentos dos tomadores de decisão [11], os quais nem sempre ficam claros nos modelos adotados na decisão.

Casos complexos de problemas de tomada de decisão podem ser encontrados, por exemplo, no planejamento estratégico de uma organização. As metas definidas no planejamento determinam em que medida os processos de decisão devem agregar valor e de que forma as ações escolhidas trarão resultados que se mostrem eficazes para que os objetivos da empresa sejam alcançados [12].

Dentre os vários estudos sobre processos de decisão encontrados na literatura, destaca-se o da escolha racional clássica [1]. Esse modelo reúne dados que representam as variáveis de controle que determinarão quais são as ações alternativas, bem como as variáveis incontroláveis que são relevantes ao problema em questão, e os critérios de decisão que podem levar à melhor ação dentre as apresentadas.

Outros modelos de busca da solução de problemas de decisão são abordados em [12]. Existem modelos que dependem de decisões programadas que, normalmente, são dirigidas por regras, procedimentos ou métodos quantitativos bem definidos e estruturados. No entanto, existem também decisões não-programadas, mais difíceis de quantificar e exigidas em situações excepcionais. São problemas desestruturados, onde a aplicação de regras ou de procedimentos nem sempre são claros. E, ainda, são encontrados problemas de decisão semi-estruturados que contemplam características estruturadas e outras não-estruturadas.

Neste artigo, são tratados modelos de decisão semi-estruturados, cujos processos de tomada de decisão, nas suas diferentes fases, permitem identificar as causas do problema, as alternativas de solução e os critérios a serem atendidos para a escolha de uma decisão. Os métodos tratados aqui são relacionados, portanto, aos problemas de decisão multicritério.

V. MÉTODOS MULTICRITÉRIO

Os métodos multicritério são técnicas que oferecem maior compreensão dos problemas de decisão e de seus processos decisórios, quando inseridos em contextos, geralmente, multidisciplinares.

Segundo [13], os métodos multicritério apóiam os processos de análise de decisão, conferindo-lhes consistência e confiabilidade nos julgamentos do decisor. De acordo com [14], dependendo de como o problema de decisão multicritério é formulado, existem métodos multicritério de escolha, classificação e ordenação das alternativas presentes no problema.

Métodos multicritério de escolha permitem que uma única alternativa seja selecionada de um subconjunto obtido do conjunto de opções disponíveis. Os métodos de classificação consistem em distribuir cada alternativa na categoria apropriada, de acordo com um valor associado à alternativa obtido por funções de utilidade. Os métodos de ordenação estabelecem uma ordem de preferência das alternativas, segundo julgamentos dos decisores que atribuem níveis de importância a cada alternativa em relação aos objetivos do problema de decisão [14] [13].

Dos inúmeros métodos multicritério de apoio à decisão, [13] destaca duas principais famílias de métodos: as que são baseadas na Teoria da Utilidade Multiatributo (MAUT – *Multiple Attribute Utility Function*) e as de subordinação (*Outranking*).

Podem ser destacados os métodos SMART (*Simple Multi-Attribute Rating Technique*), AHP ((*Analytic Hierarchy Process*), entre outros, que fazem parte da família MAUT, nos quais são aplicadas funções de utilidade que refletem a utilidade de cada critério em relação às alternativas disponíveis.

Enquanto, os métodos ELECTRE (*Elimination and Choice Translating Reality*), PROMETHE (*Preference Ranking Method for Enrichment Evaluation*), MACBETH (*Measuring Attractiveness by a Categorical Based Evaluation Technique*), TOPSIS (*Technique for Order Preference by Similarity to Ideal Solution*), entre outros, pertencem à família dos métodos *Outranking* que determinam subconjuntos ou categorias das alternativas segundo níveis de preferência.

Os métodos multicritério basicamente utilizam uma matriz, chamada matriz de decisão, onde os valores dos julgamentos do decisor são organizados. Para problemas de decisão com m critérios e n alternativas, representados na matriz de decisão (figura 2) tem-se:

		x_1	.	.	x_j	.	.	x_n
		A_1	.	.	A_j	.	.	A_n
w_1	C_1	a_{11}	.	.	a_{1j}	.	.	a_{w1}
.	.	.	.	.	.	.	.	.
w_i	C_i	a_{i1}	.	.	a_{ij}	.	.	a_{in}
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
w_m	C_m	a_{w1}	.	.	a_{wj}	.	.	a_{wn}

Figura 2 – Matriz de decisão [13]

- conjunto de critérios C = { C_i , $1 \le i \le m$ }
- conjunto de alternativas A = { A_j , $1 \le j \le n$ }
- pesos w_i que são atribuídos a cada critério C_i

- pontuação a_{ij} que representa o desempenho da alternativa A_j em relação a cada critério C_i
- valores x_j associados às alternativas A_j que determinam a classe a que pertencem.

Os pesos w_i, as pontuações a_{ij} e os valores x_j variam de método para método, são determinados por funções de utilidade de acordo com o método, devem refletir os julgamentos do decisor e são avaliados para a tomada de decisão.

VI. TABELA DE DECISÃO ADAPTATIVA E MÉTODOS MULTICRITÉRIO

Os estudos e pesquisas sobre dispositivos adaptativos e sobre métodos multicritério possibilitaram desenvolver a proposta de um modelo de Tabela de Decisão Adaptativa (TDA) para aplicações de tomada de decisão, baseado nas técnicas adaptativas e nos procedimentos de métodos multicritério.

Optou-se pelos métodos multicritério de ordenação e AHP, este proposto em [15], por apresentarem características desejáveis para a implementação do modelo TDA, tais como: processos de decisão bem estruturados, possibilidade de aplicação a problemas que apresentam dados qualitativos e quantitativos, existência de medidas de consistência dos julgamentos realizados, além de se encontrar vasta quantidade de publicações científicas e práticas dos métodos.

O modelo proposto para a TDA tem como objetivo apresentar, ao final do processo decisório, um conjunto de alternativas que determinam possíveis soluções viáveis para o problema de decisão multicritério, segundo as regras apresentadas.

Nesse sentido, a proposta para o processo decisório com o uso da TDA na tomada de decisão é descrito a seguir:

a) identificar o tipo de problema de tomada de decisão, para que haja uma compreensão clara do mesmo;
b) definir os objetivos do problema;
c) definir os critérios do problema;
d) levantar os requisitos de cada critério, para que as condições indiquem possíveis soluções que possam atender aos objetivos do problema;
e) identificar as alternativas do problema, pois diferentes abordagens das condições podem alterar o conjunto de soluções viáveis;
f) verificar se as regras especificadas atendem às necessidades do problema;
g) criar a tabela de decisão convencional composta pelos m critérios (linhas de condições), n alternativas (linhas de ações) e k regras (colunas de regras) do problema de decisão, cuja configuração inicial é mostrada na figura 3.

Na figura 3, cada d_{ip} pode assumir o valor "S" ou "N", para representar se para aquele critério C_i na regra R_p, o critério existe ou não, respectivamente.

Cada X_{jp} é igual a "X" ou está em branco para indicar se a alternativa A_j será executada na regra R_p ou não.

A partir da tabela de decisão, o processo decisório pode ser realizado de forma estática ou dinâmica. A escolha pelo processo dinâmico permite que o modelo da TDA tenha características de aprendizagem. Para isso, o modelo utiliza um algoritmo que utiliza parte de procedimentos baseados nos métodos multicritério AHP e de ordenação, e das técnicas adaptativas.

	Regras					
	R_1	R_2	...	Rp	...	R_k
Critério C_1						
Critério C_2						
...						
Critério C_i				d_{ip}		
...						
Critério C_m						
Alternativa A_1						
Alternativa A_2						
...						
Alternativa A_i				X_{jp}		
...						
Alternativa A_n						

Figura 3 – Tabela de decisão convencional

O algoritmo de realização do processo decisório utilizando a TDA proposto é descrito a seguir.

Passo 1 – Ordenar os critérios de acordo com a sua importância

1.1 Montar uma matriz de comparação entre os critérios para obter os pesos w_i (figura 4). Os pesos w_i indicam a importância relativa dos critérios C_i, como no método AHP.

Para o cálculo dos pesos w_i de cada critério C_i, optou-se por, primeiramente, calcular cada p_i como média geométrica dos valores de julgamento da linha do critério C_i, e em seguida, os pesos p_i foram normalizados para se obter os pesos finais w_i. Quanto maior o peso w_i, mais importante é o critério C_i para o decisor.

	C_1	C_2		C_J		C_m
C_1						
C_i						
C_m						

Figura 4 – Matriz de comparação entre os critérios

1.2 Ordenar os critérios de acordo com a sua importância na tabela de decisão, para obter uma nova configuração da tabela de decisão.

Passo 2 – Calcular os valores de categorização das alternativas

2.1 Montar a matriz de importância de cada alternativa A_j em relação ao critério C_i

Como no método AHP, são realizados os julgamentos para determinar as pontuações do desempenho relativo de cada

uma das alternativas em cada critério. Obtidas as pontuações, as alternativas são avaliadas usando uma das técnicas de agregação dos métodos MAUT, que resultam na matriz da figura 5, onde:

a_{ij} = importância (desempenho) da alternativa A_j em relação ao critério C_i

	A_1	A_2		A_j		A_n
C_1						
C_i				a_{ij}		
C_m						

Figura 5 – Matriz de desempenho das alternativas em relação aos critérios

2.2 Em seguida, são calculados os valores X_j associados às alternativas A_j da tabela e equivalem aos valores de categorização das alternativas.

Passo 3 – Ordenar as regras da tabela de decisão

Para ordenar as regras da tabela de decisão, são obtidos os valores Y_k associados à cada regra R_k, que dependem dos pesos w_i de cada critério C_i e dos valores X_j associados às alternativas A_j.

