Memórias do WTA 2013

VII 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
2014

Ficha catalográfica

Workshop de Tecnologia Adaptativa (7: 2013: São Paulo)
Memórias do WTA 2013. – São Paulo; EPUSP, 2013. 136p.

ISBN 978-85-86686-75-7

ISBN 978-85-86686-61-0

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

CDD 621.39

Apresentação

A sétima edição do Workshop de Tecnologia Adaptativa realizou-se em São Paulo, Brasil, nos dias 06 e 07 de Fevereiro de 2013, nas dependências da Escola Politécnica da Universidade de São Paulo. As contribuições encaminhadas na forma de artigos relacionados à Tecnologia Adaptativa, nas seguintes áreas, abrangeram, de forma não exclusiva, os tópicos abaixo:

Fundamentos da Adaptatividade

- Modelos de computação, autômatos, gramáticas, grafos e outros dispositivos automodificáveis, suas notações, sua formalização, complexidade, propriedades e comparações com formalismos clássicos.

Tecnologia Adaptativa – Técnicas, Métodos e Ferramentas

- Aplicação dos conhecimentos científicos relativos à adaptatividade e dos dispositivos adaptativos como fundamento para a formulação e para a resolução de problemas práticos.
- Ferramentas, técnicas e métodos para a automatização da resolução de problemas práticos usando técnicas adaptativas.
- Programação adaptativa: linguagens, compiladores e metodologia para o desenvolvimento, implementação e validação de programas com código adaptativo.
- Meta-modelagem de software adaptativo.
- Engenharia de Software voltada para a especificação, projeto, implementação e desenvolvimento de programas automodificáveis de qualidade.
- Adaptatividade multinível e outros conceitos introduzidos recentemente: avanços teóricos, novas idéias para aplicações, sugestões de uso prático.
- Linguagens de alto nível para a codificação de programas automodificáveis: aplicações experimentais e profissionais, práticas e extensas, das novas idéias de uso de linguagens adequadas para a codificação de programas adaptativos e suas metodologias de desenvolvimento.

Aplicações da Adaptatividade e da Tecnologia Adaptativa

- Inteligência computacional: aprendizagem de máquina, representação e manipulação do conhecimento;
- Computação natural, evolutiva e bio-inspirada;
- Sistemas de computação autonômica e reconfigurável;

- Processamento de linguagem natural, sinais e imagens: aquisição, análise, síntese, reconhecimento, conversões e tradução;
- Inferência, reconhecimento e classificação de padrões;
- Modelagem, simulação e otimização de sistemas inteligentes de: tempo real, segurança, controle de processos, tomada de decisão, diagnóstico, robótica;
- Simulação, arte por computador e jogos eletrônicos inteligentes;
- Outras aplicações da Adaptatividade, nas diversas áreas do conhecimento: ciências exatas, biológicas e humanas.

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)

Apoio

- Escola Politécnica da Universidade de São Paulo
- IEEE – Institute of Electrical and Electronics Engineers
- PCS – Departamento de Engenharia de Computação e Sistemas Digitais
- SBC – Sociedade Brasileira de Computação
- SPC – Sociedad Peruana de Computación
- Universidade de São Paulo

Memórias do WTA 2013

Esta publicação do Laboratório de Linguagens e Técnicas Adaptativas do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo é uma coleção de textos produzidos para o WTA 2013, o Sétimo Workshop de Tecnologia Adaptativa, realizado em São Paulo nos dias 06 e 07 de Fevereiro de 2013. A exemplo da edição de 2012, este evento contou com uma forte presença da comunidade de pesquisadores que se dedicam ao estudo e ao desenvolvimento de trabalhos ligados a esse tema em diversas instituições brasileiras e estrangeiras, sendo o material aqui compilado representativo dos avanços alcançados nas mais recentes pesquisas e desenvolvimentos realizados.

Introdução

Com muita satisfação apresentamos neste documento estas memórias com os artigos apresentados no WTA 2013 – Sétimo Workshop de Tecnologia Adaptativa, realizado na Escola Politécnica da Universidade de São Paulo nos dias 06 e 07 de Fevereiro de 2013.

Esta edição contou com transmissão em tempo real durante os dois dias do evento através da infraestrutura proporcionada pelo Sistema IPTV USP, permitindo participação remota. Vários grupos externos foram organizados para acompanhar os trabalhos:

Nome	Instituição
Amaury Antônio de Castro Júnior	UFMS
Ana Cristina Fricke Matte	UFMG
David Augusto Guimarães	UCDB
Guilherme Maluf Balzana	UFMG
Hemerson Pistori	UCDB
Hugo Leonardo Canalli	UFMG
Kleber Padovani de Souza	UCDB
Leonardo Freitas	UFMG
Lia Nara Balta Quinta	UCDB
Manassés Ferreira Neto	UFMG
Rafael José Puiati Bergamaschi	UFMG
Rafael Luís Caldas Almeida	UFMG
Reginaldo Inojosa da Silva Fillho	UFMS
Renata Luiza Stange Carneiro Gomes	UTFPR

Tabela 1: Lista dos professores que colaboraram na participação à distância de grupos externos.

O evento também permitiu a interação dos participantes à distância através de um canal de IRC disponibilizado pelo grupo Texto Livre do Laboratório SEMIOTEC da Faculdade de

Letras da Universidade Federal de Minas Gerais, sob coordenação da profa. Ana Cristina Fricke Matte, viabilizando em boa medida a infraestrutura e a operação da parte externa.

Participações especiais

A sétima edição do Workshop de Tecnologia Adaptativa contou com a participação especial do prof. Ricardo Luis de Azevedo da Rocha através de uma palestra especial gravada na Universidade de Alberta, Canadá, e do prof. Amaury Antônio de Castro Junior, apresentando o grupo de pesquisa PET Fronteira da Universidade Federal de Mato Grosso do Sul, campus de Ponta Porã.

Esta publicação

Estas memórias espelham o conteúdo apresentado no WTA 2013. A exemplo do que foi feito no ano anterior, todo o material referente aos trabalhos apresentados no evento estará acessível no portal do WTA 2013, 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.

Conclusão

A repetição do sucesso das edições anteriores do evento, e o nível de qualidade dos trabalhos apresentados atestam a seriedade do trabalho que vem sendo realizado, e seu impacto junto à comunidade. Somos gratos aos que contribuíram de alguma forma para o brilho do evento, e aproveitamos para estender a todos o convite para participarem da próxima edição, em 2014.

Agradecimentos

Às instituições que apoiaram o WTA 2013, à comissão organizadora, ao pessoal de apoio e a tantos colaboradores voluntários, cujo auxílio propiciou o êxito que tivemos a satisfação de observar. Gostaríamos também de agradecer aos seguintes funcionários pelo valioso auxílio para a viabilização das necessidades locais do evento:

- Daniel Costa Ferreira
- Edson de Souza
- Leia Sicília
- Nilton Araújo do Carmo

De modo especial, agradecemos ao Departamento de Engenharia de Computação e Sistemas Digitais e a Escola Politécnica da Universidade de São Paulo pelo inestimável apoio para a realização da sétima edição do Workshop de Tecnologia Adaptativa.

São Paulo, 07 de Fevereiro de 2013

João José Neto
Coordenador do evento

Sumário

Lista de autores

Parte I

Submissões

A programação Windows integrada com Delphi para o desenvolvimento de Recursos Computacionais voltados a Acessibilidade de Tetraplégicos com a fala comprometida

França, C. R.

***Abstract*— This paper presents the most relevant programming techniques used in the creation of Windows software components, whose use is not easily learned with a simple reading of the manuals or online help. We describe the technical details of implementing adaptive systems in the Windows environment, in particular on the use of certain APIs (Application Programming Interfaces), a mechanism called "Windows Hooks."**

***Keywords*— Windows environment, Toolkit of components , Adaptative Technology, Development motor impaired.**

I. INTRODUÇÃO

Neste artigo apresentam-se as técnicas mais relevantes da programação Windows utilizadas na criação de Componentes de software, cujo uso não é facilmente aprendido com a mera leitura dos manuais ou *help on-line*. São descritos os detalhes técnicos de implementação de sistemas adaptativos no ambiente Windows, em especial sobre a utilização de algumas APIs (Application Programming Interfaces), do mecanismo conhecido como "*Windows Hooks*".

Este trabalho é fruto da experiência do autor em desenvolvimento de componentes para criação de software para pessoas tetraplégicas, tendo sido aplicado no desenvolvimento de um Toolkit de Componentes de Software, denominado Toolkit Tupi. O referido trabalho foi executado para a obtenção do título de Mestre em informática do Programa de Pós-graduação do Núcleo de Computação e Eletrônica da Universidade Federal do Rio de Janeiro - NCE/UFRJ.

II. O DESENVOLVIMENTO DE SOFTWARE DE ACESSIBILIDADE, COM REUSO DE COMPONENTES DE UM TOOLKIT.

O desenvolvimento de software de acessibilidade tem como forte característica à interação com o sistema operacional. O TOOLKIT TUPI foi implementado na linguagem Delphi, mas toda a sua concepção envolve os conceitos que permeiam a programação Windows. A programação em Delphi, para a maior parte das aplicações, encapsula as chamadas ao sistema operacional na forma de componentes que ocultam os detalhes de programação do sistema operacional Windows. Porém, na programação do Tupi, esses componentes facilitadores não estavam disponíveis, sendo necessário o seu desenvolvimento. Para tanto necessitou-se de uma intensa programação no nível do sistema operacional Windows, bem como o uso de técnicas especiais para sincronismo com os elementos internos deste sistema.

O desenvolvimento do TUPI envolveu a execução de tarefas complexas de programação, que fazem uso das chamadas "interfaces com programas de aplicação" (*application program interfaces* ou API's) e técnicas que dão acesso não convencional ao sistema operacional Windows, em especial:

a) Execução de procedimentos que interferem nas filas de processamento do Windows.

b) Operações de entrada e saída não suportadas diretamente pelo sistema operacional.

A maior parte destas técnicas é muito pouco conhecida no Brasil, mesmo entre programadores experientes. Por este fato, é relevante mostrar aqui uma coletânea dos mecanismos mais complexos que foram utilizados na implementação do TUPI, com o objetivo de gerar uma referência para programadores que desejem ampliar ou realizar manutenção em sistemas adaptativos ou mesmo sistemas não convencionais que executem no ambiente Windows.

Carlos Roberto França - Universidade Federal da Fronteira Sul - Campus Realeza, carlos.franca@uffs.edu.br

Inicia-se com uma introdução para apresentar alguns pontos chaves e uma breve contextualização técnica sobre o mecanismo genérico de acionamento das APIs, o que envolve também o entendimento de alguns mecanismos como *callback (*chamadas ao programa de aplicação realizadas pelo sistema operacional) e *hooks* (ganchos utilizados para colocar uma tarefa na fila de execução do Windows). Em seguida são apresentadas as soluções que foram implementadas no TUPI relativas ao acoplamento com os *hooks*.

III. ENTENDENDO O MECANISMO UTILIZADO NAS DLLS DO WINDOWS

a) mensagens

O Windows é um sistema operacional orientado a mensagens. Muitos pedidos ao sistema operacional, bem como muitas das respostas enviadas por ele, especialmente os que são realizados de forma assíncrona, são encapsulados em "mensagens",(que são estruturas de dados padronizadas que definem os "detalhes" destes pedidos ou respostas). As mensagens são colocadas em filas por rotinas como PostMessage (envio assíncrono), SendMessage (envio síncrono) , PeekMessage (recepção assíncrona) e GetMessage (recepção síncrona).

b) DLLs

Segundo Hetch [Hetch, 2001] [1], todos sistemas Windows são construídos na forma de "bibliotecas dinâmicas" (DLL – dynamic link libraries), que podem ser compartilhadas por múltiplas aplicações. O código e os recursos contidos nas DLLs não são geralmente carregados na memória até a hora em que são necessários e podem ser descartados quando termina seu uso. A utilização de DLLs, em comparação com a técnica antiga de *linkage-edition* apresenta diversa vantagens, como:

- economia de espaço em disco;

- a possibilidade de fazer uma atualização e trocar partes de um programa sem ter que recompilar tudo;

- a possibilidade de carregar diferentes recursos e código baseado em algum critério específico, disponibilizado apenas em *runtime*.

O pequeno trecho a seguir, baseado em [Hetch,2001] [1], exemplifica o esquema básico de programação de uma rotina simples de uma das principais APIs do Windows: a USER32, responsável por grande parte da interface com o usuário, além do processamento do teclado e de mouse. Nele é mostrado a carga da DLL na memória, o uso seguro de um de seus métodos, e a descarga da DLL da memória.

```
Procedure ExecuteBeep;
Var
  DllHandle:Thandle;
  BeepFn:function(BeepType:integer): Bool sdtcall;
Begin
  // responsável por carregar a biblioteca USER32 na memória
  DllHandle:=LoadLibrary('USER32');
  // assegura-se que conseguiu carregar
  If DllHandle<>0 then
    begin
        // obtém o endereço da rotina de bip, permitindo a sua manipulação.
        @BeepFn:= GetProcAddress(DllHandle,'MessageBeep');
        // testa se a função de bip pode ser utilizada.
        if @BeepFn<> nil then
            // executa um tipo específico de bip
            BeepFn($FFFFFFFF);
        // libera a biblioteca dinâmica da memória
        FreeLibrary(DllHandle);
  End;
End;
```

Como é fácil perceber pela leitura deste trecho, é muito complicada uma interação direta com o sistema operacional, mesmo para realizar uma tarefa tão trivial como essa, dar um bip, dada a quantidade de detalhes específicos envolvidos. Assim é fundamental a criação de mecanismos de encapsulamento de detalhes que atendam às situações mais comuns.

c) Funções de chamada de retorno (callback)

Numa situação simples, o sistema operacional é chamado para realizar alguma tarefa e na maior parte das vezes o programa espera que esta tarefa seja concluída para continuar. Esse mecanismo, entretanto, não atende às necessidades de um processamento interativo assíncrono,

pela própria natureza deste processamento, onde a execução de uma tarefa não pode ser descrita por uma função de tempo linear.

As funções de chamada de retorno permitem especificar o endereço de uma função que será chamada automaticamente pelo sistema operacional ou por uma DLL na ocorrência de um evento qualquer (por exemplo, ao término da ação pedida). Desta forma é criada uma situação em que o sistema operacional não é apenas chamado pelo programa do usuário, mas ele próprio pode chamar o programa de aplicação quando necessário.

O uso mais trivial deste mecanismo é a execução de operações de Entrada e Saída, em que um erro ou término de processamento, ativa uma rotina do programa de aplicação permitindo assim que o tratamento seja realizado de forma completamente assíncrona.

d) Recursos (Resources)

Um caso particular de utilização de bibliotecas pode ser dado pelas utilizações de interfaces gráficas e textos de mensagens. Um compilador especial (*Resource Compiler)* permite a criação de uma estrutura de dados que especifica o posicionamento de botões, rótulos, *listbox*, etc, que são na verdade rotinas implementadas dentro de DLLs específicas, bem como os textos que são usados na formatação de telas e relatórios. Em tempo de execução, essa estrutura de dados é acessada pelo programa e entregue ao sistema operacional, que realiza, em *background*, as seqüências de chamadas referentes à manipulação destes componentes, inclusive as funções de *callback*.

e) componentes

A maior dificuldade de programação do Windows está no número imenso de mensagens, APIs e DLLs que existem (muitas centenas) cada uma das quais disponibilizando inúmeras funções, somado a uma fenomenal quantidade de convenções de que essas funções fazem uso (tais como códigos de ativação e interpretação de retornos, por exemplo). Segundo Piccard [2], por definição, cada APIs deve incluir uma descrição de funções realizadas pelo módulo e os mecanismos usados para passar informação para dentro e para fora do módulo. Tipicamente isso vai incluir a especificação do:

- Número e ordem dos argumentos;
- Para cada argumento, o tipo de variável e se a informação é passada por valor ou por referência.
- Informações de localidade em relação a determinados módulos.

Essa complexidade fez com que hoje em dia o uso de componentes que encapsulam ou reorganizam este imenso número de detalhes se torne cada vez mais comum na programação de Windows. Dependendo do uso pretendido, os componentes podem ser implementados na forma de bibliotecas específicas ou então num padrão conhecido como "Component Object Model", cuja interface mais usada recebe o nome genérico de Active X, [3] que é um mecanismo genérico criado pela Microsoft para permitir o uso destes componentes em diversas linguagens ou em aplicativos específicos (como editores de textos e planilhas, entre outros).

IV. ACESSO A DISPOSITIVOS

No Windows, todos os dados que são provenientes ou que se destinam a um dispositivo externo, são controlados por uma rotina específica (device driver), geralmente construída pelo fabricante do *hardware*, e que se encarrega de tratar de todos os detalhes específicos da entrada ou saída. O Windows mantém para cada dispositivo, filas de dados que serão escritos ou que foram lidos, e que provêm dos *driver*, pois a E/S quase sempre ocorre de forma assíncrona. Entre os *driver* e os *buffer* que receberão ou que contém os dados, existe um complexo mecanismo de troca de mensagens, que visa estabelecer proteção num sistema multiprogramado como o Windows.

Todo este processo é apresentado para o usuário através de uma interface de programa de aplicação (API), no qual os detalhes internos são escondidos. A API é basicamente um conjunto de rotinas ou macros que fazem acessos às funções da DLL.

O acesso direto aos *device driver* quase nunca é realizado, sendo normalmente mediado por chamadas específicas a APIs particulares do sistema operacional. Essas APIs mantém para cada dispositivo, filas de dados que serão escritos ou que foram lidos, e que provêm dos driver, pois a E/S quase sempre ocorre de forma assíncrona. Entre os *driver* e os *buffer* que receberão ou que contém os dados, existe um complexo mecanismo de troca de mensagens, que visa estabelecer proteção num sistema multiprogramado como o Windows.

V. TRATAMENTO DE INTERRUPÇÕES NO WINDOWS

Como descrito no artigo "Programming Interrupts for DOS-Based Data Acquisition on 80x86-Based Computers" [National Instruments] [4] desde a época do sistema operacional DOS uma prática para acessar o fluxo destinado aos *driver* era dado a partir de uma tabela de tratamento de interrupções de acesso, causando um desvio para uma rotina especifica e vice e versa. (fig .1)

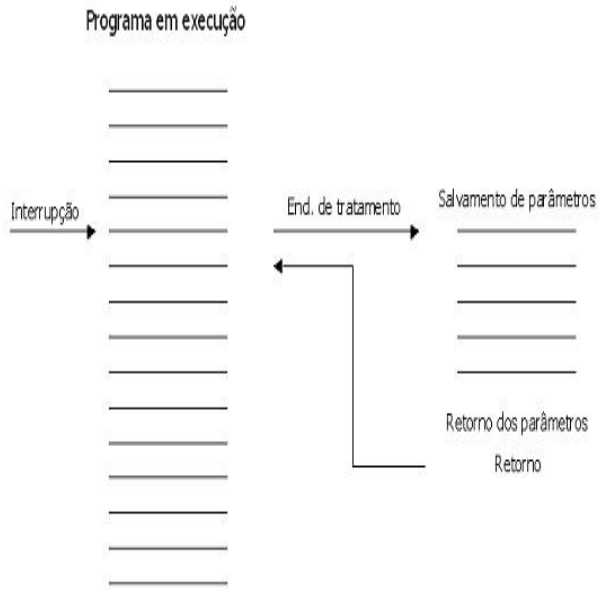


Figura 1. tratamento de interrupções

Como mecanismo de proteção, o Windows inibe, em geral, o uso dessas técnicas, permitindo que o acesso a interrupções seja feita unicamente através de rotinas certificadas (driver) e que obedecem a convenções muito específicas de sistema operacional.

1- O código precisa estar dentro de uma DLL;

2- O código precisa exibir uma execução muito rápida;

3- O programador terá que prover o encadeamento de interceptações de modo a coordenar o fluxo das mesmas.

Para acessarmos, adicionarmos ou interceptarmos alguma informação proveniente de periférico (na verdade não apenas a fila de E/S, mas até mesmo a fila de mensagens do sistema) uma alternativa viável seria a reescrita de *driver* para incorporar as tarefas especificas, de acordo com as necessidades. Esta alternativa deve, entretanto ser descartada pelo o fato das funcionalidades mais especificas serem de acesso exclusivo do fabricante dos *driver*.

O mecanismo usado para atender a estes requisitos são os *Windows Hook* que, através do qual uma sub-rotina é colocada no caminho do mecanismo normal de tratamento de mensagens do Windows.

VI . WINDOWS HOOKS

Como Rutledge descreve em seu artigo "Advanced Delphi – Windows Hooks"[5], uma "hook procedure" captura do sistema certas mensagens do Windows antes delas serem enviadas para as devidas rotinas de tratamento. No sistema operacional Windows existem vários tipos diferentes de "procedures hook", onde cada um deste tipo fornece um aspecto diferente do mecanismo de tratamento de mensagens do Windows. É, portanto uma função que pode ser introduzida no sistema de mensagem do Windows, assim uma aplicação pode acessar a mensagem corrente antes de outro processo tomar o lugar no processamento.

O quadro 1 - abaixo descreve os diversos tipos de *hooks* que são disponíveis no Windows.

WH_CALLWNDPROC	é chamada sempre que a função *SendMessage()* for chamada..
WH_CBT	usada especialmente para implementação de treinamentos, interceptando o sistema antes da chamada computacional de: ativação; criação; destruição; minimização; maximização; movimentação ou dimensionamento de uma janela; antes de completarem um comando de sistema ; antes de um novo movimento do mouse ou um evento do teclado; antes de ajustar o foco de entrada ; e antes da sincronização do sistema da fila de mensagens.

WH_DEBUG	Permite a criação de uma rotina de debug que atuará antes da chamada de um filtro instalado.
WH_GETMESSAGE	faz uma chamada de mensagem sempre que a função *GetMessage()* tenha recuperado uma mensagem na fila de aplicação
WH_HARDWARE	usada sempre que ocorre uma chamada de mensagem hook sem ser padrão de hardware, a aplicação chama as funções *GetMessage()* ou *PeekMessage()* e um evento de hardware (como um evento de mouse, teclado ou outro para processar).
WH_JOURNALRECORD	tem por função instalar uma **procedure hook** especialmente para gravar as mensagens de entrada afixadas na fila de mensagem do sistema pela *WH_JOURNALPLAYBACK*
WH_JOURNALPLAYBACK	tem por função enviar mensagens previamente gravadas pela a hook. WH_JOURNALRECORD. O tipo da função "callback", chamada de retorno, é jounalRecordProc.
WH_KEYBOARD	é chamada sempre que uma aplicação chamar as funções *GetMessage()* ou *PeekMessage()* e existir uma mensagem de teclado para processar, *WM_KEYUP* ou *WM_KEYDOWN* .
WH_MOUSE	é chamada sempre que uma aplicação chamar as funções *GetMessage()* ou *PeekMessage()* e existir uma mensagem de mouse para ser processada.
WH_MSGFILTER-	é uma chamada de mensagem hook para uma aplicação depois de uma caixa de diálogo, uma mensagem de caixa ou um menu de mensagem recuperada , porém antes a mensagem é processada.
WH_SHELL	É uma procedure de chamada da aplicação Shell , quando uma mensagem de notificação do sistema tiver sido feita.
WH_SYSMSGFILTER	Essa procedure faz uma chamada de sistemas após uma caixa de diálogo, uma mensagem de caixa, ou menu de recuperação de uma mensagem, antes que a mensagem esteja processada.

O Windows mantém internamente um "*hook chain*", que funciona como uma lista de ponteiros para as *"procedure hook*" com finalidades idênticas que os diversos programas instalaram. Este procedimento do sistema operacional Windows se faz necessário pelo fato de muitos programas instalarem uma *"procedure hook"*. Quando acontece uma chamada de mensagem no sistema, o Windows primeiro passa por cada uma das procedures no *"hook chain"* uma depois da outra. Então, caso a mensagem não tenha sido bloqueada por qualquer uma das *"procedure hook*", o Windows encaminha a mensagem para a janela adequada.

VII. IMPLEMENTANDO UMA *WINDOWS HOOK* PARA *JOURNAL PLAYBACK*: A DLL DVKBM32

Cada *Windows hook* possui um certo objetivo específico, mas existe uma grande intersecção entre as funções que elas executam. Na implementação do Tupi os requisitos para definição desses objetivos eram simples: deveria ser possível introduzir elementos que permitissem simular eventos de mouse e teclado, introduzindo novos itens nas filas respectivas.

A escolha da Windows Hook específica deveria atender também a alguns critérios importantes

a) Aplicabilidade a um grande número de situações: dada a complexidade de implementação, seria importante pensar em uma solução que pudesse ser aproveitada em situações similares, sem modificação.

b) Sincronismo de execução: O funcionamento do Hook deveria permitir um uso completamente síncrono, o que permitiria o seu uso em algumas situações ou obriga a criação de "threads de execução" para evitar o bloqueio de processamento. Apesar disso, em todo desenvolvimento do TOOLKIT TUPI, não houve nenhum momento em que esse sincronismo trouxesse maiores dificuldades à programação.

c) Encapsulamento das dificuldades: Os procedimentos e funções de ativação deveriam ser o mais invisível possível.

A solução usada foi baseada em uma implementação já existente, criada originalmente para uso nos sistemas DOSVOX e MOTRIX, mas cuja documentação inexistia até a criação do TUPI: a DLL DVKBM32. Segundo uma entrevista realizada com o desenvolvedor, professor Antonio Borges, do NCE/UFRJ "o *principal interesse era sintetizar as ações e eventos que teríamos que lidar nos nossos desenvolvimentos e principalmente evitar a necessidade do tratamento de conflitos e interrupções alheias à nossa aplicação. A escolha recaiu sobre a Windows Hook JournalPlayback por ela ser de uso amigável e poder comunicar-se sem conflitos com a maioria das APIs do Windows.* "

A DLL DVKBM32 basicamente mantém uma fila de ações desejadas e seu tempo de duração pretendido. O usuário pode inserir nesta fila pedido de ação de mouse e teclado. Uma rotina chamada PlayStart dá início ao processamento, estabelecendo o *hook*. Quando a fila está vazia, a própria rotina desestabelece o hook.

Neste ponto, é oportuno destacar que a reusabilidade e a usabilidade de componentes são algumas das principais aplicabilidades de um toolkit. A reusabilidade esta caracterizada pelo reuso de componentes sem que haja mudanças nas suas linhas de código, mas com aplicabilidades diferentes das que o originaram. Ou seja: O componente permanece com as suas características, mas a sua empregabilidade foi diversificada. Quando o componente é utilizado na resolução do problema alvo que o originou, diz-se se tratar de uma usabilidade. Na verdade, todas e quaisquer implementações que envolvem componentes de um toolkit, estarão fortemente ligadas à interface dos mesmos e as configurações que permitiram os ajustes e padronizações de acordo com os objetivos do desenvolvedor.

Por sua vez, o que garante a utilização das aplicabilidades, configurações e padronizações dos componentes de um toolkit, são os motivos que levaram a utilização do mesmo na resolução de determinados problemas. As facilidades técnicas do conjunto de ferramentas podem ser mais ou menos aproveitadas, de acordo com a habilidade do desenvolvedor. A construção de Software baseada e/ou orientada a componentes, utilizam padrões de projetos que implementam métodos como o *Catalysis*, e uma linguagem de programação de alto nível orientada a objetos, como por exemplo, o Delphi.

O quadro 2 lista as funções disponibiliza na DLL DVKBM32

PlayInit	cria um array que conterá a lista de ações desejadas. Uma das informações que será armazenada é o tempo que será gasto em cada ação.
PlayMouse	insere no array as informações sobre uma ação de mouse (mover, clicar, e as coordenadas).
PlayKey	insere no array as informações sobre a tecla a ser "pressionada"
PlayVirtKey	semelhante a PlayKey informando código virtual e não o ASCII
PlayAltKey	idem para tecla ALTs
PlayInsertKeyEvent	insere um evento de baixo nível de teclado (como a informação de apertar e desapertar)
PlayStart	ativa o hook
PlayIsActive	Função que vê se a fila ficou vazia nesta DLL

A JournalPlayback é ativada como um callback do Windows, em que são especificados os seguintes tipos de mensagem:

HC_SKIP – o windows indica que está pronto para atender a um pedido

HC_NEXT – o windows indica que acabou de atender ao último pedido

Outros tipos de mensagem podem ser enviados pelo Windows a uma JournalPlayback, mas são pouco importantes para nossa implementação.

A listagem a seguir apresenta o algoritmo utilizado no "núcleo" desta DLL

```
function PlaybackProc(Code: Integer;
        wParam: TwParam;
        lParam: TlParam): Longint; stdcall;
var
   TimeToFire: Longint;

begin
   PlaybackProc := 0;
   case Code of

      HC_SKIP:
      begin
         CurrentMsg := CurrentMsg + 1;
         if CurrentMsg >= MsgCount then
           if TheHook <> 0 then
             if UnHookWindowsHookEx(TheHook) then
               begin
                 TheHook := 0;
                 FreeMem(PMsgBuff, Sizeof(TMsgBuff));
                 PMsgBuff := nil;
               end;
         exit;
      end;

      HC_GETNEXT:
      begin
        PEventMsg(lParam)^ := PMsgBuff^[CurrentMsg];
        PEventMsg(lParam)^.Time :=
             StartTime + PMsgBuff^[CurrentMsg].Time;
        TimeToFire :=
             PEventMsg(lParam)^.Time - GetTickCount;
        if TimeToFire > 0 then
             PlaybackProc := TimeToFire;
        exit;
      end;

.......

   If code < 0  then
     PlaybackProc :=
       CallNextHookEx(TheHook, Code, wParam, lParam);
end;
```

IX. UMA INTERFACE SIMPLIFICADA PARA A DLL: DVMACRO

Para uso prático da DLL foi criada uma interface simples, que encapsulasse as funções mais utilizadas, denominada DVMACRO. A funcionalidade desta interface foi talhada para as necessidades dos software de acessibilidades para deficientes visuais (DOSVOX), e para deficientes motores (MOTRIX), e recentemente, com as necessidades do Tupi. Os procedimentos e funções da DVMACRO são responsáveis pela a ativação dos *hooks procedures* utilizadas em todos os componentes do TOOLKIT TUPI.

Além de encapsular diretamente as rotinas básicas da DVKBM32, esta interface ainda implementa diversas funções de conveniência que realizam a execução de forma síncrona de diversas ações (em outras palavras, uma seqüência de PlayInit – Ação específica – PlayStart – e espera terminar), para inserção de uma ou várias teclas, diversos tipos de cliques no mouse e de dragging de cursor.

VII. ACESSO À PORTA PARALELA DO PC EM WINDOWS NT E SUCEDÂNEOS

A porta paralela do PC, utilizada habitualmente para conexão de impressoras, é muito adequada para conexão de dispositivos não convencionais. Os sinais que são produzidos nos pinos de saída desta *interface* ou que são lidos através de seus pinos de entrada têm um sistema de proteção e amplificação de sinal que simplifica muito a conexão de chaves, leds, circuitos acionadores, e outros dispositivos cuja conexão tenha características *ON/OFF*, ou seja, baseada em interfaces digitais.

A referência [Axelson, 1998] [6] é um bom guia para programação de portas paralelas em diversas situações. Um exemplo tirado do livro de Gabriel Torres [Torres, 2001] [7], demonstra que a programação de acesso à porta paralela é uma tarefa bem simples a partir dos endereçamentos da porta paralela. Os códigos de acesso podem ser gerados em Assembly, utilizando as instruções *IN e OUT* para leitura e escrita, respectivamente, ou em Pascal como o exemplo abaixo.

Em Pascal para MS-DOS, utiliza-se o comando PORT, conforme o exemplo abaixo.

```
program portaParalela;
uses crt;
begin
   // coloca o valor 255 (ffh) na porta paralela
   port [$378]:= 255;
end.
```

Escrever programas para se comunicar via porta paralela, nos sistemas operacionais Win95/98, é tarefa relativamente fácil a partir dos comandos do MS-DOS ou utilizando o endereçamento da porta paralela, como mostrado acima. Mas quando a programação é feita

nos sistemas operacionais Win NT, Win2000 e WinXP, toda a simplicidade desaparece, pois nestes sistemas operacionais, o acesso a dispositivos é privilégio exclusivo dos *driver* do sistema operacional, e um acesso direto a portas paralelas através dessas mesmas rotinas provoca uma exceção de instrução privilegiada, geralmente associada a uma mensagem como na fig. 2

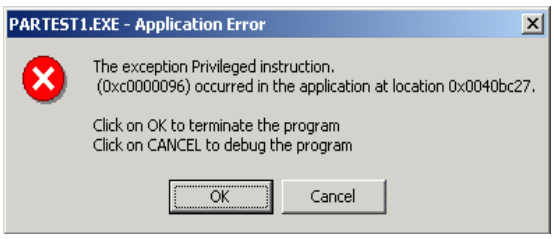


Figura 2 . Mensagem de erro

O Windows NT atribui alguns privilégios e limitações aos tipos diferentes de programas que funcionam nele a partir da classificação dos programas de acordo com o seu modo de execução. Os programas podem executar em modo usuário ou pelo *kernel mode*, e os programas serão alocados para rodarem em camadas ou anéis do núcleo *(Kernel)*.

No modo usuário, os programas rodam na camada três ou *ring3Mode*, como é mais conhecido na programação Windows, e são restritas as instruções do tipo entrada e saída (*in/out) e outras similares* no modo *kernel*, os programas rodam na camada zero ou *ring0Mode*, são programas do próprio sistema operacional. Este modo de execução não possui as restrições do modo usuário.

Segundo [Porttalk, 2000] [8] existem duas formas de solucionar essas restrições sob Windows NT. A primeira solução é um criar/utilizar um Device Driver que rode em Ring0Mode, o que lhe permite ter acesso irrestrito às portas. A segunda é alterar o "I/O permission bitmap" do sistema, estrutura de dados que é usada pelo Windows para controlar o acesso a certas portas. Essa mesma referência assinala que, mesmo sendo mais simples, essa segunda alternativa deve ser evitada, pois abre a possibilidade de acessos indevidos que removeriam a segurança do sistema.

É interessante mencionar que mesmo que se tenha conseguido encontrar um *driver* pronto para executar a tarefa desejada, instalá-lo e configurá-lo são tarefas temidas pela maior parte do staff técnico e de suporte, que tendem a olhar com desconfiança a instalação de itens de sistema operacional que não são "certificadas pela Microsoft". Por outro lado, escrever um excitador *(Driver*) não é um trabalho fácil, nem para programadores experientes, dado o enorme número de rotinas, estruturas de dados, convenções e padrões usuais de implementação que são descritas no Windows DDK (Device Driver Kit) [9] .

X. IMPLEMENTAÇÃO DO ACESSO À PORTA PARALELA NO TUPI

Assim, é interessante poder utilizar soluções prontas ou comerciais. Nossa busca na Internet demonstrou uma preponderância dois *drivers* para porta paralela específicos para determinados equipamentos: o PortTalk e o InpOut32, ambas com funcionalidade muito semelhante.

Na implementação do TUPI escolhemos a biblioteca de domínio público INPOUT32.DLL disponível em [http: //www. logix4u.net/inpout32.htm], que foi considerada uma das que melhor dá conta destas restrições, criando um acesso funcional em todas as versões do Windows (98, NT, 2000 e XP), através de um *driver* em "*kernel mode*" (*hwinterface.sys*), embutido na DLL.