Os valores Y_k são ordenados (ordem decrescente) do maior para o menor, indicando a nova ordem das regras na tabela de decisão. O objetivo de ordenar as regras visa melhorar o desempenho do processo de tomada de decisão.

A tabela de decisão, cujas regras estão ordenadas, apresenta as possíveis soluções do problema de decisão. Ela pode ser utilizada para buscar a melhor alternativa que satisfaça os critérios de entrada do usuário.

Nesse caso, as funções adaptativas da camada adaptativa poderão ser executadas.

Passo 4 – Executar as funções adaptativas

As funções adaptativas serão aplicadas para que a TDA se automodifique, criando ou eliminando regras da tabela, e desta forma, aprendendo com o processo.

As ações elementares da camada adaptativa são descritas a seguir:

1) ação de consulta à regra – com os parâmetros de uma regra, é possível fazer uma consulta para verificar a alternativa a ser executada;

2) ação de inclusão de regras – quando uma regra consultada não é encontrada na tabela, a ação adaptativa inclui uma nova regra R_p com as características solicitadas; a ação adaptativa associa um valor Y_p para a regra e assinala a alternativa, de acordo com o valor encontrado;

3) ação de exclusão de regras – após análise das regras da tabela de decisão, são excluídas as regras cujos critérios têm parâmetros que são indiferentes à regra, gerando apenas uma regra combinada; nesse caso, o correspondente parâmetro do critério fica em branco;

4) uma nova configuração da tabela é obtida com as modificações executadas pelas ações adaptativas.

Passo 5 – A nova tabela de decisão pode ser utilizada novamente, repetindo-se o passo 3.

VII. APLICAÇÃO DA TDA NA TOMADA DE DECISÃO

Um estudo de caso é apresentado neste artigo, cuja proposta é utilizar a TDA para aplicações de tomada de decisão no processo de homologação de fornecedores durante o processo de compras de suprimentos de uma empresa.

A atividade básica da função de compras estabelece contratos com fornecedores para aquisição de materiais e serviços, que podem ser utilizados diretamente na produção de bens e serviços. Outras aquisições são utilizadas para apoio às operações, tais como óleo lubrificante para equipamentos, alimentos para operários, etc, que apesar de não fazerem parte do produto, são essenciais para a produção.

Os fornecedores passam, então, a ter uma ligação importante com a empresa cliente. Portanto, os gestores devem ter uma compreensão das capacitações dos seus parceiros.

Com relação aos objetivos tradicionais da função de compras, a empresa normalmente adquire quantidades expressivas de produtos e serviços, e o volume de valores envolvido nessas transações tem aumentado à medida que as organizações têm-se concentrado em seus processos fundamentais.

A variedade de compras que uma empresa realiza, determina alguns procedimentos básicos, válidos para todos os materiais e serviços comprados. Conforme [16], para atender aos objetivos de desempenho da produção, os seguintes critérios devem ser considerados para a escolha dos fornecedores:

- os produtos devem ter qualidade que atendam as especificações do projeto e os padrões da produção;
- os produtos devem ser entregues rapidamente, se necessário, para suprir demandas imprevistas;
- os produtos devem ser entregues no momento certo e na quantidade correta, conforme as condições da compra, o que caracteriza a confiabilidade do fornecedor;
- os produtos devem ter preço adequado às necessidades da produção e do mercado;
- os fornecedores devem ter flexibilidade e serem capazes de alterar os termos de entrega e/ou quantidade dos produtos.

O relacionamento com fornecedores é um processo que envolve a seleção do parceiro adequado. É baseado na combinação de estratégias corporativas, permitindo que a eficácia da relação melhore a competitividade, reforçando a importância estratégica da gestão de suprimentos.

A escolha adequada de uma parceria certamente pode resultar em uma ação que possa impactar na lucratividade da empresa. Uma escolha inadequada trará problemas na empresa como um todo, resultando em processos de substituições, mudanças que influenciam nos custos, desgastes com a organização, perdas de sinergia, podendo afetar até mesmo em perdas de clientes.

No mercado podem-se observar empresas que adotam diversas estratégias de suprimentos, uma tendo um fornecedor exclusivo, em outras, os fornecedores competem entre si. Estudos a respeito concluem que os fornecedores devem ser

analisados em dois segmentos: fornecedores de itens estratégicos e itens necessários, mas não estratégicos. A intenção nesses regimes é manter fornecedores competindo entre si, através de estímulo de preço, resultando no benefício da diminuição de custos, permitindo aos fornecedores a maximização de economias de escala e mantendo a competição no mercado fornecedor [4].

A importância de compras tem exigido a integração da área de compras com outras áreas da organização que participam da seleção e do desenvolvimento das bases de fornecimento, assim equipes de compras, engenharia de produto, qualidade, logística e finanças decidem quem irá fornecer, sendo considerados os seguintes fatores qualificadores de fornecimento:

- certificação de qualidade (geralmente ISO);
- capacidade financeira para garantir a continuidade do fornecimento e atualização tecnológica;
- qualificação no processo produtivo;
- capacidade de engenharia (desenvolvimentos de produtos);
- histórico de fornecimento;
- menor preço;
- localização física do fornecedor (cabe observar que em algumas organizações encontra-se o termo *global sourcing*, ou seja fornecedores localizados até mesmo em outros países).

Baseado nesse referencial teórico, é apresentado a seguir um exemplo de aplicação da TDA no processo de decisão para a homologação de fornecedores de suprimentos.

Na aplicação "Homologação de Fornecedores" são considerados os seguintes critérios:

- Qualidade do produto a ser fornecido;
- Flexibilidade de alteração na quantidade pedida;
- Distância do fornecedor.

As alternativas consideradas para a homologação dos fornecedores são:

- Fornecedor Aprovado;
- Fornecedor Reprovado.

A tabela de decisão criada para o problema é mostrada na figura 6, onde são estabelecidas algumas regras e condições.

		regra 1	regra 2	regra 3	regra 4	regra 5
C1	Qualidade	s	s	n	n	s
C2	Flexibilidade na quantidade	s	s	s	s	n
C3	Distância próxima	s	n	s	n	s
A1	Fornecedor Aprovado	x	x	x		
A2	Fornecedor Reprovado				x	x

Figura 6 – Tabela de decisão do problema

A matriz de comparação entre os critérios e seus valores é apresentada na figura 7, cujos valores são obtidos de acordo com o método AHP.

	C1	**C2**	**C3**
C1	1,00	4,00	3,00
C2	0,25	1,00	2,00
C3	0,33	0,50	1,00

Figura 7 – Matriz de comparação entre os critérios

Após a obtenção dos pesos w_i de cada critério, as pontuações a_{ij} relativos ao desempenho de cada alternativa A_j em relação a cada critério C_i, foram calculados os valores x_j para categorizar cada A_j. A figura 8 apresenta a matriz de decisão do problema.

			x1	**x2**
			0,868	0,132
			A1	**A2**
w1	0,630	**C1**	0,889	0,111
w2	0,218	**C2**	0,857	0,146
w3	0,151	**C3**	0,800	0,200

Figura 8 – Matriz de decisão

Os valores da matriz anterior foram utilizados para calcular os pesos Y_k de cada uma das regras da tabela, o que provocou uma reordenação das regras (figura 9).

		regra 1	regra 3	regra 2	regra 5	regra 4
C1	Qualidade	s	n	s	s	n
C2	Flexibilidade na quantidade	s	s	s	n	s
C3	Distância próxima	s	s	n	s	n
A1	Fornecedor Aprovado	x	x	x		
A2	Fornecedor Reprovado				x	x

Figura 9 – Nova configuração da tabela de decisão

Em seguida, simulou-se uma consulta com as seguintes condições, para a qual não se conhecia a alternativa a ser assinalada (figura 10):

		regra 6
C1	Qualidade	s
C2	Flexibilidade na quantidade	n
C3	Distância próxima	n

Figura 10 – Regra a ser pesquisada na tabela de decisão

A tabela é percorrida e a regra não é encontrada, logo uma ação adaptativa de inclusão de regras é executada. A ação adaptativa além de incluir a nova regra, determina qual a alternativa possível para as condições da regra, assim como o seu peso (figura 11).

		regra 1	regra 3	regra 2	regra 5	regra 6	regra 4
C1	Qualidade	s	n	s	s	s	n
C2	Flexibilidade na quantidade	s	s	s	n	n	s
C3	Distância próxima	s	s	n	s	n	n
A1	Fornecedor Aprovado	x	x	x			
A2	Fornecedor Reprovado				x	x	x

Figura 11 – Tabela de decisão após a inclusão da regra 6

Em seguida, uma ação adaptativa de remoção de regra é aplicada para verificar se existem regras cujos critérios são indiferentes. A regra 3 e a regra 2 podem fundir-se em uma única regra, gerando nova configuração da tabela de decisão (figura 12).

		regra 1	regra 3 + regra 2	regra 5	regra 6	regra 4
C1	Qualidade	s	-	s	s	n
C2	Flexibilidade na quantidade	s	s	n	n	s
C3	Distância próxima	s	-	s	n	n
A1	Fornecedor Aprovado	x	x			
A2	Fornecedor Reprovado			x	x	x

Figura 12 – Nova configuração da tabela de decisão

A partir desta configuração, a tabela de decisão pode ser utilizada novamente para um próximo ciclo decisório.

Os resultados do estudo de caso mostram que as técnicas adaptativas, aliadas aos procedimentos dos métodos multicritério, possibilitam o uso de Tabelas de Decisão Adaptativas aos processos de tomada de decisão, o que pode representar mais uma alternativa de sistema.

VIII. CONCLUSÃO

Neste artigo são apresentados os mecanismos de Tabela de Decisão Adaptativa e multicritérios para tomada de decisão, originados pela combinação de tabelas de decisão convencionais com ações adaptativas e procedimentos de métodos multicritério. É mostrado como esses dispositivos, pelo seu caráter genérico e com aplicabilidade ampla, podem apoiar na tomada de decisão, diante de critérios estabelecidos a cada ciclo do processo decisório.

Para efeito de aplicação da tomada de decisão considerando métodos adaptativos, foi apresentado um estudo de caso na homologação de fornecedores, cuja aprovação depende de alguns critérios de apoio à ação de executivos da área de suprimentos.

A partir da identificação do objetivo "Homologação de Fornecedores", foi criada uma Tabela de Decisão Adaptativa, cuja operação é baseada nos métodos multicritério, tais como AHP e de ordenação, o que permitiu a análise da importância dos critérios de homologação para as alternativas de aprovação ou reprovação do fornecedor.

Naturalmente, embora o modelo da TDA proposto seja uma ferramenta para a tomada de decisões na homologação de fornecedores, as decisões ainda devem ser tomadas pelos profissionais responsáveis pela análise e são, portanto, sujeitas aos julgamentos dos decisores humanos.

O tomador de decisão, sendo o maior conhecedor de seu negócio, possuidor de uma sensibilidade intuitiva decorrente de sua experiência, pode desenvolver uma ação mais direcionada ao seu objetivo no atendimento das necessidades de suprimento de forma eficiente e eficaz, visando amenizar o impacto das incertezas, através da utilização de processos sistemáticos de tomada de decisão.

No entanto, deve-se observar que esse projeto ainda está em fase de experimentação, e a confiabilidade do modelo adaptativo deverá ser alcançada pelo formalismo e rigor matemático proposto. Desta maneira, acredita-se que o modelo da Tabela de Decisão Adaptativa e multicritérios para tomada de decisão trará contribuições, tanto nos aspectos práticos de aplicação da Tecnologia Adaptativa quanto nos processos de tomada de decisão.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] PIDD, M., *Modelagem empresarial: ferramentas para tomada de decisão*. Trad. Gustavo Severo de Borba et al. Porto Alegre: Artes Médicas, 1998.

[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] LIMA, J. C.S. *Um Estudo Sobre a Reconfiguração da Função de Compras em Empresas do Setor Automotivo*. Tese de Doutorado - Escola Politécnica da USP, 2004.

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

[6] HUGHES, M. L., SHANK, R. M., STEIN, E. S. *Decision Tables*. Midi Publications, Management Development Institute, Divisions of Information, Industries, Inc., Wayne, Pennsylvania, 1968.

[7] GILDERSLEEVE, T. R. *Decision Tables and Their Practical Application in Data Processing*. Englewood Cliffs, N. J., Prentice Hall, 1970.

[8] 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, p. 45-57, Nov 2003.

[9] TCHEMRA, A. H. *Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão*. I WTA – Workshop sobre Tecnologia Adaptativa, Jan 2007 - www.pcs.usp.br/~lta.

[10] CERTO, S. C., *Modern management: diversity, quality, ethics and the global environment*. New Jersey: Prentice Hall, 2000.

[11] THOMAZ, J. P. *Concepção de um Modelo Multicritério de Apoio à Decisão*. Dissertação de Mestrado, Universidade Lusíada. Lisboa, Portugal, 2000.

[12] STAIR, R. M. e REYNOLDS, G. W. *Princípios de Sistemas de Informação – Uma abordagem gerencial*. Trad. Alexandre M. Oliveira. 4ª ed. LTC, Rio de Janeiro, 2002.

[13] FÜLÖP, J. *Introduction to Decision Making Methods*. Laboratory of Operations Research and Decision Systems, Computer and Automation Institute, Hungarian Academy of Sciences, 2005.

[14] ROY, B. *Multicriteria methodology goes decision aiding*. Kluwer Academic Publishers, 1996.

[15] SAATY, T. L. *Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process*, vol VI, Pittsburgh: RWS Publications, 1994.

[16] SLACK, N.; CHAMBERS, S.; JOHNSTON, R. *Administração da Produção*. Rio de Janeiro: Atlas 2007.

Angela Hum Tchemra é bacharel e licenciada em Matemática pela Universidade Presbiteriana Mackenzie e mestre em Administração de Empresas pela mesma universidade. Atualmente é professora de cursos de graduação na Universidade Presbiteriana Mackenzie, Faculdade de Engenharia São Paulo e Fundação Armando Álvares Penteado. É Doutoranda do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

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 Universidade Presbiteriana Mackenzie. É Doutorando do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

COMUNICAÇÕES

Actividades conjuntas entre el Laboratorio de Lenguajes y Tecnologías Adaptativas (LTA) del Departamento de Ingeniería de Computación y Sistemas Digitales de la Escuela Politécnica de la Universidad de São Paulo (USP) y el área de Lenguajes de Programación de la Sección Ingeniería Informática (SII) de la Pontificia Universidad Católica del Perú (PUCP).

***Resumen**: Profesores y estudiantes de pre - grado de Ingeniería Informática de la PUCP han desarrollado, con el apoyo del profesor Dr. João Jose Neto y del LTA, diversas actividades en las que se cuentan: un curso, proyectos conjuntos, una pasantía, asesorías académicas, proyectos de fin de carrera (tesis de pre - grado) y un artículo conjunto en un evento internacional. Las relaciones institucionales se han fortalecido como resultado de estas actividades y se espera que así continúe.*

El establecimiento de la relación institucional entre el LTA y la SII, se remonta a Octubre del 2004, cuando el prof. Neto, en su paso por LIMA de retorno del CLEI, sostiene una reunión con algunos profesores con quienes intercambian ideas para poder hacer trabajos conjuntos. Algunas de las ideas se concretaron y fueron desarrolladas en estrecha coordinación entre ambas instituciones.

Las actividades se inician en el 2005 cuando se organiza el curso "Introducción a los Compiladores Adaptativos" en donde se introdujo a los participantes en los temas de autómatas de pila estructurados. En dicha oportunidad se establecieron dos actividades. La primera es la traducción del portugués al español del libro "Introducción a los Compiladores" de autoría del Prof. Neto y que fuera completado por profesores del SII; y la segunda es el desarrollo del proyecto de construcción de un meta-compilador para la notación Wirth utilizando la técnica de autómatas de pila estructurados: WirthCC: Un Metacompilador para Lenguajes Libres del Contexto, desarrollado con estudiantes de pre-grado.

Durante el 2006, durante una visita del Prof. Neto, se iniciaron dos proyectos: (i) el primero denominado "Formalización Completa de un Lenguaje y Automatización de su Implementación" y el otro "Implementación de un Sistema de Control Basado en Tecnología Adaptativa para un Robot Móvil que Genera Representaciones Abstractas de un Ambiente Desconocido"; ambos proyectos trabajados entre profesores y estudiantes de la PUCP.

En el 2007, el Prof. Otto Pflücker realizó una estancia breve en el LTA y como resultado de ello se preparó el artículo conjunto "GAMA-NW: Un enfoque reutilizable para un metalenguaje y un metacompilador", el mismo que fue presentado en la XXXIII Conferencia Latinoamericana de Informática. Se tiene previsto la estancia de otros profesores para fortalecer las relaciones y coordinar diversos proyectos.

Durante estos años, ambas instituciones han colaborado para poder impulsar el tema de la tecnología adaptativa en el Perú y se ha contado con el respaldo de las autoridades e instancias de investigación de ambas Universidades.

Lima, Enero de 2008

Prof. Abraham Davila
Pontificia Universidad Católica del Perú

Actividades conjuntas entre el Laboratorio de Lenguajes y Tecnologías Adaptativas (LTA) del Departamento de Ingeniería de Computación y Sistemas Digitales de la Escuela Politécnica de la Universidad de São Paulo (USP), la Sociedad Peruana de Computación (SPC) y la Pontificia Universidad Católica San Pablo, en Arequipa (UCSP).

***Resumen**: Profesores y estudiantes de pre - grado de Ingeniería Informática de la UCSP han desarrollado, con el apoyo del profesor Dr. João Jose Neto y del LTA, diversas actividades en las que se cuentan: cursos y una pasantía. Las relaciones institucionales se han fortalecido como resultado de estas actividades y se espera que así continúe.*

La primera vez que prof. Joao José Neto estuvo en Perú fue en Enero del 2003, en el Congreso Internacional de la Sociedad Peruana de Computación, presentando sus investigaciones en Tecnología Adaptativa, y participando en un panel sobre la orientación de la carrera de Ciencias de la Computación en Perú.

De ahí nació una conexión con el Perú que lo ha llevado ya varias veces por aquí, a participar en varios eventos en Perú y dictar cursos en la Pontificia Universidad Católica del Perú y en la Universidad Católica San Pablo.

En la Pontificia Universidad Católica del Perú él ha dictado en Febrero de 2005 un curso de Compiladores, con énfasis en una técnica basada en Autómatas de Pila Estructurados.

Con la universidad Católica San Pablo hay una amistad que nos ha llevado a realizar otro curso allí, en Octubre de 2006, que envolvió algunos tópicos de la Tecnología Adaptativa,

Como una importante consecuencia de la realización de esos cursos en Perú, los Autómatas de Pila Estructurados han sido incorporados en el sílabo del curso de Teoría de la Computación de la Universidad Católica San Pablo.

Además, estamos uniendo esfuerzos en la realización de proyectos conjuntos, principalmente en la iniciación de una investigación para la aplicación de tecnologías adaptativas en el campo de métodos formales en la ingeniería de software.