A característica proeminente de Inpout32.dll, é poder trabalhar com todas as versões do Windows sem nenhuma modificação no código do usuário ou na própria DLL. A DLL verificará a versão do sistema operacional quando as funções são chamadas, e se o sistema se operacional for WIN9X, a DLL usará o *inp ()* e funções do *outp* para leitura/escrita a partir da porta paralela.

As duas funções principais que são exportadas pela inpout32.dll são:

1) ' **Inp32** ', lê dados de um registro especifico da porta paralela.

2) ' **Out32** ', escreve dados em um registro especifico da porta paralela.

As outras funções executadas pela Inpout32.dll estão expostas no quadro 3 abaixo:

a) Funções	**Atribuições**
DllMain	É uma função chamada quando a DLL é carregada ou descarregada. Quando a DLL é

	carregada, verifica a versão do sistema operacional e carrega *hwinterface.sys* se necessário for.
Closedriver	Esta função fecha o *driver* que realizou a chamada, antes de baixar o *drvier* que foi chamado.
Opendriver	É a função que abre um driver de chamada para a função *hwinterface*.
Inst	Tem por função extrair a função ' hwinterface.sys' do *driver* da raiz do sistema de diretórios binários e criam um serviço. Esta função é chamada quando a função ' Opendriver ' não abre um *driver* de chamada válido ao serviço ' do hwinterface '.
Start	Tem por função iniciar o serviço da função hwinterface utilizando o gerenciador de APIs(Application services) do controle de serviço.
SystemVersion	Esta função verifica a versão do sistema operacional e retorna o código apropriado.

Quadro 3 – Funções executadas pela Inpout32.dll

XI. CONSIDERAÇÕES FINAIS

Na implementação do TUPI foram encontrados muitos casos que envolvem interfaces pouco conhecidas com o Windows. Na maior parte dos casos, as soluções são pouco documentadas, mas conceitualmente simples ou de implementação rápida. Apresentou-se neste artigo, os dois casos mais característicos, e que foram mais difíceis de encontrar solução e implementá-la.

Julga-se, outrossim, importante que o toolkit seja mantido em "código aberto", para que, através do estudo aprofundado de seu código, outros programadores possam obter subsídios para solucionar grande número de situações simples que ocorrem no desenvolvimento de software adaptativos. Esse código servirá de documentação complementar ao material disponível nos manuais e livros textos, provendo exemplos práticos de implementações de um grande número de situações em especial relacionadas à programação no nível de software básico ou de sistema operacional no ambiente Windows.

XII. REFERÊNCIAS

[1] Joe C. Hecht, Borland Technical Support Group – Manual técnico da Borland

[2] http://oak.cats.ohiou.edu/~piccard/mis300/osapis.htm

[3]http://www.et.utt.ro/public/Docs/ActiveX%20Programming%20Unleashed/

[4] http://zone.ni.com/devzone/conceptd.nsf/webmain/

[5]http://www.prestwood.com/community/delphi/usergroup/newsletter/1998oct.html

[6] Axelson, J. - Parallel Port Complete - Lakeview Research – 1998

[7] http://geocities.yahoo.com.br/conexaopcpc/artigos/portas_paralelas.htm

[8] www.beyondlogic.org/**porttalk**/**porttalk**.htm

[9] Microsoft, www.microsoft.com/whdc/**ddk**/win**ddk**.mspx**.**

Carlos Roberto França, Mestre em Informática pelo IM/NCE/UFRJ, tendo defendido a sua dissertação intitulada: " Especificação e Desenvolvimento de um Toolkit para Construção de Ferramentas de Acessibilidade para Deficientes Motores ", em 2005, tem 17 anos de experiência em desenvolvimento de Tecnologia Adaptativa. Atualmente é Professor Assistente do quadro da Universidade Federal da Fronteira Sul - UFFS, Campus Realeza - PR

Uma Abordagem por Técnicas Adaptativas de Segmentos de Retas em Navegação Robótica

L. C. Barros Neto and A. H. Hirakawa

***Abstract*— Com enfoque em adaptatividade, foi investigada uma arquitetura de registro de dados que possibilite a construção de representações cartesianas abstratas de ambientes desconhecidos por robôs móveis. As tecnologias adaptativas são formalismos da ciência da computação capazes de alterar seu comportamento dinamicamente, sem a interferência de agentes externos, em resposta a estímulos de entrada. Assim, este trabalho introduz uma nova estrutura de dados, não cartesiana, que permite a representação do ambiente e do movimento do robô mais próximo do movimento natural esperado, com base em segmentos de linhas retas digitalizadas adaptativos (SLRDA).**

***Keywords*— Adaptive Systems, Robotics, Automata, Computational Geometry, Pattern Recognition, Error Correction.**

I. Introdução

Classicamente, os agentes robóticos são normalmente equipados com sistemas de sensores e de atuadores, para percepção e atuação, respectivamente, no espaço físico ao qual estão integrados. Em robótica móvel, além dos sistemas de sensores e atuadores, os robôs são providos também de sistemas de deslocamento para que se movimentem pelo ambiente a fim de executar atividades de maior complexidade que as da robótica tradicional. Entende-se navegação como o processo ou atividade de planejamento de um caminho e posterior movimentação e direcionamento nesse caminho [1] para que um robô autônomo se desloque com segurança de um local para outro, sem se perder ou colidir com outros objetos, a fim de executar determinada tarefa. [2] afirma que em navegação robótica existem três problemas gerais que são localização de um objetivo ou alvo a alcançar, planejamento da rota ou caminho das partes móveis para atingir o objetivo, e controle do movimento na rota. Dentre estes três problemas, [2] argumenta que o planejamento de um caminho é uma das questões mais importantes no processo de navegação de robôs móveis autônomos, sendo uma área ativa de pesquisas desde a década de 1970 .

Uma aproximação comumente utilizada em planejamento de trajetórias é considerar o ambiente como um espaço cartesiano a fim de facilitar a discriminação de configurações requeridas. No entanto, a implementação dos algoritmos respectivos em um espaço requer transformações de coordenadas cartesianas para conjuntos em tempo real, tendo em vista que o controle de movimentos do manipulador é feito prevendo as articulações mecânicas. Em tempo real significa para [4] que a trajetória é implementada por um método com baixo custo computacional à medida que ocorre o movimento, próximo do ideal por não envolver atrasos ou escalas de tempos significativos que poderiam comprometer o controle da dinâmica do manipulador por seus atributos como tempo, posição, velocidade, aceleração. Portanto, a complexidade computacional envolvida na transformação de coordenadas é um dos aspectos importantes dos algoritmos de planejamento de trajetórias no espaço cartesiano para viabilizar aplicações em tempo real [3] [4].

Conforme [3], com base em especificações iniciais contendo um caminho intermediário e um caminho final, o planejamento de trajetórias leva em conta fatores como a posição articular das partes móveis, a velocidade e a aceleração de um manipulador robótico. Em [5] e [2] apresenta-se um resumo geral de técnicas e algoritmos, relacionadas ao controle do movimento e ao planejamento da trajetória de robôs, tais como: *i*) de esquema motor, *ii*) da velocidade do obstáculo, *iii*) método do cone de colisão, *iv*) abordagem por curvas polinomiais polares (baseadas em sensores), *v*) métodos de computação inteligentes tais como inteligência artificial (IA) ou redes neurais, *vi*) método de campo potencial artificial, e *vii*) método gráfico de visibilidade.

Independentemente de técnicas e algoritmos para controle do movimento e planejamento de trajetórias, sempre ocorrem interferências espúrias e aproximações que interferem com os modelos teóricos. [4] menciona que os trabalhos iniciais em planejamento de movimento de robôs concentraram-se principalmente em otimizações de tempos associadas a aproximações padronizadas deste movimento. As aproximações indicadas por [4] são de segmentos lineares combinados com curvas, por exemplo, arcos de forma parabólica, entre pontos de comutação selecionados da trajetória.

Comumente, o planejamento de trajetórias no espaço cartesiano costuma aproximar caminhos cartesianos por trajetórias compostas de segmentos de retas, desde que os erros de translação e rotação gerados pelo desvio entre um caminho cartesiano e as trajetórias correspondentes interpoladas possam ser ajustados para satisfazer tolerâncias especificadas. [3] restringe aproximação de caminhos cartesianos por segmentos de retas apenas para os que apresentem suavidade, ou seja, significando que um dado caminho cartesiano possa ser parametrizado por uma variável *t* de tempo e seja, pelo menos, derivável em segunda ordem com relação a *t*.

Ainda na opinião de [4], trabalhos posteriores estudaram requisitos mecânicos importantes, tais como limitações

L. C. de Barros Neto, Universidade de São Paulo (USP), São Paulo, Brasil, leonclarobr@ymail.com

A. H. Hirakawa, Universidade de São Paulo (USP), São Paulo, Brasil, arhiraka@usp.br

vibracionais de robôs em aplicações industriais, tendo em vista que uma trajetória suave, sem vibrações, é importante para reduzir o desgaste do manipulador robótico e melhorar a precisão e velocidade de rastreamento do mesmo. Limitação vibracional (e, proporcionalmente, da taxa de torque) resulta também em carga suavizada para os atuadores, o que efetivamente reduz as frequências ressonantes de excitação do manipulador, consequentemente minimizando desgastes do atuador. Além disso, tal requisito é importante para aplicações específicas, inclusive por questões de maior segurança.

A fim de se obter trajetórias mais suaves, a comutação de retas para arcos tem sido efetuada também por curvas, especialmente de terceiro grau, definidas matematicamente por dois ou mais pontos de controle (splines). Polinômios de ordem superior não são geralmente utilizados devido à sua tendência a oscilar, gerando movimentos retrógrados. Entretanto, [4] descreve um método que utiliza uma concatenação de polinômios de quinta ordem para produzir uma trajetória suave entre dois pontos do trajeto, aproximando segmentos lineares entre pontos de controle. A interligação destes pontos de controle com polinômios de quinto grau resulta em uma trajetória controlada que não oscila, considerada viável para utilização *on-line*. No que se refere às interpolações de quarto grau, o trabalho de [3] apresenta métodos para aproximar caminhos no espaço cartesiano, resultados do planejamento de trajetórias de robôs por conjunto de segmentos de retas, em que procedimentos sistemáticos são propostos por algoritmos recursivos de planejamento. A abordagem proposta é ilustrada por um exemplo numérico.

[6] examina o planejamento do caminho mais curto sob uma direção métrica euclidiana ponderada. Motivado pelo problema de encontrar as melhores rotas para veleiros, [6] mostra que o caminho mais curto sempre consiste em dois segmentos de retas, e por estas resolve-se o problema por figuras poligonais. Na prática de [6], o percurso de um veleiro é uma sequência finita de segmentos de retas, em que investiga a complexidade de várias instâncias do problema do marinheiro em um veleiro, incluindo o caso da trajetória ser uma poligonal. A seguir, a trajetória sempre consiste em segmentos de retas, e por estas resolve-se o problema por figuras poligonais. Finalmente, o percurso, composto por uma sequência finita de segmentos de retas, é compensado devido aos erros e aproximações.

Em determinadas aplicações críticas, os erros e aproximações de trajetórias por linhas retas ou por curvas com curvatura constante (arcos) têm que ser compensados nos modelo e implementações. Por exemplo, em [7] é apresentada uma abordagem baseada em trajetória de navegação para o controle lateral de veículos autônomos que percorrem uma fila e têm que seguir o caminho do carro líder por meio de um sistema de processamento de imagens sem utilizar qualquer infraestrutura adicional de comunicação ou de sistema de posicionamento global (GPS). A solução encontrada para que o veículo autônomo não se desvie da trajetória do veículo líder, devido aos erros e aproximações, foi um procedimento que envolve armazenar o histórico do posicionamento do veículo líder e a velocidade do veículo autônomo com relação ao tempo.

Nesse estudo de [7], a transformação de coordenadas de posição em um sistema de coordenadas em repouso envolveu compensar o movimento do veículo a ser controlado usando a sua velocidade e seu ângulo da direção da trajetória. Conhecendo-se a posição do veículo a ser controlado em um sistema de coordenadas, pode-se selecionar o ponto de trajetória do veículo líder que está mais próximo da frente do veículo sob controle, como entrada para o controle lateral. No entanto, controladores laterais reais têm certo atraso para garantir a estabilidade e compensar erros de medição dos dados dos sensores. Para levar tais erros em consideração, o algoritmo sempre seleciona o ponto de trajetória que exceda certa distância mais adiante, medida a partir do veículo sendo controlado [7].

No aspecto de erros e aproximações de trajetórias por digitalizações, [8] apresenta um esquema em tempo discreto de planejamento de movimento para determinar os pontos definidores de trajetória em linha reta. A motivação de [8] é que os esforços anteriores em planejamento de trajetórias causam que o braço do manipulador pode não estar no caminho desejado, mas sobre um caminho interpolado por polinômios. Adicionalmente, devido a aproximações discretas de parâmetros como velocidade, aceleração e vibrações permitidas, tais soluções de otimização por meio de polinômios envolvem cálculos errôneos, sendo também computacionalmente intensivo, o que tende a inviabilizá-las em aplicações correntes. [8] opta por uma solução algorítmica formulando o problema de planejamento de trajetória como de maximização da distância cartesiana entre dois instantes no tempo com a restrição de que componham um determinado caminho em linha reta sujeito a limitações em torque e de suavidade da trajetória.

[5] pesquisa algoritmos de planejamento de trajetórias pelo método de campo potencial e método de Monte Carlo, para evitar obstáculos na navegação de robô móvel em um ambiente dinâmico do tipo industrial. O algoritmo de planejamento de trajetória é dividido em dois sub-módulos: o planejamento de trajetória global que usa grafos de visibilidade e o planejamento de caminho local pelo método de campo potencial para evitar os obstáculos. O processamento de imagens é usado para obter informações sobre o ambiente de trabalho a partir de câmera com acesso ao ambiente.

Tendo em vista que os erros no processo de digitalização envolvem também detalhes de calibração do sensor, por exemplo, uma câmera montada no robô, [9] argumenta que o planejamento no espaço euclidiano tende em resultar em trajetórias inadequadas no espaço da imagem e vice-versa. Estas dificuldades são devidas à transformação em perspectiva da câmera; à perda de uma dimensão, devido à projeção sobre o plano da imagem; e ao fato de que, em termos práticos, é considerada viável apenas uma aproximação grosseira de parâmetros da câmera. Assim, [9] propõe um esquema de planejamento de trajetórias por imagens tal que o comportamento em linha reta é assegurado tanto no espaço da

imagem quanto no espaço efetivo do mundo em que se movimenta o robô. Esse esquema permite trajetórias compactas em linhas retas, ou seja, para o centro óptico da câmera relativamente a um ponto escolhido arbitrariamente no plano da imagem.

Focado em técnicas sintáticas para compensar erros de calibração de sensores, o trabalho de [10] descreve uma abordagem de segmentação de imagens usando linhas retas, na verdade aproximações de arcos curvos por linhas retas codificadas por cadeias de símbolos ou *chain code*, com a finalidade de reduzir significativamente o número de primitivas, bem como diminuir o esforço computacional e a taxa de erros em segmentação. Os algoritmos propostos são parte de um sistema de computação de profundidade a partir de sequências de imagens monoculares tomadas de uma sequência de pontos de vista diferentes por uma câmera montada em um manipulador de robô. Tendo em vista que cada grupo de duas imagens consecutivas é considerado como uma imagem estéreo, o trabalho se propõe a reduzir os problemas de calibração imprecisa e intensidade variável de sinal recebida pela câmera visando fundamentar os trabalhos futuros em planejamento de trajetórias.

Sob o aspecto adaptativo para solução do problema de modelagem em navegação, [11] descreve evoluções de uma abordagem que parte da constatação de que um conjunto inicial de parâmetros de projeto ou de controle de um sistema muito frequentemente torna possível inferir o comportamento do sistema sob um cenário envolvendo conjunto de parâmetros diversos daquele inicial. Isto sugere fortemente o potencial para "aprendizado" sobre o comportamento global de um sistema de navegação, simplesmente partindo de um caminho considerado como amostra. Por sua vez, tal fato implica o potencial para ajustar vários parâmetros críticos, como um refinamento de modo a ajustar continuamente o desempenho do robô por reações a alterações e mudanças no ambiente. Assim, [11] é um estudo centrado em autômatos estocásticos para que o comportamento de um robô em navegação possa ser inferido a partir de um caminho único como amostra.

[12] enfatiza determinadas habilidades dos robôs autônomos tal como a capacidade de se adaptarem aos ambientes de operação, aprender conforme adquirem mais experiência e realizar escolhas frente às mudanças ou novas situações desses ambientes.

Concluindo, de modo geral a baixa fidelidade de representação utilizando métodos tradicionais com base em mapas ambientais cartesianos, resulta em pouco conhecimento armazenado sobre o ambiente ao redor do robô, conhecimento este comumente obtido por sensores de processamento de imagens associado a procedimentos de detecção de linhas retas e arcos, tal como em [13].

A hipótese inicial deste trabalho é que o formalismo de SLRDA apresentado em [14] e [15] seja um modelo mais preciso de representação do movimento e do ambiente em que está inserido o robô.

A. Organização do texto

A seção I analisa os trabalhos principais que embasam conceitualmente esta pesquisa. A seção II apresenta conceitos necessários para o entendimento deste trabalho, tais como segmentos de linhas retas digitalizadas (SLRD), segmentos de linhas retas digitalizadas adaptativos (SLRDA) e autômato finito adaptativo (AFA). A seção III apresenta o enunciado do problema estudado por este trabalho, os desafios e os meios e métodos aplicados para superar os desafios. A seção IV apresenta a síntese do método proposto, com detalhes técnicos sobre o comentado no item anterior. A seção V especifica a implementação de um banco de autômatos que implementam SLRDA. A seção VI apresenta justificativas para o método de comparação nos experimentos e o posicionamento da pesquisa com relação ao estado da arte. A seção VII descreve os experimentos efetuados. A seção VIII relaciona perspectivas de trabalhos futuros. A seção IX apresenta as considerações finais deste relato e contribuições.

II. FUNDAMENTOS

Este tópico apresenta conceitos básicos necessários ao entendimento desta pesquisa por uma síntese de [15].

A. Conceitos básicos sobre SLRD

Em [16] o *chain code* foi apresentado como um descritor de contornos, de apenas um pixel de espessura, e um modelo definidor de retas digitais foi também conjecturado por Freeman. Nesse modelo, dado um ponto digital, as vizinhanças principais e imediatas desse ponto estão mostradas na Fig. 1, que mostra o relacionamento dos símbolos do chain code com a vizinhança-4 e vizinhança-8.

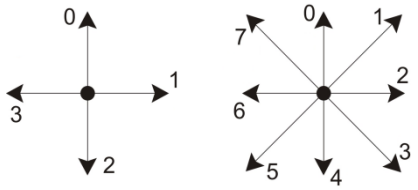


Figura 1. À esquerda estão os símbolos de 0 a 3 do *chain code* para vizinhança-4. À direita, os símbolos de 0 a 7 do *chain code* para vizinhança-8 (adaptado de [21]).

Um arco digital é reto se for o resultado da digitalização de um segmento em linha reta euclidiano. No modelo de Freeman, as cadeias ideais que representam linhas retas obedecem a três propriedades, em uma codificação utilizando números de 0 até 7, em vizinhança-8:

--Prop1: No máximo dois tipos de símbolos, representando direções distintas no código do *chain*, podem estar presentes, e estes são números consecutivos correspondentes do *chain*, módulo oito;

--Prop2: Um dos dois símbolos sempre ocorre isoladamente, solitário;

--Prop3: As ocorrências sucessivas do símbolo isolado são tão uniformemente espaçadas quanto possível entre códigos do outro valor, que ocorre em grupos ou corridas (runs).

O significado de Prop1 a Prop3 é representar a linha por uma sequência de vetores com inclinações múltiplas de 45º e cujos comprimentos são unitários (se horizontal ou vertical) ou $\sqrt{2}$ (se diagonal).

A noção de linha reta está associada à propriedade da

corda, pois [17] demonstrou que a condição necessária e suficiente para que um arco digital qualquer seja um SLRD ideal é atender à propriedade da corda.

Definição 2: A Propriedade da Corda: Diz-se que um arco digital C, representando "objetos sólidos" delgados em uma imagem digitalizada, apresenta a propriedade da corda se, para cada dois pontos digitais c e d pertencentes a C, e para cada ponto p = (x, y) em $\overline{cd}$, existe um ponto e = (h, k) pertencente a C tal que $\mathbf{max}\{|\boldsymbol{x}-\boldsymbol{h}|, |\boldsymbol{y}-\boldsymbol{h}|\} < \mathbf{1}$, onde $\overline{\mathbf{cd}}$ é o segmento de reta entre c e d [17].

A Definição 2 implicou na demonstração da existência de uma estrutura hierárquica composta de números consecutivos correspondentes às corridas dos símbolos especificados por Prop1 e Prop2. Essa estrutura de números consecutivos é expressa por uma propriedade adicional Prop4: Quanto à direção referente ao símbolo que ocorre em grupos (não isolado), as corridas correspondentes podem ocorrer com apenas dois valores, os quais diferem de uma unidade (por exemplo, P e P+1).

Pela propriedade da corda, define-se um SLRD ideal como a seguir.

Definição 3: Linha Reta Digitalizada Ideal (SLRD ideal): Uma linha reta digitalizada ideal é aquela que atende à propriedade da corda, na vizinhança respectiva.

Também os conceitos de unidade de segmento e ângulo de orientação são importantes para o entendimento das propriedades locais e globais dos SLRD.

Definição 4: Unidade de Segmento Digital em Linha Reta (abreviado USLR) é o menor segmento possível em que um SLRD pode ser subdividido a fim de manter o correspondente ângulo de orientação principal [20].

Definição 5: Ângulo de Orientação Principal: Entende-se como ângulo de orientação principal do SLRD, denominado θ_S, como aquele ângulo que se destaca relativamente ao eixo *x*, por algum critério, dentre a distribuição de ângulos das USLR individuais do segmento.

O processo de digitalização de uma linha reta euclidiana específica resulta que o correspondente SLRD não atenda à propriedade da corda, resultando em muitos segmentos curtos na vizinhança de um pixel, devido às influências espúrias. Portanto, é necessário algum tipo de medida, denominada métrica, apropriada para avaliar se dois SLRD pertencem a uma única estrutura linear [18] [22]. Por essa métrica, pode-se definir uma função de vizinhança escolhida de modo a expressar como agrupar SLRD aproximados.

Contudo, "aproximado" é entendido no tocante a linhas retas visualmente corretas dentro de uma tolerância, não necessariamente no tocante à propriedade da corda. Portanto, de agora em diante, a menos que de outra maneira seja especificado, a nomenclatura deste texto não faz nenhuma distinção entre um SLRD ideal e as linhas "quase" retas nas proximidades da primeira, considerando que tanto SLRD ideais quanto SLRD aproximados são reconhecido pelo dispositivo adaptativo deste trabalho.

B. Codificação

Nesta pesquisa, os SLRD não são codificados por meio de cadeias de dígitos, bastando considerar $\boldsymbol{a} = \mathbf{0}$ e $\boldsymbol{b} = \mathbf{1}$ no segmento exemplificado pela Fig. 2 a fim de atender Prop1. Uma cadeia é uma sequência de zero ou mais símbolos pertencentes ao alfabeto $\boldsymbol{\Sigma}$. Esses símbolos são também denominados como primitivas, *tokens*, elementos do *chain code* ou simplesmente estímulos. O conjunto de todas as cadeias possíveis com $\boldsymbol{\Sigma}$ é denotado por $\boldsymbol{\Sigma}^*$. O comprimento de uma cadeia qualquer *S* é denotado por $|\boldsymbol{S}|$. A cadeia vazia, de comprimento zero, é representada por $\boldsymbol{\varepsilon}$. O *i*-ésimo símbolo de uma cadeia $S = s_1 \ldots s_n$ é representado por s_i.

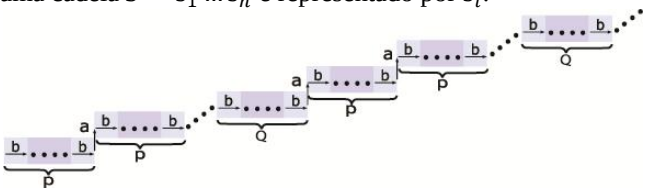


Figura 2. Exemplo de SLRD genérico na faixa $\mathbf{0} < \theta_{\mathbf{u}} < \boldsymbol{\pi}/\mathbf{4}$ ($\boldsymbol{\theta}_{\mathbf{u}}$ é o ângulo de inclinação relativo ao eixo *x* de USLR), com corridas de P e Q símbolos *b*, tão uniformemente espaçados quanto possível entre *tokens a*, que ocorrem isolados (adaptado de [20]).

Caso nada em contrário seja especificado, sem qualquer redução em generalidade, a vizinhança-4 é utilizada neste trabalho como padrão, com os símbolos da propriedade Prop1 consecutivos, módulo quatro. Mais precisamente, os símbolos que compõem as cadeias pertencem a $\boldsymbol{\Sigma} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$ tal que, para atender Prop1, basta considerar módulo 4, juntamente com ***a=0, b=1, c=2, d=3***, relativamente à vizinhança-4 da Fig. 1.

Uma cadeia qualquer $\boldsymbol{S} = \boldsymbol{s_1} \ldots \boldsymbol{s_n}$ pode também ser representada no formato da Expressão 1.

$$\boldsymbol{S}: \boldsymbol{s_i}; \boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{n}. \quad (1)$$

Caso todos os elementos de S da Expressão 1 sejam idênticos, $\boldsymbol{S} = \boldsymbol{s_1} = \boldsymbol{s_2} = \cdots = \boldsymbol{s_{n-1}} = \boldsymbol{s_n} = \boldsymbol{s}$, uma representação compacta é

$$\boldsymbol{S} = \boldsymbol{s^n}. \quad (2)$$

Por exemplo, na Fig. 2, as cadeias das USLR são do tipo ab^n com *n* inteiro, *n*=0, 1, 2, 3.....; resultando a expressão para o ângulo $\Theta_u = \arctan(1/n)$.

C. O Autômato Finito Adaptativo (AFA)

Definição 1: O Autômato Finito Adaptativo (AFA) é um dispositivo auto-modificável, com poder computacional equivalente à máquina de Turing, da forma **AFA = (ND0, AM)** onde ND0 é a camada subjacente, e AM é a camada adaptativa associada, formalizada nos mesmos moldes que ND0. O AFA é caracterizado por ter o autômato finito (AF) como formalismo da camada subjacente. Bibliografia e evolução do AFA e adaptatividade estão em [32].

A camada adaptativa da Definição 1 integra o conjunto de ações adaptativas, responsáveis por alterar a camada subjacente dinamicamente em resposta aos estímulos. As ações adaptativas podem ser interpretadas como chamadas de função, a função adaptativa (FAD), em que esta pode ser paramétrica.

A Fig. 3 mostra uma representação gráfica estática genérica de transição do AFA do tipo: ***(x, i) : R→y : S*** onde ***x*** é o estado atual do autômato antes da transição entre estados; ***y*** é o estado de destino do autômato após a transição; *i* é o estímulo

de entrada; R e S são as FAD dos tipos anterior e posterior respectivamente por serem ativadas "antes" e "depois" da transição do estado ***x*** para o estado ***y*.**

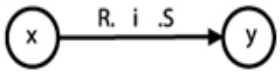


Figura 3. Uma transição adaptativa genérica, onde R e S são FAD opcionais do tipo anterior e posterior, respectivamente.

Por sua vez, as funções adaptativas são formadas por conjuntos de ações adaptativas elementares ou primitivas a serem aplicadas para definir o processo adaptativo do autômato. Pela Tab. I, tais ações adaptativas elementares podem ser de três modalidades, de acordo com o prefixo de uma transição especificada entre colchetes, denominada gabarito. Por esse gabarito, as transições em uso devem ser testadas por um processo de busca da seguinte maneira:

1) Ações de consulta, denotadas pelo prefixo "?": efetuam uma busca por regras que casem com o gabarito especificado, sem alterar o conjunto de regras;
2) Ações de remoção, denotadas pelo prefixo "-": removem do conjunto as regras que casam com o gabarito especificado;
3) Ações de inserção, denotadas pelo prefixo "+": inserem o gabarito especificado no conjunto de regras.

TABELA I
FORMATO DAS AÇÕES ADAPTATIVAS ELEMENTARES.

MODADALIDADE	PREFIXO	SIMBOLOGIA
CONSULTA	?	?[*(x, i) : R→y :S*]
REMOÇÃO	-	-[*(x, i) : R→y :S*]
INSERÇÃO	+	+[*(x, i) : R→y: S*]

Observe-se que, havendo mais de uma ação adaptativa elementar a ser executada, independentemente da ordem em que foram declaradas, têm precedência as consultas. Em seguida são efetuadas as remoções, e posteriormente as inserções. Possíveis transições em vazio são sempre executadas por último, após as inserções. Ações elementares que referenciam elementos indefinidos não são consideradas.

C1. Formato das Funções Adaptativas (FAD)

Opcionalmente, as FAD podem ser paramétricas. Apesar de opcionais, se parâmetros forem especificados, terão de ser fornecidos para ativar a correspondente FAD. Em outras palavras, os parâmetros formais de FAD são variáveis especiais, utilizados para que valores reais (argumentos) sejam atribuídos, sempre que determinada FAD seja ativada. Assim, aos parâmetros da FAD são atribuídos os valores recebidos como argumentos antes de ativar a FAD, permanecendo inalterados durante toda a sua execução.

No caso geral, além dos parâmetros, uma FAD pode ser especificada por variáveis e geradores, também opcionais, com os significados seguintes:

Variáveis: identificadores que recebem um valor de acordo com o resultado de ações adaptativas elementares de consulta ou exclusão. As variáveis são utilizadas para representar entidades existentes em transições do AFA tais como estados ou estímulos.

Geradores: identificadores que recebem um valor único (isto é, nunca antes utilizado) no início da ação adaptativa e permanecem com este valor até o final da ação. São espécies de variável que contém a identificação do próximo estado a ser incluído na topologia do AFA.

Graficamente, uma determinada FAD R é representada por R**.** (R seguido de ponto**)**, caso seja do tipo anterior. Similarmente, uma FAD S é representada por **.**S (ponto seguido de S**)**, caso seja do tipo posterior.

D. O Segmento Digitalizado Adaptativo

Em um primeiro aspecto, SLRDA são implementados nesta pesquisa por meio de AFA a fim representar as diferentes instâncias do modelo ideal de SLRD afetada pelos desvios em ângulo. Isso requer que o SLRDA atue numa determinada faixa de ângulos, definindo uma vizinhança, para comprimentos de arcos digitalizados teoricamente quaisquer. Tal vizinhança associa um conjunto de arcos a um SLRDA correspondente, que reconhece esse conjunto.

Em um segundo aspecto, SLRDA são implementados também relacionados a uma vizinhança adaptativa, tal que as vizinhanças dos SLRDA são alteradas adaptativamente em função dos respectivos comprimentos dos arcos.

A Fig. 4 mostra um SLRDA, tal que com o primeiro símbolo b, a FAD B é ativada, alterando-se a configuração do AFA para a Fig. 5, passando o AFA a consumir as USLR subsequentes, sendo que cada USLR da cadeia de entrada ativa a FAD RB. Ou seja, com o consumo da $USLS_1$, recebendo-se o primeiro símbolo *b*, a configuração do AFA altera-se para a Fig. 5. Na Fig. 5, da mesma maneira que na Fig. 4, com o acionamento da FAD RA, uma transição em vazio é removida. A FAD RB insere novamente as transições em vazio, preparando o AFA para o recebimento de outra USLR.

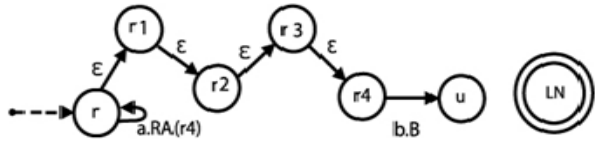


Figura 4. Configuração inicial de AFA que implementa SLRDA a fim representar as diferentes instâncias do modelo ideal de SLRD afetada pelos desvios em ângulo. A cada ativação de RA, uma das transições em vazio é removida, reconhecendo a primeira unidade de segmento: $USLR_1$.

Ainda na Fig. 5, expecificando-se a quantidade de estado do *loop* (exemplificado pelos estados u, u_1, u_2, u_3, u_4, u_5) e a quantidade de transições em vazio esse autômato foi implementado para que reconheça cadeias de entrada com USLR do tipo $USLS_i = \{a^n b: 3 \leq n \leq 5\}$, com *i* inteiro, *i*= 0, 1, 2, 3....

Observe-se que transições em vazio causam não determinismo, podendo-se implementar um autômato determinístico equivalente sem transições em vazio, alterando-se o AFA das Fig. 4 e 5 utilizando-se transições com marcadores tal como Δ sendo que $\Delta \notin \Sigma$.

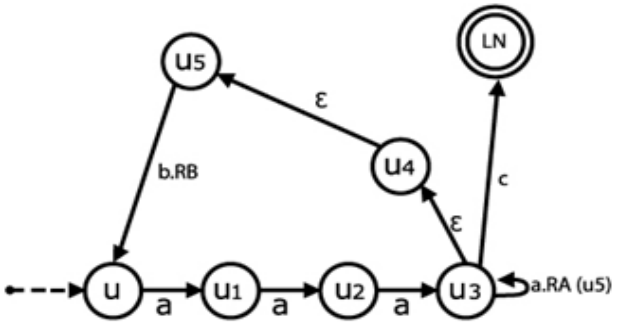


Figura 5. Configuração de AFA após ativar a FAD B da Fig. 4. A faixa de ângulos reconhecida por este SLRDA é ajustada pela quantidade de transições em vazio e transições que consomem o símbolo *a* que compõem o *loop* compostos, como exemplo, pelos estados u, u_1, u_2, u_3, u_4, u_5.

III. Enunciado do Problema

O presente trabalho investiga um novo modelo de representação computacional do ambiente por um robô, tarefa considerada fundamental para a robótica móvel conforme trabalhos relevantes da área citados no tópico I. Dentre os trabalhos comentados no tópico I, [2] relata vários algoritmos de navegação em um ambiente envolvendo, normalmente, leitura do mapa representativo do ambiente ou espaço de trabalho e, subsequentemente, tentar criar caminhos livres para que o robô percorra a área de trabalho sem colidir com os objetos e obstáculos. Existem métodos tradicionais de representar o ambiente para navegação. Por tais métodos, os ambientes tendem a não serem representados de forma ideal, pois normalmente geram um mapa abstrato representativo correspondendo a uma grade fixada *a priori* por coordenadas cartesianas. O assunto se relaciona com os temas clássicos da geometria computacional e do planejamento de movimento e navegação [6], tendo em vista que um percurso a ser percorrido por um agente robótico é composto por pontos interligando um local de partida com uma localização final, compondo uma rota de referência. Considerando as diversas aplicações existentes na atualidade, a linha reta, interligando o ponto de partida e o ponto final, pode ser considerada como a melhor rota, a título de referência dos algoritmos de navegação a serem implementados por este trabalho.

No contexto do planejamento de rotas em navegação, o agente robótico deve detectar os elementos em seu ambiente e o significado de cada situação tática para definir rotas alternativas. Ou seja, o agente robótico utiliza restrições de espaço, localização do alvo ou ponto no espaço a alcançar pela navegação e obstáculos existentes a serem evitados durante o movimento para definir alterações da rota de referência. Na reavaliação dinâmica do ambiente, o agente robótico tem que se adaptar a uma nova rota reiteradamente, sempre que ocorra alteração do ambiente ou mudança de situação tática.