Enero de 2008

Profa. Abigail Parisaca Vargas
Universidad Católica San Pablo
Arequipa, Perú

Mesa Redonda

Resumo — **Esta Mesa Redonda, que encerrou os trabalhos do WTA 2008, foi presidida pelo Prof. Dr. André Riyuiti Hirakawa e contou com a participação de quatro convidados: Profa. Cristina Tomie Motohashi Matsusaki (FSA – Fundação Santo André), Prof. Aparecido Valdemir de Freitas (IMES – Universidade Municipal de São Caetano do Sul), Prof. Amaury Antônio de Castro Júnior (UFMS – Universidade Federal de Mato Grosso do Sul, campus Coxim), e Prof. Luciano Gonda (UCDB – Universidade Católica Dom Bosco, Campo Grande – MS). Numa primeira etapa, os membros da Mesa Redonda expuseram suas visões e experiências, e na segunda, apresentaram sugestões, para maior aproximação entre os interessados na área e para a motivação e viabilização de iniciativas em Tecnologia Adaptativa.**

VIII. INSTALAÇÃO DA MESA REDONDA

(Prof. André)

Inicialmente o presidente desta Mesa Redonda, o Prof. Dr. André Riyuiti Hirakawa, da Escola Politécnica da USP, convidou os demais componentes a comporem a mesa:

- Professora Cristina Toshie Motohashi Matsusaki da FSA, Faculdade de Engenharia do Centro Universitário Fundação Santo André, SP.
- Professor Aparecido Valdemir de Freitas, do IMES, Universidade Municipal de São Caetano do Sul, SP.
- Professor Amaury Antônio de Castro Junior, da UFMS, Universidade Federal de Mato Grosso do Sul, campus de Coxim, MS.
- Professor Luciano Gonda, da UCDB, Universidade Católica Dom Bosco, campus de Campo Grande, MS.

Deu então início à sessão, apresentando o formato dos trabalhos: a Mesa Redonda seria realizada em duas partes:

- Na primeira parte, os membros da Mesa fariam um relato das atividades relacionadas com Tecnologia Adaptativa, conduzidas em suas instituições.
- Na segunda, apresentariam propostas de idéias e de sugestões para fortalecer os grupos e estimular trabalhos colaborativos em Tecnologia Adaptativa.

O Prof. Hirakawa ressaltou o fato de, na presente edição, o WTA ter demonstrado o prestígio que este evento vem adquirindo, aqui materializado pela participação de entidades consagradas na área, tais como o IEEE, a SBC, a SPC, etc.

A seguir, apresenta-se uma transcrição livre do pronunciamento dos membros da Mesa Redonda, bem como dos questionamentos apresentados pela platéia e das respostas dadas pelos membros da Mesa.

IX. PRIMEIRA PARTE

A. *(Profa. Cristina)*

Na nossa instituição, a Fundação Santo André, a Faculdade de Engenharia, onde trabalho, é uma faculdade nova - nós estamos com a primeira turma de formandos -, e em termos de pesquisa, ela é ainda emergente.

O que temos lá, em termos produção acadêmica e de linhas de pesquisa, são os trabalhos individuais dos docentes, e também os trabalhos de iniciação científica e trabalhos de formatura, desenvolvidos pelos alunos.

A minha área de interesse é a Automação, os Sistemas de Automação, por exemplo, de Manufatura, os Sistemas de Comunicação, Robótica, etc. (ou seja, a Mecatrônica).

Sendo sistemas a eventos discretos, nota-se que diversas ferramentas da matemática discreta, tais como autômatos, redes de Petri, máquinas de estados etc., têm sido utilizadas há décadas na modelagem do comportamento e na especificação do comportamento desses sistemas.

Nesse sentido, a Tecnologia Adaptativa se mostra muito promissora, porque ela permite representar de forma concisa e "enxuta" as inúmeras alternativas de comportamento que podem ser exibidos por tais sistemas atuais.

Por exemplo, em um sistema de manufatura flexível tem-se conjuntos de máquinas que podem processar diferentes tipos de produtos, gerando inúmeras possibilidades de combinação no momento de programar a produção: a peça que vai para cada máquina, a seqüência que se deseja executar, etc.

O emprego da Tecnologia Adaptativa tem se mostrado, nesse sentido, bastante promissora, por exemplo através da representação desses fenômenos por meio de autômatos adaptativos, que conseguem representar confortavelmente a abstração desses comportamentos.

Quando se fala da realização de um processo do controle, com um software sendo executado em um processador, que por sua vez está ligado a sensores e atuadores, enfim ao meio físico que são os sistemas de manufatura, os robôs etc, surgem inúmeros problemas, desde comportamentos imprevisíveis, que não são desejados, até os problemas de ocupação de memória, que constituem os maiores desafios.

Então, lá na Fundação Santo André, estamos trabalhando com um grupo de alunos que começaram a verificar quais poderão ser as aplicações de Tecnologia Adaptativa na Automação.

A primeira investida que fizemos foi a de tentar imitar um CLP (controlador lógico programável), o qual pudesse ler um Autômato Adaptativo em vez das linguagens padrão usualmente empregadas para os CLPs, e rodar esse Autômato Adaptativo, e com isso controlar um sistema.

Esta é, em resumo, a atividade que estamos desenvolvendo.

B. *(Prof. Aparecido)*

Vou fazer um breve resumo da área de pesquisa em que estou trabalhando, que envolve a programação adaptativa, e o grande objetivo do meu trabalho tem sido conceituar linguagens de programação adaptativas e colocar alguns pontos desejáveis essenciais, que caracterizam uma linguagem adaptativa.

Como foi apresentado ontem pelo Éder, código adaptativo tem um estreito relacionamento com código automodificável: em essência, um código adaptativo é capaz de se alterar durante sua execução, e portanto apresenta o comportamento de um código automodificável.

Alguém pode objetar que o conceito de código automodificável já existe e tem sido usado há muito tempo pelos programadores.

De fato, a automodificação, embora uma técnica de bastante antiga, tem uso restrito devido às dificuldades de manutenção, depuração e legibilidade que acarreta.

No nosso trabalho de projeto de linguagens de programação adaptativas, estamos canalizando esforços para que a adaptatividade se incorpore ao código automodificável.

Mais recentemente, pode-se perceber, na literatura, um reaquecimento do uso de códigos automodificáveis. Existem muitas aplicações, por exemplo, nas áreas de segurança, ofuscação de código, otimização de código, etc.

Então, por exemplo, como foi visto nas apresentações, na linha de atuação da UFSCar há trabalhos interessantes ligando segurança com automodificação de código.

Então, com o nosso projeto, imaginamos que a programação adaptativa deva contribuir para que tais aplicações possam ser desenvolvidas e concretizadas de maneira mais consistente.

Em nosso projeto nós optamos por usar, como linguagem subjacente, uma linguagem funcional, e aproveitar, em nosso projeto, o recurso de extensão já oferecido por essa linguagem.

O mecanismo de extensão nativo do Lisp foi usado como ferramenta para construir alguns operadores novos, a partir dos quais foi formado o núcleo da camada adaptativa, o qual foi, portanto, desenvolvido todo ele na linguagem Lisp subjacente.

Esta camada adaptativa tem, assim, a responsabilidade de efetuar todo o tratamento adaptativo em tempo de execução.

Estamos trabalhando com a linguagem Lisp, finalizando a camada adaptativa, e também, paralelamente, trabalhando em uma aplicação, tipicamente adaptativa, que incorpora um conjunto de regras que se modifica em tempo de execução, e no projeto nós estamos mapeando esse conjunto dinâmico de regras em um código, que se automodifica.

Para materializar essa adaptatividade, a linguagem de programação adaptativa deve ser capaz de executar as operações de automodificação, alterando esse conjunto de regras, através da aplicação dessas funções implementadas na camada adaptativa anteriormente construída.

O assunto é complexo, mas gratificante, pelos seus potenciais desdobramentos, e o projeto dá margem a um grande número de outras investigações interessantes, uma das quais relacionada com o estilo de programação.

Conversando anteriormente com o Amaury sobre paradigmas, fizemos alguns paralelos. Lembramos, por exemplo, que a programação orientada a objetos exige que se pense de forma aderente ao paradigma orientado a objetos; que a programação paralela exige que se pense de forma aderente ao paradigma paralelo, e assim por diante. Analogamente, para programar adequadamente em linguagens adaptativas, será necessário também pensar de forma aderente ao estilo adaptativo de programação.

Será importantíssimo que seja elaborado algum método ou técnica que auxilie o programador a pensar e a escrever de forma adaptativa. No nosso trabalho tenho procurado contribuir para um progresso nessa direção. No site do LTA estão disponíveis algumas publicações que fizemos sobre o assunto, e estamos à disposição para discuti-los com todos aqueles que se interessarem por essas idéias.

C. (Prof. Amaury)

Há na UFMS um programa de mestrado, no campus de Campo Grande. Na última avaliação da CAPES, recebemos o conceito 4. Há várias linhas de pesquisa na instituição, com diferentes focos tais como: algoritmos paralelos e distribuídos, teoria da computação, bio-informática, computação gráfica, geometria computacional, entre outros, e hoje há também diversos projetos em desenvolvimento na UCDB.

Fiz meu mestrado na Universidade Federal, em Campo Grande, trabalhando com algoritmos paralelos, e na ocasião, enquanto trabalhava na UCDB, conheci meu colega de trabalho e também meu amigo pessoal, Hemerson Pistori, que infelizmente não pôde comparecer a essa segunda edição do WTA. Foi ele quem me apresentou o Prof. João, com quem na ocasião tive uma conversa, e eu então comecei a conhecer a área de Tecnologia Adaptativa. Ele me falou de autômatos automodificáveis, e eu comecei a me interessar muito pelo assunto. Aí começamos a desenvolver uma série de estudos, e me envolvi, juntamente com o Hemerson, em alguns projetos, que hoje estão sendo executados na UCDB, como o Luciano depois irá detalhar. Atualmente eu, como o Aparecido e o Éder, estou trabalhando na parte de linguagens adaptativas.

Em particular, estou pesquisando a adaptatividade no paradigma imperativo, com programação estruturada, e com os conceitos de extensibilidade e de linguagem adaptativa estruturada, e meu objetivo é o de implementar todo o mecanismo de extensão modelado com o autômato adaptativo, e com ele desenvolver, para uma linguagem adaptativa estruturada, comandos que permitam exprimir a alteração de código, em complemento, assim às pesquisas do Éder, que desenvolve ambientes de execução adaptativos de baixo nível, e do Aparecido, que trabalha com a implementação da adaptatividade na linguagem funcional Lisp.