A. Desafios científicos e tecnológicos

Os desafios científicos e tecnológicos desta pesquisa podem ser relacionados às áreas envolvidas com o trabalho proposto, principalmente reconhecimento de padrões, geometria computacional ou geometria digital e compiladores, comentados a seguir:

A1. Desafios científicos

Os desafios científicos são devido ao ineditismo da proposta que é o uso do formalismo de SLRDA para representar ambientes e o movimento de robôs em navegação, como alternativa ao modelo cartesiano. Destaque-se que, apesar de existir na literatura uma infinidade de trabalhos sobre arcos e retas digitalizados (SLRD), que tiveram início nas proximidades da década de 1970, constata-se que os resultados obtidos com os métodos sintáticos em áreas diversas tendem a apresentar restrições quando relacionados a cenários sujeitos a influências espúrias (erros e ruído), tais como em navegação robótica, restringindo os trabalhos, por exemplo, às linguagens regulares [19] [23].

Em particular, ao longo dos últimos anos, os estudos em planejamento de trajetórias ou caminhos têm aumentado sensivelmente, pois robôs móveis são hoje utilizados em variadas aplicações envolvendo operar em presença de incerteza em vários domínios tais como atuação nas áreas industriais e residenciais [5], as operações de resgate em situações de catástrofe, exploração marítima e planetária [24], defesa e a execução de tarefas agrícolas [25][1].

Essas modernas aplicações implicam em utilizar os avanços tecnológicos em sistemas digitais. Considerando que as modelagens de sistemas por equações de diferenças e diferenciais há muito tempo têm fornecido um quadro rico para a análise, projeto e controle dos processos, a característica orientada a eventos da tecnologia digital conduz a um comportamento dinâmico em que a abordagem centrada na modelagem de sistemas por equações de diferenças e diferenciais é simplesmente inadequado [11], se não considerar os erros envolvidos entre as modelos contínuos e discretos e as perturbações espúrias ou ruído inerentes à aplicação.

A2. Desafios tecnológicos

Há 3 vertentes principais de desafios tecnológicos para a navegação robótica no contexto apresentado. A primeira é de natureza de obtenção de informações do mundo real: definir a trajetória, identificar os pontos críticos na trajetória e ângulo de curvaturas locais, para finalmente efetuar ainda uma análise global da trajetória para detectar o comprimento de subtrajetórias entre pontos críticos, descartando ou não ângulos de curvatura abaixo de determinado valor. A outra vertente é a dos desafios envolvidos no modelo para representação do ambiente: alterar o modelo cartesiano com granularidade fixa especificada a priori, para uma representação dinâmica do ambiente com granularidade que se altera adaptativamente de acordo com os estímulos.

A terceira é a dos desafios envolvidos na atualização dinâmica das informações e das representações, face aos estímulos e obstáculos, levando em conta o movimento do robô pelo ambiente, em que o processo de digitalização é afetado por erros na fonte de dados, de erros no processamento anterior à transformação para o formato de cadeia, ou até mesmo erros na transformação final para cadeia, e por ruído em todo o processamento. Outras fontes de ruído e erros são causadas por efeitos como qualidade da imagem não ideal, alterações do foco da câmera afetando as dimensões e ângulos

de segmentos, pré-processamento imperfeito, distorções afetando as escalas, aproximações em modelos e equações. Como resultado, a navegação por SLRDA impõe formalismos de gramáticas irrestritas [15], tipo 0, integradas a cenários em robótica sujeitos a distorções e ruído.

Esclarecendo melhor sobre o poder computacional requerido por este trabalho, os SLRD implicam em linguagens tipo 1, entretanto, devido aos comprimentos de segmentos representado pelos SLRDA, que podem ser em diferentes escalas, tendendo teoricamente até infinito, esta pesquisa impõe formalismos com poder computacional equivalentes à da máquina de Turing (detalhes adicionais sobre esse aspecto estão em [14]).

B. Meios e métodos para superar os desafios.

Os meios e métodos para superar os desafios são os seguintes:

B.1 Superação dos desafios científicos

A superação dos desafios científicos se efetua aplicando o formalismo adaptativo apresentado por [14] para o registro e tratamento das informações adquiridas pelo conjunto de sensores de um robô móvel. A arquitetura apresentada em [26] é aperfeiçoada por esta pesquisa para incluir os registros de dados de erros e ruído em rotas de qualquer ângulo relativamente a uma direção de referência ou azimute, conforme esta proposta.

O formalismo de [14] é aplicado na superação dos desafios científicos principalmente nos seguintes pontos: *i*) Utilizar o formalismo de SLRDA para representação não cartesiana do ambiente e do movimento do robô, que considere a curvatura e comprimento dos arcos de navegação; *ii*) Aplicar a adaptatividade para representar os erros, as tolerâncias e os parâmetros envolvidos; *iii*) Utilizar uma vizinhança adaptativa a fim de incorporar dinamicamente ao modelo as tolerâncias e erros nos parâmetros, tais como ângulo e comprimento, otimizando a capacidade de representação computacional da informação; *iv*) Utilizar a facilidade de escalas adaptáveis para se obter trajetórias de navegação que podem ser em diferentes escalas, com comprimentos relativos tendendo teoricamente até infinito.

A escala adaptável possibilita alterar a representação geométrica discreta do ambiente e do movimento do robô, alterando os erros de quantização adaptativamente em função dos comprimentos e curvaturas dos arcos de navegação. Com esse procedimento, detalhes detectados para um determinado espaçamento da grade representativa do ambiente, não serão detectados em outros, e vice-versa. De acordo com a precisão requerida, a grade digital mencionada deve ser suficientemente fina para preservar o raio de curvatura mínimo dos arcos das trajetórias.

B.2 Superação dos desafios tecnológicos

Com relação aos agentes robóticos propriamente ditos, o presente trabalho emprega AFA para definição da trajetória de robôs, nos moldes de [26], o qual relaciona várias referências sobre pesquisas que empregam máquinas de estados e autômatos, isoladamente ou conjuntamente a outras funcionalidades da robótica móvel. Em outras palavras, determinados AFA neste trabalho estarão associados a agentes robóticos em navegação, tendo em vista que [14] demonstrou a viabilidade em aplicar o formalismo dos AFA em geometria digital para modelagem de SLRD.

De conformidade com esse processo de associar agentes robóticos a AFA, a superação dos demais desafios tecnológicos se realiza por simulações, correspondentes a provas de conceito objetivando a validação, teste e análise dos resultados obtidos com a proposta. A escolha dos ambientes simulados e rotas de navegação de referência utilizados nas provas de conceito se baseiam na representatividade em função de características específicas visando abranger grande parte das situações possíveis.

IV. Síntese do método proposto

Este tópico apresenta detalhes técnicos sobre o comentado no item anterior.

Quanto à modelagem de software, este é representado como um sistema de máquinas de estados para cobrir o espaço ambiental do robô, esquema adaptado de [36]. Essa modelagem é conveniente também devido à necessidade de disponibilizar um banco de autômatos (composto de AFA e AF) para cobrir a faixa de ângulos, variável de 0 a 360 graus, para as trajetórias possíveis de robôs. A quantidade de SLRDA implementados em cada setor depende de escolha considerando critérios tais como precisão requerida nas trajetórias.

Observe-se que os autômatos permanecem invariáveis com alterações adaptativas da grade, pois esta, em última análise, envolve alterar adaptativamente o comprimento das primitivas da Fig. 1, relativamente às dimensões efetivas do ambiente em que está inserido o robô.

A Tabela II mostra 8 setores que cobrem $2\boldsymbol{\pi}$ radianos (rad), em que cada setor corresponde a SLRDA, cuja expressão de USLR está também indicada na tabela, com *n* inteiro *n*=0, 1, 2.....

Tabela II
Faixas de ângulos Θ_U relativo ao eixo *x* de SLRDA

Setor	Expressão de USLR	faixa de ângulo (rad)
1	ab^n	$0 < \Theta_u < \pi/4$
2	ba^n	$\pi/4 < \Theta_u < \pi/2$
3	da^n	$\pi/2 < \Theta_u < 3\pi/4$
4	ad^n	$3\pi/4 < \Theta_u < \pi$
5	cd^n	$\pi < \Theta_u < 5\pi/4$
6	dc^n	$5\pi/4 < \Theta_u < 3\pi/2$
7	bc^n	$3\pi/2 < \Theta_u < 7\pi/4$
8	cb^n	$7\pi/4 < \Theta_u < 2\pi$

Para os ângulos de USLR $(0 + \frac{k\pi}{2})$ radianos, com k inteiro k=0, 1, 2.....; as cadeias representando segmentos de retas são do tipo *aaaaaaa...; bbbbb....; ccccc...; ddddd.....;* ou seja, são reconhecíveis por 4 AF.

Para os ângulos $(\frac{\pi}{4} + \frac{k\pi}{2})$ radianos, com k inteiro k=0, 1, 2.....; as cadeias representando segmentos de retas são do tipo *abababababab*.....; *bcbcbcbcbcbcb*.....; *dcdcdcdcdcdcd*.....;

cbcbcbcbcbcbcbcbc....; ou seja, são também reconhecíveis por 4 AF.

Complementando o comentado em II.B, as faixas de ângulo da Tabela 2, são obtidas pelas cadeias das USLR. Por exemplo, no setor 1, as USLR são do tipo ab^n com *n* inteiro, *n*=0, 1, 2, 3.....; resultando que o ângulo $\Theta_u = \text{arctang}(1/n)$. Para o setor 2, USLR são do tipo ba^n implicando $\Theta_U = \text{arctang}(n)$. E assim por diante para todos os setores.

Sintetizando, as estruturas de dados propostas nesta fase de pesquisa envolvem o seguinte: *i*) Utilização de SLRDA para representar as várias opções de trajetórias variáveis em ângulo e comprimento; *ii*) Alterar adaptativamente a resolução da grade (denominada grade adaptativa), ou seja, dos comprimentos das primitivas da Fig.1, relativamente às dimensões efetivas do ambiente em que está inserido o robô; *iii*) Rotação adaptativa das primitivas da Fig. 1, relativamente ao ambiente em que está inserido o robô, fim de evitar as descontinuidades da função arco tangente em $\frac{\pi}{2}$ e $\frac{3\pi}{2}$ radianos; *iv*) Os autômatos permanecem inalterados.

V. Implementação

Para acionar o banco de autômatos, foi implementado ambiente para simular cenários de navegação composto de duas camadas adaptativas:

- O Sistema Gerente é uma camada adaptativa hierarquicamente superior, que tem a função de monitorar e coordenar SLRDA, sendo que a quantidade de SLRDA altera-se conforme os estímulos pelo gerenciamento dessa camada. O Sistema Gerente recebe as informações dos SLRDA pelos correspondentes estados finais, bem como pela informação se a cadeia de entrada foi aceita ou rejeitada;
- O Sistema Escravo é a camada adaptativa comandada pelo Sistema Gerente ou camada adaptativa superior, composta por um ou mais SLRDA.

No caso mais geral, a camada adaptativa superior comanda os SLRDA por meio de cadeias do tipo $\boldsymbol{W} = \Psi\, \boldsymbol{C}$ indicadas na Fig. 6. A Fig. 6 esquematiza um AFA, associado a agentes robóticos por este trabalho, ou seja, reitere-se que $\boldsymbol{W} = \Psi\, \boldsymbol{C}$ é a cadeia de entrada para um SLRDA.

Na Fig. 6, Ψ é uma cadeia recebida pelo AFA para que este ajuste uma faixa de tolerância relacionada à cadeia *C*. Ψ corresponde a um fator para compensar imprecisões no processo de digitalização. O assunto de imprecisões e erros em digitalização é aprofundado em [14], o qual relaciona vários trabalhos sobre essa questão.

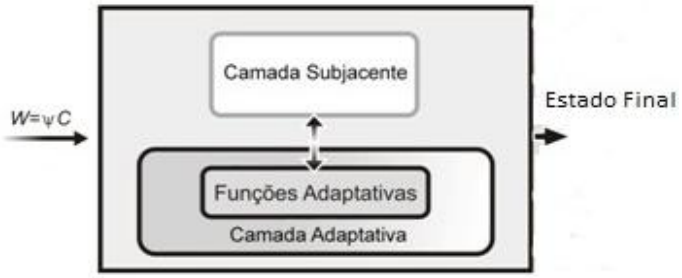


Figura 6. Representação de segmento digitalizado adaptativo, associado a agente robótico por esta pesquisa.

A. Implementação de autômatos

Para implementação de autômatos, utilizam-se padrões de projeto, mantendo-se a topologia de autômatos de um setor da Tabela II como padrão, alterando-se os símbolos convenientemente, obtendo os SLRDA dos demais setores.

Um dos padrões de projeto é o SLRDA do tópico II.D implementado para reconhecimento de SLRD submetidos a erros em inclinação (ângulo relativo ao eixo *x*), exemplificando para cadeias de entrada com USLR do tipo $USLS_i = \{a^n b: 3 \leq n \leq 5\}$, com *i* inteiro, *i*= 0, 1, 2, 3....

Uma sequência básica para implementação é a seguinte. Inspeciona-se a Tabela II, que mostra os intervalos em ângulo dos SLRDA e expressões respectivas das USLR que auxiliam no procedimento de implementar AFA a partir do padrão de projeto (AFA das Fig. 4 e 5). A seguir, definem-se os símbolos convenientemente do SLRDA do outro setor, seguindo as propriedades do modelo de Freeman, ajustando-se a correspondente faixa de ângulos a ser reconhecida. Tal ajuste se faz pela quantidade de transições em vazio e transições que consomem o símbolo que se repete, as quais compõem o loop exemplificado pelos estados u, u_1, u_2, u_3, u_4, u_5 das Fig. 4 e 5.

Outros exemplos de padrões de projeto são os seguintes, utilizados para implementação dos AF descritos na tópico IV-Síntese. *i*) O AF que reconhece cadeias do tipo tipo *aaaaaaa*... da Fig. 7; *ii*) O AF que reconhece cadeias do tipo tipo *abababababab*..... da Fig. 8.

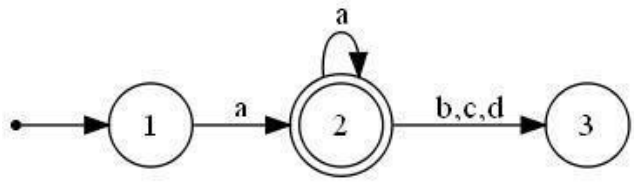


Figura 7 AF que reconhece cadeias do tipo tipo *aaaaaaa*...

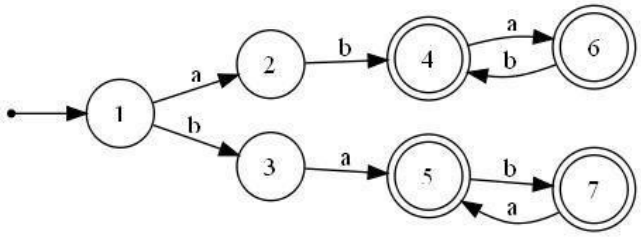


Figura 8 AF que reconhece cadeias do tipo *abababababab*.....

B. O banco de autômatos

Cada setor da Tab. II contribui com quatro AFA, denominados A, B, C e D. Por exemplo, para o primeiro quadrante composto pelo setor 1 e setor 2 da Tab. II, os seguintes SLRDA foram selecionados:
Setor 1 (ab^n): $2 \leq n \leq 4$ (Autômato A); $5 \leq n \leq 7$ (Autômato B); $8 \leq n \leq 12$ (Autômato C); $13 \leq n \leq 20$ (Autômato D).
Setor 2 (ba^n): $2 \leq n \leq 4$ (Autômato A); $5 \leq n \leq 7$ (Autômato B); $8 \leq n \leq 12$ (Autômato C); $13 \leq n \leq 20$ (Autômato D).

Para os demais setores a topologia dos AFA é a mesma dos setores 1 e 2. Resulta em 32 AFA e 8 AF, totalizando um banco de 40 autômatos.

VI. JUSTIFICATIVA PARA OS EXPERIMENTOS

Este tópico apresenta justificativas para o método de comparação nos experimentos, comparando-se com trabalhos selecionados, posicionando esta pesquisa com relação ao estado da arte, descrita a seguir.

A. Posicionamento com relação ao estado da arte.

Com referência ao comentado no item I, destaque-se que as trajetórias de robôs requerem ângulos variáveis de trajetórias (denominada por [34] como "navegação em qualquer ângulo"). Assim, [34] afirma que as trajetórias relativas à grade cartesiana tendem a não serem os caminhos mais curtos (isto é, os caminhos mais curtos no terreno), pois se restringem artificialmente a ângulos múltiplos de 45 graus (para vizinhança-8) e múltiplos de 90 graus (para vizinhança-4). Em outro trabalho com o mesmo propósito de ângulos variáveis em trajetórias, [36] formaliza um espaço de busca de rotas alternativas por meio de algoritmo de busca em grafos.

Quanto às técnicas adaptativas utilizando autômatos, certamente o estado da arte é representado pelo trabalho [26] que se restringe a trajetória em apenas quatro direções (norte, sul, leste e oeste), suficientes para o propósito de mapeamento de ambientes. A motivação de [26] em utilizar a adaptatividade foi devido à elevada quantidade de memória para registro de informações para mapeamento de ambientes, implicando em alto poder computacional de seus sistemas de planejamento de trajetória.

Como breve resumo desse trabalho, [26] utiliza dois AFA, um deles denominado Autômato de Exploração, em que a informação de saída indicada pela execução deste autômato determina qual a direção do próximo movimento a ser executado pelo sistema motor do robô, nas quatro direções. Em sequência, tal informação é transmitida para o outro AFA, denominado Autômato de Mapeamento, responsável pelo registro dos dados sobre a topologia do ambiente que está sendo mapeado.

Pelo registro topológico do ambiente, constrói-se, por estados de AFA: *i*) Uma trajetória de retorno de um local já conhecido que envolva um menor número de deslocamentos do robô durante o processo de exploração; *ii*) O mapa do ambiente construído durante o processo de navegação do robô, registrado na memória do agente robótico.

A metodologia desta pesquisa envolve também a formalização de um espaço de busca de rotas com certa analogia com [36], porém tal espaço é representado nesta pesquisa por uma vizinhança adaptativa, por estados do AFA.
Este estudo propõe aplicar o formalismo dos SLRDA para possibilitar navegação em qualquer ângulo, atendendo ao requerido por [34] e [36], integrando o ponto positivo obtido por [26] por meio da adaptatividade viabilizando a utilização progressiva da memória do sistema, consumida de acordo com a área já mapeada.

Complementarmente, o formalismo dos SLRDA apresenta vantagens adicionais em utilização de memória, considerando que basta o registro de coordenadas de apenas dois pontos, os pontos extremos dos segmentos digitalizados representando trajetórias, e não mais o registro de todos os pontos das trajetórias para construção do mapa, conforme [26].

Pelo fato de que as trajetórias variam em ângulo com qualquer comprimento, em que este comprimento é variável teoricamente até infinito, inviabiliza-se a utilização de autômatos finitos com os propósitos deste estudo. Resulta que esta proposta se posiciona na pesquisa de dispositivos e algoritmos guiados por regras (com adaptatividade hierárquica multi-nível) cujo conjunto de regras é variável apresentando funções adaptativas modificáveis [32].

B. O Método Clássico

Os experimentos utilizam trajetórias variáveis em ângulo e comprimento, comparando-se com o método clássico em que se inclui o trabalho de [26]. Isso porque a denominação "método clássico" significa neste trabalho a não utilização da adaptatividade, ou adaptatividade parcial por existência de restrições.

Desse modo, o método clássico é constituído normalmente por uma ou mais das seguintes características dos dispositivos ou algoritmos: não utilização da adaptatividade, movimento do robô em apenas umas poucas direções (comumente quatro direções ortogonais ou oito direções pela inclusão das direções diagonais, tais como as direções definidas pela vizinhança-4 ou vizinhança-8, respectivamente), registro de todos os pontos das trajetórias para construção do mapa, comprimentos das trajetórias limitados, indefinidos ou não formalizados, recuperação de erros em trajetórias não formalizada.

Assim, levando-se em conta que [26] representa o estado da arte quanto às técnicas adaptativas utilizando autômatos, mais especificamente o AFA, os experimentos são comparados com esse trabalho. Dessa forma, o método de comparação usado nos experimentos mostra a trajetória por [26], composto de pequenos movimentos nas quatro direções ortogonais (norte, sul, leste e oeste) e a trajetória pelo método proposto resultante de cadeia de entrada para AFA atuando como autômato de movimento.

Por meio dessas comparações, chega-se a conclusões quanto à robustez da representação de ambientes de navegação de robôs móveis por segmentos de linhas retas digitalizadas (SLRD) pelo modelo de SLRDA, baseado em AFA.

VII. EXPERIMENTOS

Os experimentos de [14] para ensaios do SLRDA das Fig. 4 e 5 utilizaram as cadeias de entrada da Fig. 9, especificadas na Tab. III, mostrando o reconhecimento adequado

independentemente de desvios em ângulo (em uma determinada faixa de tolerância) e variações em comprimento.

Figura 9. Exemplos de SLRD modelados por um único SLRDA.

TABELA III
CADEIAS DA FIGURA 9 (DA ESQUERDA PARA A DIREITA) MODELADAS POR UM ÚNICO SLRDA.

CADEIAS DA FIG. 7	CODIFICAÇÃO
1	a^2b
2	a^2ba^3b
3	ba^3b
4	$a^3ba^3ba^4ba^3b$
5	$a^3ba^3ba^4ba^3ba^3ba^3ba^4ba^3b$
6	$a^3ba^3ba^4ba^3ba^5ba^4ba^3ba^4b$
7	$a^3ba^5ba^3ba^4ba^3ba^3ba^4ba^3ba^5ba^4b$

Os experimentos deste trabalho apresentam analogia aos de [14], integrando-se o movimento do robô.
O deslocamento do robô é previsto inicialmente por este trabalho como da ordem da dimensão do robô, ou seja, esse fato envolve uma questão de escala compatível entre cada passo para movimento do robô e as suas respectivas dimensões.

A. Comparação de trajetórias com o método clássico

A Fig. 10 mostra em tracejado a trajetória percorrida pelo robô pelo método clássico, o qual é apresentado numa escala não compatível com o passo de seu deslocamento, definido pela grade.

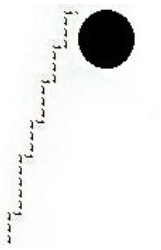


Figura 10. O robô é representado pela bola preta, fora de escala compatível com o passo de sua trajetória, indicada pelo tracejado.

A Fig. 11 mostra em pontilhado a trajetória pelo método clássico do mesmo robô, em duas posições diferentes, indicado em escala compatível com o passo da trajetória. A trajetória por esta pesquisa é representada pela linha reta.

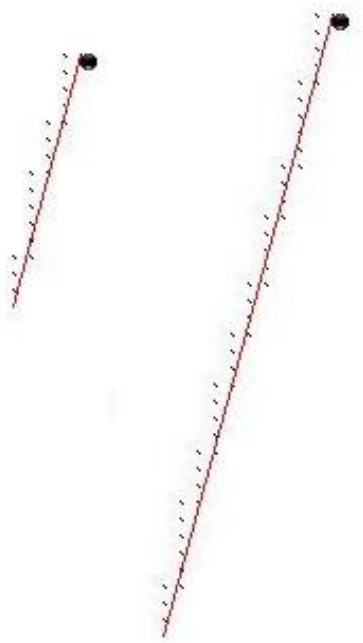


Figura 11. A figura mostra o mesmo robô em duas posições diferentes da mesma trajetória. O robô se desloca em trajetória indicada pelos pontos pelo método clássico [26]; enquanto que, pelo método proposto por esta pesquisa, o robô se desloca em trajetória representada pela linha reta. A figura à esquerda mostra o robô em uma posição intermediária e, à direita, a posição final para a cadeia de entrada $a^3ba^5ba^3ba^4ba^3ba^3ba^4ba^3ba^5ba^4b$, reconhecida pelo AFA das Fig. 4 e Fig. 5.

Uma maneira didática para descrever o método proposto por esta pesquisa é considerar que o robô se move a cada USLR da cadeia de entrada (correspondente a um SLRD), ao invés de se mover em cada símbolo pelo método clássico. Implica que o robô percorra uma trajetória composta de pontos correspondentes a cada USLR. Resulta que, dada SLRD de entrada com *n* USLR:
{$USLR_1$, $USLR_2$, $USLR_3$, $USLR_4$, $USLR_5$,..., $USLR_n$}, o robô terá *n* deslocamentos curtos até atingir o ponto final.

Outro modo, computacionalmente mais efetivo que o anterior, é o robô se direcionar diretamente ao ponto final, pois, neste caso envolve a troca de informações com o robô apenas de coordenadas indicativas da localização no mapa do ambiente de apenas um ponto. Isso é esquematizado na Fig. 11 pelas linhas retas, representado a trajetória pelo método proposto.

B. Grade Adaptativa

A Fig. 12 mostra pelas linhas retas as trajetórias percorridas pelo robô, em duas grades diferentes. Nesse caso de alteração adaptativa da grade, o banco de autômatos permanece inalterado.

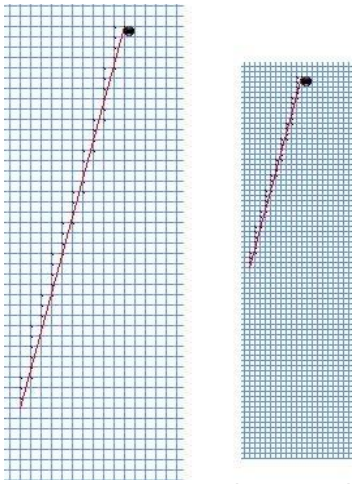


Figura 12. As linhas retas são exemplo de trajetórias pelo método proposto por esta pesquisa, em duas grades diferentes. A cadeia de entrada SLRD = $a^3ba^5ba^3ba^4ba^3ba^3ba^4ba^3ba^5ba^4b$ é reconhecida pelo AFA das Fig. 4 e Fig.5.

C. Segmentação de trajetórias por segmentos adaptativos em escalas adaptáveis

A Fig. 13 mostra um robô em uma trajetória segmentada em três SLRD (cadeias 1, 2 e 3).

A cadeia 1 é $SLRD_1$ = $a^2ba^4ba^3ba^4ba3ba^3ba^4ba^3ba^4ba^4b$ reconhecida pelo Autômato B-setor 2.

A cadeia 2 é $SLRD_2$ = $(cb^5)(cb^6)(cb^6)(cb^7)$ reconhecida pelo Autômato B – setor 8. A cadeia 3 é $SLRD_3$ = $(c^8b)(c^9b)(c^{12}b)^4$ reconhecida pelo Autômato C-setor 7.

Na Fig. 13, no centro, as cadeias estão em uma mesma escala. Na Fig. 13, à esquerda e à direita, as escalas da cadeia 3 são 0,5 e 1,5 respectivamente.

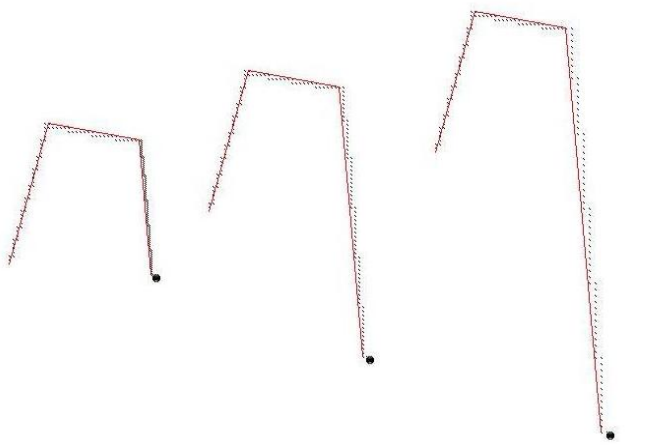


Figura 13 A figura mostra uma mesma trajetória, segmentada em 3 SLRD (cadeia 1, cadeia 2 e cadeia 3), com a cadeia 3 em grades diferentes. As trajetórias pelo método clássico são indicadas pelos pontilhados, enquanto as linhas retas mostram as trajetórias pelo método desta pesquisa.

VIII. TRABALHOS FUTUROS

Trabalhos futuros envolvem estudos de algoritmos para navegação robótica em cenários com obstáculos, investigando a necessidade de SLRDA de exploração e mapeamento. O SLRDA de Mapeamento armazena os dados indicativos da trajetória que está sendo percorrida pelo conjunto de unidades, principalmente comprimentos dos arcos; O SLRDA de Exploração tem a função de atuar nas mudanças de direção do próximo deslocamento efetuado pelo robô, durante a exploração da trajetória prevista, decidindo favoravelmente ou não pela mudança. Por exemplo, no caso de uma região de alta curvatura e comprimento muito pequeno relativamente ao arco global, o SLRDA de Exploração evitaria a mudança de direção. Esta decisão é feita com base em regiões de suporte variáveis adaptativamente, dos dados coletados sobre a existência ou não de mudanças de curvaturas ou direções, nas informações previamente registradas pelo SLRDA de Mapeamento e no algoritmo de exploração executado por este autômato.

Quanto aos algoritmos dos SLRDA de mapeamento e exploração, em [14] foram estudados modelos de cadeias de SLRD, associados aos correspondentes SLRDA. Para tais modelos, esses algoritmos envolvem técnicas de inferência de SLRD, a fim de que os mesmos se auto ajustem, levando em consideração a similaridade em padrões repetitivos de segmentos. Estudos de inferência gramatical pela adaptatividade foram apresentados em [28] e [29].

IX. CONSIDERAÇÕES FINAIS.

A despeito da relevância da representação computacional de SLRD, inclusive sendo uma área ativa de pesquisas há quase meio século conforme os levantamentos de [30] e [31], esse assunto ainda não foi explorado em navegação robótica, considerando cenários sujeitos a influências espúrias, com os recursos adaptativos listados em [32]. Esta pesquisa preenche tal lacuna, incorporando os fundamentos da geometria discreta aritmética de [33] ao método sintático em navegação robótica, por meio de técnicas adaptativas.

Este trabalho mostrou que o formalismo apresentado em [14] traz novas possibilidade à navegação robótica, agregando duas das principais vantagens verificadas na proposta de SLRDA: *i*) Simplicidade de modelagem e relativa facilidade de implementação, associadas a alto poder computacional; *ii*) Modelos fáceis de entender, relativamente simples de programar e flexíveis para aceitar mudanças em seu comportamento.

REFERÊNCIAS

[1] Guoyu Zuo, Peng Zhang, and Junfei Qiao, “Path planning algorithm based on sub-region for agricultural robot,” in Anais... march 2010, vol. 2, pp. 197 –200, 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR).

[2] N. Sariff and N. Buniyamin, “An overview of autonomous mobile robot path planning algorithms,” in Anais... 4th Student Conference on Research and Development, June 2006.

[3] Yeong-Hwa Chang, Tsu-Tian Lee, and Chang-Huan Liu, “On-line approximate cartesian path trajectory planning for robotic manipulators,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 22, no. 3, may 1992.

[4] Sonja Macfarlane and Elizabeth A. Croft, “Jerk-bounded manipulator trajectory planning: Design for real-time applications,” IEEE Transactions on Robotics, vol. 19, no. 1, pp. 42–52, February 2003.

[5] Dhananjay Bodhale and Nitin Afzulpurkar, “Path planning for a mobile robot in a dynamic environment,” in Anai... IEEE International Conference on Robotics and Biomimetics, 2008.

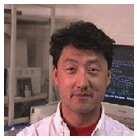
[6] J. Sellen, “Direction weighted shortest path planning,” in Anais... may 1995, vol. 2, pp. 1970 –1975 vol.2, IEEE International Conference on Robotics and Automation.

[7] Stefan K. Gehrig and Fridtjof J. Stein, “A trajectory-based approach for the lateral control of car following systems,” in Anais..., 1998, vol. 4, pp. 3596 – 3601.

[8] B. H. Lee, “Algorithmic approach to straight line trajectory planning for mechanical manipulators,” in Anais..., June 1986, pp. 121 – 126.

[9] F. Schramm, A. Micaelli, and G. Morel, “Calibration free path planning for visual servoing yielding straight line behaviour both in image and work space,” in Anais.. aug. 2005, pp. 2216 – 2221, International Conference on Intelligent Robots and Systems, (IROS 2005).
[10] Rudiger Befit, Dietrich Paulus, and Michael Harbeck, “Segmentation of lines and arcs and its application for depth recovery,” in Anais... april 1997, vol. 4, pp. 3165 –3168 vol.4, IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-97.
[11] C.G. Cassandras, “Rapid learning techniques for discrete event systems,” in Anais.. jun 1993, pp. 7/1 –7/8, IEE Colloquium on Discrete Event Systems: A New Challenge for Intelligent Control System.
[12] A. H. R. Costa, Robótica móvel inteligente: progressos e desafios, Tese de livre docência, Escola Politécnica da Universidade de São Paulo, 2003.
[13] Eliana P. L. Aude, Ernesto P. Lopes, Cristiano S. Aguiar, and Mario F. Martins, “Door crossing and state identification using robotic vision,” in Anais... 2006, 8th International IFAC Symposium on Robot Control.
[14] Leoncio Claro de Barros Neto, Modelagem em geometria digital aprimorada por técnicas adaptativas de segmentos de retas, Ph.D. thesis, Escola Politécnica da Universidade de São Paulo (USP), Junho 2011.
[15] Leoncio C. de Barros Neto, André R. Hirakawa, and Antonio M. A. Massola, “An adaptive model applied to digital geometry to enhance segment straightness,” IEEE Latin America Transactions, vol. 9, pp. 956 – 962, Oct. 2011.
[16] H. Freeman, “Boundary encoding and processing,” Picture Processing and Psychopictorics, pp. 241–266, 1970,
[17] Azriel Rosenfeld, “Digital straight line segments,” IEEE Transactions on Computers, vol. C-23, no. 12, pp. 1264–1269, December 1974.
[18] Peter F.M. Nacken, “Metric for line segments,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 12, pp. 1312–1318, December 1993.
[19] F. Feschet, “The lattice width and quasi-straightness in digital spaces,” in Anais..., Tampa, FL, December 2008, International conference on Pattern Rrecognition - ICPR, pp. 1–4.
[20] Shu Xiang Li and Murray H. Loew, “Analysis and modeling of digitized straight-line segments,” in Anais..., Rome, Italy, 1988, Proceedings of International Conference on Pattern Recognition, pp. 294–296, Publ by IEEE, Piscataway, NJ. S.M.
[21] S.M. Aghito and F.S. Forchhammer, “Context-based coding of bilevel images enhanced by digital straight line analysis,” IEEE Transactions on Image Processing, vol. 15, no. 8, pp. 2120–2130, August 2006.
[22] D. Proffitt and D. Rosen, “Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges,” Computer Graphics and Image Processing, vol. 10, no. 4, pp. 318–332, 1979.
[23] Kai Ching You and King Sun Fu, “A syntactic approach to shape recognition using attributed grammars,” IEEE transactions on systems, man, and cybernetics, vol. 9, no. 6, pp. 334–345, June 1979.
[24] B. Garau, A. Alvarez, and G. Oliver, “Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach,” in Anais.. april 2005, pp. 194 – 198, IEEE International Conference on Robotics and Automation, ICRA 2005.
[25] M. Tarokh, “A genetic robot path planner with fuzzy logic adaptation,” in Anais... july 2007, pp. 388 –393, 6th IEEE/ACIS International Conference on Computer and Information Science.
[26] M. A. A. Sousa, “Mapeamento de ambientes desconhecidos por robôs móveis utilizando autômatos adaptativos,” M.S. thesis, Escola Politécnica da Universidade de São Paulo, 2006.
[27] Leoncio C. de Barros Neto, André R. Hirakawa, and Antonio M. A. Massola, “Adaptive modeling of digital straightness applied to geometric representation enhancement,” International Journal of Computer Applications, vol. 10, no. 2, pp. 31–39, November 2010.
[28] Margarete Keiko Iwai, Um formalismo gramatical adaptativo para linguagens dependentes de dependentes de contexto., Doutorado, Escola Politécnica da Universidade de São Paulo, São Paulo, 2000.
[29] Ivone Penque Matsudo, “Um estudo dos processos de inferência de gramáticas regulares e livres de contexto baseados em modelos adaptativos.,” Mestrado., Escola Politécnica da Universidade de São Paulo, 2006.
[30] R. A. Klette and A. B. Rosenfeld, “Digital straightness: a review,” Discrete Applied Mathematics, vol. 139, no. 1-3, pp. 197–230, April 2004.
[31] Partha Bhowmick and Bhargab B. Bhattacharya, “Fast polygonal approximation of digital curves using relaxed straightness properties” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, pp. 1590–1602, September 2007.
[32] J. J. Neto, “Um levantamento da evolução da adaptatividade e da tecnologia adaptativa,” Revista IEEE América Latina, vol. 5, no. 7, pp. 496–505, Novembro. 2007.
[33] J. P. Reveillès, Géométrie discrète, calcul en nombres entiers et algorithmique, Ph.D. thesis, Université Louis Pasteur, Strasbourg, 1991.
[34] Kenny Daniel et. al. A*: Any-Angle Path Planning on Grids. Journal of Artificial Intelligence Research, October 2010.
[35] Kamil Tulum et. al. Situation Aware UAV Mission Route Planning. IEEE Aerospace conference, March 2009.
[36] Ferdinand Wagner et. al. Modeling Software with Finite State Machines: a Practical Approach. Auerbach Publications, 2006.