Tentei relacionar em uma lista os assuntos que mais me chamam a atenção na Tecnologia Adaptativa. Pretendo, uma vez terminando o meu doutorado, iniciar na UFMS uma linha de pesquisa na área de Tecnologia Adaptativa, a partir da qual se possam formar mais pessoas com condições de desenvolver mais trabalhos nesta área de interesse. Nessa lista, os principais temas são os seguintes:

- A revista do IEEE contém todos os artigos apresentados na primeira edição do WTA. Há aí um tutorial muito interessante do Prof. João.
- Apresentei nessa edição do WTA um trabalho sobre a questão dos não-determinismos em Autômatos Adaptativos, seu tratamento.

- Há nessa publicação também um estudo inicial sobre as características de Autômatos Adaptativos de nível k, os quais têm um poder de expressão superior e permitem representar diversos fenômenos complexos com maior facilidade que com os Autômatos Adaptativos convencionais.
- Outro assunto interessante é a relação entre a Estatística e a Tecnologia Adaptativa. Por exemplo, as Cadeias de Markov e os Autômatos Probabilísticos podem ser estudados pela ótica dos aspectos teóricos da Tecnologia Adaptativa, e esse estudo pode render trabalhos, ferramentas e aplicações muito interessantes.
- Outro tema, anteriormente apresentado por Rodrigues e Rodrigues no presente evento, refere-se ao estudo de aplicações da Tecnologia Adaptativa em biologia computacional, bio-informática e processamento de cadeias, e forma uma das áreas de interesse que já temos no programa de mestrado em Campo Grande, e que muito me chamou a atenção durante o meu período de créditos, quando comecei a ler a respeito e coletar material para o desenvolvimento da minha tese. Em particular, houve uma interessante dissertação de mestrado, que foi apresentada no IME-USP, sobre o uso de autômatos nessa área.
- Um assunto teórico muito atraente refere-se à criação e ao estudo de um modelo de computação adaptativo, baseado totalmente em autômatos adaptativos. A exemplo do modelo RAM de Von Neumann, para computação seqüencial, e do modelo PRAM, para computação paralela, etc., esse seria um modelo de computação mais apropriado para a representação de fenômenos relacionados com a adaptatividade.
- A extensibilidade de linguagens de programação é outro assunto a ser explorado sob a ótica da adaptatividade.
- Constitui também um tema de grande interesse o estudo das linguagens adaptativas que prometem surgir, e seu uso para a descrição de aplicações que usem código automodificável.
- Daí surgiu a idéia da arquitetura de um hardware aderente ao modelo de computação adaptativa, e portanto apropriado para dar suporte à adaptatividade e à execução de aplicações adaptativas e a aplicações desenvolvidas com o emprego das emergentes linguagens de programação adaptativas.
- Então, de posse de alguma linguagem adaptativa, uma possibilidade seria desenvolver uma linguagem de descrição de hardware, como essas utilizadas para a programação de CLPs e de outros hardwares, mas voltada às necessidades específicas da adaptatatividade, e que favorecesse a implementação eficiente de aplicações adaptativas.
- Surge então a idéia de um paradigma adaptativo, de uma forma de pensar na programação adaptativa. Há sempre uma certa resistência e dificuldade de se mudar a forma de pensar, que exige uma adaptação do programador às novas maneiras de estudar o problema e de chegar a uma solução. Isso é muito comum, e tem ocorrido em diversas situações, como aconteceu, por exemplo com os programadores em relação às linguagens de programação. Tanto na época do aparecimento da programação estruturada como depois, ao surgirem as linguagens orientadas a objetos, foi exigida uma radical mudança no modo de pensar dos desenvolvedores, tanto na fase do projeto como na da codificação dos programas.
- Outro assunto, que está sendo bastante estudado em visão computacional, é o reconhecimento e processamento de padrões, e a mudança de padrões, com aplicações muito interessantes nos aspectos de segurança usando padrões adaptativos.
- Na tese do Prof. Hemerson Pistori, citam-se alguns problemas específicos, alguns muito complexos e de grande interesse, ainda não resolvidos, como é o caso do tratamento da semântica das ações adaptativas elementares de consulta, o problema da satisfação seqüencial de restrições, o tratamento de geradores e variáveis, e com base nisso ele desenvolveu o software AdapTools, ferramenta que tem sido muito usada pelos interessados em Tecnologia Adaptativa.
- Mais uma aplicação de grande importância para a Tecnologia Adaptativa é o Processamento de Linguagem Natural, que tem sido assunto de alguns temas tratados no LTA, e que constituem uma área de aplicação muito adequada ao uso da adaptatividade.
- Uma última aplicação muito interessante, e muito própria para se aplicar a Tecnologia Adaptativa, é a do uso da adaptatividade em redes de sensores sem fio, que será comentada a seguir pelo Prof. Luciano Gonda.

D. (Prof. Luciano Gonda)

A minha experiência com Tecnologia Adaptativa está apenas começando, mas eu vou procurar falar um pouco dos projetos em Tecnologia Adaptativa desenvolvidos na UCDB.

Existe ali instalado um grupo de pesquisa, o GPEC – Grupo de Pesquisa em Engenharia de Computação, e nós temos aí desenvolvido diversas aplicações de Tecnologia Adaptativa, voltadas para a área de visão computacional.

Infelizmente, hoje não está aqui o Prof. Hemerson Pistori, que é o coordenador desses projetos, e que poderia fornecer mais detalhes sobre os nossos desenvolvimentos em Tecnologia Adaptativa, mas em sua ausência procurarei dar uma visão geral do que ocorre nesses projetos.

Em um primeiro projeto, o tema é o reconhecimento e classificação de peças de couro. Hoje o Brasil é exportador de couro cru, e o GPEC tem um projeto cujo objetivo é a detecção de problemas e defeitos no couro. O projeto tem parceria com a EMBRAPA, do estado de Mato Grosso do Sul.

Atualmente, a classificação do couro é feita subjetivamente, por uma pessoa, através de inspeção visual. O que se está desenvolvendo é um software, na área de reconhecimento de padrões, que captura uma imagem do couro, inspeciona-a automaticamente, classificando o couro em três classes, como resultado de um reconhecimento de padrões, efetuado com o auxílio de uma árvore de decisões adaptativa.

Nesse projeto mesclam-se os métodos tradicionais de reconhecimento de padrões por visão computacional, com as tomadas de decisão através das árvores de decisão adaptativas.

Os algoritmos adaptativos empregados na classificação das peças de couro derivam do software AdapTree, desenvolvido pelo Prof. Hemerson Pistori em sua tese de doutorado.

Um segundo projeto desenvolvido pelo GPEC na UCDB é voltado a pessoas com necessidades especiais. É o projeto SIGUS, que também emprega o AdapTree para o reconhecimento de padrões. Trata-se de uma plataforma com duas linhas: uma delas refere-se ao reconhecimento de comunicações feitas na Linguagem Brasileira de Sinais (LIBRAS) e a outra, ao auxílio a tetraplégicos mediante a detecção e execução de comandos, fornecidos pelo tetraplégico, através da movimentação dos seus olhos. Hoje já existe um software de demonstração, que foi desenvolvido com patrocínio da FINEP, e que pode ser treinado, permitindo ao usuário tocar bateria somente com os comandos (definidos em uma fase de treinamento), capturados a partir de uma imagem da cabeça do interessado.

Um terceiro projeto foi denominado Topolino, e refere-se a um software destinado ao estudo do comportamento de animais em laboratório. Há neste projeto três pesquisas, todas elas usando AdapTree na tomada de decisões.

A primeira é um software de auxílio ao estudo do comportamento de camundongos sob o efeito de medicamentos. Administra-se o medicamento ao animal e monitoram-se seus movimentos, determinando a partir dos sinais captados a correlação existente entre o medicamento administrado e o comportamento decorrente.

A segunda refere-se à análise do comportamento observado de serpentes, e a terceira, o mais novo projeto dentro dessa pesquisa, refere-se ao monitoramento de larvas do mosquito da dengue, de cujo comportamento se pretende determinar a fase evolutiva em que a larva é mais frágil, e que seria a mais indicada para a aplicação de inseticidas com a intenção de controlar os focos da doença. Este projeto é feito em conjunto com grupos de biotecnologia, e também tem financiamento externo.

São basicamente esses os projetos que temos lá. Só para comentar, repito algo que é sempre dito pelo Amaury: em quase todas as aplicações que podemos imaginar, há espaço para a aplicação da Tecnologia Adaptativa.

Na segunda rodada desta Mesa Redonda, deverei falar um pouco mais sobre outras das muitas aplicações da Tecnologia Adaptativa, uma das quais, a do uso da adaptatividade em redes de sensores sem fio.

X. CONTRIBUIÇÕES DO PLENÁRIO

A. (Prof. Jaime Sichman)

Gostaria de ter uma idéia do que seriam esses operadores adaptativos em linguagens automodificáveis ou adaptativas. Seria remover linhas, trocar comandos, ou algo assim?

B. (Prof. Aparecido)

No projeto em que estou trabalhando, estou utilizando como linguagem subjacente uma linguagem de programação funcional. Criamos para ela uma camada adaptativa, na qual, com a ajuda do mecanismo de extensão, estamos, usando a própria linguagem, criando alguns construtos novos, os quais permitem, em tempo de execução, imprimir à linguagem as características desejadas de adaptatividade.

Em Lisp, os programas podem ser representados por uma árvore equivalente, e os construtos adaptativos assim introduzidos permitem efetuar edições nessa árvore, que permite representar, de outra maneira, o próprio programa Lisp.