Leoncio Claro de Barros Neto é formado em Engenharia Elétrica pela Escola de Engenharia Mauá, São Caetano do Sul, SP, em 1979. Concluiu o mestrado em 1994 e obteve o título de Doutor em Engenharia Elétrica em 2011, ambos pela POLI, USP. Em 1981 foi aprovado em concurso público de âmbito nacional, integrando o Corpo de Engenheiros Navais da Marinha do Brasil (MB). Na MB exerceu atividades técnicas em Setores de Manutenção e de Pesquisas, iniciando programa de doutorado na USP em 2006, após transferência para a reserva. Suas áreas de interesse e pesquisa abrangem automação, sensores inteligentes, eletrônica de potência, guerra eletrônica e suas aplicações.



André Riyuiti Hirakawa recebeu os títulos de Engenheiro Eletricista e correspondente mestrado pela USP em 1990 e 1992 respectivamente. Em 1992, como pesquisador, foi integrado no programa de doutorado da Yokohama National Universidade, em Yokiohoma, Japan, tendo recebido o título de PhD em 1997. Em 1998, iniciou atividades no Departamento de Engenharia de Computação e Sistemas Digitais da USP, sendo atualmente Professor Associado. Suas áreas de interesse e pesquisa abrangem automação e robótica, sensores inteligentes, eletrônica de potência, planejamento de rotas, AVG‘s, e sistemas sem fio, tanto aplicados à automação agrícola quanto em outros ambientes.

Projeto de Leiaute Semi-Automático utilizando Dispositivo Adaptativo

P. R. M. Cereda

***Resumo*—Este artigo apresenta um estudo preliminar sobre a utilização de dispositivos adaptativos no processo de elaboração de leiautes. Este caso específico pode ser generalizado para o problema de particionamento de regiões. Durante a fase de projeto, foi utilizado um modelo semi-automático para geração de um leiaute simplificado baseado em um estudo de caso.**

***Palavras-chave:*—Editoração de texto, Diagramação, Dispositivo adaptativo, Particionamento de regiões**

I. Introdução

A diagramação de texto consiste no processo de distribuir elementos textuais na área disponível de uma página. Os textos podem compartilhar seu espaço com outros elementos gráficos, tais como imagens e formas. A diagramação é uma etapa primordial na publicação de materiais gráficos, tais como jornais, revistas e periódicos [1].

O profissional designado para a tarefa de diagramação deve levar em consideração uma série de critérios para realizar a distribuição dos elementos em uma página, tais como tamanho dos elementos e nível hierárquico. A diagramação costuma ser um processo oneroso em tempo e esforço.

Este artigo procura realizar um estudo preliminar sobre a utilização de dispositivos adaptativos no processo de diagramação, na tentativa de torná-lo mais fluente, simplificado e proporcionar meios para predição de espaços sub-ótimos para a acomodação dos elementos textuais e gráficos. Uma visualização mais abstrata do processo de diagramação pode ser generalizada para o problema de particionamento de regiões, tema recorrente em diversas áreas de pesquisa, incluindo Inteligência Artificial e Pesquisa Operacional.

A organização deste artigo é a seguinte: na Seção II, o processo de diagramação é apresentado, abrangendo suas características principais. Na Seção III, o projeto de leiaute semi-automático é contextualizado. A Seção IV apresenta uma implementação do modelo proposto. Os experimentos e análises são apresentados na Seção V. As considerações finais são apresentadas na Seção VI.

II. Diagramação

O processo de diagramação define um conjunto de categorias de elementos que determinam seus níveis hierárquicos e aspectos tipográficos em uma disposição de página. Além disso, existem critérios específicos para cada tipo de publicação, nos quais os elementos podem receber regras específicas que têm prioridade sobre outras regras. Alguns dos elementos textuais incluem títulos, subtítulos, corpos de texto, epígrafes, notas de rodapé, caixas de texto e blocos de citações. Cada um destes elementos possui características de formatação e disposição hierárquica distintas [1].

Para que os elementos textuais sejam incorporados na área disponível de uma página, é necessário que exista um projeto de leiaute, isto é, um conjunto de diretrizes e critérios que determine a disposição. Algumas dessas diretrizes incluem número de colunas, margens de página, espaçamento entre elementos, hifenização, alinhamento textual e orientação. A Figura 1 apresenta um exemplo de leiaute de página contendo alguns elementos textuais e gráficos, a saber: *(1)* dimensão total da página, *(2)* corpo de texto, *(3)* figura inserida ao lado do texto, *(4)* linha divisória, e *(5)* margens da página.

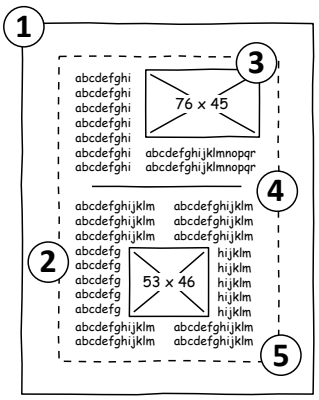


Figura 1. Exemplo de leiaute de página, contendo alguns elementos textuais e gráficos. A área tracejada denota as margens da página.

Um projeto de leiaute deve também levar em consideração aspectos tipográficos e de ergonomia. Tamanhos de fontes pequenos ou famílias de fontes com traços irregulares podem comprometer a visão do leitor, dificultando a leitura e a própria compreensão do texto. É também interessante que respeite-se, sempre que possível, o fluxo natural de um texto – nos países ocidentais, de cima para baixo, da esquerda para a direita – e que linhas orfãs ou parágrafos curtos não sejam isolados em blocos distintos.

III. Projeto de leiaute semi-automático

A área de inteligência artificial apresenta vários estudos para oferecer soluções automáticas para o problema de particionamento de regiões e, em particular, na resolução da disposição hierárquica de elementos textuais e gráficos em uma página. Os modelos propostos apresentam resultados interessantes, inclusive na avaliação de leiautes com um alto nível de complexidade [2], [3], [4], [5], [6].

Os autor pode ser contatado através do seguinte endereço de correio eletrônico: cereda@users.sf.net.

Apesar de extremamente eficientes, os modelos tradicionais de projeto de leiaute automático requerem um conjunto de regras imutáveis, o que limita a abrangência do algoritmo e dificulta sua extensão para outras classes, além de demandarem tempo e esforço computacionais consideráveis [3]. O processo de diagramação exige rapidez e agilidade na disposição dos elementos na página por parte do profissional; algoritmos eficientes mas onerosos em tempo e recursos podem comprometer seriamente o fluxo de trabalho dos corpos editoriais.

Com o advento da Internet e popularização das redes sociais, os conteúdos de jornais, revistas e periódicos passaram a ser disponibilizados em vários formatos. O processo de diagramação sofreu uma transformação para contemplar vários projetos de leiaute em um mesmo documento, de acordo com cada formato de distribuição, de modo simultâneo - meio eletrônico, impressão monocromática ou colorida, conteúdo para dispositivos móveis, livros digitais, entre outros. A necessidade de automação do processo de leiaute tornou-se muito mais evidente, principalmente em grandes veículos de informação [3].

A tecnologia adaptativa tem obtido resultados interessantes na utilização de dispositivos adaptativos aplicados em problemas tradicionalmente tratados com técnicas de inteligência artificial [7]. A simplicidade e o poder computacional de tais dispositivos contribuem para sua ampla utilização, além de serem eficientes em tempo e espaço [8]. Este artigo propõe um modelo computacional utilizando um dispositivo adaptativo para construir um projeto de leiaute semi-automático, ao invés das técnicas tradicionais de inteligência artificial geralmente empregadas nesta particular classe do problema de particionamento de regiões.

O modelo proposto é dito *semi-automático* porque é esperado que o diagramador forneça sua ordem inicial preferida para os elementos textuais a serem inseridos na página. O dispositivo adaptativo então tentará manter-se o mais fiel possível a essa ordem.

O leiaute escolhido para o modelo proposto é o mais simplificado possível: respeitando as margens da página, o dispositivo adaptativo tentará dispor uma quantidade arbitrária de blocos de texto independentes entre si em duas colunas não-balanceadas. Os blocos de textos podem continuar em outras páginas, caso seja necessário. A Figura 2 apresenta um esboço do leiaute a ser obtido.

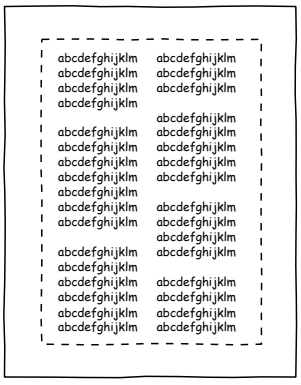


Figura 2. Esboço do leiaute a ser obtido através do modelo proposto.

Algumas diretrizes foram impostas ao modelo para representar o contexto de um projeto de leiaute real. Dois grupos de leiaute foram definidos no escopo deste artigo, cada um com suas próprias diretrizes, apresentados a seguir.

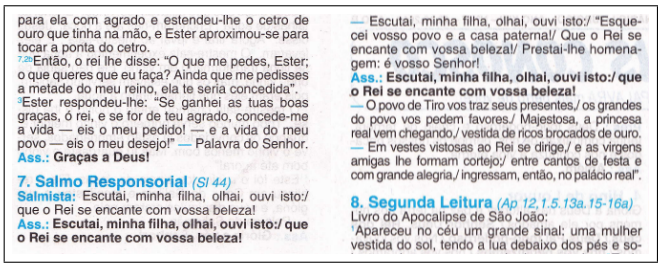
O primeiro grupo consiste em um leiaute para blocos de textos com eventual troca de posição entre blocos, sem quebra de texto para a página seguinte. As diretrizes definidas são:

1) A troca de posição entre blocos de textos é permitida, mas o dispositivo adaptativo tentará manter-se o mais fiel possível à ordem inicial.
2) Um bloco de texto pode iniciar em uma coluna e continuar em outra, mas não pode iniciar em uma página e continuar em outra.

O segundo grupo contempla um leiaute para blocos de textos respeitando a ordem dos blocos, com eventuais repetições de partes dos textos, de acordo com o contexto e disposição. As diretrizes definidas são:

1) A troca de posição entre blocos de textos é proibida, de forma que a ordem dos elementos seja sempre preservada.
2) O dispositivo adaptativo poderá repetir partes do texto de acordo com uma semântica estabelecida previamente. Por exemplo, refrões podem ser repetidos quando o bloco de texto continuar na página seguinte.
3) Um bloco de texto pode fluir por todo o leiaute da página, inclusive continuando em outra página, se necessário. O dispositivo adaptativo deve apenas evitar linhas órfãs e deslocamento de parágrafos curtos.

A Figura 3 apresenta um exemplo de leiaute real com diretrizes semelhantes às do segundo grupo. Observe que o bloco em negrito é repetido na página seguinte para facilitar a leitura. O leiaute em questão é proveniente do semanário litúrgico *Deus Consco*, uma publicação da Editora Santuário contendo o ordinário de missa do dia, e amplamente utilizado por algumas comunidades católicas para acompanhar as missas.


para ela com agrado e estendeu-lhe o cetro de ouro que tinha na mão, e Ester aproximou-se para tocar a ponta do cetro.
7.2bEntão, o rei lhe disse: "O que me pedes, Ester; o que queres que eu faça? Ainda que me pedisses a metade do meu reino, ela te seria concedida".
3Ester respondeu-lhe: "Se ganhei as tuas boas graças, ó rei, e se for de teu agrado, concede-me a vida — eis o meu pedido! — e a vida do meu povo — eis o meu desejo!" — Palavra do Senhor.
Ass.: Graças a Deus!

7. Salmo Responsorial *(Sl 44)*
Salmista: Escutai, minha filha, olhai, ouvi isto:/ que o Rei se encante com vossa beleza!
Ass.: Escutai, minha filha, olhai, ouvi isto:/ que o Rei se encante com vossa beleza!

— Escutai, minha filha, olhai, ouvi isto:/ "Esquecei vosso povo e a casa paterna!/ Que o Rei se encante com vossa beleza!/ Prestai-lhe homenagem: é vosso Senhor!
Ass.: Escutai, minha filha, olhai, ouvi isto:/ que o Rei se encante com vossa beleza!
— O povo de Tiro vos traz seus presentes,/ os grandes do povo vos pedem favores./ Majestosa, a princesa real vem chegando,/ vestida de ricos brocados de ouro.
— Em vestes vistosas ao Rei se dirige,/ e as virgens amigas lhe formam cortejo;/ entre cantos de festa e com grande alegria,/ ingressam, então, no palácio real".

8. Segunda Leitura *(Ap 12,1.5.13a.15-16a)*
Livro do Apocalipse de São João:
1Apareceu no céu um grande sinal: uma mulher vestida do sol, tendo a lua debaixo dos pés e so-

Figura 3. Exemplo de leiaute real com diretrizes semelhantes às do segundo grupo. O leiaute em questão é proveniente do semanário litúrgico *Deus Consco*.

A Tabela I resume os grupos de leiaute utilizados para a definição do modelo de projeto de leiaute semi-automático e suas diretrizes.

O dispositivo adaptativo escolhido para representar o modelo de projeto de leiaute proposto foi o autômato adaptativo, devido à sua simplicidade e poder computacional [9], [10]. É importante destacar que outros dispositivos adaptativos poderiam ser utilizados neste caso; o autômato foi escolhido por uma preferência do autor.

Tabela I
GRUPOS DE LEIAUTE DEFINIDOS NO ESCOPO DO ARTIGO.

	Grupo 1	Grupo 2
Troca de posições	Sim	Não
Respeitar ordem	Se possível	Sim
Fluir em outra página	Não	Sim
Blocos como átomos	Sim	Não
Partes convencionais	Não	Sim
Partes de repetição	Não	Sim

A primeira etapa da definição do modelo é realizar a divisão do espaço disponível de uma página em regiões menores. Para o modelo proposto, optou-se pela divisão em oito regiões de dimensões iguais, com quatro regiões por coluna, conforme ilustrado na Figura 4. Considerando as dimensões das margens como $M = a \times b$, onde a denota a altura e b a largura, cada região i terá as dimensões $R_i = c \times d$, onde $c = \frac{a}{4}$ e $d = \frac{b}{2}$.

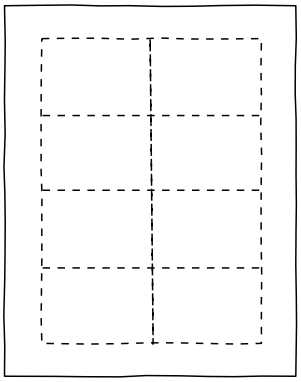


Figura 4. Esboço do leiaute a ser obtido através do modelo proposto.

A etapa seguinte consiste em discretizar os blocos de texto a serem inseridos em termos das regiões calculadas no passo anterior. Cada bloco de texto i a ser inserido no espaço disponível da página terá sua área A_i calculada e depois comparada com a área da região, $A_{\text{região}} = cd$, para determinar o número de regiões que o bloco i ocupará. O cálculo do número de regiões N_i é apresentado na Fórmula 1.

$$N_i = \left\lceil \frac{A_i}{A_{\text{região}}} \right\rceil \quad (1)$$

É importante salientar que o modelo definido neste artigo não suporta valores de N_i maiores do que oito regiões, isto é, textos que estrapolam uma página inteira. Esta decisão foi tomada por questões de simplicidade.

O autômato adaptativo é modelado de acordo com os blocos de textos disponíveis. Cada bloco de texto será representado por um estado q_i do autômato. O alfabeto de entrada Σ adota as letras do alfabeto latino em ordem crescente denotando o número de regiões de cada bloco, isto é, todos símbolos e seus respectivos significados apresentados na Tabela II estarão disponíveis.

A modelagem a seguir contempla o primeiro grupo de leiaute. Cada transição do automato adaptativo do modelo consome um símbolo $\sigma_i \in \Sigma$ que denota a quantidade de regiões

Tabela II
SÍMBOLOS DO ALFABETO DE ENTRADA Σ E SEUS REPECTIVOS SIGNIFICADOS NO PRIMEIRO GRUPO DE LEIAUTE.

Símbolo	N_i	Símbolo	N_i
a	1	e	5
b	2	f	6
c	3	g	7
d	4	h	8

ocupadas pelo bloco de código seguinte – o estado de destino da transição corrente. As funções adaptativas do autômato mapeiam as diretrizes do modelo de projeto de leiaute; caso um bloco de texto apresente uma inconsistência de leiaute, provavelmente devido ao número de regiões necessário para sua correta inclusão, a função adaptativa associada à transição corrente realiza uma modificação na topologia do autômato, alterando as transições seguintes ou inserindo estados novos.

A criação de um estado novo no conjunto de estados disponíveis Q indica que o modelo de projeto de leiaute não pôde encontrar um bloco de texto que tenha o número mínimo de regiões para ser inserido na posição corrente. Para não comprometer a diagramação dos elementos subsequentes, o autômato adiciona um estado preenchendo a região corrente tantas vezes quanto for necessário até que um bloco de texto tenha o número de regiões válidas para então ser inserido.

O autômato adaptativo do modelo tem uma execução extremamente compacta. As funções adaptativas são executadas de acordo com o símbolo corrente da cadeia de entrada e seu respectivo significado. Por exemplo, se existem apenas dois blocos disponíveis na página e o símbolo corrente denota um número de regiões maior do que esse valor, a função adaptativa será executada e tentará encontrar um bloco de texto que atenda ao requisito; caso não existam blocos de texto com o número mínimo de regiões, um novo estado denotando um bloco de texto *dummy* é então inserido, e o reconhecimento do restante da cadeia prossegue.

Como exemplo, considere oito blocos de textos já discretizados para serem avaliados pelo autômato adaptativo. A Tabela III apresenta o número de regiões de cada bloco.

Tabela III
NÚMERO DE REGIÕES DE CADA BLOCO.

Bloco	N_i	Bloco	N_i
1	2	5	1
2	3	6	2
3	2	7	2
4	2	8	2

A partir do número de regiões de cada bloco e de sua ordem estabelecida, e de acordo com a Tabela II, a cadeia a ser submetida ao autômato terá a forma $\langle bcbbabbb \rangle$. Uma representação do autômato adaptativo do primeiro grupo de leiaute é apresentado na Figura 5, transposto no leiaute das páginas, para fácil visualização. As transições foram omitidas, e as cores preenchem a quantidade de regiões ocupadas por cada bloco, denotado por cada estado.

É possível notar que, de acordo com a Figura 5, existe

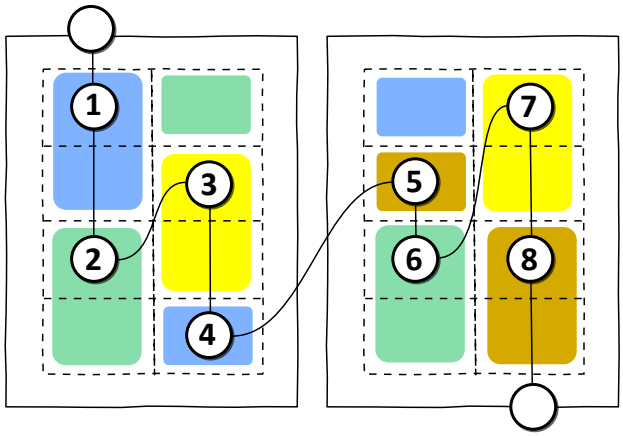


Figura 5. Representação do autômato adaptativo do primeiro grupo de leiaute.

um bloco de texto violando uma das diretrizes do primeiro grupo de leiaute para o modelo – o bloco número quatro está iniciando o texto na primeira página e encerrando na segunda. Para que a diretriz sobre praticidade e ergonomia seja contemplada, o autômato adaptativo tentará, por meio das funções adaptativas, redefinir a posição do bloco de texto problemático.

Uma possível disposição dos oito blocos de textos nas duas páginas, sem violar as diretrizes e tentando manter-se o mais fiel possível à ordem dos elementos, é apresentada na Figura 6. É importante observar que essa ilustração apresenta, de modo simplificado, a lógica de como as funções adaptativas atuarão sobre os estados e transições.

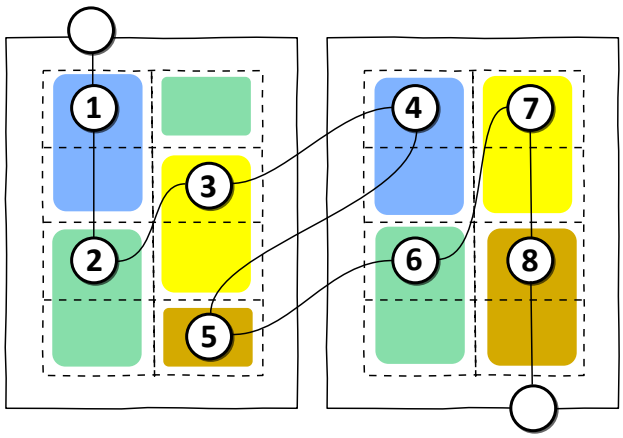


Figura 6. Possível disposição dos oito blocos de textos nas duas páginas, sem violar as diretrizes.

O autômato adaptativo resultante após o reconhecimento da cadeia $\langle bcbbabbb \rangle$ é apresentado na Figura 7. Note que as funções adaptativas removeram as transições $q_3 \rightarrow q_4$, $q_4 \rightarrow q_5$ e $q_5 \rightarrow q_6$, e as novas transições $q_3 \rightarrow q_5$, $q_5 \rightarrow q_4$ e $q_4 \rightarrow q_6$ foram inseridas. A configuração final do autômato – ou melhor, a ordem de seus estados – determina a disposição dos blocos de textos nas páginas.

As funções adaptativas atuam sobre a disponibilidade de regiões na página corrente, reconfigurando os blocos de textos para compôr uma região válida, ou simplesmente marcar

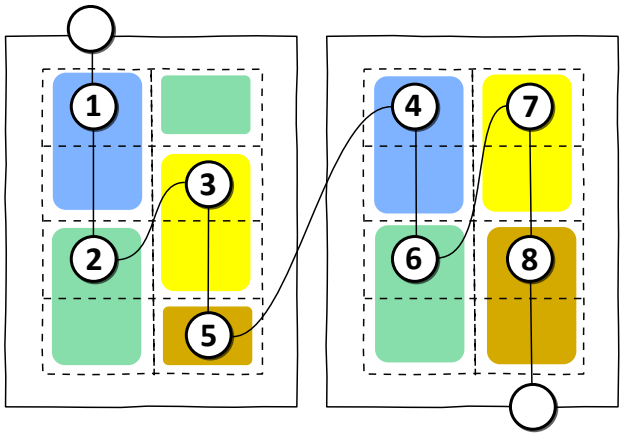


Figura 7. Representação da configuração final do autômato adaptativo do primeiro grupo de leiaute.

o espaço como insuficiente, através da inserção de novos estados.

A modelagem do autômato adaptativo do segundo grupo de leiaute é semelhante à do primeiro grupo, com a exceção do tratamento dos blocos de textos. No primeiro grupo, os blocos de textos eram tratados como elementos atômicos; no segundo grupo, existe uma subdivisão dentro do próprio bloco de texto. Para simplificar a compreensão, a subdivisão foi definida como o número de regiões N_i (Fórmula 1) do bloco de texto i, isto é, um bloco de texto i com $N_i = 4$ terá quatro subdivisões (ou partes) dentro do bloco.

Uma parte do bloco de texto pode admitir repetição. Por exemplo, suponha que um canto tenha três estrofes e um refrão (estrofe principal); o último pode ser repetido entre cada estrofe. Um jogral pode também admitir repetições de partes. Por uma questão de ergonomia, a repetição pode ser útil em alguns casos – se o refrão encontra-se em uma página e as estrofes seguintes em outra, é necessário alternar páginas para acompanhar a leitura; a repetição do refrão na nova página poupa esforços e facilita o manuseio e compreensão.

O alfabeto de entrada Σ, no segundo grupo de leiaute, adota as letras do alfabeto latino, no qual cada letra denota uma parte determinada. Letras iguais referem-se a um bloco de texto. O tamanho desta subcadeia denota o número de regiões do bloco de texto; por exemplo, $\langle aaa \rangle$ denota o bloco de texto a com o número de regiões $N_a = |aaa| = 3$.

As letras minúsculas denotam partes convencionais do bloco de texto; analogamente, as letras maiúsculas referem-se às partes de repetição. Por exemplo, $\langle bBbb \rangle$ pode denotar um bloco de texto com três estrofes e um refrão.

Como exemplo, considere a cadeia de entrada na forma $\langle aABbCcDddeEFfGg \rangle$ a ser submetida ao autômato, denotando sete blocos de textos, cada qual com suas partes convencionais e de repetição. Uma representação do autômato adaptativo do segundo grupo de leiaute é apresentado na Figura 8, transposto no leiaute das páginas, para fácil visualização. As transições foram omitidas, e as cores preenchem a quantidade de regiões ocupadas por cada bloco, denotado por cada estado. As marcações listradas denotam as partes de repetições.

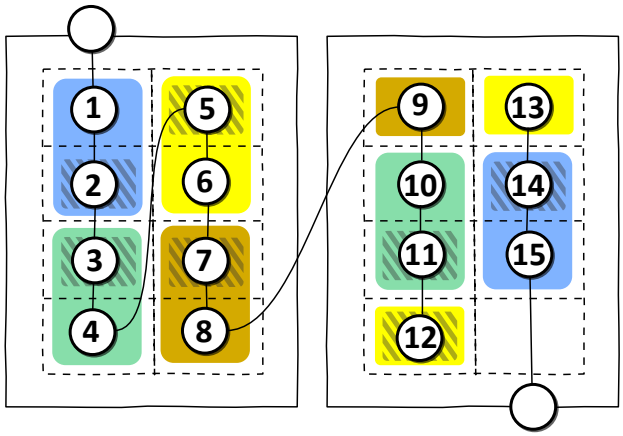


Figura 8. Representação do autômato adaptativo do segundo grupo de leiaute.

É possível notar que, de acordo com a Figura 8, existe um bloco de texto violando uma das diretrizes do segundo grupo de leiaute para o modelo – o bloco número quatro inicia em uma página, com uma parte de repetição, e termina em outra página, com apenas uma parte convencional. Para que a diretriz sobre ergonomia seja contemplada, o autômato adaptativo tentará, por meio das funções adaptativas, reproduzir a parte de repetição na segunda página.

O autômato adaptativo resultante após o reconhecimento da cadeia $\langle aABbCcDddeEFfGg \rangle$ é apresentado na Figura 9. Note que as funções adaptativas removeram a transição $q_8 \rightarrow q_9$ que liga duas partes convencionais, criaram um novo estado q_{16} idêntico a q_7 – uma parte de repetição – e as novas transições $q_8 \rightarrow q_{16}$ e $q_{16} \rightarrow q_9$ foram inseridas. A configuração final do autômato – ou melhor, a ordem de seus estados – determina a disposição das partes dos blocos de textos nas páginas.

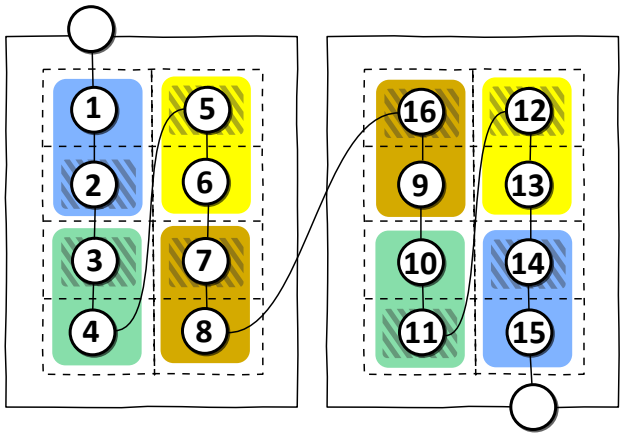


Figura 9. Representação da configuração final do autômato adaptativo do segundo grupo de leiaute.

As funções adaptativas atuam sobre a distribuição semântica das partes de um bloco de texto na página corrente, reproduzindo partes de repetição entre partes convencionais para compôr um bloco de texto válido, ou simplesmente evitar linhas órfãs e parágrafos curtos.

O autômato adaptativo do modelo pode representar várias classes de projetos de leiaute, incluindo novas diretrizes e critérios hierárquicos de inserção. A página também pode ser subdividida e ter autômatos independentes atuando em cada partição, com regras e restrições exclusivos de cada área.

IV. Implementação

O modelo do projeto de leiaute semi-automático apresentado na Seção III foi implementado utilizando a linguagem Python e executado em um ambiente Linux de 64 bits. O modelo foi disponibilizado como um módulo, isto é, um arquivo contendo definições e declarações da linguagem. A utilização do módulo requer apenas sua importação, através do comando:

```
>>> import aalayout
```

Na fase de definição dos aspectos de implementação, optou-se por traduzir as diretrizes de projeto de leiaute diretamente no código-fonte. Estuda-se a possibilidade de torná-las independentes, escritas em uma linguagem de marcação – por exemplo, XML – e carregadas em tempo de execução pelo módulo de leiaute. Por ora, apenas as diretrizes dos dois grupos de leiaute apresentados na Seção III estão disponíveis.

A utilização do módulo é semelhante ao processo de reconhecimento de uma cadeia w pelo autômato adaptativo do modelo; é suficiente fornecer a cadeia de entrada e o identificador do grupo de leiaute escolhido:

```
>>> aalayout.generateLayout('bcbbabbb', 1)
- Input:
[1, 2, 3, 4, 5, 6, 7, 8]
- Output:
[[1, 2, 3, 5], [4, 6, 7, 8]]
```

A saída da função `generateLayout` é uma lista de listas. Cada lista interna representa uma página, com seus respectivos blocos de textos. Caso existam blocos de textos *dummy*, eles serão representados pela letra `X`, indicando que não existiam blocos de texto com o número mínimo de regiões para concluir a diagramação da página corrente:

```
>>> aalayout.generateLayout('cccc', 1)
- Input:
[1, 2, 3, 4]
- Output:
[[1, 2, X], [3, 4, X]]
```

Quando o segundo grupo de leiaute é escolhido, a saída da função `generateLayout` é ligeiramente diferente:

```
>>> aalayout.generateLayout('aABbCcDddeEFfGg',
    2)
- Input:
[[1, 2], [3, 4], [5, 6], [7, 8, 9], [10, 11],
    [12, 13], [14, 15]]
- Output:
[[[1, 2], [3, 4], [5, 6], [7, 8]], [[7, 9],
    [10, 11], [12, 13], [14, 15]]]
```

Neste caso, a lista mais interna representa blocos de textos, com suas respectivas partes convencionais e de repetição. A próxima lista no nível superior representa uma página, contendo os blocos de textos. Caso existam partes a serem repetidas, seu identificador é repetido nas regiões correspondentes.

O módulo é extremamente compacto e compatível com as séries 2 e 3 da linguagem Python. É possível utilizá-lo autonomamente, na forma de script, ou incluí-lo em programas já existentes.

V. Experimento e Análise

Foi realizado um experimento semelhante aos exemplos sintéticos apresentados na Seção III para confirmar os resultados obtidos. O experimento consistiu na diagramação de dois folhetos de cantos religiosos de uma comunidade católica do interior do estado de São Paulo. O primeiro folheto consistia em oito cantos, dispostos em duas colunas, ao longo de duas páginas no formato A5. As diretrizes do projeto de leiaute utilizadas conicidiram com as do primeiro grupo de leiaute. O segundo folheto consistia em sete cantos, dispostos em duas colunas, ao longo de duas páginas no formato A5. As diretrizes do projeto de leiaute utilizadas coincidiram com as do segundo grupo de leiaute.

O experimento foi iniciado com a diagramação do primeiro folheto. Na primeira etapa, os cantos foram discretizados para determinar suas dimensões em cada região disponível da página, considerando as margens. A classificação dos oito cantos do primeiro folheto é apresentada na Tabela IV.

Tabela IV
Número de regiões para cada canto do primeiro folheto do experimento.

Bloco	N_i	Bloco	N_i
1	2	5	1
2	3	6	2
3	2	7	2
4	2	8	2

Como comparação, a Figura 10 apresenta uma diagramação linear do primeiro folheto sem levar em conta as diretrizes do projeto de leiaute. É possível observar a quebra do quarto canto para a próxima página – os traços horizontais no folheto indicam a separação entre blocos de cantos.

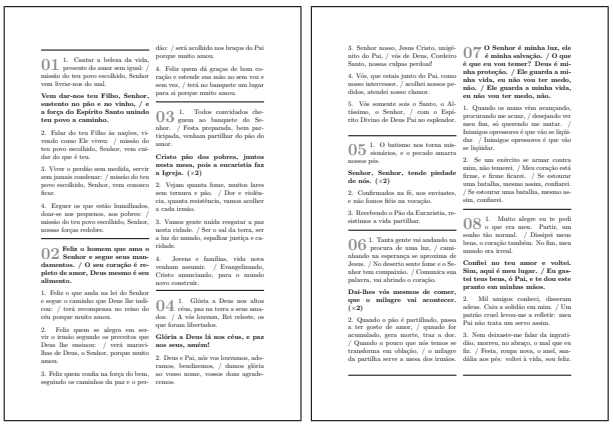


Figura 10. Diagramação linear do primeiro folheto, sem diretrizes.

De acordo com a Tabela IV, a cadeia a ser submetida ao autômato adaptativo também apresentou a forma $\langle bcbbabbb \rangle$. A topologia inicial do autômato é praticamente idêntica à representação do autômato da Figura 5.