Os operadores adaptativos assim disponíveis permitem então fazer, essencialmente, a edição dessa árvore através de inserção e remoção de subárvores, e o programa, em tempo de execução, vai assumindo sucessivas instâncias diferentes de código, instâncias essas refletidas nas diversas formas sucessivamente assumidas, a partir de uma forma inicial, pela árvore que o representa.

C. (Prof. Jaime Sichman)

Uma vez que, normalmente, o Lisp já permite a modificação das listas durante a execução, então qual é a diferença?

D. (Prof. Aparecido)

Na concepção do projeto, as atividades adaptativas são implementadas com o auxílio da função eval (evaluate) do Lisp. Como, no Lisp original, essa utilização, pelo programador, nem sempre é feita de forma obrigatoriamente disciplinada, optou-se por forçar que as operações adaptativas fossem efetuadas de uma forma mais organizada, tornando compulsório o uso da camada adaptativa como intermediária em todas as operações de automodificação, garantindo-se assim que essas operações sejam efetuadas de maneira segura e disciplinada.

E. (Prof. Amaury)

Complementando, talvez nas linguagens imperativas em que não disponha do recurso da extensibilidade, fica mais evidente a necessidade desse recurso para a implementação da automodificação. Nessas linguagens, tal capacidade de extensão deve ser, no entanto, implementada externamente, e acrescentada à linguagem antes que possa ser utilizada.

O mesmo ocorre em relação às instâncias que o código assume, em tempo de execução, as quais também decorrem da ação das instruções adaptativas, que são novas instruções, adicionadas por extensão à linguagem original, que se incumbem finalmente de promover a automodificação do código, em tempo de execução.

F. (Prof. Zorzo)

Fico satisfeito de ver a mesa contar tantos trabalhos relacionados com a Tecnologia Adaptativa, e feliz de ver os frutos desse projeto, mas eu gostaria de saber quantas pessoas trabalham, em Campo Grande, nesse grande número de projetos?

G. (Prof. Luciano)

São quatro professores e até o fim do ano passado tínhamos cerca de dez alunos de Iniciação Científica trabalhando nesses projetos. Seis deles se formaram e foram aceitos em todos os programas de mestrado em que se inscreveram. Todos os projetos são desenvolvidos com

financiamento externo, com bolsas de Iniciação Científica, o que faz com que haja uma grande produção.

H. (Bal. Denys Gonçalves dos Santos)

Gostaria de saber mais um pouco sobre esses roteadores para a rede dos sensores sem fio. Onde se usa a Tecnologia Adaptativa neles? Seria na parte de controle, em cima de um roteador e este roteador faria todo o controle, ou então se faria uma rede overlay desses sensores e a partir disso se faria o controle?

I. (Prof. Luciano)

Esta pesquisa está sendo desenvolvida no LAA – Laboratório de Automação Agrícola da EPUSP. Nessa rede de sensores sem fio não existe a figura do roteador, uma vez que todos os nós têm capacidade de fazer roteamento. Então a pesquisa, que está numa fase bem inicial, procura desenvolver um algoritmo para construção de topologias para a melhoria do desempenho da rede.

Há, no caso, limitações no uso das baterias, e a vida útil da rede depende do tempo de vida da bateria que alimenta o nó. Cada nó é pequeno, e alimentado por uma bateria AA, que dura em média oito meses, em operação de monitoramento.

Não há roteamento permanente, mas existem as rotas criadas, e sempre que um nó opera, num dado momento, como roteador, como a comunicação é a atividade que mais gasta energia, então esse nó, naquele momento, vai apresentar um maior consumo de energia, e sua vida útil tenderá a se reduzir por isso.

Deseja-se propor um mecanismo adaptativo que não só preveja esse tipo de problema, ou que execute a operação de balanceamento de carga, mas também preveja outros tipos de situações, como, por exemplo, aquelas em que um nó é destruído como resultado de algum acidente, prejudicando assim a rota a ele associada e o caminho lógico ao qual pertence o tal sensor, podendo-se em resposta, nesse caso, identificar a avaria e, de forma adaptativa, re-arranjar a topologia da rede para compensar os danos constatados.

Mas isso, por enquanto, está no início. Há muito a fazer ainda, e estamos estudando a maneira como se poderia modelar a rede de sensores sem fio para usar adequadamente os formalismos adaptativos nos algoritmos de roteamento para as redes de sensores sem fio.

Há uma diferença entre uma rede tradicional, em que a preocupação maior é com a transmissão de dados, e as redes de sensores sem fio, nas quais é fundamental uma preocupação adicional com a transmissão, por causa do conseqüente consumo adicional de energia que acarreta.

J. (Prof. João)

Gostaria de lembrar que este é um problema muito similar a uma série de outros, que surgem nas mais variadas áreas. Por exemplo, o problema de *scheduling* em sistemas operacionais, o problema da rota mais curta e o problema do caixeiro viajante, todos esses são problemas de complexidade matemática elevada, portanto problemas considerados matematicamente difíceis.

O papel da Tecnologia Adaptativa, quando empregada na resolução desse tipo de problemas consiste basicamente em não buscar obrigatoriamente um roteamento ótimo, que acarretaria um custo elevado, mas uma rota subótima, que possa ser dinamicamente alterada em função da variação de parâmetros ou do contexto em que os fatos ocorrem.

Essa é, como se pode notar, mais uma manifestação evidente das dependências de contexto nas aplicações da adaptatividade na Engenharia de Computação.

Este e muitos outros problemas similares têm todos a mesma raiz. A apresentação de uma resolução boa para essa classe de problemas (não precisa ser ótima, pois seria exageradamente complexa do ponto de vista matemático) representará uma contribuição valiosa para todas essas áreas.

Eu recomendaria que esses outros problemas fossem estudados, em busca de alguma eventual solução tradicional que possa ser aproveitada, ao menos parcialmente, em conjunto com a Tecnologia Adaptativa, suscitando soluções alternativas, híbridas, de boa qualidade.

K. (Prof. Almir)

Gostaria de perguntar à Profa. Cristina sobre a utilização das redes de Petri. Eu apresentei um trabalho sobre um visualizador gráfico para redes de Petri, que pode ser utilizado para a apresentação de outros dispositivos também.

Quando eu usei redes de Petri, eu as empreguei como um estudo de caso de dispositivos adaptativos.

Queria saber se o emprego de redes de Petri adaptativas trariam algum benefício para a área de automação, e se já foi feito algum estudo sobre esse assunto.

L. (Profa. Cristina)

Uma das aplicações das redes de Petri na automação é na representação e modelagem do comportamento do sistema-alvo a ser controlado.

Considerando as fases do projeto de um sistema de controle: concepção, especificação das leis de controle, design, no projeto e na codificação, as redes de Petri mostram-se úteis principalmente na fase inicial de especificação funcional.

Logicamente, as redes de Petri deverão ser codificadas para poderem ser executadas no sistema computacional. O grande desafio que se coloca é determinar o melhor método para gerar uma boa rede, a que melhor represente o comportamento do sistema.

O desafio mesmo refere-se às técnicas de projeto: como converter a rede abstrata em um código executável equivalente? Como tratar os não-determinismos? Usa-se a técnica do Amaury e sevadota a versão determinística?

Outro ponto a considerar é que o código gerado, sendo adaptativo, e portanto, automodificável, é capaz de se expandir.

Em sistemas embarcados, há a necessidade de um comportamento autônomo, há a necessidade de estratégias de gerenciamento de memória, é preciso garantir um tempo de processamento que seja compatível com os tempos de risco da planta. Existem ainda vários difíceis desafios para que uma rede adaptativa possa ser usada com segurança como um sistema de controle.

Entretanto, agora restrita à primeira fase de especificação de sistemas de controle, ela é bastante promissora, porque consegue representar, de forma muito elegante e compacta, as características que os sistemas possuem, principalmente quando estes são grandes e apresentam muitas alternativas de comportamento.

A Tecnologia Adaptativa apresenta a importante característica de permitir que sejam mantidas mascaradas ou escondidas essas inúmeras possibilidades, evidenciando apenas aquela(s) que mais interessa(m), escolhendo uma delas e executando aquela que mais interesse num dado momento. É nesse sentido que as redes de Petri são bastante úteis e se aplicam na automação e controle.

XI. SEGUNDA PARTE

A. Abertura da segunda parte (Prof. André)

A Tecnologia Adaptativa hoje é internacional. O Prof. João esteve por algum tempo como professor convidado em duas universidades no Peru, e noutra oportunidade, em Portugal, onde presidiu uma sessão inteira sobre Tecnologia Adaptativa em um importante evento internacional.

Isso foi nos anos que passaram, mas hoje nós vivenciamos também uma expansão adicional bastante grande dessa tecnologia, o que é muito bom para todos.

O Prof. Amaury mencionou que o Prof. João ficou muito contente porque estava "perdendo o controle da situação" da Tecnologia Adaptativa. Então, para colaborar ainda mais com esse efeito, vamos pedir aos componentes da mesa que apresentem suas idéias sobre como fazer colaboração, idéias de pesquisa, e iniciativas adicionais, que possam auxiliar nesse sentido.

B. (Profa. Cristina)

Já é mais que evidente para todos que a Tecnologia Adaptativa tem um enorme campo de aplicação. Na área de automação ocorre o mesmo.

Em todos os níveis de automação eu observo possibilidades – e não é tão no futuro assim – de emprego da Tecnologia Adaptativa. Por exemplo, na área de planejamento e programação da produção, definição de rotas, etc., creio que a Tecnologia Adaptativa possa ser conjugada com outras técnicas para a geração de rotas ótimas e sub-ótimas, avaliação, enfim, das alternativas também.

No nível de coordenação, supervisão ou um pouco mais abaixo, a grande esperança que a Tecnologia Adaptativa propicia é a da viabilização de sistemas com comportamentos mutáveis. Imagine só, poder mudar a estratégia de atuação de um robô que é autônomo, e está a 3000m de profundidade no mar. Esta é uma possibilidade que Tecnologia Adaptativa nos dá, bastante real.

Ou, por exemplo, uma situação em que este robô submarino remoto perde parte de sua estrutura – sensores, atuadores, motores, etc – por algum acidente, mas consegue assim mesmo atingir seu objetivo graças à auto-adaptação do seu software.

Em níveis operacionais, também é muito real termos sistemas não só reativos, que alternam sua operação em reação à mudança do cenário, mas também termos sistemas mais inteligentes, que possam mudar suas próprias regras de mudança de estratégia, usando para isso a Tecnologia Adaptativa em várias camadas.

Tudo isso depende de transpormos algumas desafios, alguns já citados, principalmente pela falta de metodologia ou da técnica do conhecimento de como gerar, projetar o autômato que vai especificar o comportamento, e também transpor o desafio na parte de projeto e implementação do código, enfim de determinar as restrições que sejam suficientes para que ele possa se tornar operacional.

Acho que é isso. Bastante trabalho a fazer.

C. (Prof. Aparecido)

Em primeiro lugar, gostaria de realçar a importância deste Workshop, a oportunidade que ele proporciona de a gente manter contato com diversas instituições, com vários colegas.

Apesar de toda a dificuldade logística de se conduzir um Workshop – as atividades dos bastidores aqui não aparecem, mas quem organiza sabe – e das dificuldades operacionais, é muito importante que se mantenha regularmente este Workshop, que, sem dúvida, é uma ocasião sem igual, muito propícia para se trocar idéias.

Eu, por exemplo, tive aqui a oportunidade de conhecer e de fazer novos amigos, e de estreitar laços de amizade.

Certamente um evento como este contribui muito para que a Tecnologia Adaptativa se desenvolva, a passos cada vez mais largos.

Outra sugestão: a minha instituição foi elevada à categoria de universidade há pouco tempo, e por isso não temos linhas de pesquisa ainda, mas nós temos trabalhado no sentido de buscar parcerias empresariais. Isso também não é uma atividade fácil, porque envolve toda a burocracia das universidades, mas tivemos algumas experiências felizes, em que nós abolimos a burocracia fazendo contratos "fio de bigode", aqueles contratos amigáveis, e tivemos alguns resultados animadores.

Algumas empresas se interessam por determinados projetos – e em Tecnologia Adaptativa, já se tem uma grande quantidade de softwares para apresentar às empresas, como se pôde observar claramente na mostra de software que tivemos a oportunidade de visitar neste evento. Certamente, deve haver no mercado empresas interessadas nesses softwares.

Muitas vezes, uma parceria dessas se torna atrativa por não envolver custos – nós fizemos três ou quatro projetos lá na universidade, nos quais empresa vem à instituição, participa do desenvolvimento do projeto em um grupo envolvendo alunos da instituição.

Para a empresa isso não resulta em custos diretos, mas sua participação se resume na disponibilização de alguns profissionais da empresa, que cedem parte do seu tempo para acompanhar o projeto com os alunos.

Coloco então essa idéia como sugestão. Às vezes podemos viabilizar um projeto fugindo da burocracia através de esquemas como esse.

D. (Prof. Amaury)

Coloquei aqui um conjunto de áreas: aspectos teóricos, incluindo não-determinismo, adaptativo-adaptativo,

aplicações em bio-informática, modelo de computação, extensibilidade, paradigma, aspectos de hardware, processamento de linguagem natural, redes de sensores sem fio, etc. Muito importante iniciativas. São muitas idéias e vários temas. Há certamente outros, que vocês deverão colocar ao final dessa rodada.

Acredito que algo muito importante para o desenvolvimento da área será a melhoria da nossa comunicação. Temos a página do LTA, agora remodelada, temos uma lista de discussão, em que todos os que desejarem poderão inscrever-se de uma forma muito simples, clicando apenas em um botão, através da própria página do LTA, e a partir daí colocar as suas idéias e sugestões nessa lista de discussão.

Outra idéia seria incluir na página do LTA uma sessão: "Tome um café com o Prof. João". Esse "café" realmente abre a área de Tecnologia Adaptativa de uma forma incrível. Eu, quando venho aqui, faço sempre questão de ir tomar café com o Prof João. Sei que, em alguns momentos, eu faço aquela expressão do Prof. Ítalo: "acho que me perdi...", mas depois eu me reencontro.

Entre outras coisas, temos uma seção de eventos na página, com anúncios de congressos e outras chamadas de trabalhos, na área e em áreas correlatas.

A SBC lançou há um ou dois anos um documento com os dez grandes desafios da área de computação, da área de tecnologia, para os próximos dez anos, e existem alguns desafios nos quais os trabalhos na área de Tecnologia Adaptativa se encaixam perfeitamente.

A questão dessas novas tecnologias que estão surgindo, a questão do processo de criação de software, como citou a Cristina, a forma de pensar em um sistema, um produto da área de tecnologia, tudo isso acaba estimulando o surgimento de novas idéias.

Isso me lembra a frase do Paulo Cereda, em sua primeira apresentação neste evento: "a maturidade de uma área é comprovada a partir do momento em que ela começa a resolver problemas em outras áreas" e em Tecnologia Adaptativa realmente é isso o que está acontecendo, e é por isso que sou um dos grandes defensores dessa tecnologia. Estarei aqui no terceiro, quarto, quinto WTA, espero que não mais como aluno, mas sempre participando.

Existem outros eventos, em áreas correlatas, dos quais vale a pena participar. Um deles surgiu no Congresso da SBC de 2006. É o LACS, de Computação Autonômica, associado à IBM, que foi uma empresa que tentamos trazer para este evento. Infelizmente não deu certo, mas acredito que poderemos ter outros contatos com outras pessoas que possam trabalhar conosco nesses assuntos.

Para qualquer sugestão, crítica, ou idéia em relação a isso, visitem nossa página, inscrevam-se em nossa lista, enviem-nos suas idéias. Isso realmente é muito importante.

E. (Prof. Luciano)

Como disse, eu ainda sou novo nesta área, mas posso citar algumas aplicações que visualizo. Assim como o Prof. Aparecido, acho que este Workshop tem uma importância muito grande, e assim como o Prof Amaury, o café com o Prof. João é muito produtivo. Decidi em um café com o Prof. João aplicar Tecnologia Adaptativa em redes de sensores sem fio.

Aplicações futuras: gosto da área de redes e de aplicações de segurança. Aí me lembrei de um amigo meu, que fez doutorado na Unicamp, e cuja tese visava modelar o sistema imunológico humano para construir a partir disso um sistema de segurança em redes.

Pode um dia surgir uma doença nova que o nosso sistema não conhece ainda. Aí pensei em uma aplicação de sistema de controle de segurança baseado em Tecnologia Adaptativa.

A própria parte de redes de sensores sem fio com uma futura AdapNet, e a modelagem de processos da cadeia produtiva em piscicultura poderão também ser projetados usando Tecnologia Adaptativa.

Nosso estado é forte em piscicultura, e há outro professor nosso, o Prof. Jefferson Pistori, trabalhando neste assunto, sob a orientação do Prof. André. À medida que se vai aprofundando em Tecnologia Adaptativa, vão aparecendo novas idéias.

Completando a resposta para o Prof. Zorzo, em relação ao grupo, temos poucos professores, mas temos diversas parcerias, incluindo o LTA, algumas empresas inclusive uma de São Paulo, que é nossa parceira na parte de pesquisa. Assim, não são apenas os professores e os alunos, mas temos também as parcerias.

Gostaria de convidar a todos que quiserem conhecer a UCDB. E os que quiserem ser nossos parceiros, se desejarem trabalhar conosco em visão computacional, visitem nosso site www.gpec.ucdb.br

F. (Prof. André)

Finalizada a segunda rodada, gostaria de declarar que eu também sou fá da Tecnologia Adaptativa e aprendi a ser fã da Tecnologia Adaptativa em um dos cafés com o Prof. João, e convido a todos a fazerem parte desse café com o Prof. João.

A Tecnologia Adaptativa é muito poderosa e tem a capacidade de ser determinística, simples, auxiliar na tomada de decisão, ser flexível ou adaptativa, e devido a essas características ela substitui ou trabalha da mesma forma das outras lógicas computacionais clássicas para modelar ou obter informações sobre algum comportamento – lógica nebulosa, redes neurais, algoritmos genéticos – essas lógicas são muito boas para tentar modelar algum comportamento, mas elas às vezes apresentam dificuldades quanto aos aspectos de determinismo e de convergência. E, embora a Tecnologia Adaptativa não nasça determinística, é possível levá-la a comportar-se deterministicamente, permitindo assim que seja aplicada em diversas situações em que essas lógicas tradicionais são empregadas.

Passo agora a palavra à platéia para os seus comentários e para ver para quando será agendado o café com o Prof. João.

XII. CONTRIBUIÇÕES DO PLENÁRIO

A. (Prof. Zorzo)

Eu também sou pupilo do Prof João. Tomei muito café com ele. Isso foi após meu doutorado. Quero parabenizar. O

ano passado eu não pude estar presente, apesar de saber do workshop. Na verdade é uma iniciativa que só fortalece a área. Conheci o Prof. João quando ele começou a trabalhar com esses autômatos adaptativos, e ele já me contava todo o seu entusiasmo e motivação um bom tempo atrás.

Passados 10 anos, é uma satisfação ver o grupo deste tamanho participando de um workshop desta natureza. É uma satisfação pelo fato de gostar muito da pessoa do Prof. João ver que profissionalmente ele conseguiu crescer e fortalecer dessa maneira esta área.

É satisfação grande ver o grupo de Mato Grosso do Sul está com um grupo grande, e vem difundindo o que parte tudo desse projeto inicial do Prof. João.