Após a submissão da cadeia $\langle bcbbabbb \rangle$, o autômato resultante apresenta a disposição desejada dos cantos no folheto, conforme ilustrado na Figura 11. A topologia final do autômato também é idêntica à representação do autômato da Figura 7.

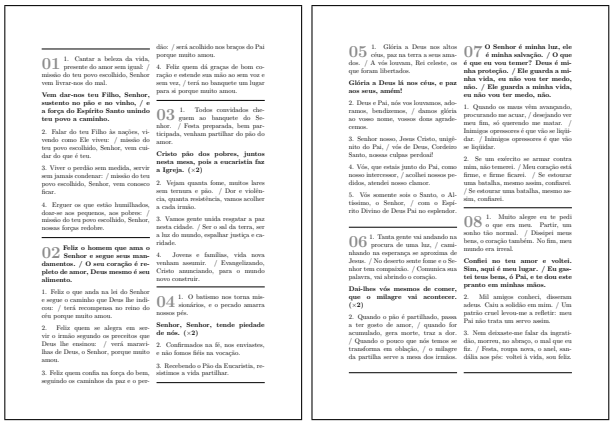


Figura 11. Diagramação final do primeiro folheto, de acordo com a topologia do autômato adaptativo resultante do modelo do primeiro grupo de leiaute.

A segunda parte do experimento consistiu na diagramação do segundo folheto, utilizando as diretrizes do segundo grupo de leiaute. Na primeira etapa, os cantos foram discretizados para determinar suas dimensões em cada região disponível da página, considerando as margens. A classificação dos sete cantos do segundo folheto é apresentada na Tabela V.

Tabela V
Número de regiões para cada canto do segundo folheto do experimento.

Bloco	N_i	Bloco	N_i
1	2	5	2
2	2	6	2
3	2	7	2
4	3		

Como comparação, a Figura 12 apresenta uma diagramação linear do segundo folheto sem levar em conta as diretrizes do projeto de leiaute. É possível observar a quebra do quarto canto para a próxima página, sem a repetição do refrão na nova página.

De acordo com a Tabela V, a cadeia a ser submetida ao autômato adaptativo também apresentou a forma $\langle aABbCcDddeEFfGg \rangle$. A topologia inicial do autômato é praticamente idêntica à representação do autômato da Figura 8.

Após a submissão da cadeia $\langle aABbCcDddeEFfGg \rangle$, o autômato resultante apresenta a disposição desejada dos cantos no folheto, conforme ilustrado na Figura 13. A topologia final do autômato também é idêntica à representação do autômato da Figura 9.

Adicionalmente, foram realizadas avaliações de desempenho com a implementação do modelo do projeto de leiaute semi-automático, apresentada na Seção IV. Os testes consistiram em submeter um conjunto de cantos litúrgicos, devidamente discretizados e classificados, ao módulo de leiaute e calculando seu tempo de execução. O tamanho do conjunto de

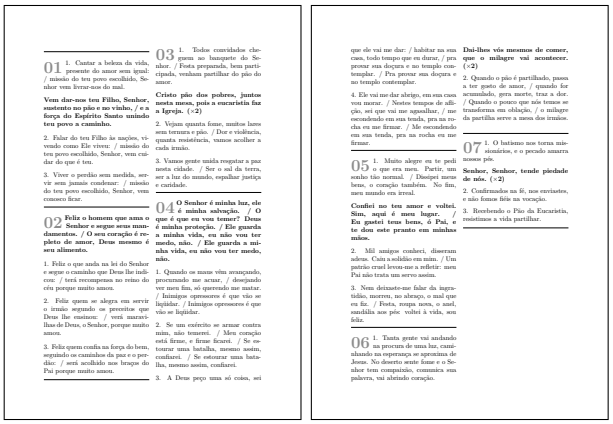


Figura 12. Diagramação linear do segundo folheto, sem diretrizes.

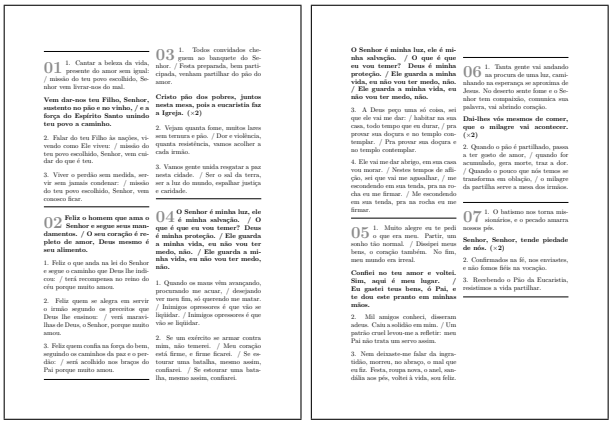


Figura 13. Diagramação final do segundo folheto, de acordo com a topologia do autômato adaptativo resultante do modelo do segundo grupo de leiaute.

teste cresceu de 100 a 1500 cantos, de tamanhos arbitrários, utilizando os dois grupos de leiaute apresentados na Seção III. Os resultados obtidos estão ilustrados na Figura 14. O eixo x denota o tamanho do conjunto de cantos, iniciando em 100 e encerrando em 1500, e o eixo y denota o tempo de execução do módulo, em segundos. A Tabela VI apresenta os valores obtidos na avaliação.

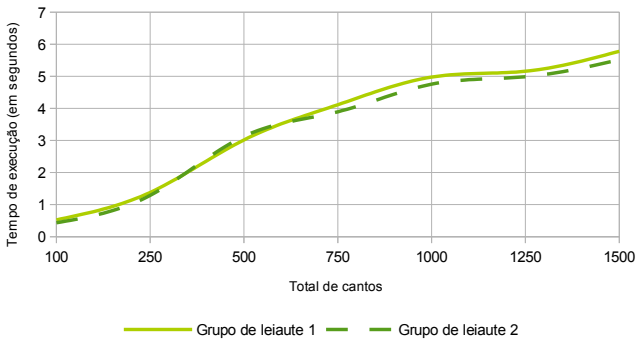


Figura 14. Avaliação de desempenho da implementação do modelo do projeto de leiaute semi-automático.

De acordo com a Figura 14, é possível observar que a execução do módulo foi linear, independentemente do grupo de leiaute escolhido. O primeiro grupo de leiaute apresentou

Tabela VI
VALORES OBTIDOS NA AVALIAÇÃO DE DESEMPENHO DA IMPLEMENTAÇÃO DO MODELO DO PROJETO DE LEIAUTE SEMI-AUTOMÁTICO.

	Total de cantos						
	100	250	500	750	1000	1250	1500
Grupo 1	0,524	1,378	3,015	4,112	4,975	3,157	5,780
Grupo 2	0,432	1,287	3,125	3,894	4,754	4,987	5,512

uma ligeira diferença de tempo em relação ao segundo grupo, devido ao fato de eventualmente realocar as posições dos cantos durante a diagramação, mas tal variação não compromete o desempenho total.

O modelo apresentado mostrou-se extremamente eficiente em tempo e espaço computacionais. Obteve-se uma configuração sub-ótima para o problema de diagramação em uma ordem $\mathcal{O}(n)$ de tempo de reconhecimento da cadeia, onde n denota o comprimento da cadeia a ser submetida ao autômato.

A simplicidade do modelo proposto permite sua extensão para projetos de leiaute mais complexos, incluindo um conjunto de diretrizes arbitrárias e regiões irregulares – por exemplo, uma figura com texto à sua volta. Além disso, o conjunto de regras – funções adaptativas – pode ser substituído em tempo de reconhecimento da cadeia por outro conjunto, caso seja necessário. Esta característica confere uma mutabilidade ao projeto de leiaute, sem entretanto, exaurir recursos computacionais.

VI. CONSIDERAÇÕES FINAIS

Este artigo apresentou um estudo preliminar sobre a utilização de dispositivos adaptativos para a geração de modelos computacionais para projetos de leiaute, uma classe específica do problema de particionamento de regiões. O autômato adaptativo do modelo proposto mostrou-se interessante na resolução de problemas complexos, como a diagramação de blocos de textos de tamanho arbitrário.

Os dispositivos adaptativos proporcionam soluções computacionais viáveis para problemas complexos, mantendo a simplicidade e a eficiência em tempo e espaço. Para problemas em que uma solução sub-ótima já é suficiente, como no caso apresentado neste artigo, os dispositivos adaptativos são uma alternativa extremamente interessante à utilização de técnicas tradicionais de inteligência artificial.

AGRADECIMENTOS

O autor agradece ao professor João José Neto, da Escola Politécnica da Universidade de São Paulo, pelas valiosas contribuições para a escrita deste artigo.

REFERÊNCIAS

[1] A. T. Turnbull and R. N. Baird, *The Graphics of Communication: Typography–Layout–Design*. Holt, Rinehart and Winston, Inc., 1975.
[2] F. Vico, "Automatic design synthesis with artificial intelligence techniques," *Artificial Intelligence in Engineering*, vol. 13, pp. 251–256, 1999.
[3] S. Lok and S. Feiner, "A survey of automated layout techniques for information presentations," 2001.
[4] S. K. Feiner, "A grid-based approach to automating display layout," in *Proceedings on Graphics interface 1988*, 1988.

[5] L. M. Pereira, "Artificial intelligence techniques in automatic layout design," *Artificial Intelligence and Pattern Recognition in Computer Aided Design*, 1978.
[6] S. M. Casner, "Task-analytic approach to the automated design of graphic presentations," *ACM Trans. Graph.*, vol. 10, pp. 111–151, 1991.
[7] J. J. Neto, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa," *IEEE Latin America Transactions*, vol. 5, no. 7, pp. 496–505, 2007.
[8] P. R. M. Cereda and J. J. Neto, "Mineração adaptativa de dados: Aplicação à identificação de indivíduos," in *WTA 2012: Workshop de Tecnologia Adaptativa*, 2012.
[9] J. J. Neto, "Contribuições à metodologia de construção de compiladores," Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.
[10] ——, "Adaptive automata for context-dependent languages," *SIGPLAN Notices*, vol. 29, no. 9, pp. 115–124, 1994.

Paulo Roberto Massa Cereda é graduado em Ciência da Computação pelo Centro Universitário Central Paulista (2005) e mestre em Ciência da Computação pela Universidade Federal de São Carlos (2008). Atualmente, é desenvolvedor de software, atuando em diversos segmentos de mercado, com enfoque principal em software livre. É membro ativo do repositório de código fonte *SourceForge* desde 2005, contribuindo com bibliotecas e softwares de propósito geral. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: inteligência artificial, linguagens de programação, teoria da computação e tecnologia adaptativa.

Proposta de protocolo de roteamento de dados e camada de supervisão adaptativos em rede de sensores com nós móveis

I. M. Santos e C. E. Cugnasca

***Resumo*— Rede de Sensores Sem Fio é um tipo especial de rede *ad hoc*, formada por vários sensores capazes de coletar e processar informações do ambiente em que estão distribuídos. Essa tecnologia tem sido empregada em diversos tipos de problemas, inclusive no monitoramento em agricultura de precisão. Este trabalho apresenta um protocolo de roteamento de dados para o contexto da aplicação de pulverização agrícola, considerando a comunicação entre a rede de sensores e um nó móvel embarcado no veículo pulverizador. Desenvolver protocolos de roteamento para contextos específicos é importante pois visa a otimização do funcionamento da rede de sensores e a eficiência da aplicação. O protocolo proposto compreende duas etapas: uma de construção da topologia inicial e outra de sua manutenção. A tecnologia adaptativa atua na fase de manutenção da topologia da rede de sensores, buscando decidir pelas ações mais adequadas e evitando o consumo excessivo de energia dos nós sensores. Além disso foi incorporada uma camada supervisora adaptativa que monitora diferentes aspectos do protocolo de roteamento, podendo emitir alertas ao usuário ou mesmo atuar sobre o protocolo de roteamento de modo a otimizá-lo. O trabalho apresenta a especificação do protocolo de roteamento, sua Tabela de Decisão Adaptativa e a arquitetura do sistema, considerando a camada supervisora adaptativa.**

***Palavras-chave*— redes de sensores sem fio, protocolo de roteamento, nós móveis, pulverização agrícola, tecnologia adaptativa, supervisão.**

I. Introdução

As Redes de Sensores Sem Fio (RSSF) são um dos primeiros exemplos de computação pervasiva, esse tipo de rede é uma tecnologia emergente que tem sido aplicada em diferentes problemas que envolvem monitoramento, automação e controle. Uma RSSF é um tipo especial de rede *ad hoc*, formada por muitos (dezenas, centenas ou até milhares) sensores distribuídos sobre uma área de interesse, com comunicação sem fio e capaz de coletar e processar informações do ambiente nos quais os sensores estão distribuídos [1, 2, 3].

As RSSF têm sido aplicadas em diferentes áreas, como em sistemas de monitoramento ambiental, militar, na agropecuária, no monitoramento médico, em sistemas de suporte pessoal, mobilidade, segurança, controle logístico, processos industriais, sistemas embarcados, entre outros [1, 4, 5, 6, 7, 8, 9, 10, 11].

Uma RSSF típica possui uma infraestrutura limitada e o monitoramento do ambiente é baseado no esforço colaborativo entre os nós que compõem a rede de sensores. Suas características e propriedades específicas as diferem das redes de computadores tradicionais, principalmente quanto aos recursos disponíveis [12]. As principais limitações das RSSF estão relacionadas com a fonte de energia, processamento, alcance de comunicação, banda de transferência de dados e armazenamento de dados.

O uso racional e otimizado desses recursos é fundamental, especialmente da fonte de energia, pois determina o tempo de vida útil (funcionamento) do sistema. Na rede de sensores, cada nó é autônomo e responsável por utilizar de maneira eficiente seus recursos.

Nesse sentido, o protocolo de roteamento de dados em uma RSSF determina o comportamento (processamento, envio e recebimento de mensagens) dos nós que compõem a rede e consequentemente o gasto energético. O desenvolvimento de protocolos de roteamento para RSSF deve ter como objetivo viabilizar sistemas capazes de gerenciar a comunicação entre os sensores de uma rede e propagar os dados para o usuário. [7] afirmam que o protocolo de roteamento deve encontrar meios de gerenciar com eficiência os recursos disponíveis, visando prolongar o tempo de atividade da RSSF, garantindo os aspectos da tolerância a falhas, escalabilidade e segurança.

Em razão da autonomia dos sensores, do dinamismo da topologia nas RSSF e do vasto conjunto de aplicações com diferentes contextos e objetivos, não há um conjunto de protocolos de roteamento e algoritmos computacionais amplamente eficientes para qualquer contexto de aplicação das redes de sensores. Dessa forma, é necessário escolher ou desenvolver protocolos de roteamento eficientes para contextos específicos, com base nos objetivos de roteamento e na demanda da aplicação.

Além das restrições inerentes aos sensores que compõem a rede, existem numerosos desafios de projeto e comunicação em RSSF que têm influência sobre os protocolos de roteamento, pois degradam seu desempenho. Dessa forma, além da natureza da aplicação, alguns aspectos devem ser levados em conta no desenvolvimento de um protocolo de roteamento, sendo eles: distribuição dos nós; meio de comunicação; conectividade; cobertura; tolerância a falhas; escalabilidade; e agregação de dados. [13] discute em seu

I. M. Santos, Universidade de São Paulo (USP), São Paulo, São Paulo, Brasil, ivairton@usp.br

C. E. Cugnasca, Universidade de São Paulo (USP), São Paulo, São Paulo, Brasil, carlos.cugnasca@poli.usp.br

trabalho diferentes estratégias que podem ser adotadas para definir um protocolo de roteamento. Por exemplo, podem-se priorizar rotas que utilizam sensores com maior disponibilidade de energia, ou aquelas rotas com menor tráfego de dados, ou simplesmente definir uma rota que passe pelo menor número de sensores.

Decidir sobre a melhor estratégia a ser adotada depende diretamente do contexto da aplicação no qual a rede de sensores será utilizada. Este trabalho usa como contexto de referência o monitoramento agrícola, que consiste em observar continuamente uma área de plantio, com o objetivo de avaliar as mudanças ocorridas nesse ambiente, especialmente as condições climáticas (temperatura, umidade, etc.).

Um dos desafios na produção agrícola é a aplicação eficiente de pesticidas com baixo custo e sem contaminar o meio ambiente e pessoas [14, 15]. O principal efeito a ser evitado durante a aplicação de pesticidas é a deriva [16]. Deriva é o deslocamento horizontal que sofrem as gotas desde o seu ponto de lançamento até atingirem o solo ou as plantas. Para minimizá-la é preciso configurar corretamente o equipamento de pulverização (escolha dos bicos de pulverização, pressão e mistura dos produtos) e conhecer as condições ambientais no momento da aplicação, especialmente do vento, temperatura e umidade [16, 17, 18]. As RSSF podem ser empregadas como suporte no processo de pulverização agrícola, monitorando as condições ambientais no momento da aplicação do pesticida [19, 20, 21].

Este trabalho apresenta um protocolo de roteamento de dados em RSSF, com base no contexto do problema da pulverização agrícola, utilizando de tecnologia adaptativa para a manutenção das rotas no roteamento de dados. Além disso é proposta uma camada supervisora adaptativa que monitora diferentes aspectos do protocolo de roteamento e atua com o objetivo de melhorar o desempenho do protocolo.

O trabalho está organizado de modo que a Seção II apresenta uma contextualização do monitoramento agrícola e como as RSSF podem sem empregadas, com foco no processo de pulverização agrícola. A Seção III apresenta e discute o protocolo de roteamento de dados, descrevendo como a tecnologia adaptativa é empregada, tanto no protocolo de roteamento quanto na camada de supervisão. Finalmente na Seção IV são apresentadas as considerações finais.

II. Definição do contexto de utilização da RSSF

A agricultura de precisão consiste em dividir o terreno da área cultivada em parcelas e tratá-lo de modo diferenciado (específico), buscando atender as necessidades de cada área. Dessa forma, espera-se o aumento da produção, menor custo e menor impacto ao meio ambiente [22]. O uso das RSSF na agricultura de precisão possibilita um melhor monitoramento da cultura e das propriedades do ambiente de cultivo [23].

O processo de pulverização agrícola traz alguns desafios dentre os quais se destaca o efeito deriva, especialmente quando a aplicação é realizada por aeronaves.

Os fatores que interferem na formação e intensidade da deriva são o tamanho (diâmetro) e peso das gotas, o vento, a temperatura, a umidade relativa do ar e a altura de lançamento. O principal aspecto de controle da deriva é o tamanho das gotas, determinado por meio dos ajustes nos bicos pulverizadores instalados no equipamento de pulverização. Outro fator importante no controle da deriva, foco deste trabalho, é a velocidade e direção do vento. Como nenhuma gota consegue se mover contra o vento, o planejamento da aplicação, que leva em conta a direção do vento, é vital para o controle da deriva. Pulverizar com o vento na direção favorável pode garantir a segurança de áreas sensíveis, como culturas vizinhas, pastos, cursos d'água, vilas, cidades, entre outros.

Considerando a capacidade das redes de sensores e os fatores que interferem diretamente no controle da deriva no processo de pulverização agrícola, as RSSF se mostram como uma tecnologia candidata a dar suporte no processo de pulverização. Nesse contexto, considera-se que os nós sensores da rede estarão distribuídos sobre a área a ser pulverizada, monitorando informações como temperatura, umidade e condições do vento (direção e velocidade). O veículo pulverizador (trator, avião, ou até um veículo aéreo não tripulado) atua como nó móvel coletor de informações (sorvedouro).

As informações das condições ambientais coletadas pela RSSF serão utilizadas pelo veículo pulverizador, determinando correções na sua rota. Dessa forma espera-se que ao considerar as variações nas condições ambientais (especialmente do vento) e ajustando a rota do veículo pulverizador durante a aplicação do defensivo agrícola, de modo a atingir adequadamente a área alvo, obtenha-se uma aplicação mais eficiente, com o controle adequado da praga ou doença, sem desperdício de produto e com menor contaminação do ambiente.

Um dos principais desafios nesse processo de integração entre a rede de sensores e o nó coletor embarcado no veículo pulverizador é a definição e manutenção do roteamento de dados, de modo a garantir que os sensores que compõem a rede possam entregar corretamente os dados registrados por eles ao nó coletor móvel.

O desenvolvimento de protocolos de roteamento para RSSF deve ter como objetivo viabilizar sistemas capazes de gerenciar a comunicação entre os sensores de uma rede e propagar os dados para o sensor coletor. A seção seguinte descreve o protocolo de roteamento proposto e como a tecnologia adaptativa é utilizada.

III. Descrição do protocolo de roteamento de dados e do modelo adaptativo

O protocolo de roteamento de dados em uma RSSF é responsável por estruturar logicamente toda comunicação, seja ela entre os sensores, ou entre os sensores e o usuário.

O protocolo de roteamento proposto considera que a rota a ser percorrida pelo veículo pulverizador será definida previamente, antes do início do seu movimento. Essa informação será enviada para a RSSF que irá utilizá-la no processo de determinação do roteamento dos dados. A Fig. 1 ilustra essa representação inicial da rota do nó coletor móvel na RSSF. Os nós sensores são representados pelos círculos, o nó coletor móvel pelo triângulo e sua trajetória pela linha pontilhada. A rota é determinada por meio das coordenadas

dos pontos de partida e chegada do nó móvel, respectivamente os pontos A e B.

O nó coletor móvel (veículo pulverizador) pode atuar junto à rede de sensores por meio de duas perspectivas distintas: a primeira seria uma ação passiva, na qual o nó coletor estaria em um estado constante de recebimento dos dados, recebendo informações enviadas proativamente pela rede de sensores; a segunda é uma ação ativa, na qual o nó coletor móvel executa uma consulta à RSSF e então aguarda o processamento e o retorno da resposta da rede de sensores. Este trabalho adota o segundo contexto, no qual a RSSF aguarda a passagem e consulta do nó coletor móvel para então executar a respectiva consulta e entregar os dados solicitados.

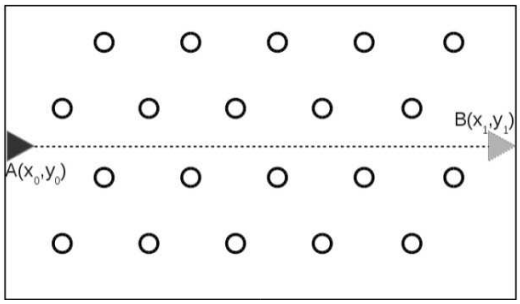


Figura 1. Representação da rota do nó sensor coletor móvel junto à rede de sensores.

Assumindo esses princípios, o protocolo de roteamento possui duas fases distintas, sendo elas a construção inicial da tabela de roteamento dos dados e a manutenção da topologia da rede.

A. Construção da tabela de roteamento

Após a alocação dos sensores na área a ser monitorada, a primeira tarefa a ser executada para o seu funcionamento é a construção da tabela de roteamento, que segue um algoritmo (estático) composto por três passos: representação da rota do nó coletor móvel; seleção dos nós cabeça de *cluster*; e definição da árvore de roteamento inicial.

Representação da rota do nó coletor móvel: a rota do nó coletor será representada por meio da especificação das coordenadas dos pontos de entrada e saída do nó na área a ser percorrida (conforme ilustrado na Fig. 1). Com esses pontos é possível utilizar a equação fundamental da reta ($y_A - y_B = m.(x_A - x_B)$) e geral (determinante via regra de Sarrus), para representação da rota do nó móvel e posteriormente estabelecer a relação de distância entre a rota e a posição dos nós sensores (que têm suas coordenadas conhecidas por meio de GPS).

Seleção dos nós cabeça de *cluster*: o protocolo de roteamento fará uso de *clusters*, que são aglomerados de sensores que se organizam localmente, elegendo um nó de referência denominado cabeça de *cluster* (CH). Essa estratégia busca reduzir o número de mensagens transitando na rede, aumentar a escalabilidade e hierarquizar a organização dos nós, consequentemente reduzindo o consumo de energia. A seleção dos nós CH levará em consideração a carga de energia disponível dos sensores (priorizar aqueles com mais energia) e sua posição em relação à rota do nó coletor móvel. Nessa etapa pode ser utilizada a equação da distância entre um ponto e a reta ($d = |ax_i+by_i+c| \,/\, \sqrt{a_2 + b_2}$), priorizando aqueles mais próximos de onde o nó coletor irá passar. Apesar de um parâmetro inicial, o número dos nós CH deve ser definido dinamicamente durante a execução do algoritmo de manutenção da topologia da rede. Inicialmente é sugerido que esse número fique em 5% do número total de nós da rede. Esse processo será monitorado pela camada supervisora adaptativa.

Definição da árvore de roteamento: a definição da árvore de roteamento inicial pode se basear em diferentes algoritmos. Será aqui considerado o algoritmo *Earliest First Tree* [24], com uma estrutura adicional que contém a lista dos possíveis nós "pais" para cada nó da rede. O algoritmo consiste nas seguintes etapas: cada CH envia uma mensagem do tipo "inicialização" no modo *broadcast*. A árvore de roteamento é construída com base na propagação e no tempo de envio desta mensagem. Cada nó da rede, ao receber a primeira mensagem de inicialização, determina o remetente como seu nó "pai" e reencaminha essa mensagem no modo *broadcast*. Ao receber outras mensagens do tipo "inicialização" oriundas de outros nós, o nó constrói sua lista de possíveis nós "pais", determinando a prioridade de acordo com a ordem de chegada das mensagens.

A Fig. 2 ilustra a definição da árvore de roteamento com foco em um nó da rede. No primeiro momento (à esquerda) o nó em destaque com a cor negra recebe três mensagens de inicialização em diferentes tempos (t_1, t_2 e t_3). Em um segundo momento (à direita), dado que $t_1 < t_2 < t_3$, o nó passa a ter como seu nó "pai" o nó A, construindo sua lista de possíveis nós "pais" com os nós {B, C} nessa ordem de prioridade.

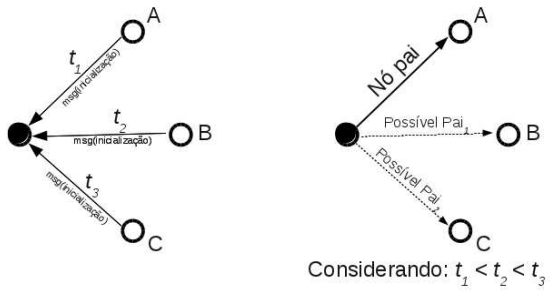


Figura 2. Estratégia de definição da topologia inicial baseada no algoritmo *Earliest First Tree*, no qual o nó pai de um determinado nó é aquele que primeiro envia a mensagem de inicialização, mais a construção de uma lista de possíveis nós "pais".

A Fig. 3 ilustra um exemplo de uma tabela de roteamento inicial, com a representação da rota do nó coletor móvel, os nós CH (em destaque com cor negra) e as rotas entre os nós sensores e os nós CH.

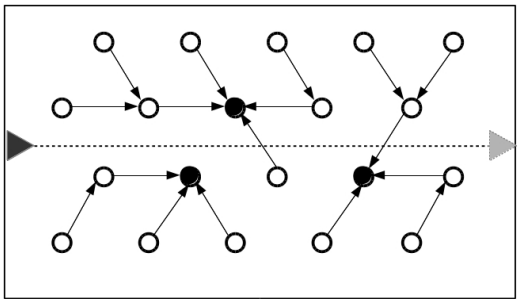


Figura 3. Exemplo de configuração inicial da topologia da rede de sensores após execução do algoritmo de inicialização.

B. Manutenção da topologia da rede

Na etapa de manutenção da topologia da rede será empregada a tecnologia adaptativa como mecanismo de decisão entre quais ações devem ser executadas, de acordo com as ocorrências identificadas pela rede de sensores. Essas ocorrências podem ser, por exemplo, falha, inclusão e baixo nível de energia nos nós sensores, entre outras. Este modelo é baseado no trabalho de [25].

Na manutenção da topologia da RSSF poderão ser adotadas três ações distintas, sendo elas:

Ação A1 – Reconstrução global da topologia. Neste caso o processo segue estritamente o mesmo algoritmo de inicialização da RSSF (conforme apresentado anteriormente).

Ação A2 – Reconstrução da topologia do *cluster*. Nesta ação o processo de ajuste da topologia fica limitado aos nós que compõe um *cluster*.

Ação A3 – Reconstrução local. Somente o nó especificado redefine seu nó "pai".

Para subsidiar o processo de decisão entre quais ações devem ser executadas, é apresentada uma Tabela de Decisão Adaptativa (TDA) que irá correlacionar diferentes critérios por meio de uma regra.

Inicialmente são sugeridos seis critérios; entretanto é possível que novos critérios sejam adicionados à TDA caso necessário. Todos os critérios empregados na TDA assumem valores dicotômicos "Sim" ou "Não". Os critérios propostos são:

Critério C1 – Nó CH: identifica se o nó é CH.

Critério C2 – Nó roteador: identifica se o nó é um nó intermediário na árvore de roteamento.

Critério C3 – Falha de um nó: contexto que identifica se algum nó da rede está apresentando falha.

Critério C4 – Inserção de novo nó: contexto que representa que um novo nó foi adicionado na RSSF.

Critério C5 – Exclusão por desligamento: contexto que identifica que um nó foi removido da rede.

Critério C6 – Nó com baixa energia: identifica que um nó está com seu nível de energia em estado crítico (baixa disponibilidade), com desligamento iminente.

Dado o conjunto de possíveis ações {A1, A2, A3} a serem executadas e o conjunto inicial de critérios a serem avaliados {C1, C2, C3, C4, C5, C6}, propõe-se uma configuração inicial de regras que irão compor a TDA.

A Tabela I demonstra um exemplo inicial para a TDA, com possibilidade de ter seu número de regras ampliado por meio das funções adaptativas. Essa TDA descreve três regras iniciais. Como exemplo, na primeira regra (R1), os critérios C1 e C3 possuem valor "Sim", enquanto que os demais possuem valor "Não". A regra determina que nesse contexto (do nó ser um CH (C1) e estar com falha (C3)) deverá ser executada a ação A1 (reconstrução global da topologia).

TABELA I
Exemplo de configuração inicial da TDA.

	R1	**R2**	**R3**	**...**
C1	S	N	N	...
C2	N	S	N	...
C3	S	S	N	...
C4	N	N	S	...
C5	N	N	N	...
C6	N	N	S	...
A1	X			...
A2		X		...
A3			X	...

É importante notar que caso ocorra um contexto que combine diferentes valores para os critérios sem uma regra correspondente, uma ação adaptativa será executada criando uma nova regra que irá compor a TDA.

O Algoritmo I descreve a lógica de manutenção da topologia da RSSF. Ele é executado por todos os nós da rede e implementa a camada de roteamento adaptativa.

ALGORITMO I
Algoritmo para manutenção da topologia da rede de sensores

```
01- Nós CH enviam mensagem "inicio" para a RSSF
02- se nó é CH então
03-    aguarda mensagem de "controle"
04-    se mensagem é do tipo "alerta" então
05-       //** verifica decisão por meio da TDA **//
06-       se critérios existem em uma regra Ri então
07-          executa regra Ri
08-       se não, se critérios existem, mas regra Ri não existe então
09-          adiciona regra na TDA por meio de função adaptativa
10-          propaga nova regra para os outros nós CH
11-       se não, se critério não existe então
12-          adiciona critério na TDA por meio de função adaptativa
13-          propaga novo critério para os outros nós CH
14- se nó não é CH então
15-    se evento ocorreu então
16-       envia mensagem de "alerta" correspondente
```

Para subsidiar a função adaptativa pode-se aplicar pesos aos critérios. Critérios considerados críticos, que demandam uma reconstrução global da topologia da rede de sensores, recebem pesos maiores, enquanto que critérios menos críticos recebem pesos menores.

Na definição de uma nova regra para a TDA deve-se estabelecer uma relação entre a soma dos pesos dos critérios com resposta "Sim" e a soma dos pesos de todos os critérios estabelecidos. O resultado desta relação irá determinar qual ação será executada, sendo que um valor alto corresponde à ação de reconstrução global da topologia. Um valor baixo corresponde à ação de menor impacto (reconstrução local) e

resultados intermediários correspondem respectivamente às ações intermediárias.

C. *Camada supervisora adaptativa*

A tecnologia adaptativa pode ser empregada em diferentes níveis de abstrações. Uma possibilidade é especificar uma camada supervisora, que monitore diferentes aspectos do protocolo de roteamento, diagnosticando e atuando em diferentes contextos.

A Fig. 4 ilustra a arquitetura com a camada supervisora adaptativa. No nível mais baixo estão os nós sensores que compõe a RSSF. Acima deles está o protocolo de roteamento e atuando sobre o protocolo de roteamento tem-se a camada supervisora, que inclui diferentes funções, cada uma monitorando e/ou interferindo sobre um aspecto específico do protocolo.

Figura 4. Arquitetura do sistema com o protocolo de roteamento, a camada supervisora adaptativa e suas funções de supervisão.

O número de funções de supervisão é dinâmico, podendo ser especificadas novas funções à medida que seja necessário. Inicialmente pode-se definir um conjunto de funções para: avaliação dos pesos associados aos critérios (redefinindo o valor do peso); verificação da aplicação das ações (identificar se alguma ação está sendo executada excessivamente); verificação de regiões dominantes (por exemplo, se há áreas que apresentam maior gasto de energia); ajuste na seleção dos nós CH; definição da quantidade ideal de nós CH; ajuste na redefinição das coordenadas da rota do nó coletor móvel; e avaliação da eficiência do protocolo de roteamento, sugerindo por exemplo a inclusão de um novo critério a ser considerado na TDA.

Para implementar a camada supervisora adaptativa deve-se selecionar alguns nós da rede de sensores para atuarem como supervisores. Esses nós irão executar as funções de supervisão. Um nó não deve executar necessariamente todas as funções de supervisão definidas, é desejável que as funções estejam distribuídas nos nós disponíveis.

Os nós sensores que estiverem atuando na camada de supervisão terão autonomia de enviar informações de diagnóstico ao usuário do sistema ou mesmo atuar sobre o protocolo de roteamento, com o objetivo de otimizá-lo.

IV. Conclusões

As RSSF têm apresentado potencial de aplicação em diversas áreas. Um dos seus principais desafios é a economia de energia, sem que isso signifique perda de eficiência na utilização da rede de sensores. Desenvolver protocolos de roteamento eficientes, que promovam a economia de energia, garantam a entrega dos dados, sejam robustos e adequados ao contexto da aplicação, tem sido um desafio na área de pesquisa em RSSF.

Este trabalho apresentou um protocolo de roteamento de dados em RSSF com foco no problema de pulverização em agricultura de precisão. O protocolo de roteamento possui duas fases distintas, a de construção da topologia de rede inicial e a fase de manutenção. A tecnologia adaptativa foi utilizada na fase de manutenção de modo a proporcionar decisões adequadas entre quais ações devem ser executadas pelo protocolo de roteamento para manutenção da topologia da rede de sensores. Foi apresentada a modelagem, a TDA e o algoritmo que implementam a camada adaptativa.

Foi proposto também uma camada supervisora adaptativa que tem a função de monitorar o comportamento do protocolo de roteamento. A camada supervisora adaptativa é composta por diferentes funções de supervisão, cada uma com um foco em específico. Essas funções podem emitir alertas para o usuário ou mesmo atuar sobre o protocolo de roteamento com o objetivo de otimizar seu funcionamento.

Nos trabalhos futuros, a eficiência e robustez do protocolo de roteamento de dados devem ser exploradas, verificadas e mensuradas em um ambiente de simulação computacional. A camada supervisora adaptativa deve ter seus detalhes técnicos para seu funcionamento definidos, bem como suas funções de supervisão. Além de ser verificada sua aplicação e eficiência por meio de simulação computacional.