B. (Prof. João)

Acho que isso não pode ter essa conotação tão pessoal. Acredito ser um instrumento voltado para que essa linha de trabalho e de interesse evolua. Tenho tentado fazer meu papel de catalisador, e por isso invisto no tal café. Essas conversas no café são momentos de descontração, e proporciona o ambiente favorável e descontraído, que nos estimula a termos idéias, e então, juntando dois cérebros, conseguem-se resultados melhores que com um só.

Isso faz as coisas acontecerem se houver dos dois lados interesse, empenho, e boa vontade – e boa vontade é o que não falta por aqui – não se consegue manter até esta hora numa sexta feira, véspera de carnaval, cerca de 40 pessoas numa sala como esta, se não houver boa vontade. Começamos o dia com perto de 100, a agora ainda está aqui presente um número expressivo de pessoas (que a sala grande dilui um pouco e não deixa perceber que há tanta gente aqui). Isso é uma coisa que me deixa realmente feliz. Vocês estão fazendo uma parte fundamental desse processo todo. Sem a participação de cada um, só o meu entusiasmo não resolveria nada disso, e eu gostaria de poder continuar a ser este catalisador.

Vocês podem contar comigo para o que for necessário, eu tenho vontade de participar de todo este processo com o maior prazer. Há a componente técnica, que é um assunto que me interessa muito, me anima e me entusiasma muito até hoje. Depois de mais de vinte anos trabalhando nisso tenho o mesmo entusiasmo que tinha no primeiro dia. Então este assunto é algo que me cativa muito. E o fato de as pessoas se animarem também com isso é que me mantém entusiasmado.

É tão difícil superar umas tantas dificuldades que se nos apresentam por aí devido à falta de terreno fértil em tantas áreas, que quando se encontra uma terra boa vale a pena investir, e eu encontro aqui essa terra, e por isso estou interessado em investir nela, continuando a promover este Workshop. Gostaria que no ano que vem haja de novo este evento, se vocês quiserem, naturalmente.

E agora, para o próximo ano, tenho uma proposta a todos: gostaria que tivéssemos, para a próxima edição deste WTA, no mínimo uns dez artigos selecionados pela revista latinoamericana do IEEE.

O número mágico dez tem uma explicação: é o número considerado adequado para que possa ser publicado novamente um número especial da revista, a exemplo daquele número que publicamos no Workshop do ano passado, resultado do trabalho realizado no ano que passou: foram publicados dez trabalhos que a revista selecionou, fizemos muitas revisões, foram muitas idas e voltas, porque inicialmente o material não foi elaborado para essa finalidade, mas acabou resultando um número muito interessante, que compilou um conjunto de artigos que realmente vale a pena.

Quisera ter repetido o feito este ano, mas infelizmente não tivemos o número suficiente de artigos submetidos à revista, para que dez fossem selecionados, mas teremos uma edição eletrônica de todos os artigos que foram apresentados aqui, que ficarão disponíveis na nossa página, e o Prof. Ricardo está providenciando para essa publicação um número de ISBN, que está sendo devidamente registrado. Poderemos ter também publicação em papel se as pessoas assim desejarem. Para quem ainda quiser esta revista do WTA 2007, está sendo cobrado preço de custo para uma edição encomendada.

Outro ponto que desejaria mencionar refere-se às iniciativas que estamos tomando aqui, e que podem ser repetidas também nas instituições de cada um: promover a Tecnologia Adaptativa através de palestras, workshops, convidar os colegas que apresentaram trabalhos para falarem lá também, etc. São muitas as formas possíveis de promover a Tecnologia Adaptativa dentro do país. No exterior, tenho tido algum contato com dois grupos peruanos, um na Pontifícia Universidad Católica del Perú, em Lima, e outro, na Universidad Católica San Pablo, em Arequipa. São dois grupos que estão formando núcleos de pesquisa em Tecnologia Adaptativa, com quem temos articulado projetos colaborativos. Algumas coisas já foram iniciadas, e outras, encaminhadas para que tenhamos ao menos algum patrocínio que viabilize materializar essas coisas, ao menos parcialmente, de forma presencial.

C. (Prof. Almir)

É importante ver o Segundo WTA acontecendo, com muitas coisas ocorrendo, com muitas coisas discutidas, tendo como ponto forte o tutorial do Prof. Ítalo, que abriu a área para a Orientação a Objetos e Engenharia de Software, área pela qual que tive sempre interesse, pela minha formação e trabalho.

Vejo que uma das coisas interessantes seria termos algumas pessoas se dedicando ao desenvolvimento de aplicações comerciais ou coisas similares, usando Tecnologia Adaptativa.

Já existem muitas aplicações de voz, imagens, jogos, mas pouco são exploradas as tantas aplicações comerciais complexas que existem.

Pretendo investir em Tecnologia Adaptativa nessas aplicações. Seria um ganho para as empresas, e utilidade das linguagens adaptativas no desenvolvimento dessas aplicações.

D. (Prof. André)

Algumas idéias adicionais, que surgiram nesta Mesa Redonda: A primeira, tornar o WTA itinerante; a outra, para quem está chegando agora, esta revista e as suas referências dão boa idéia da área. Quem os escreveu, relata

aplicações, com linguagem generalista e próxima das aplicações.

O Prof. Ricardo está obtendo um ISBN para todos os trabalhos apresentados neste evento, que serão publicados portanto na forma de um livro eletrônico, mostrando mais uma vez a importância deste evento.

E. (Eng. Danilo Bellini)

Tenho umas sugestões, para aplicações na área musical, do uso de Tecnologia Adaptativa para resolver problemas até comuns na área, alguns foram citados na apresentação do projeto Orfeu. Foi citado o OCR para partituras gráficas e converter em music xml ou similar.

Outro é conversão de som matricial arquivos .mp3, .wav ou outro em partitura. Isso necessita 2 passos: o primeiro refere-se à conversão de arquivos de som para o formato midi, sem informações sobre interpretação e o segundo, que se destina a converter para o formato de partitura um som vetorial, sem informações de compasso ou de bloco rítmico. Este poderia usar muito bem a Tecnologia Adaptativa.

Uma terceira proposta é a do projeto de um hardware, motivada pelo fato de eu tocar teclado. O problema básico é que os teclados são caros, mas existem programas sintetizadores por computador, que poderiam bem usar no computador versões Midi da música, mas com isso seriam perdidos muitos recursos musicais.

Por exemplo, para ter dois ou três teclados em conexão com dois computadores, a idéia seria usar as místicas "réguas midi", que permitiriam a conexão de vários instrumentos midi, com vários sintetizadores, para que se fizessem conversões de dados de entradas midi antes de converter para o formato de saída, por exemplo trocando o canal midi, ou pegando uma faixa do teclado para ser um timbre e outra, para ser outro, e isso seria uma aplicação muito interessantes que explorasse os recursos proporcionados pela Tecnologia Adaptativa.

XIII. ENCERRAMENTO

A. (Prof. João)

Bem, acabou a festa. Missão cumprida, terminamos nossos trabalhos, gostaria de agradecer pessoalmente a cada um dos componentes da Mesa Redonda, que abrilhantou este evento.

Com isso, gostaria de deixar para vocês esse desafio de que para o ano que vem todos os que tenham projetos na área se animem a escrever. Tenho a maior boa vontade de participar, discutir as idéias.

Gostaria de estimulá-los a gerar essas dez publicações no mínimo, para que posamos manter viva essa linha de pesquisa, de interesse de nossa pequena comunidade, que começa a se expandir.

O importante é que cada um de nós seja um veículo de disseminação da Tecnologia Adaptativa. Todos têm idéias fantásticas, muito interessantes, mas muitíssimas infelizmente acabam se perdendo, por falta de registro.

Então, a idéia do Prof. Amaury, de ter na página um espaço para isso, cria um ambiente natural para as pessoas se comunicarem, e para que se possa prosseguir nesse trabalho que estamos fazendo.

Eu queria deixar aqui um abraço a cada um de vocês, com meu agradecimento pela presença e pela participação, que foi fundamental para o sucesso deste evento, e o convite, desde agora, para participarem da próxima edição do WTA.

Temos o e-mail de vocês, e iremos usá-lo para nos comunicarmos. Por favor não apaguem as nossas correspondências, pois gostaríamos de prosseguir e manter forte a nossa ligação, que está ficando cada vez maior. Então, muito obrigado, e até o ano que vem!

ESTA TRANSCRIÇÃO LIVRE, DAS FALAS DA MESA REDONDA, FOI PRODUZIDA PELO PROF. DR. JOÃO JOSÉ NETO A PARTIR DE GRAVAÇÃO EM VÍDEO REALIZADA DURANTE O EVENTO. PROCUROU-SE RESPEITAR FIELMENTE TUDO O QUE FOI PROFERIDO NA OCASIÃO, PORÉM FORAM NECESSÁRIAS PEQUENAS ALTERAÇÕES NAS FALAS, COM A FINALIDADE DE PRIORIZAR O CONTEÚDO TÉCNICO, BEM COMO ELIMINAR REDUNDÂNCIAS E EXPRESSÕES COLOQUIAIS OU OUTRAS, PRÓPRIAS DA LINGUAGEM FALADA, MAS INADEQUADAS A UM REGISTRO ESCRITO DESTA NATUREZA.

ÍNDICE DE AUTORES

Barros Neto, L. C. _____ 1
Bellini, D. J. S._____ 39
Camargo, R._____ 44
Camolesi, A. R. _____ 31
Cereda, P. R. M. _____ 9
Gomes, W. C. M._____ 31
Hirakawa, A. R._____ 1
Koike, F. R. _____ 35
Lopes, V. D._____ 14
Massola, A. M. A. _____ 1
Mignon, A. S. _____ 18
Rocha, R. L. A. _____ 14, 27
Rodrigues, E. S. C. _____ 27
Rodrigues, F. A._____ 27
Shibata, D. P._____ 35
Tavella, A. C. B. _____ 39
Tchemra, A. H. _____ 5, 44
Werneck, N. L. _____ 22
Zorzo, S. D. _____ 9