Agradecimentos

Os autores agradecem a Fundação de Amparo à Pesquisa do Estado de Mato Grosso – FAPEMAT – pelo apoio a este trabalho via projeto de pesquisa e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES.

Referências

[1] I. F. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, E. CAYIRCI. Wireless sensor networks: a survey, *Computer networks*. 38, p.393-422, 2002.

[2] M. TUBAISHAT, S. MADRIA. Sensor networks: an overview. *IEEE Potentials*. 22 (2), 2003.

[3] P. GAJBHIYE, A. MAHAJAN. A survey of architecture and node deployment in Wireless Sensor Network. In: *Applications of Digital Information and Web Technologies*, ICADIWT, p.426-430, 2008.

[4] D. ESTRIN, L. GIROD, G. POTTIE, M. SRIVASTAVA. Instrumenting the world with wireless sensor networks. In: *International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, USA, 2001.

[5] G. J. POTTIE, W. J. KAISER. Wireless integrated network sensors. *Communications of the ACM* 43, p.51–58, 2000.

[6] D. ESTRIN, R. GOVINDAN, J. HEIDEMANN, S. KUMAR. Next century challenges: Scalable coordination in sensor networks. In: *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks* (MobiCom'99), Seattle, Washington, USA, ACM Press, 1999.

[7] J. YICK, B. MUKHERJEE, D. GHOSAL. Wireless sensor network survey. *Computer Networks*, v. 52, n. 12, p. 2292-2330, 2008.

[8] K. SOHRABY, D. MINOLI, T. ZNATI. *Wireless Sensor Networks – Technology, Protocols, and Applications*. Wiley, 2007.

[9] F. ZHAO, L. GUIBAS, *Wireless Sensor Networks – An information processing approach*. Elsevier, 2004.

[10] P. SIKKA, Wireless ad hoc sensor and actuator networks on the farm. Nashville-USA: *ACM*. 2006. p. 492-499.

[11] I. M. SANTOS, M. A. DOTA, C. E. CUGNASCA. Visão Geral da Aplicabilidade de Redes de Sensores Sem Fio no Monitoramento Agrícola no estado de Mato Grosso. Anais do *Congresso Brasileiro de Agricultura de Precisão* – ConBAP 2010. Ribeirão Preto/SP, Setembro de 2010.

[12] T. CAMP, J. BOLENG, V. DAVIES. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications e Mobile Computing* (WCMC), v. 2, p. 483-502, 2002.

[13] BOUKERCHE, A. *Algorithms and protocols for wireless sensor networks*. Ed. Wiley, 2008.

[14] How dangerous is pesticide drift? *Scientific American*. At <http://www.scientificamerican.com/article.cfm?id-pesticide-drift>, 17 de setembro de 2012.

[15] PIGNATI, W. A., MACHADO, J. M. H., CABRAL, J. F. Acidente rural ampliado: o caso das 'chuvas' de agrotóxicos sobre a cidade de Lucas do Rio Verde-MT. *Ciência & Saúde Coletiva*, v. 12, p. 105-114, 2007.

[16] CHAIM, A. *Manual de Tecnologia de Aplicação de Agrotóxico*. Embrapa, 2009.

[17] CUNHA, J. P. A. R., TEIXEIRA, M. M., COURY, J. R., FERREIRA, L. R. Avaliação de estratégias para redução da deriva de agrotóxicos em pulverizações hidráulicas. *Planta Daninha*. 21: p. 325–332, 2003.

[18] HEWITT, A. J. Spray drift: impact of requirements to protect the environment. *Crop Protection*. 19: p. 623–627, 2000.

[19] KWONG, K. H., SASLOGLOU, K., GOH, H. G., WU, T. T., STEPHEN, B., GILROY, M., TACHTATZIS, C., GLOVER, I. A., MICHIE, C., ANDONOVIC, I. Adaptation of wireless sensor network for farming industries. *6th INSS*. p. 1–4. Pittsburgh, 2009.

[20] QIAO, X., ZHANG, X., WANG, C., REN, D., HE, X. Application of the wireless sensor networks in agriculture. *Transactions of the Chinese Society of Agricultural Engineering*. 21: p. 232–234, 2005.

[21] SANTOS, I. M., CUGNASCA, C. E. Pesticide drift control with wireless sensor network. In: *11th International Conference on Precision Agriculture*, Indianapolis. 2012.

[22] J. P. MOLIN, Tendências da agricultura de precisão no Brasil. In: *Congresso Brasileiro de Agricultura de Precisão*, 2004. Anais do Congresso Brasileiro de Agricultura de Precisão - ConBAP 2004. Piracicaba, p. 1-10, 2004.

[23] I. M. SANTOS, M. A. DOTA, C. E. CUGNASCA. Modelagem de Autômato Adaptativo para a definição dinâmica do intervalo de amostragem em Rede de Sensores Sem Fio. In: *V Workshop de Tecnologia Adaptativa*, 2011, São Paulo. 2011.

[24] SOHRABI, K., GAO, J., AILAWADHI, V., POTTIE, G. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5):16–27, 2000.

[25] GONDA, L., CUGNASCA, C. E., NETO, J. J. Uso de tabelas de decisão adaptativas em redes de sensores sem fio. In: *Workshop de Tecnologia Adaptativa*, 4, São Paulo-SP, 2010.

Ivairton Monteiro Santos é graduado em Ciência da Computação pela Universidade Federal de Mato Grosso (2002) e mestre em Ciência da Computação pela Universidade Federal Fluminense (2005). Atualmente é aluno de doutorado da Escola Politécnica da USP, sob a orientação do Prof. Dr. Carlos Eduardo Cugnasca. É membro do Laboratório de Automação Agrícola da Escola Politécnica da USP e professor da Universidade Federal de Mato Grosso, no Campus Universitário do Araguaia. Tem experiência na área de Ciência da Computação e seus interesses em pesquisa concentram-se em Redes de Sensores Sem Fio e otimização combinatória.

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

Semiótica e Tecnologia Adaptativa: Esquema de Comunicação

A. C. F. Matte

***Abstract*— This article discusses the interface between Greimasian Semiotics and Adaptive Technology. The communication scheme of Ignacio Assis Silva is presented as a starting point for rules allowing the automatic generation of dialogues based on the semiotic content. The scheme is appropriated for allowing a dynamic process of communication in which the sender and recipient actants are responsible for constant adjustments, making the system a strong candidate for one adaptive communication scheme.**

***Keywords*— semiotics, communication scheme, dinamic patterns, content, robot, chat.**

I. INTRODUÇÃO

ESTE trabalho traz parte de uma pesquisa em andamento sobre a construção de um bot para atendimento online de professores para uso de recursos educacionais on-line livres, o projeto Livrinho [1].

A Semiótica Greimasiana é uma teoria com importante potencial de aplicabilidade interdisciplinar, como mostram trabalhos em diferentes áreas do conhecimento, especialmente na área das Ciências Humanas. O esquema de comunicação de Ignácio Assis Silva [2], aqui focalizado, é mais um processo que um esquema, pois permite visualizar os deslizes presentes e intrínsecos ao fazer comunicativo, no lugar das estabilidades.

Considerando que a adaptatividade é a capacidade de um programa automodificar-se para atender a situações inicialmente não previstas em suas regras [3], estamos estudando esse esquema de comunicação como base para a introdução da adaptatividade no projeto Livrinho, especialmente no que diz respeito à geração automática de diálogos escritos. O trabalho de Alfenas e Pereira-Barretto [4] é um forte indicativo da produtividade da utilização da Tecnologia Adaptativa para gerenciamento de diálogos.

A capacidade de automodificação, em essência, busca simular a habilidade humana de adaptar-se a diferentes situações a fim de obter um mesmo resultado [3].

Numa situação controlada, como o texto de uma notícia de jornal ou uma fábula, nos quais começo, meio e fim estão dados a priori, a análise semiótica, por sua vez, pode tomar o texto como um todo e, a partir deste todo, realizar sua análise, que acaba sendo, em virtude dessa característica finita previamente selecionada, de natureza discreta mesmo no que tange a elementos contínuos da construção do sentido. A análise de diálogos espontâneos, no entanto, foge a este controle e exige um tratamento não só menos linear da sequência discursiva, como também mais maleável no que diz respeito ao "todo" que define o texto, já que este muda a cada nova intervenção.

Este problema, fascinante para a Semiótica, a nosso ver é o mesmo problema básico da adaptatividade quando trabalha com problemas complexos, com entrada de dados de um conjunto finito mas com infinitas possibilidades de resultados finais a partir de infinitas possíveis relações, já que o tamanho máximo do resultado é ilimitado.

A proposta de Silva transforma o famoso esquema de comunicação de Roman Jakobson [5] em um processo dinâmico, envolvendo o código – no caso, a língua –, o sinal – no caso, a escrita –, e o conteúdo semiótico – no caso, a construção do sentido na conversação no chat –, sendo particularmente adequado para tratar do problema aqui apresentado.

O presente artigo está organizado em 8 partes. Em "II. Semiótica e Tecnologia Adaptativa" busca-se apresentar a viabilidade desse diálogo teórico para pesquisadores de ambas as áreas, por suas semelhanças. "III. Um pouco de história" referenda o tópico anterior com base na história da interface entre a Semiótica e a Inteligência Artificial. "IV. Comunicação pelo viés de Ignácio Assis Silva" discute o esquema proposto pelo pesquisador. O tópico "V. À guisa de metodologia" trabalha no sentido de prover ao esquema de comunicação ferramentas para sua aplicação prática na interface entre os estudos da linguagem e a computação. Os exemplos 1 a 4 (tópico VI) procuram aplicar, de forma breve, essa metodologia, embora ainda incipiente, em textos com diferentes problemas de ambiguidade. Finalmente, o tópico VII apresenta alguns exemplos de análise de chat numa abordagem preliminar do que seria a aplicação adaptativa deste esquema de comunicação. A conclusão traz uma síntese do artigo, com foco na concepção do esquema aqui discutido como proposta para um esquema adaptativo de comunicação.

II. SEMIÓTICA E TECNOLOGIA ADAPTATIVA

A Semiótica Greimasiana sempre esteve no limite entre a linguagem e a tecnologia, limite mesmo da ciência e do fazer científico, sendo uma teoria da linguagem sempre pronta a disputar lugar de destaque nos campos interdisciplinares e na pesquisa de ponta. Sua base estruturalista, embora seja alvo de inúmeras críticas por defensores de teorias concorrentes na Área de Humanas, é, a nosso ver, um dos principais motivos pelos quais esta Semiótica é altamente favorável a estudos interdisciplinares, inclusive com a área de Exatas [6].

Trabalhando com uma separação metodológica entre imanência e manifestação, entre conteúdo e expressão e entre forma e substância, conceitos caros a Hjelmslev [7], a análise semiótica busca apreender o sentido em imanência, nas profundidades, e jogá-lo de forma organizada para a superfície. Seu grande sucesso na literatura e outras artes deve-se ao fato de que sua metalinguagem permite redimensionar o objeto analisado de forma quase tão artística quanto ele próprio foi construído, o que acaba muitas vezes "borrando" a imagem inicial, não porque a Semiótica não possa ser nítida, mas porque o sentido é mais complexo do que aparenta na superfície. Assim, muitos trabalhos de análise semiótica

A.C.F.Matte, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, anacrisfm@ufmg.br

acabam sendo, eles mesmos, quase que novas obras artísticas, no sentido em que a arte, para dar sentido ao mundo, reescreve seus eventos. Disso podem decorrer, e em alguns casos é efetivamente o que acontece, análises que, em escopos teóricos não literários, parecem visões distorcidas desse mundo, como por uso de óculos imperfeitos.

Esse modo de trabalhar com Semiótica, portanto, não é o mais apropriado para trabalhos interdisciplinares, especialmente quando se trata da interface com ciências ditas mais duras. Nesse caso, pensamos, o melhor processo é o que, a partir de um palpite teórico baseado na Semiótica e seguindo a metodologia semiótica à risca, vai desconstruir o texto de forma organizada, esvaziando o texto de suas camadas mais superficiais em busca dos traços, daquilo que gostamos de chamar de caricaturas, pelo caráter mimético e por sua maleabilidade como meta objeto, permitindo que um texto seja compreensível por pessoas diferentes com histórias diferentes e, portanto, por pessoas com diferentes construções do que seja a própria linguagem. Trata-se da mesma ideia que, em 2005, buscou Silva [8] no conceito de boi mínimo, retratado em "Metaforfoses de um touro" por Pablo Picasso: trata-se de buscar a humanidade e a civilidade mínimas, as quais acabam oscilando fortemente entre o inteligível e o sensível, ponto nevrálgico e forte da teoria Semiótica. O tema foi bastante bem abordado por Silva [8], com uma síntese completa no capítulo *Balizas*.

III. UM POUCO DE HISTÓRIA

A relação entre Semiótica e Inteligência Artificial foi primeiramente abordada em alguns trabalhos publicados na série Bulletin (hoje continuada pelo periódico Actes Sémiotiques) na década de 80 em especial dois números dedicados ao tema: "Intelligence artificielle et théorie sémio-linguistique", de 1985 [9], e "Intelligence Artificielle, II: Approches cognitives du texte", de 1986 [10]. Naqueles trabalhos, no entanto, a abordagem do tema foi bastante indireta: em parte porque alguns artigos deslizaram da Semiótica para as ciências cognitivas ao realizar a análise da Inteligência Artificial, e em parte porque, quando deram maior destaque à Semiótica propriamente dita, optaram por focalizar a relação homem máquina que estaria sendo simulada nos artefatos de Inteligência Artificial, de forma incipiente na época.

O acompanhamento da bibliografia produzida desde então indica uma espécie de abandono dessa interface, que nos parece tão produtiva. Este abandono pode ser explicado em virtude da incipiência de ambas as teorias na década de 80. A Semiótica data do início dos anos 70 e, na década de 80, começava a trabalhar com problemas relativos às paixões, enquanto a Inteligência Artificial apenas começava a passar da fase de promessas para uma fase de realizações, ainda limitadas em termos de aplicação prática pelo baixo poder de processamento dos computadores na época. Os dois volumes do Bulletin, citados acima, são reflexo claro desse quadro.

O que nos trouxe a esse palco foi uma pesquisa iniciada no doutorado na USP [11] e continuada em pós-doutoramento na UNICAMP [12] sobre expressão da emoção na fala. Do ponto de vista semiótico, a emoção não é um conteúdo, é uma "expressão comprometida por uma paixão". Para a Semiótica, paixão é um conjunto, passível de moralização, de modalizações e comportamentos de um sujeito, os quais destoam do quadro de valores socialmente aceito, incluindo ódio, amor, desejo de vingança e compaixão, dentre muitas outras. Assim, para explicar emoção com outras palavras: quando uma paixão qualquer afeta um sujeito, sua textualização reflete isso e é esse "afetar" o que define a emoção, a qual é definida como uma perturbação corporal perceptível (voz trêmula, por exemplo). A emoção, portanto, é quantificável e pode ser medida se for observada em relação àquele padrão socialmente aceito, como é o padrão linguístico, por exemplo.

Os principais resultados dessa pesquisa de pós-doutoramento são i) uma proposta de medida do que chamamos Modalização Temporal Tensiva (reflexo da emoção no texto, em qualquer linguagem) [13], ii) um método para identificar e descartar amostras afetivamente comprometidas no sinal acústico, a fim de tornar o corpus adequado ao padrão de referência linguístico [14] e, finalmente, iii) um software para trabalho em fonética acústica, o Setfon [15].

Muitas vezes, durante nosso trabalho na interface com a fonética acústica, em busca de desvelar os mecanismos de produção de sentido emotivo na fala, a Inteligência Artificial foi cogitada como parceira alternativa para os estudos semióticos da comunicação, mas somente no WTA2012 encontramos, na Tecnologia Adaptativa, o tipo de abordagem que, acreditamos, seja o adequado para o tipo de análise que aqui se quer fazer.

A Tecnologia Adaptativa, como comentado na introdução [3], visa à modificação de regras, em softwares, num sistema quase minimalista baseado em operações de inclusão, remoção e consulta. Essa ideia de uma sintaxe básica e abstrata, a qual garante a aplicação da TA a diferentes linguagens e sistemas computacionais [16], é comparável à forma como a Semiótica trabalha sua própria sintaxe, especialmente no nível narrativo, no qual as relações são de natureza lógica. O nível narrativo possui uma estrutura bastante cristalizada, pois foi o primeiro a ser desenvolvido, e a forma de organizá-lo e compreendê-lo, dada essa anterioridade, afeta a forma com que são abordados os outros níveis de produção do sentido.

Respeitadas as diferenças das linguagens-objeto com que trabalham a Semiótica e a Tecnologia Adaptativa, em ambas as teorias a sintaxe mínima aparece e multiplica-se em cada objeto, não só em extensão, como também em diferentes instâncias (níveis ou camadas), aumentando a complexidade do sistema sem, no entanto, causar uma multiplicação desnecessária e indesejável das unidades sintáticas mínimas.

IV. COMUNICAÇÃO NO VIÉS DE IGNÁCIO ASSIS SILVA

A análise da construção do sentido pela Semiótica possui, atualmente, um arsenal de recursos analíticos coeso e suficientemente amplo para dar conta dos mais diferentes objetos e linguagens, no que tange à construção do sentido. Assim, a teoria passou a ocupar-se de problemas para os quais não havia, inicialmente, fundamentação suficiente para sua abordagem, apesar de previstos desde o princípio das investigações semióticas. Trata-se de questões como continuidade, plano da expressão, percepção e, o que nos interessa aqui, a comunicação em processo.

Os esquemas de comunicação, na grande maioria, pecam por manterem-se fiéis àquilo que se propõem ser, pois um esquema não precisa descrever a dinâmica do processo. Um esquema pode ser como uma foto, estática, e a grande maioria dos esquemas de comunicação segue esse estilo [20]. É justamente por explorar o caráter dinâmico do processo

comunicativo que o esquema de Ignácio Assis Silva é uma das opções mais adequadas à análise da comunicação. Neste artigo apresentamos alguns exemplos dessa dinâmica tendo em vista o estudo de regras relativas ao conteúdo do texto para geração automática de diálogos, na interface entre a Semiótica e a Tecnologia Adaptativa.

O esquema de comunicação aqui focalizado (Fig. 1), que parte do conjunto Destinador (formado por Fonte, mensagem como imagem e transmissor) e Destinatário (formado por receptor, mensagem como imagem e destino) foi proposto por Ignácio Assis Silva em sua tese de doutorado, em 1972 [17], e reapresentado por Diana Barros no livro de Introdução à Linguística publicado pela FFLCH/USP [2]. Imagem, aqui, possui o mesmo sentido que "imagem acústica" em Saussure [27] e Mattoso Câmara [23].

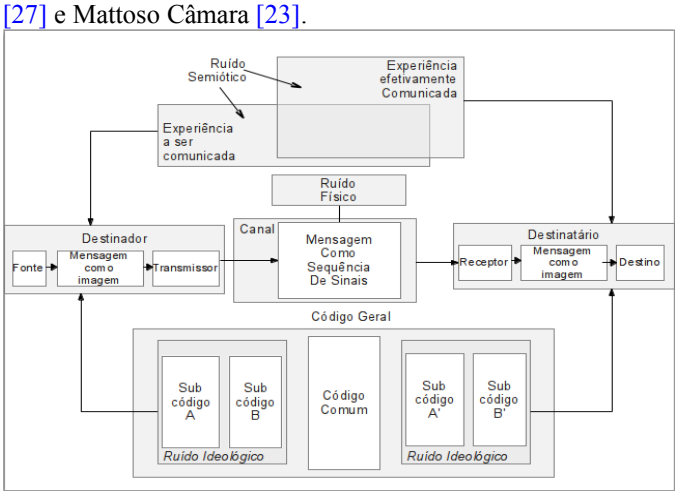


Figura 1: Esquema de Comunicação de Ignácio A. Silva (1972).

Em 2008 [18], analisamos o esquema, passando a nomear as três vias de construção da comunicação: a via do código, a via do sinal e a via semiótica. Naquele artigo, discutimos o conceito de "ruído", que é nada mais do que uma diferença potencial entre o que cada actante do processo comunicativo (Destinador e Destinatário) institui para cada texto em cada uma das vias. O ruído, que já fazia parte da proposta inicial de Silva, é o responsável pela dinâmica do processo: é exatamente porque cada actante sabe que existe um ruído intrínseco em cada uma das vias que a comunicação é possível.

Assim, pode-se afirmar que existem dois sistemas em choque, produzindo um terceiro, que é o da comunicação em si.

O primeiro sistema é do Destinador e é formado pela mensagem que ele deseja comunicar, pelo seu conhecimento do código, pela imagem que faz do código do destinatário e, finalmente, pela capacidade de produzir o sinal necessário no processo de textualização, no caso dos nossos exemplos, produzir a escrita ou a fala e transmiti-la ao Destinatário.

É importante notar que esse esquema, embora concebido inicialmente para a fala, não predetermina nem o tipo de suporte para o envio do sinal e nem a linguagem utilizada. No caso da presente pesquisa sempre estamos trabalhando com a linguagem verbal escrita, o que significa que o código é verbal e o sinal é visual e digitalizado.

O segundo sistema é o do Destinatário, que, sem esperarmos que se trate de um processo linear, recebe o sinal, decodifica-o e interpreta o sinal decodificado. Esse sistema é formado, portanto, pela capacidade de receber o sinal, pelo seu conhecimento do código, pela imagem que tem do código usado pelo Destinador e pela possibilidade de preencher as lacunas inerentes à mensagem recebida.

A *via do código* (a primeira de baixo para cima na Fig. 1) é a via da língua, no caso de linguagens verbais, e o ruído pode aparecer desde sutis diferenças individuais na conceitualização do vocabulário até diferenças entre as línguas dos falantes. O ruído produzido pela diferença entre o que o Destinador pensa ser o sub-código do Destinatário (e que define o padrão linguístico adotado por ele) e o que o Destinatário pensa ser o sub-código utilizado pelo Destinador (que é sua referência para decodificação) é chamado de *ruído ideológico* (Fig. 2) por Silva.

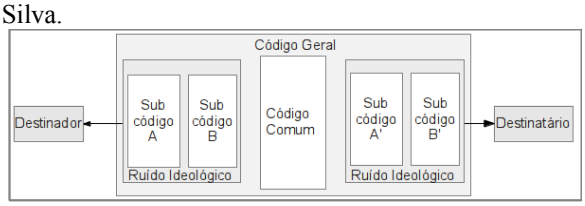


Figura 2: Ruído na via do código, no esquema de I.A.Silva, 1972.

A *via do sinal* (representada como central no esquema) é a do plano da expressão. No caso da língua (linguagem verbal), pode tanto ser sonora (fala) quanto ser visual (escrita). O ruído que afeta essa via é o *ruído físico* (Fig. 3), podendo ser uma má dicção, uma caligrafia problemática, o ruído no telefone e até problemas de conexão causando perda de sinal.

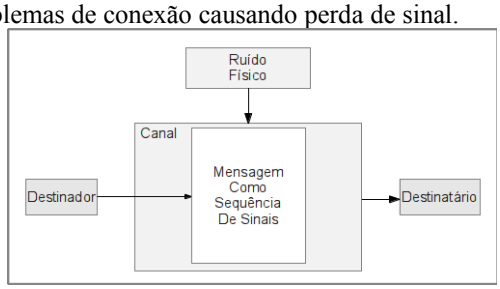


Figura 3: Ruído na via do sinal, no esquema de I.A.Silva, 1972.

Silva, ainda nos primórdios da Semiótica Greimasiana, não chegou a aprofundar a última via, a *via semiótica* (nomeada segundo nossa proposta de 2008 [18]), na qual o elemento chave é a *experiência*. Existe uma diferença intrínseca entre o que se quer dizer e o que se diz de fato, em qualquer processo comunicativo, por inúmeros fatores que não cabem no escopo do presente artigo. Podemos explicar de forma resumida essa diferença intrínseca pelo que decidimos chamar de *lacunas*.

As *lacunas* são inerentes ao processo de comunicação, no que tange à *via semiótica* do esquema. Podemos apoiar essa hipótese num exemplo bastante corriqueiro: como se consegue contar um filme que dura duas horas? Deixando elementos de fora da narração. Escolhemos para omitir os elementos que supomos possam ser facilmente recuperados pelo Destinatário o qual, por sua vez, sabe que sempre receberá uma "pintura incompleta" e precisará "pintar por conta própria" boa parte da "tela" a fim de obter o quadro completo. A Semiótica explica o sucesso deste processo pela existência de uma cadeia de pressupostos e pressuponentes disponíveis como pistas para tal preenchimento nos três diferentes níveis do percurso gerativo do sentido. Isso evita situações problemáticas como um Destinador que decidisse contar um filme em todos os detalhes (além da história, temos música, iluminação, perfil dos personagens, ritmo das cenas, cores, formas, tomadas de câmera etc), levando, assim, não duas horas, mas dois anos

para terminar a narração, o que é humanamente inapropriado, para dizer o mínimo.

Assim, é imprescindível que, para comunicar qualquer experiência, sejam deixadas *lacunas*, não importa a linguagem e nem o suporte utilizados. Pode-se afirmar que parte da própria competência do Destinador é determinada por sua capacidade de escolher as *lacunas* corretas a serem deixadas para ser preenchidas pelo Destinatário.

Ao efeito das *lacunas* sobre o processo de comunicação optamos por designar como *ruído semiótico* (Fig. 4), já que trata do conteúdo da comunicação.

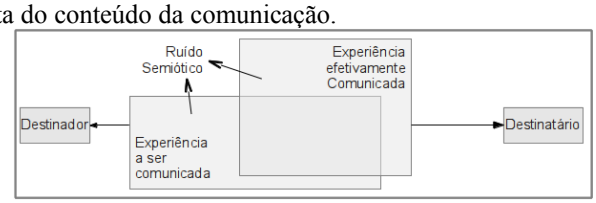


Figura 4: Ruído na via semiótica do esquema de I.A.Silva, 1972.

Em todas as vias, portanto, temos um ruído correspondente intrínseco e até necessário como parte do conjunto esperado:

- *Via semiótica → ruído semiótico*: causado pela diferença entre o que foi omitido na mensagem pelo Destinador e o que foi utilizado pelo Destinatário para preencher as *lacunas* deixadas por essa omissão – intencional ou não (Fig. 4);
- *Via do sinal → ruído físico*: causado pela perturbação provocada no plano da expressão por falhas na transmissão, impropriedade do suporte físico e/ou habilidade limitada dos actantes (Fig. 3);
- *Via do código → ruído ideológico*: causado pela interseção entre dois conjuntos de códigos com um código geral. O conhecimento do código geral por ambos os actantes é o que permite a comunicação, embora sua falta não resulte necessariamente em sua interrupção e o código comum aos dois nunca seja igual à totalidade do código geral; além disso, cada actante conta com uma variante pessoal do código e uma suposição do que seja a variante do outro actante. A intersecção da variante pessoal de cada actante com a variante suposta pelo outro produz o *ruído ideológico* (Fig. 2).

O ruído, em qualquer das três vias, varia em grau e intensidade, desde o imperceptível, quando não causa nenhum efeito importante e, assim, não requer nenhuma adaptação, até o insuportável, quando nenhuma adaptação possível pode restaurar o fluxo comunicativo e a comunicação é interrompida.

Ousamos mesmo afirmar que se trata de um sistema adaptativo por natureza, já que pode ser descrito como um sistema baseado em regras passível de automodificação pela inclusão ou remoção previsível de regras. Transições adaptativas são adequadas a qualquer uma das etapas visíveis na Fig. 1, para além mesmo dos ruídos previstos por Silva em 1972:

1. No Destinador:
   - da "fonte" à "mensagem como imagem";
   - da "mensagem como imagem" ao "transmissor";
2. Do Destinador ao Destinatário pela *via do sinal:*
   - do "transmissor" à "mensagem como sequência de sinais";
   - na relação entre a "mensagem como sequência de sinais" com o "canal" (suporte físico da textualização);
   - da "mensagem como sequência de sinais" ao "receptor";
3. Do Destinador ao Destinatário pela *via do código:*
   - da "mensagem como imagem" ao "código do destinador";
   - da "mensagem como imagem" ao "código do destinatário";
   - do "código do destinador" ao "código do destinatário suposto pelo destinador";
   - do "código do destinatário suposto pelo destinador" ao "código do destinatário";
   - do "código do destinador" ao "código do destinador suposto pelo destinatário";
   - do "código do destinador suposto pelo destinatário" ao "código do destinatário";
4. Do Destinador ao Destinatário pela *via semiótica:*
   - da "mensagem original" à "mensagem com lacunas";
   - da "mensagem com lacunas" à "mensagem preenchida";
   - da "mensagem original" à "mensagem preenchida".

O esquema de Silva pressupõe que qualquer comunicação seja afetada por ruído, variando apenas o grau ou intensidade com que esse ruído afeta o processo. Desse modo, toda comunicação é sempre um processo de adaptação entre os actantes, explicando os ajustes constantes realizados pelas partes envolvidas durante todo o evento comunicativo.

Também é possível explicar, assim, a preferência histórica da Semiótica pelos objetos "acabados", tais como notícias de jornais ou romances, pois um dos actantes (o Destinador) passa a pressuposto, não sendo mais acessível senão por marcas deixadas no texto (marcas da enunciação) e que, enquanto marcas, são estáticas. Desta forma, apenas um dos lados do esquema permanece sujeito às oscilações causadas pelo ruído.

Mas o que nos interessa não é esse tipo de objeto: o objetivo do estudo, cujos pressupostos estão sendo discutidos no presente artigo, é a comunicação em processo, sincrônica e sujeita a ruídos constantes e ajustes provenientes de todos os participantes, sejam eles Destinador ou Destinatário.

Compreendemos o ruído como o espaço para aplicação da Tecnologia Adaptativa no esquema, escolhido para abordar essa comunicação em processo: não é o código ou o sinal ou a mensagem o foco das modificações, mas o desequilíbrio causado pelas duas forças em jogo, a de quem comunica e a de quem interpreta.

V. À GUISA DE METODOLOGIA

Na interface entre a análise textual computadorizada e a análise semiótica, enfrentamos um problema que não pode ser resolvido senão de forma arbitrária. Para o computador, o texto completo não pode ser a unidade mínima. Para a Semiótica, por outro lado, o texto é um todo dotado de sentido, qualquer quebra provocando mudanças no sentido produzido. A fim de buscar um equilíbrio, optamos por dividir o texto em sentenças, pois cada uma possui um sentido

próprio, mas sem perder de vista que esse sentido depende do texto como um todo [19].

Acontece que, mesmo para o esquema de Silva, cada uma das vias pede uma divisão diferente do texto-objeto.

A análise da frase termo a termo geralmente não é a mais adequada para a análise do *ruído semiótico*, pois as lacunas aparecem em trechos muitas vezes maiores do que a frase, compreendendo o texto inteiro. Em outras palavras, sendo o sentido dado no texto como um todo, uma análise termo a termo não seria adequada porque não seria capaz de recuperar a informação total do texto, nem aquela que se quer dizer nem aquela que se pode apreender. O sentido do texto não é dado pela soma de suas partes.

Mesmo se comparadas a via do sinal e a via do código, trata-se de duas unidades mínimas de análise de dimensões bem diferenciadas, seja qual for a linguagem em foco. No caso da língua verbal falada, por exemplo, a unidade mínima do sinal é a sílaba, enquanto, no nível do código, são termos muitas vezes do tamanho de palavras, seja do ponto de vista semântico, seja do ponto de vista sintático.

Como realizar, então, o cruzamento de três vias de comunicação cujas unidades mínimas não coincidem?

Pensamos ser necessário determinar um modelo de análise, o que geralmente é feito pelo caminho indutivo, com uma pesquisa de casos, antes da proposição de um modelo. No entanto, ao se tomar a teoria Semiótica como ponto de partida, essa limitação muda, pois a teoria permite adotar uma postura dedutiva. Propomos que o ponto de vista seja a narrativa, berço das principais lacunas necessárias ao fazer comunicativo e que, por sua natureza lógica, não possui uma ligação direta com a textualização, que é, em última análise, uma junção de escolhas no código que só são manifestadas pela produção de um sinal específico. Ou seja, o nível narrativo seria o mais próximo da imanência e, desta forma, o menos afetado pelas outras duas vias do processo comunicativo, as quais estão intimamente ligadas à textualização.

Temos assim uma possibilidade de análise que vai partir da semiose para a textualização e, da imanência, à manifestação. Esse processo é o processo básico da geração de diálogos, do evento da comunicação em si: a rigor, o algo a ser dito pré-existe ao dito, que está pressuposto pela interpretação do dito.

Uma situação interessante para pensar esse processo seria justamento o oposto. Num jogo de formação de palavras a partir de letras, por exemplo, o sinal existe antes do código e este existe antes do sentido. Será? Semioticamente falando, o primeiro código em andamento é o do próprio jogo, cujas peças e regras criam um sistema cujo sentido final é o sentido de vencedor e perdedor – e, afinal, uma linguagem é um sistema para produzir um sentido. Nesse caso, não existiria nesse exemplo o processo oposto, mas isso não significa que o processo não exista.

Fizemos questão de levantar um aparente exemplo para mostrar uma das características essenciais do esquema processual de comunicação que estamos utilizando: ele baseia-se na premissa semiótica de que o homem é um animal imerso na linguagem e só tem acesso ao mundo por meio da linguagem e é com base nessa premissa que vamos analisar o exemplo seguinte.

As nuvens no céu formam desenhos que ninguém desenhou. Não existe um Destinador, como pensar, nesse caso, em comunicação? Toda a teoria Semiótica baseia-se numa tendência do homem em ver o mundo como espelho, antropomórfico e antropo centrado. Como se pode depreender de Klinkenberg [20], qualquer reta da qual o sujeito apareça como um ponto será, para ele, sempre uma reta que para ele aponta ou dele sai. É um sentido primário da própria narratividade, em que o sujeito é sempre centro.

Assim, não importa se existe ou não um Destinatário das nuvens, se o ator não ocupa o papel de Destinador, ele vai se colocar na outra posição, a de Destinatário, automaticamente, ficando o Destinador, se não aparente ou explícito, simplesmente pressuposto. Então, qualquer configuração que se encaixe, mesmo por acaso, em alguns dos códigos dominados pelo ator "leitor do mundo" (ou códigos nos quais está imerso), vai ser vista como uma manifestação textual e, portanto, manifestação geradora de sentido. É assim que a criança enxerga cachorros e flores em milho de pipoca estourado, de forma aparentemente aleatória.

Em primeiro lugar, é necessário lembrar que, embora o esquema de comunicação de Silva seja adequado a qualquer linguagem, para os fins do presente trabalho a linguagem é sempre verbal, na modalidade escrita.

A cadeia de entrada, portanto, é texto escrito digitalizado. Já dispomos de um sistema para realizar essa segmentação: o módulo de preprocessamento morfossintático do DadosSemiotica [21], software no qual um processo de análise e síntese será realizado nas etapas posteriores ao trabalho apresentado neste artigo [22], das quais temos uma amostra preliminar no tópico VII.

As unidades dessa fita são definidas como termos. A super segmentação em sentenças, necessária à análise da *via semiótica*, segue o padrão morfossintático definido em [19]. A análise linguística, baseada no mesmo padrão, produz uma segmentação em palavras para determinar as unidades mínimas para a análise da *via do* código e esta é, segundo a hipótese atual, a unidade mínima da fita de entrada para todas as análises. Uma subsegmentação silábica pode vir a ser necessária para a análise da *via do sinal*.

A *via semiótica* sempre trabalhará com unidades iguais ou maiores do que a sentença, o que não impede que as análises, como veremos, busquem, nas palavras, pontos de ruptura produtores de sentido. A análise desta via vai muito além do que seria pertinente analisar no escopo do presente artigo, buscamos estão uma análise simplificada apenas para ilustrar alguns dos muitos pontos de ruptura geradores de ruído.

A análise do sinal pode ter, como ponto de partida, a escrita fonológica. O programa Setfon [15] contém um método (baseado em [28]) para transformação do texto escrito em escrita fonológica e produz uma segmentação vowel-to-vowel (V-V), necessária à análise fonético acústica a que o programa se propõe, mas que pode ser adotada como base para a subsegmentação e análise do sinal para os propósitos desse trabalho, já que a notação fonológica adotada, baseada em Mattoso Câmara [23] [24] [25], trabalha com metassegmentos e, portanto, não considera variações da manifestação acústica, mas, sim, as possibilidades previstas para essa manifestação, em bloco.

A metodologia aqui proposta visa encontrar o ponto passível de gerar ruído e, portanto, o ponto em que uma regra da análise da cadeia de entrada pode ser alterada tendo em vista a adaptação do sistema aos atores e mensagens envolvidos. Nos exemplos a seguir apresentamos, para cada um, uma análise trecho a trecho da cadeia de entrada e um esquema mostrando em qual das vias a produção do ruído gera necessidade de ajuste.

VI. EXEMPLOS

Tomemos a frase “Ontem ele apagou a velhinha” (Fig. 5).

	Ontem	ele	apagou	a velhinha	.	
Código	Advérbio	Sujeito	Verbo passado	Objeto	Ponto final	
Sinal	/oNteIN/	/elI/	/apagoU/	/avelinhA/	// (prosódia: terminativo)	
Semiose	Passado recente	Outrém	Dupla possibilidade isotópica (assassinato ou aniversário?)	Apagamento da velinha Assassinato da velhinha	Declaração terminada.	

Figura 5: Leitura da sentença "Ontem ele apagou a velhinha" segundo as três vias de comunicação do Esquema de Silva.

A Fig. 6 mostra as três análises necessárias: a análise do código, a análise do sinal e a análise da semiose (como sentido produzido ou passível de ser produzido). A análise morfossintática apresentada para a *via do código* está simplificada, pois o processamento gera um resultado para a sentença com elementos em árvore [22]. No caso desta sentença, o resultado é:

(S (ADVP (adv-*-ontem Ontem)) (SUBJ (NP (pronpers-M=3S=NOM-ele ele))) (P (VP (vfin-PS=3S=IND-apagar apagou))) (ACC (NP (art-F=S-o a) (n-F=S-velho velhinha))) (PUNCT .))

Note que cada nome simplificado da análise do código está definido como um ramo na análise morfossintática:

- advérbio: (S (ADVP (adv-*-ontem Ontem))
- sujeito: (SUBJ (NP (pronpers-M=3S=NOM-ele ele)))
- verbo passado: (P (VP (vfin-PS=3S=IND-apagar apagou)))
- objeto: (ACC (NP (art-F=S-o a) (n-F=S-velho velhinha)))
- ponto final: (PUNCT .))

A análise do sinal pelo Setfon resulta nos segmentos:

- o'Nt
- eN
- e'l
- e
- Ap
- Ag
- o'U
- a'v
- elh
- i'nh
- A

Optamos por incluir um ponto na sequência de entrada para deixar determinados os subsegmentos V-V dentro do conjunto composto por eles na expressão.

A análise da *via semiótica*, simplificada neste artigo, mostra uma possível bifurcação no que tange ao sentido do verbo apagar. Trata-se de uma bifurcação isotópica (com respeito a temas e figuras e altamente vinculada à semântica das palavras) que indica que o modelo seria bem representado por uma árvore de decisão pois, dependendo da escolha feita nesse elemento, determina-se um ou outro efeito de sentido.

Semiose é um processo, sinal e código são conjuntos de estados e regras que estão em processo. Nesse exemplo, se a pessoa interpretar a isotopia do aniversário, ela terá entendido algo parcialmente diferente do que foi dito, por mais significante que seja essa diferença. Ao deparar-se com o termo seguinte (velhinha), o leitor pode ter duas reações: perceber a incongruência da escolha realizada e modificá-la, voltando um passo atrás, ou realizar uma leitura em bloco e ignorar a ortografia que diferencia velhinha de velinha (se fosse fala, não haveria diferença perceptível).

No esquema de Silva temos um ruído semiótico passível de acontecer em função de uma homofonia que levaria à leitura em bloco (e não letra a letra) do termo “velhinha”, trocando-o por velinha (Fig. 6).

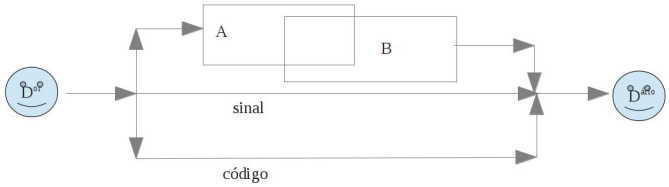


Figura 6: No exemplo 1, o ruído acontece pelo desajuste entre a mensagem a ser comunicada e a mensagem efetivamente comunicada, na via semiótica do esquema.

A figura 6 mostra a intersecção parcial entre os conjuntos A e B, sendo A o que o Destinador queria dizer (contexto de assassinato) e B o que o Destinatário pode entender (contexto de aniversário). Sempre que um Destinador fala algo, ele deixa algumas lacunas (não ditos no canal semiótico), presumindo que o Destinatário possa preenchê-las corretamente. Nesse caso, ele não explicou que o contexto específico era o de um assassinato e permitiu ao Destinatário fazer uma escolha diferente de isotopia para a palavra “apagou”. O exemplo permite perceber que a lacuna mal preenchida no processo de semiose foi causada porque o sinal continha uma quase homofonia (também permitindo a escolha errada) e porque no código nenhuma das escolhas causaria estranheza, pois tratam-se de cadeias aceitas tanto no código que o Destinador presume ser o do Destinatário, quanto no código que o Destinatário presume ser do Destinador.

A frase do exemplo 2 é uma sentença dita por uma criança em fase de aquisição de fala e anotada informalmente: “Um auau!”.

A cadeia de entrada para a Semiótica precisa ser composta por elementos sintáticos com sentido próprio. Por isso “Um auau” pertence à mesma célula da fita (Fig. 7).

	Um auau	!	
Código	Demonstrativo/objeto	Ponto exclamação	
Sinal	/uNaUaU/	// (prosódia: terminativo exclamativo)	
Semiose	Adulto: cachorro Criança: classe mais ampla	Declaração terminada com emoção.	

Figura 7: Leitura da sentença "Um auau!", conforme as três vias de comunicação.

Sendo quem fala um adulto (Destinador) e quem escuta uma criança (Destinatário) (poderia ser o contrário), o subconjunto correto é menor para o adulto do que para a criança. Daí o espanto esperado no caso do adulto observar a criança chamando de “auau” a um cavalo, caso em que a lacuna, do ponto de vista do adulto, teria sido mal preenchida (Fig. 8).

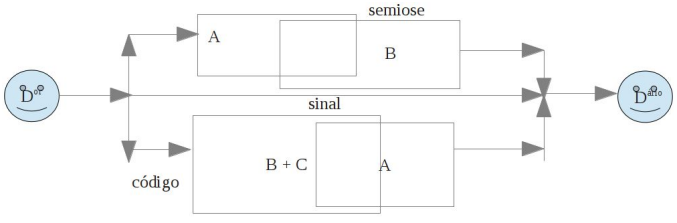


Figura 8: Esquema para a sentença "Um auau!". O ruído afeta duas das três vias de comunicação.

Nesse caso, a lacuna foi mal preenchida porque o vocabulário de um não corresponde ao vocabulário do outro: o ruído que causou o erro de interpretação veio do código. Para o adulto, "auau" corresponde ao conjunto:

A={cachorro}={quadrúpede, rabo, pelo, tamanho limitado, latido}

enquanto para a criança, "auau" corresponde a um conjunto de características menos específico:

B={quadrúpede, rabo, pelo, qualquer tamanho, qualquer som}

O cavalo corresponde a C para o adulto:

C={quadrúpede, rabo, pelo, tamanho grande, relincho}

Desse modo, cavalo não pertence a A, mas pertence a B.

A palavra "auau", portanto, existe no código comum, mas tem sentidos diferentes nos subcódigos do Destinador e do Destinatário.

É importante notar que se trata de dois pontos de geração de ruído: o semântico (no código, no sentido da "palavra") e o semiótico (no sentido do texto).

O próximo exemplo tem como base um relato de experiência de primeiro contato com computador, no qual a sentença "Aperte o botão para enviar", dita pelo instrutor, causou o ato de apertar o botão (hardware) para desligar a máquina.

A reação esperada para "Aperte o botão para enviar" é, por meio da movimentação do mouse, levar o cursor que aparece na tela até a posição do link para o comando (enviar), representado na forma de um botão de máquina, e clicar sobre ele com outro movimento do mouse. Ou seja, não é só o botão que não foi reconhecido pelo usuário leigo, mas também a representação de apertar ou pressionar que, nesse caso, é uma metáfora para "clicar".

Por isso, trata-se novamente de um ruído gerado em dois pontos de não coincidência: o semântico (sentido da palavra "botão", ruído ideológico) e o semiótico (o sentido da sentença na relação entre os sentidos de software e hardware, ruído semiótico).

O último exemplo busca raízes na teoria semiótica, explorando um pouco mais a *via semiótica*. A sentença do exemplo 4, "Subiu a escada voando.", tem como destinador e destinatário dois adultos, falantes de língua portuguesa, letrados e sem qualquer tipo de dislexia.

Ou seja, nenhum tipo de ruído é esperado nem na via do sinal (a não se que algum incidente de natureza externa afete o sinal; por exemplo, a frase for escrita na areia e uma onda a apagar parcialmente), nem na via do código.

Mesmo na via semiótica não é esperado um ruído comprometedor, mas o exemplo serve para compreender um outro tipo de ruído, sempre presente, cuja intensidade pode eventualmente afetar a comunicação.

- "subiu a escada" → tempo passado, direção de baixo para cima, alçar degraus, mudança de estado
- "voando" → aspectualização temporal correspondendo a velocidade.

Enquanto "subiu a escada" significa um percurso que poderia ser descrito como: "apoiou-se num dos pés, colocou o outro pé no primeiro degrau da escada, passou o peso para ele, levantou o pé liberado do peso para o degrau seguinte e assim por diante até alcançar o topo da escada", "voando" indica que isso foi feito rapidamente.

No entanto, voar pode significar algo completamente diferente. Se o sujeito de "subiu" for um bruxo, ou um super-herói, por exemplo, "voando" pode significar "sem tocar os degraus da escada". Assim, mesmo que a frase estivesse sendo lida num contexto de história fantástica e o personagem voasse o tempo todo, um leitor menos atento ao contexto poderia considerar que o conteúdo da expressão "subiu a escada" definia o ato de tocar os degraus e se sobreporia às capacidades sobrenaturais do personagem.

Se o texto for um romance, como comentamos acima, caso o leitor realize interpretações inadequadas, espera-se que esses ruídos sejam corrigidos durante a leitura, mas se o texto é uma conversa num chat, o ruído só será suavizado por alterações e verificações realizadas durante a troca de mensagens pelos participantes, sentença a sentença. Se no primeiro caso a compreensão é dada pelo texto (romance) como um todo dotado de sentido, no segundo caso o todo dotado de sentido é um processo e não um produto final, de modo que, a cada nova intervenção, todo o sistema está sujeito a alterações e os participantes podem, inclusive, reinterpretar outras sentenças escritas antes daquela reveladora do ruído. No exemplo abaixo, tomado do próximo tópico, a consulta feita por *Papagaio*, embora não tivesse o objetivo de verificar o gênero do nick *abelha*, permitiu a identificação do ruído e consequente adaptação do sistema, possibilitando a reinterpretação de toda a interação anterior ao trecho e modificando o sentido de sua continuação.

1. [12:41:26] <Papagaio> Oi abelha, quem é o abelha?
2. [12:42:12] <Gato> Papagaio, eh uma amiga do linux

Dito desta forma há pouca diferença entre participar de um chat ou ler um romance; a grande diferença está no fato de que o chat pressupõe constante troca de papéis entre Destinador e Destinatário e isso, se por um lado torna a análise semiótica mais complexa, por outro lado pode-se dizer que viabiliza a geração automática de diálogos.

Além disso, há que se considerar que, quando um dos atores é um robô, a relação de adaptação e troca de papéis é a mesma do chat [26], provavelmente com maior fidelidade num sistema inteligente adaptativo, de modo que a atuação de ambos atores pode e deve ser tratada da mesma forma, como comunicação em processo.

VII. ANÁLISE PRELIMINAR DE IDENTIDADES NO CHAT

Para concluir, apresentamos uma breve análise de identidades .

O *corpus* foi obtido num chat do IRC [30], na rede Freenode [31], com logs coletados pelo cliente de IRC Konversation, para Linux (http://konversation.kde.org/). Para manter a privacidade dos usuários, optou-se por trocar seus nicks por nomes de animais (no caso de nicks compostos em que uma parte foi utilizada anteriormente, optou-se por manter o nome do animal ou parte dele, conforme feito originalmente, acrescentando o aposto original, para não prejudicar a lógica da troca escolhida pelo usuário.

Os dados foram coletados em 4 dias diferentes, não sequenciais, somando 81 horas e 41 minutos de registro. Do número total de entradas (312), 268 correspondem a entradas de texto pelos usuários (a que chamamos "fala") e 28 são notificações de mudanças no nick (troca de nick, entrada e saída), sendo o restante relativo a outros tipos de notificações características deste protocolo de chat. Utilizou-se o programa Dadossemiotica [21] para organizar as análises; o Módulo de Preprocessamento Morfossintático provido por este software dividiu as entradas de texto que continham pontuação e outros indícios de final de sentença, totalizando 348 sentenças para a

análise, das quais 305 são sentenças pertencentes a dados entrados pelos usuários ("fala").

O texto foi tratado pelo Módulo de Chat, que forneceu, dentre outras coisas, informações sobre a demora de uma entrada em relação à entrada anteriormente registrada, em minutos, tipo de notificação e nick. A análise manual marcou:

- *nick atual*: o nick do responsável pela entrada de texto em questão ou o nick sobre o qual refere-se a notificação de entrada, saída ou troca de nick. Se for notificação de saída, recebe valor nulo;
- *nick anterior*: o nick que o mesmo sujeito tinha na sua última interação; em caso de notificação de entrada, recebe valor nulo;
- *Principal lacuna*: a partir da análise de outras categorias, também manuais (via do *sinal*, via do *código* e via *semiótica*), cada sentença foi classificada conforme sua principal lacuna (a com maior probabilidade de provocar um ruído relevante para a comunicação) fosse código, semiótica, sinal ou ruído improvável (um cumprimento "oi", por exemplo, no contexto de entrada de um nick, foi considerado como ruído improvável).

A identidade no chat também é indicada pela citação de nicks pelos usuários, mas, dado o escopo estrito deste trabalho, estas citações não foram abordadas.

O excerto de chat contou com 12 nicks diferentes, conforme o gráfico da Fig. 9. O código numérico é gerado também automaticamente pelo programa: o DadosSemiotica faz esta conversão para, rodando o R (http://cran.r-project.org/) em background, calcular alguns dados descritivos (média, mediana, desvio padrão e variança) e gerar um histograma da categoria especificada. A relação informada pelo programa sobre a conversão dos nicks para números foi: 1) abelha, 2) "", 3) Gato, 4) Papagaio, 5) Lobinho, 6) vespa_amarela, 7) abelha_away, 8) Papa_Ja_Volta, 9) Papa_Voltou, 10) Papa_foi_de_novo, 11) Papa_voltou_de_n, 12) gato e 13) Formiga.

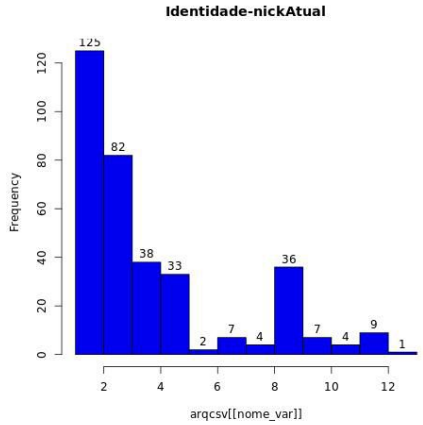


Figura 9: Distribuição das amostras segundo o nick utilizado no momento da interação.

Foram registradas 7 entradas de nicks, 7 saídas e 14 trocas, nas 81 horas de registro, de modo que se pode concluir que é um chat frequentado por poucos usuários e a observação dos dados mostra que a maioria permaneceu on-line por muitas horas. Papagaio utilizou 5 nicks diferentes durante o registro, todos eles composições do nick inicial. Gato utilizou o mesmo nick com letra inicial minúscula e abelha também utilizou um segundo nick, composição do primeiro.

O nick mais ativo foi *abelha*, que, somando-se sua entrada com o nick composto *abelha_away*, totalizou 129 intervenções. Os 3 nicks mais utilizados em seguida foram *Gato* (38 entradas), *Papa_Ja_Volta* (36 entradas) e *Papagaio* (33 entradas). A soma das entradas dos nicks utilizados por *Papagaio* é de 56 entradas e as entradas do Gato (somadas a gato) totalizam 47 manifestações.

Esta amostra apresenta baixa correlação (0,06, método de Pearson) entre o tempo de resposta e o tipo de lacuna principal; como se trata de dados sincrônicos mas cuja leitura e resposta pode acontecer aleatoriamente (já que os participantes em geral não estão conectados tendo como objetivo principal sua participação no chat, mas outras atividades on-line e off-line), esse resultado é previsível.

A análise destes dados será feita considerando-se o esquema de comunicação, tendo como objetivo observar a construção de uma *identidade no chat*, tema para o qual este trabalho apresenta-se como uma pesquisa piloto.

A identidade no chat não é simplesmente um nick: trata-se de uma estrutura complexa, composta principalmente por um ou vários nicks, um estilo de escrita, tipo de interação, assuntos preferidos e frequência de acesso e participação. Podemos pensar essa construção como um autômato que, a cada nova ocorrência, pode adaptar-se e modificar-se. Para cada participante do chat, essas construções são diferentes, pois cada um tem uma experiência particular determinada, inclusive, pelas características mesmas de sua própria identidade no chat. Por esse motivo, é adequado pensar que, para análise da construção dessa identidade, é necessário arbitrar um ponto de vista, ou seja, escolher um dos participantes como observador da evolução da identidade dos outros participantes do chat. Este participante pode ser um chatterbot, por exemplo, que colheria os dados do chat, limitado à sua presença on-line; são os limites desta presença que delimitam o contexto de criação das identidades: em outras palavras, não se levanta hipóteses sobre a identidade de um determinado nick se ele só entrar na sala de bate-papo quando o participante observador não estiver nela.

Semioticamente falando, diríamos que só existe um D^{or} se ele ocupar esse papel para um D^{ário}, numa existência de dependência recíproca. Assim, o esquema de comunicação só opera quando houver tal condição satisfeita. Considerando-se esta premissa, pode-se assumir que o estado zero do chat passa a existir quando ele é iniciado para o participante que, doravante, chamaremos de Observador.

Sua entrada no chat é registrada pelo programa Konversation da seguinte forma:

[qua 25 abr 2012] [12:37:54] Entrada (~nickObservador@unaffiliated/nickObservador) juntou-se ao canal #canal .

[qua 25 abr 2012] [12:37:54] Tópico O tópico do canal é Canal do grupo XX fale de tudo e todos ao mesmo tempo! Eletrônica, política, carros, programas....

[qua 25 abr 2012] [12:37:54] Tópico O tópico foi definido por Lobinho em 06-10-2011 20:16.

[qua 25 abr 2012] [12:37:54] Modo Modos do canal: F, nenhuma cor permitida, não receber mensagens de fora, proteção de tópico

[qua 25 abr 2012] [12:37:54] Criado Este canal foi criado em 06-10-2011 19:14.

[qua 25 abr 2012] [12:38:07] URL URL do canal: http://wwww.site.xxx

Note que estas linhas são visualizadas pelo Observador na interface de acesso do programa quando de sua entrada no canal. Ou seja: o primeiro registro acessado pelo Observador é exatamente o de sua própria entrada no canal (Fig. 10).

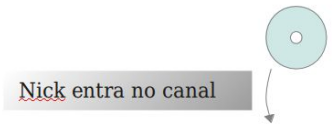


Figura 10: Estado zero.

A partir de então, cada nova entrada de texto no chat, seja pelos usuários, seja na forma de uma notificação do sistema, corresponde a uma mudança nesse estado, o qual vai, neste modelo, bifurcar-se com a criação de um clone a cada nova identidade com a qual o Observador interagir. O número de variáveis que afeta o desenvolvimento de cada clone é imprevisível, mas vamos procurar aqui nos ater apenas a uma delas. Temos três tipos de ocorrências nesta amostra, com :

1. Entrada e saída do usuário sem troca de nick:
   - [qua 25 abr 2012] [22:23:04] Entrada Formiga juntou-se a este canal (~Formiga@IP.provedor).
   - [qua 25 abr 2012] [22:33:40] Sair Formiga deixou este servidor (Ping timeout: 260 seconds).

2. Troca de nick para nick composto e vice-versa:
   - [qua 25 abr 2012] [14:27:08] Apelido Papagaio está conhecido agora como Papa_Ja_Volta.

3. Troca de capitalização no nick, sem mudança do nick, só registrado neste *corpus* durante um processo de saída e entrada:
   - [qua 25 abr 2012] [16:50:53] Sair Gato deixou este servidor (Quit: Fui embora).
   - [qua 25 abr 2012] [19:24:37] Entrada gato juntou-se a este canal (~gato@IP).

Ocultamos os dados de IP e provedor para proteger a identidade do informante. Para fins do presente artigo, vamos nos ater ao primeiro tipo de ocorrência, com o trecho inicial da amostra (nota-se que a última notificação de início de chat, citada acima, ocorreu às 12:38:07, um segundo antes da primeira manifestação dos usuários). A data foi retirada dos trechos seguintes porque os participantes só vêem a hora, não a data, durante o uso do chat; a data fica registrada apenas pelo programa nos logs. As manifestações foram numeradas apenas para referência no presente artigo:

3. [12:38:08] <Gato> eee
4. [12:38:15] <abelha> :)
5. [12:39:09] <Gato> abelha, como esta o seu projeto?
6. [12:40:44] <abelha> qual deles, Gato?
7. [12:41:26] <Papagaio> Oi abelha, quem é o abelha?
8. [12:42:12] <Gato> Papagaio, eh uma amiga do linux
9. [12:42:22] <Gato> nao participa da lista
10. [12:42:23] <abelha> oi, Papagaio
11. [12:43:17] * abelha odeia referencias bibliográficas... escrevendo uma ementa :/
12. [12:43:36] <Gato> abelha, o projeto dos servidores
13. [12:43:55] <Gato> lembra q eu te mostrei um link sobre o open hardware
14. [12:44:01] <Gato> desculpa
15. [12:44:04] <Gato> open compute
16. [12:44:05] <Lobinho> putz neutro travestido de terra, dificil falar a linguagem da nbr5410 hein..
17. [12:44:53] <Lobinho> boa tarde abelha !!
18. [12:45:08] <abelha> ah, Gato, ta parado, tivemos que parar pra focar num outro projeto mais urgente... pior que daí veio uma avalanche de coisas mais urgentes pegando carona :/
19. [12:45:15] <abelha> oi, Lobinho :)
20. [12:45:49] <Lobinho> :)

No excerto acima, o Observador (*abelha*) é conhecido de um dos integrantes do chat (*Gato*), que o chamou para aquele canal a partir de outro no qual conversavam, portanto este aguardava a entrada daquele no canal. Caso *abelha* entrasse num canal desconhecido em que os participantes não esperam sua chegada, a manifestação 1 jamais seria compreendida por *abelha* como sendo uma recepção para si mesmo. Provavelmente seria recebida como se sua entrada tivesse ocorrido no meio de uma interação entre os outros participantes. Nesse caso, o *emoticon* (manifestação 2) com o qual *abelha* respondeu ao *Gato* teria outro sentido: enquanto aqui "eee" é uma manifestação de boas vindas e o emoticon é um agradecimento, no outro contexto possível esse *emoticon* não seria uma resposta ao "eee", mas um cumprimento simpático, semelhante a "oi, pessoal".

Vamos, portanto, limitar, a duas situações como possibilidades para a interpretação desta brevíssima interação em momento inicial do chat: A) *abelha* entrou convidada por *Gato*, ambos frequentadores de um outro chat e B) *abelha* não conhecia ninguém no chat onde entra pela primeira vez.

Antes de ir para a situação real do chat (cuja compreensão é possível a partir do próprio trecho acima, apesar das informações serem insuficientes para definir o "lugar" de onde vieram *abelha* e *Gato*, podendo se tratar até mesmo do espaço de uma única sala física na qual ambos acessassem o chat por computadores diferentes), vamos refletir sobre a opção B.

A expressão "eee" é insuficiente não só para contextualizar um assunto, mas também para definir seu próprio sentido. Isolada, não tem sentido. Poderia ser dita com ironia, sarcasmo, alegria, decepção, dentre outras possíveis paixões. Dado que pode estar veiculando uma mensagem tanto negativa quanto positiva, é até menos elucidativa que o emoticon com o qual *abelha* lhe responde (um sorriso, se não positivo, no máximo uma ironia, nunca uma decepção). A título de exemplo de análise das categorias de comunicação, para esta sentença foi indicada, como maior lacuna, a semiótica, pois, a despeito da inegável lacuna no código trazendo consigo possibilidade de ruído ideológico, a vagueza da expressão, com pouquíssima informação, cria uma grande possibilidade de que a experiência que se deseja comunicar seja substancialmente diferente da experiência comunicada, como explicado a seguir.

O Observador (na situação B, em que não é esperado), ao entrar no chat e se deparar com esse "eee" imediato, depara-se com um conjunto de possibilidades: a irrupção abrupta e sem contexto de uma expressão com inúmeros e contraditórios sentidos possíveis devido a um ruído no canal do sinal, sua própria aparição inesperada no meio: havia uma conversa antes? Se havia, qual o assunto e qual a posição de *Gato* a respeito? O canal estava silencioso demais e *Gato* ficou feliz com a entrada de um possível interlocutor? *Gato* é simplesmente alguém que gosta de escrever coisas sem sentido no canal? Nenhuma dessas possibilidades (e mesmo muitas outras aqui não elencadas) poderia ser descartada por *abelha* apenas com base na manifestação 1 ("eee").

Nota-se que o foco de atenção é o *Gato*: é sua identidade que está iniciada para *abelha*, num estado inicial quepodemos chamar de estado 1: ao manifestar-se, *Gato* estimula a geração de uma identidade-Gato (I^{Gato}) para o Observador. Isso leva a uma importante constatação: no escopo deste trabalho, e com base na teoria semiótica das paixões, *a identidade é*

construída a partir do próprio sujeito, formada por todo o processo desde a constituição do sujeito semiótico, sendo parte do papel do Observador apenas a etapa de Moralização [29].

Durante o trecho citado acima seriam, portanto, criados mais dois clones identitários ($I^{papagaio}$ na manifestação 5 e $I^{Zebrinha}$ na manifestação 14) para o Observador. É importante notar que o Observador também possui uma identidade no chat, cuja interação com as outras identidades tem potencial de modificação e sua identidade pode modificar-se a partir dessa interação, mas ela possui um estatus privilegiado já que, como ponto de referência, não precisa explicar-se a si próprio: isso fica ainda mais claro se admitirmos que o Observador poderia ser um robô. Por isso usamos uma denominação diferenciada, chamando a identidade-abelha de O^{abelha}.

Em termos da adaptatividade, trata-se de uma situação que requer do O^{abelha} uma verificação a fim de permitir diminuir o número de possíveis sentidos para a expressão "eee". É um problema relativo ao código, determinado pelo ruído físico, que, ao eliminar o contexto, provocou uma grande lacuna na via do código, prejudicando a transmissão da experiência. Manter o sentido em aberto impediria a comunicação, excluindo O^{abelha} do diálogo em curso, pelo menos até que o número e tamanho das lacunas fosse aceitável. Aqui já temos o desenho possível da identidade em forma de clone (Fig. 11).

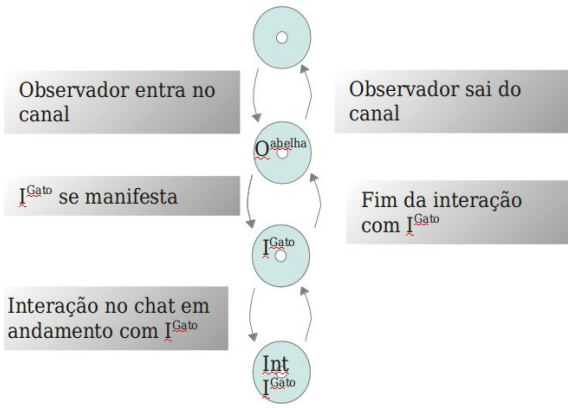


Figura 11: Mapa da interação um a um no chat, desde o início dos registros pelo observador, com um único interlocutor.

Não cabe aqui insistir nas inúmeras possibilidades decorrentes de tal situação, já que o objetivo é analisar a ocorrência realmente manifestada neste trecho, a opção A, em que *Gato* convidara *abelha* a conhecer o canal no qual se desenvolve a interação. É importante apontar, no entanto, que essa mudança de perspectiva carrega a manifestação 1 com um sentido específico, diminuindo o ruído físico a ponto da resposta, expressa pelo O^{abelha} na manifestação 2 (o *emoticom*), seja não só adequada como, também, tenha sentido suficientemente restrito. Desse modo, a diferença entre a experiência a ser comunicada é muito semelhante à experiência efetivamente comunicada para cada um dos interlocutores. Em outras palavras, a eficiência da comunicação está diretamente ligada à desambiguação, que requer limites claros para a leitura.

Nas manifestações 3 e 4, I^{Gato} e O^{abelha} acrescentam, para qualquer participante do chat, a informação sobre esse contexto prévio que os une. Assim, do ponto de vista desse observador, as identidades $I^{papagaio}$ na manifestação 5 e $I^{Zebrinha}$ na manifestação 14 são criadas do zero (Fig. 12), mas O^{abelha} e I^{Gato} já estão mais desenvolvidas, o que só pode ser depreendido de uma análise do conteúdo da interação de 1 a 4.

Nas manifestações 5 a 7, a identidade de $I^{papagaio}$ é gerada para o O^{abelha} mas, simultaneamente, é incrementada no chat a identidade O^{abelha} com duas informações: é uma amiga de *Gato*, de uma comunidade de linux e não está na lista. É curioso que *abelha*, embora possivelmente não soubesse nada sobre a lista, não tenha perguntado "qual lista" (como fez para o projeto). Sua manifestação 9 mostra que ela não responde à vagueza relativa à "lista", mas responde à negatividade em si desta afirmação "não está na lista", mostrando saber usar os recursos do chat (no caso, um /me que ecoa a mensagem diferentemente das mensagens simples que todos estão usando, usado para expressar pensamentos ou dar destaque em algo, uma frase sobre si mesmo pois começa sempre com o nick de quem deu o comando), ou seja: mostra ser novato no canal mas não novato no protocolo de chat usado. E *Lobinho* usa estratégia semelhante: antes de cumprimentar o recém chegado, fala de um assunto que o define no canal e fala isso de forma casual, como alguém que: a) sabe do que está falando e b) não é novato no canal.

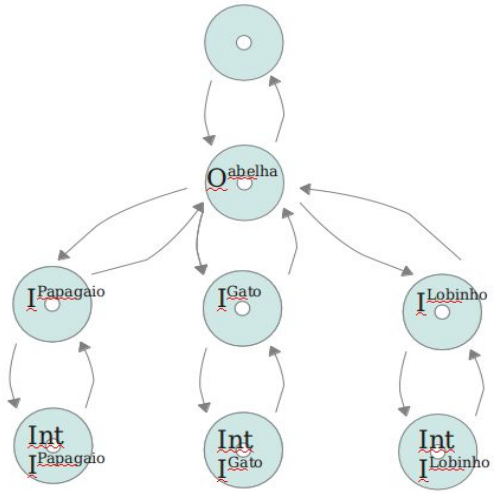


Figura 12: Mapa das identidades dos 3 participantes ativos, do ponto de vista do Observador.

Essas informações incrementam cada uma das identidades no canal para O^{abelha} podendo ou não acionar ações adaptativas. Neste pequeno trecho, a interação efetiva em relação aos temas tratados é, de fato, bem pouca: somente acontece entre *Gato* e *abelha*, visto que as outras manifestações sobre assuntos que poderiam render longas discussões (isotopia acadêmica na manifestação 9, isotopia técnica de elétrica na manifestação 14) não são desenvolvidos. Uma informação, porém, acarreta uma ação adaptativa que só será sentida em outros momentos do chat: na manifestação 5, *Papagaio* se refere a *abelha* usando o gênero masculino, o que é imediatamente corrigido por *Gato* na manifestação 6. No ambiente de IRC o gênero predominante é o masculino, assim ele é a premissa básica: se o nick do sujeito não revelar o contrário, assume-se que seja masculino, mesmo porque não se dispõe de recursos visuais ou sonoros que possam completar a informação sobre o gênero. Assim, o clone inicial é sempre masculino e somente será acionada uma ação adaptativa que modifique, não só a interação a partir de então, mas também toda a memória da interação até então, se essa decisão não for adequada em algum momento (o que pode acontecer na verificação inicial

do gênero do nick ou em qualquer outro momento da interação, como pode nem chegar a acontecer).

Cabe notar que este mesmo trecho ilustrou a exemplificação de um processo de consulta baseado no esquema comunicativo, no final do tópico anterior, e ambas as explicações estão tratando exatamente do mesmo processo: a automodificação do sistema a partir de uma consulta.

O trecho a seguir, que acontece um pouco mais tarde no mesmo dia, mostra uma discussão sobre troca de nicks.

1. [16:25:28] Apelido Papa_foi_de_novo está conhecido agora como Papa_voltou_de_n.
2. [16:25:39] <Gato> n?
3. [16:25:43] <Gato> onde fica?
4. [16:25:43] <Papa_voltou_de_n> de novo :P
5. [16:25:48] <Papa_voltou_de_n> nick Papa_Voltou
6. [16:25:50] <Papa_voltou_de_n> ops
7. [16:25:53] <Gato> uahuahuahau
8. [16:25:53] Apelido Papa_voltou_de_n está conhecido agora como Papa_Voltou.
9. [16:26:00] <Gato> ae
10. [16:26:06] <Gato> Papa_Voltou, Papa_Voltou Papa_Voltou
11. [16:26:12] <Gato> la la la

A lógica da troca de nicks usadas pela identidade $I^{papagaio}$ baseia-se na sua presença ativa no chat (Fig. 11): ele não desconecta, mas espera que o nick acuse se está ou não ativo (em frente ao computador, por exemplo).

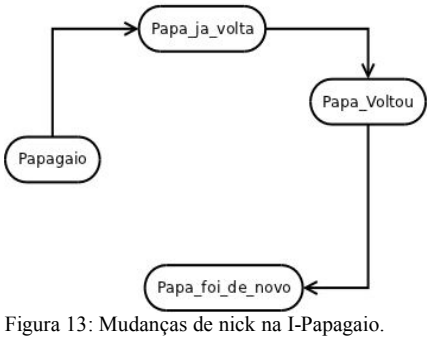


Figura 13: Mudanças de nick na I-Papagaio.

Quando ele muda seu nick de *Papa_foi_de_novo* para *Papa_voltou_de_n*, deixa uma lacuna sobre o significado de "n" que, se o interlocutor seguir a lógica proposta pela $I^{papagaio}$, será facilmente preenchida: Papagaio está presente novamente no chat, então n = novo. No entanto, *Gato*, com base no sentido de "voltar" como deslocamento físico, preenche a lacuna com outro sentido também possível: n = lugar de onde $I^{papagaio}$ estaria voltando. Trata-se de um ruído semiótico o qual, sem uma verificação (feita por *Gato* nas manifestações 20 e 21), permaneceria afetando a compreensão de I^{Gato} sobre a manifestação de $I^{Papagaio}$.

Isso ilustra nossa hipótese de que o sistema de comunicação proposto precisa exclusivamente de três operações para funcionar: consulta, inclusão e remoção das regras que cada um, D^{or} e $D^{ário}$, usa no processo. Nenhum dos dois actantes é capaz de manter qualquer processo comunicativo sem que essas regras sejam afetadas por estas operações, em maior ou menor escala e em maior ou menor número de vezes. Trata-se de um sistema complexo e dinâmico, portanto, que não pode ser resolvido e nem descrito eficientemente por regras fixas.

VIII. CONCLUSÃO

Longe de uma resposta final sobre o tema, procuramos indicar, no presente artigo, a linha reflexiva que pretendemos seguir para um trabalho de geração de diálogos (textualização) na interface entre Semiótica Greimasiana e Tecnologia Adaptativa. Enquanto esta entra como reguladora da relação polarizada entre o destinador e o destinatário, o esquema de comunicação de Silva entra como modelo para indicação de pontos nos quais um ajuste adaptativo se faz necessário a cada intervenção do *chatterbot* e seu(s) interlocutor(es). As análises do código, do sinal e da semiose são responsáveis por indicar os focos de ruído e, assim, acionar transições adaptativas.

O artigo procura mostrar que o esquema de comunicação de I.A.Silva é um ótimo candidato a esquema de comunicação adaptativo. A proposta metodológica, no entanto, justamente pela incipiência dessa investida, deve ainda passar por análises de *corpora* e categorizações semióticas mais finas a fim de se prestar, com maior propriedade, a embasar a produção automática de diálogos prevista no projeto Livrinho.

IX. AGRADECIMENTOS

O primeiro agradecimento que requer este trabalho é póstumo, a Ignácio Assis Silva, mestre cujos ensinamentos mobilizam a maior parte das reflexões da autora e autor do esquema aqui estudado.

Agradecemos com a mesma intensidade a João José Neto quem, desde o WTA2012, vem acompanhando e orientando nossos estudos e ensaios nesse novo campo interdisciplinar que hoje se nos apresenta e quem, por suas inúmeras provocações, figura hoje entre nossas maiores influências.

Esta pesquisa não teria sido possível sem o apoio da FAPEMIG (Processo PPM-00206-10: DadosSemiotica: programa para coleta e análise de dados) e do Grupo de Pesquisa Texto Livre: Linguagem e Tecnologia (http://dgp.cnpq.br/buscaoperacional/detalhegrupo.jsp?grupo=0333801U4BKW6D).

REFERÊNCIAS

[1] http://www.sourceforge.net/projects/livrinho/
[2] D. L. P. De Barros, "A Comunicação Humana" in: J. L. Fiorin.(Org). Introdução à Lingüística. São Paulo: Contexto, 2002. p. 25-53.
[3] J. José Neto. Solving complex problems with Adaptive Automata. In: CIAA'2000 Fifth International Conference on Implementation and Application of Automata, 2001, London, Ontario CANADA. Lecture Notes in Computer Science. New York: Springer Verlag, 2000. v. 2088. p. 340-342.
[4] D.A.Alfenas, M.R.Pereira-Barretto. Adaptatividade em Robôs Sociáveis: uma Proposta de um Gerenciador de Diálogos.
[5] R. Jakobson. Lingüística e comunicação. São Paulo: Cultrix, 1969.
[6]A. C. F. Matte, G. M. P. Lara. Um panorama da semiótica greimasiana. In: Alfa: Revista de Linguística (UNESP. São José do Rio Preto. Impresso), v. 53, p. 339-350, 2009.
[7] L. Hjelmslev. La structure fondamentale du langage. Prolégomènes a une theorie du langage./Anne-Marie Lénoard (trad.). Paris: Minuit, 1968.
[8] I. A. Silva. Figurativização e metamorfose. O mito de Narciso. São Paulo: Editora da Universidade Estadual Paulista, 1995.
[9] P. Stockinger, M. Arnold, P. Boudon, J. P. Desclés, F. Rastier. Intelligence artificielle et théorie sémio-linguistique. Actes Sémiotiques Bulletin vol. VIII, n.o 36, 1985 .
[10] P. Stockinger, G. Denhière, J. Fontanille, A. Piolat, M. Zock. Intelligence Artificielle, II: Approches cognitives du texte. Actes Sémiotiques Bulletin vol. IX, n.o 40, 1986.
[11] A. C. F. Matte. Vozes e Canções Infantis Brasileiras: emoções no tempo. Tese de doutoramento. Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo. São Paulo: USP, 2002.
[12] A. C. F. Matte. Para uma investigação do padrão prosódico-emocional no Português do Brasil. Pós-doutoramento. Instituto de Estudos da Linguagem da Universidade Estadual de Campinas. Campinas: UNICAMP, 2004.
[13] A. C. F. Matte. Análise Quantitativa da Tensividade no Conteúdo Verbal tendo em vista o Estudo da Expressão da Emoção na Fala e o

Modelamento Prosódico. In: Cadernos de Estudos Linguísticos vol. 46, n.o 1, 2004. p. 53-69.
[14] A. C. F. MATTE. Existe Fala Neutra para a Poesia?. DELTA. Documentação de Estudos em Lingüística Teórica e Aplicada, v. 24, p. 159-174, 2008.
[15] A. C. F. Matte, A. R. Meireles, R. T. Ribeiro. SETFON: O Problema da Análise de Dados Prosódicos, Textuais e Acústicos. In: Revista (con) textos linguísticos (UFES), v. 1, p. 8-30, 2011.
[16] J. José Neto. 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, 2003.
[17] I. A. SILVA A deixis pessoal. Tese de doutoramento. Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo. São Paulo: USP, 1972.
[18] A. C. F Matte. O Processo Semiótico de Comunicação. Sobre o Esquema de Comunicação de Ignácio Assis Silva. In: CASA Cadernos de Semiótica Aplicada Vol. 6.n.2, dezembro de 2008 URL: http://seer.fclar.unesp.br/casa/article/view/1206 Acessado em 12/10/2012.
[19] A. C. F. Matte, W. D. C. M. Silva, H. L. Canalli, R. T. Ribeiro. DadosSemiotica: coleta e processamento de análises semióticas de texto escrito. In: Worshop Software Livre, 2012, Porto Alegre. Anais do WSL 2012. Porto Alegre: Sociedade Brasileira de Computação, 2012. v. 1.
[20] J-M. Klinkenberg. À quoi servent les schémas? Tabularité et dynamisme linéaire. In: Protée, vol. 37, n.o 3, 2009.
[21] http://dadossemiotica.textolivre.org
[22] A.C.F.Matte. Dadossemiotica 1.0: elaboração de projeto. submetido.
[23] J. Mattoso Câmara Jr. Estrutura da Língua Portuguesa. 35.a edição. Petrópolis: Vozes, 2002.
[24] J. Mattoso Câmara Jr. Problemas de Linguística Descritiva. 19.a edição. Petrópolis: Vozes, 2002.
[25] J. Mattoso Câmara Jr. Dispersos de J. Mattoso Câmara Jr/C.E.F.Uchoa (org). Coleção Dispersos. Nova edição revisada e ampliada. Rio de Janeiro: Lucerna, 2004.
[26] M. E.K. Buzato. Será que ler um robô desrobotiza um leitor? In: Trab. linguist. apl. [online], vol.49, n.2, 2010. URL: <http://www.scielo.br/scielo.php?pid=S0103-18132010000200004&script=sci_arttext>. Acessado em 12/10/2012.
[27] F. de Saussure. Curso de Lingüística Geral. São Paulo: Cultrix/EDUSP, 1969.
[28] A.C.F. Matte, A.R.Meireles, C.C.Fraguas. SIL Web - analisador fonológico silábico-acentual de texto escrito. Revista de Estudos da Linguagem, v. 14, p. 31-50, 2006.
[29] A.J.Greimas, J. Fontanille. Semiótica das Paixões./Tradução M.J.Coracini. São Paulo, editora Ática, 1993, pp. 155-156.
[30] http://en.wikipedia.org/wiki/Internet_Relay_Chat
[31] http://freenode.net

Ana Cristina Fricke Matte é mestre e doutora em Semiótica e Linguística Geral pela Universidade de São Paulo (USP) em 2002 e realizou pós doutorado em Fonética Acústica na UNICAMP, concluído em 2004. É professora da Faculdade de Letras da Universidade Federal de Minas Gerais (UFMG) desde 2004 e trabalha na linha de pesquisa em Linguagem e Tecnologia do POSLIN/UFMG desde 2006. Fundou o grupo de pesquisa Texto Livre: Semiótica e Tecnologia, registrado sob sua coordenação no Diretório de Grupos do CNPq. Atualmente coordena o grupo interinstitucional responsável pelo projeto Portal do Professor Livre na Rede, atuando na equipe de desenvolvimento. Possui interesse de pesquisa nas seguintes áreas de pesquisa: semiótica tensiva, semiótica das paixões, desenvolvimento de software livre educacional, novas tecnologias para a educação, licenças livres, EAD e documentação em software livre.

Uso de tecnologia adaptativa para implementação de sistema de aprendizagem de algoritmos baseado na plataforma Google Android

G. C. Farto

***Abstract* — The purpose of this article is to present the development of an algorithms learning system based on Google Android platform, in order to provide a new resource of teaching and learning for students of IT courses. Concepts of adaptive technology were used to make the application more dynamic.**

Keywords – sistema de aprendizagem; Mobile Learning; algoritmos; tecnologia adaptativa; Java; Google Android

I. INTRODUÇÃO

[1]As disciplinas de Algoritmos e Lógica de Programação, geralmente lecionadas nas primeiras etapas dos cursos de Tecnologia da Informação (TI), são consideradas desafiadoras por grande parte dos alunos, pois exige a formulação e desenvolvimento de soluções de problemas utilizando-se os conceitos base de matemática e lógica.

Por meio de um levantamento realizado em parceria com a Fundação Educacional do Município de Assis (FEMA) e Instituto Municipal de Ensino Superior de Assis (IMESA), obteve-se indicadores sobre a quantidade de alunos aprovados e reprovados de cursos de tecnologia de 2009, 2010 e 2011.

O gráfico ilustrado na Figura 1 apresenta as estatísticas de alunos aprovados, reprovados com exame, reprovados por frequência e reprovados sem exame que frequentavam a disciplina de Algoritmos e Estruturas de Dados I dos cursos de Análise e Desenvolvimento de Sistemas (ADS) e Tecnologia em Processamento de Dados (TPD) no período analisado.

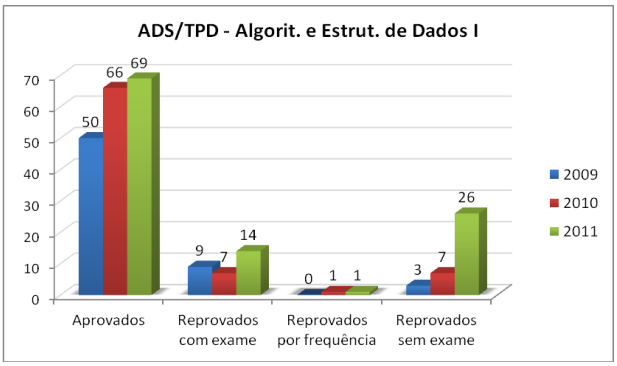


Figura 1. Gráfico de situações para os cursos de ADS e TPD

G. C. Farto, Fundação Educacional do Município de Assis (FEMA), Assis, São Paulo, Brasil, guilherme_computacao@yahoo.com.br

Apesar de ser em quantidade menor, o gráfico ilustrado na Figura 2 apresenta as estatísticas de alunos reprovados que frequentavam a disciplina de Algoritmos e Estruturas de Dados I do curso de Bacharelado em Ciências da Computação (BCC) no período analisado.

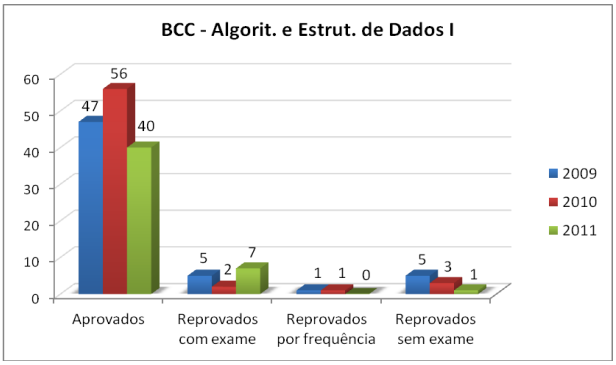


Figura 2. Gráfico de situações para o curso de BCC

Pode-se observar que o índice de reprovações é considerado alto, comparado a outras disciplinas do mesmo curso, entretanto reflete uma situação real da disciplina que é a tendência a dificuldades de aprendizado.

Na busca de soluções para esse problema recorrente nos cursos de tecnologia, diversas pesquisas enfocam tempo e recursos necessários em abstração, projeto e construção de ferramentas computacionais capazes de auxiliar o aluno nas fases iniciais de aprendizagem de algoritmos e lógica.

O principal desafio no processo de ensino de algoritmos e lógica se deve ao fato de que, na maioria das vezes, o conteúdo é aplicado a grupos heterogêneos de participantes, cada qual com seus talentos, modo de trabalhar e pensar, assim como métodos de aprendizagem diferentes. Uma alternativa para solucionar essa dificuldade está na possibilidade do próprio aluno gerir e conduzir boa parte da evolução de seu aprendizado.

Esse modo de aprender pode ser instituído por meio de um sistema de aprendizagem interativo, onde o participante, além de obter conteúdos e instruções de determinado assunto, é capaz de interagir com a aplicação, propondo soluções para resolver um determinado problema apresentado durante o processo de aprendizado.

A partir dessa ideia, os ambientes ou sistemas de aprendizagem interativos devem ser baseados em quatro princípios: o estudante deve construir conhecimento; o controle do sistema é feito, de forma mais significativa, pelo estudante; o sistema é individualizado para cada estudante; e o feedback é gerado em função da interação do estudante com o ambiente.

Outro assunto que motiva o tema proposto é o mercado de dispositivos móveis que cresce cada vez mais, correspondendo a um grande percentual da população mundial. Entre as plataformas disponíveis, destaca-se a do Google Android, por ser o primeiro ambiente de desenvolvimento de aplicativos móveis completamente livre e open-source, representando uma grande vantagem para sua evolução, uma vez que desenvolvedores podem contribuir com melhorias para a arquitetura e o sistema operacional.

Portanto, a tecnologia Google Android é um recurso muito importante e deve ser considerada nas pesquisas e sistemas de aprendizagem interativos, pois um aplicativo móvel permite ao aluno modelar e executar algoritmos tanto em aulas presenciais como a distância, assim como estar disponível a qualquer hora, local e dispositivo portátil, como um smartphone ou tablet, aumentando as oportunidades e maneiras de ser utilizado para fins de aprendizado.

Atualmente, há uma grande intensificação no desenvolvimento de metodologias de Ensino a Distância (EAD), descrevendo a interação entre professor e aluno durante o processo de ensino-aprendizagem. Porém, em sua grande maioria, a proposta do modelo de ensino é apenas a de transmitir, ao aprendiz, informações e conteúdos definidos pelo tutor por meio de lições pré-estabelecidas.

Sabendo-se da necessidade de metodologias de ensino a distância baseadas na geração de conteúdos em um sistema de aprendizagem, uma alternativa é fazer uso de conceitos da tecnologia adaptativa para realizar a implementação de aplicações mais dinâmicas, contribuindo com o aprendizado do aluno em uma determinada área de conhecimento.

Os resultados proporcionados pelo uso de tecnologia adaptativa são caracterizados por apresentar uma estrutura dinâmica, objetivando a automodificação provocada por interações com o meio externo, sendo ele real ou virtual. Esta habilidade de mudança comportamental é essencial para a construção de máquinas e programas de computadores capazes de evoluir e gerar novas situações com a própria experiência.

O principal diferencial da tecnologia adaptativa é tornar possível, de maneira razoavelmente simples, o reaproveitamento e a ampliação das capacidades de teorias e técnicas existentes e consolidadas, principalmente nas áreas de Inteligência Artificial e Teoria da Computação.

Este artigo aborda o desenvolvimento de um sistema de aprendizagem de algoritmos baseado na plataforma Google Android, com a finalidade de ser uma ferramenta de estudo para alunos das disciplinas de Algoritmos e Lógica de Programação.

Objetivando tornar a aplicação mais dinâmica, fez-se uso de conceitos da tecnologia adaptativa para possibilitar a geração dinâmica de problemas que desafiam o aluno a criar, a cada enunciado proposto, novas soluções computacionais, sem a necessidade de tal situação estar pré-estabelecida.

II. AS DIFICULDADES DE APRENDIZAGEM DE ALGORITMOS

Presente na literatura há diversos trabalhos que identificam justificativas para a dificuldade, quase que intrínseca, na tarefa de aprender os conceitos de lógica e programação.

Entre os argumentos propostos, deve-se considerar que o aprendizado de programação, assim como em outras áreas de conhecimento, é obtido por meio um processo lento e gradual, onde novos fundamentos e ideias, até então desconhecidos, devem ser transformados em algo familiar, facilitando a sua assimilação [1].

Ressalta-se, também, que uma das principais dificuldades reside na técnica de compreender e, em particular, aplicar as noções básicas, como estruturas de controle, para a criação de algoritmos capazes de resolver situações reais [2].

No âmbito de aprendizado de programação, uma possível causa de dificuldade se deve ao fato de que muitos dos alunos não apresentam interesse neste tipo de disciplina. Tal desmotivação pode ser explicada pelo grande volume de conhecimentos abstratos relacionados à atividade de programar, da mesma forma que muitas linguagens e tecnologias estão cada vez mais sofisticadas [3].

Devido às dificuldades enfrentadas pelos alunos, as disciplinas de Algoritmos e Lógica de Programação têm apresentado altos índices de desistência e reprovação. A evasão, além de distanciar o aluno da formação intelectual e profissional, incita a desconfiança sobre a qualidade dos cursos superiores, contribuindo por impedir a entrada de novos alunos, atrasando, consequentemente, o crescimento da área de computação [4].

Para obter sucesso no aprendizado de programação, deve-se realizar um treino intensivo em resolução de problemas e exigir uma precisão e atenção a detalhes muito mais elevada do que a requerida por grande parte de outras disciplinas [1], [5].

Além dos argumentos apresentados anteriormente, alguns trabalhos referem-se que, ao invés de haver uma dificuldade inerente ao aprendizado de algoritmos, há alunos que não possuem as aptidões necessárias para a área de programação, especificamente na resolução de problemas envolvendo lógica e matemática [6], [7], [8].

Entre outras, há várias causas para o insucesso em disciplinas de computação, como a dificuldade de abstração e compreensão, a falta de competências necessárias para resolver problemas, o uso inadequado de metodologias didáticas se comparado ao modelo de aprendizagem dos alunos, além de que as linguagens e tecnologias são detentoras de sintaxes complexas para aprendizes com pouca ou nenhuma experiência [9].

Muitos dos temas relacionados ao processo de aprendizagem são bastante questionáveis, já que o método de ensino abrange tanto alunos quanto professores, assim como as metodologias aplicadas na sala de aula e o certo grau de dificuldade inerente da área de tecnologia.

Há diversos outros motivos, de natureza didática, que podem ser considerados a origem da dificuldade de aprendizado, como o grande número de alunos por turma, dificuldade de o professor compreender a lógica formulada pelo aluno, níveis diferentes de experiência e ritmo entre os alunos, assim como a ausência de bons materiais para disciplinas introdutórias e a própria dificuldade da escolha do curso superior.

Alcançando um novo patamar, tambem podem ser analisados problemas de naturezas cognitiva, como ausência de perfil necessário para a resolução de problemas, e afetiva, como problemas de ordem pessoal que impedem a concentração durante explanações de conteúdo.

Analisando as dificuldades encontradas por alunos dos cursos de tecnologia e a fundamental importância dessa base inicial de conhecimento e considerando que os modos de pensar e aprender são pessoais e que não é possível nem viável ao tutor adequar-se às necessidades de cada aluno, tornam-se justificáveis a abstração, modelagem e implementação de um sistema de aprendizagem de algoritmos.

III. TECNOLOGIA ADAPTATIVA: ESTADO DA ARTE E APLICAÇÕES

A terminologia da palavra "adaptatividade" se aplica a diversos conceitos, porém, dentro do contexto de tecnologia e computação, refere-se à possibilidade de se automodificar, adaptando-se, de maneira instintiva, sem a necessidade de qualquer outro tipo de recurso.

A tecnologia adaptativa originou-se da notação científica de autômatos, onde a formalização é designada pelos conceitos de autômatos finitos, visto que a base das linguagens regulares é especificada por estados, transições e uma cadeia de entrada, definindo-se os conceitos e fundamentos.

O domínio da tecnologia adaptativa requer o conhecimento de três vertentes:

- Teoria: responsável por fornecer os fundamentos matemáticos;
- Ferramentas e ambientes: responsáveis por facilitar o desenvolvimento de aplicações adaptativas;
- Aplicações: responsáveis pela resolução eficiente de problemas das mais variadas áreas de interesse.

Entre os primeiros estudos realizados com tecnologia adaptativa, destaca-se o uso de conceitos de adaptatividade para a definição e construção de compiladores, aplicando mecanismos adaptativos na análise sintática e em geradores de reconhecimento sintático [10].

Devido ao avanço nas pesquisas, tornou-se possível a incorporação de funções de transdução sintática, apresentando um aperfeiçoamento de reconhecedores sintáticos, baseando-se nos autômatos de pilha estruturada [11]. Pouco tempo depois, por meio de um novo trabalho sobre transdutores adaptativos, definiu-se uma classe de máquinas de estados finitos armazenados em uma pilha e com memória organizada que demonstra a alteração dinâmica de sua configuração, de acordo com o aprendizado realizado nas transições, acrescentando um poder maior na representação dos modelos matemáticos propostos [12].

Com base nos trabalhos realizados anteriormente, novos estudos foram iniciados com o objetivo de aplicar a tecnologia adaptativa em projetos de sistemas reativos, ou seja, regidos basicamente por estímulos internos ou externos. Neste contexto, um formalismo para sistemas reativos é o statechart, que, em sua versão adaptativa, resultou na implementação da primeira ferramenta computacional para a construção e simulação de dispositivos adaptativos nomeada de STatecharts ADaptativos (STAD) [13].

Ao aplicar as técnicas de adaptatividade em um statechart, o formalismo de um sistema reativo começa a apresentar a capacidade de modificar sua configuração em relação às entradas fornecidas ao sistema.

Continuando as pesquisas para a implementação de sistemas reativos adaptativos, uma evolução do STAD, adicionando recursos de sincronização de processos, baseados em redes de Petri, foi implementada em um sistema chamado de Statecharts Adaptativos Sincronizados (SAS) [14].

Os conceitos da tecnologia adaptativa também têm sido abordados em pesquisas na área de ensino a distância, definindo propostas para sistemas baseados no modelo adaptativo de Architectural Modelling Box for Enterprise Redesign, também conhecido como AMBER-Adp [15]. O modelo AMBER é uma ferramenta de apoio que visa auxiliar o projeto e desenvolvimento de sistemas distribuídos e, por meio das técnicas fornecidas pela adaptatividade, seu poder de expressão é aumentado, possibilitando a modelagem natural de sistemas com regras que se modificam dinamicamente.

A referência [16] propõe um gerador de ambiente, também chamado de meta-ambiente, que possibilita a geração automática de ambientes para projetos baseados em aplicações adaptativas. Por meio deste trabalho, foram documentados o método para definição de dispositivos adaptativos, a arquitetura geral de um ambiente para projeto de aplicações adaptativas e a arquitetura para um gerador de ambientes para a modelagem de aplicações utilizando um dispositivo adaptativo específico.

Desde então, tem-se estudado e aplicado os fundamentos de adaptatividade, com a finalidade de criar mecanismos que, por meio de um dispositivo e conjunto de regras, podem modificar o comportamento de aplicações, objetivando desenvolver softwares mais dinâmicos e eficientes na resolução de diversos problemas existentes na computação.

A aplicação e uso de dispositivos adaptativos abrangem muitas áreas, tais como:

- Educação: uso de fundamentos de adaptatividade para a modelagem e implementação de softwares de apoio ao ensino e aplicativos educacionais;
- Inteligência artificial: modelos adaptativos utilizados para representar e manipular conhecimento, capazes de aprender por meio de informações pré-definidas e de um algoritmo de tomada de decisões;
- Segurança e privacidade: criptografia, controle de acesso, classificação de dados, reconhecimento de padrões, entre outros tópicos;
- Robótica: uma vasta área de aplicação da tecnologia adaptativa, intensificando seu uso em protocolos de roteamento, redes móveis sem fio e estudos ligados à navegação robótica autônoma;
- Jogos e simuladores: mecanismos adaptativos que podem ser acoplados a jogos e simuladores, fornecendo realismo e situações mais dinâmicas para um determinado contexto.

Conforme demonstrado por meio desta breve introdução, a tecnologia adaptativa objetiva o estudo de técnicas computacionais, tanto de modelagem quanto de desenvolvimento, para a construção de modelos ou dispositivos com características automodificáveis, dificilmente encontradas quando utilizada grande parte de métodos que buscam atender a mesma finalidade.

IV. AMBIENTES DE APRENDIZAGEM BASEADOS EM MOBILE LEARNING

O uso de ambientes de ensino, especialmente os baseados em ensino a distância, começou, há pouco tempo, a ingressar para a terceira onda tecnológica, chamada de Mobile Learning ou Aprendizagem Móvel.

Essa modalidade de aprendizado caracteriza-se pelo uso de equipamentos portáteis, como smartphones e tablets, em um contexto de "computação pervasiva", amparado pela mobilidade global, conectividade ubíqua, independência do dispositivo e ambiente computacional do usuário disponível em quaisquer locais e horários [17].

Além das perspectivas citadas anteriormente, é possível relacionar algumas características chaves para facilitar o entendimento e contribuir com a definição de Mobile Learning, destacando-se:

- Prover acesso a conteúdos didáticos educacionais em qualquer local e a qualquer momento, por meio de recursos de conectividade do dispositivo portátil utilizado;
- Expandir os limites internos e externos da sala de aula ou da empresa, de forma ubíqua, ou seja, integrando a computação com as ações e comportamentos naturais das pessoas;
- Fornecer os meios necessários para o desenvolvimento de metodologias inovadoras de ensino e treinamento, por meio de novos recursos da computação e portabilidade.

Portanto, de maneira resumida, pode-se conceituar Mobile Learning como qualquer tipo de aprendizado que faz uso de dispositivos móveis como ferramentas de ensino a qualquer contexto educacional, seja ele na área acadêmica ou organizacional.

A Figura 3 ilustra a evolução dos sistemas de aprendizagem a distância, definindo Mobile Learning como sendo uma extensão de E-Learning e, consequentemente, do Distance Learning [18].

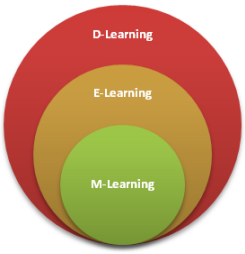


Figura 3. Evolução dos sistemas de aprendizagem a distância [18]

Partindo dessa mesma classificação de sistemas ou metodologias de ensino a distância, pode-se afirmar que o objetivo do Mobile Learning é fazer com que o ambiente de E-Learning seja implementado em dispositivos computacionais móveis, valendo-se de diversos recursos até então inexistentes ou, por muitas vezes, inexplorados.

Essa abordagem visa prover ubiquidade e um processo de aprendizagem significativo para o aprendiz por meio da criação de um contexto mais dinâmico e motivador, fazendo-se uso de conteúdos de multimídia e de uma alta interatividade [19].

Há diversas ações já iniciadas nas instituições de ensino do Brasil com o objetivo de introduzir os primeiros indícios da computação móvel e, com isso, definir as bases principais para novos estágios relacionados à computação pervasiva e ubíqua. Por esse e outros motivos citados anteriormente, motivou-se a elaboração deste trabalho, como alternativa para fornecer novos meios de ensino-aprendizagem de disciplinas de computação.

V. ESTUDO DE CASO

O processo de ensino utilizando dispositivos computacionais teve início no final da década de 1950, quando o psicólogo americano B. F. Skinner, fundamentado na teoria comportamentalista, propusera uma metodologia de ensino, chamando-a de "máquina de ensinar". Nessa metodologia, um conjunto de conhecimentos a ser ensinado se divide em módulos sequenciais onde o aluno deve responder, corretamente, as questões propostas para avançar de etapa durante o processo de ensino-aprendizagem.

Por meio desse método, foram desenvolvidas as primeiras aplicações de computador específicas para o ensino, sendo conhecidas como Computer Aided Instruction (CAI) ou Instrução Assistida ou Auxiliada por Computador.

Nos sistemas de ensino baseados no modelo CAI, o aprendiz segue uma série finita e predeterminada de passos. A cada etapa, novos conhecimentos são adquiridos, testados e, se correspondendo ao resultado correto, um novo conjunto de informações é transmitido; caso contrário, o conhecimento ainda não assimilado é novamente apresentado e o teste é refeito. Esse ciclo se repete até que o estudante responda corretamente ao teste, resultado este que comprova a aquisição do conhecimento exposto pelo sistema CAI.

Ao longo do tempo, das pesquisas realizadas e da constante evolução das tecnologias computacionais existentes, diversos sistemas de apoio à aprendizagem de programação foram desenvolvidos, como representações gráficas de algoritmos, sistemas tutores inteligentes, ambientes de aprendizagem a distância, companheiros de aprendizagem, entre outros.

Sabendo acerca dos problemas existentes e das dificuldades enfrentadas por alunos durante o aprendizado de conceitos fundamentais, torna-se viável a utilização de sistemas de aprendizagem em diversas disciplinas lecionadas nos cursos de computação, destacando-se, nesse cenário, Algoritmos e Lógica de Programação.

A proposta e um dos objetivos principais para a realização deste trabalho é, além de contribuir de forma teórica para os ambientes computacionais especializados na área de educação, o de implementar um sistema de aprendizagem de algoritmos baseado na plataforma móvel Google Android, disponibilizando um novo recurso de ensino-aprendizagem a ser utilizado para auxiliar e motivar os alunos dentro e fora de instituições de ensino e empresas.

Dessa forma, o projeto computacional, proposto e desenvolvido neste trabalho, contém, como premissas e requisitos iniciais, características relevantes presentes em um sistema de aprendizagem, resultando em uma aplicação capaz de fornecer um ambiente e metodologia para o estudo de algoritmos. Portanto, com o uso dessa ferramenta, cria-se a possibilidade de os alunos adquirirem e compreenderem as diversas etapas de construção de um algoritmo, além de permitir, a partir de sua implementação, testes, validações e possíveis correções em suas próprias soluções algorítmicas para um determinado problema sugerido pelo sistema tutor.

A. Arquitetura do sistema de aprendizagem de algoritmos

Durante a etapa de análise e modelagem do sistema de aprendizagem de algoritmos, cinco componentes básicos, herdados dos métodos de construção de sistemas CAI, foram identificados:

- Interface: camada responsável por intercambiar ou trocar informações entre o sistema de aprendizagem e o aluno. É por meio da interface que ocorre a interação do aprendiz com os conhecimentos expostos pelo sistema;
- Controlador de eventos: camada responsável por gerenciar e efetuar a troca de informações entre os demais módulos da arquitetura;
- Modelo do aluno: camada responsável por gerenciar o conhecimento do aluno, registrando informações sobre seus acertos e erros, assim como quais conteúdos já foram assimilados;
- Base de domínio: camada responsável por conter e descrever os conhecimentos de um especialista na área de domínio do sistema, contribuindo para a evolução do modelo do aluno;
- Modelo pedagógico: camada responsável por gerenciar as instruções ou regras de ensino, assim como executar um diagnóstico baseado no conhecimento do aluno para decidir quais as estratégias de ensino serão adotadas durante o processo de aprendizagem.

A Figura 4 ilustra a arquitetura simplificada do sistema de aprendizagem de algoritmos proposto neste trabalho, destacando os cinco componentes básicos de uma aplicação de ensino: interface, controlador de eventos, modelo do aluno, base de domínio e modelo pedagógico.

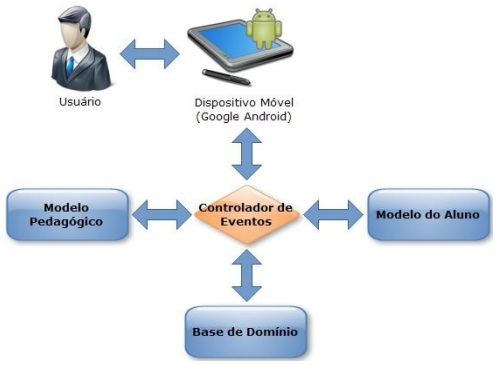


Figura 4. Arquitetura simplificada do sistema de aprendizagem de algoritmos

Apesar da simplicidade da arquitetura adotada, conforme ilustrada na Figura 4, as camadas definidas são suficientes para uma abordagem eficaz e eficiente dentro do contexto de um sistema de aprendizagem.

A base de domínio, como citado anteriormente, é responsável pelos conhecimentos da área de ensino de um sistema de aprendizagem. Portanto, para o ensino de algoritmos, essa camada deve prover as estruturas dos comandos disponíveis assim como a correta e melhor forma de utilizá-los.

A Tabela I relaciona todas as instruções ou comandos existentes que podem ser utilizados para a criação de soluções algorítmicas no sistema de aprendizagem proposto neste trabalho. A tipagem de dados, utilizada na definição de constantes e variáveis, é restrita aos valores ¨booleano´´, “inteiro”, “real” e “texto”, enquanto os operadores condicionais são: igual a (==), diferente de (!=), menor que (<), maior que (>), menor ou igual a (<=) e maior ou igual a (>=).

TABELA I. COMANDOS DISPONÍVEIS NO SISTEMA DE APRENDIZAGEM DE ALGORITMOS

Comando	Padrão para utilização
Declaração de Algoritmo	ALGORITMO <nome do programa>
Início de Algoritmo	INICIO:
Fim de Algoritmo	FIM.
Declaração de Área de Constantes	CONSTANTES:
Declaração de Constante	<nome> = <valor> : <tipo>;
Declaração de Área de Variáveis	VARIAVEIS:
Declaração de Variáveis	<nome> : <tipo>;
Atribuição de Valor	<variável> = <valor>;
Estrutura Condicional Se... Então... Senão...	SE (<condição>) ENTAO ... SENAO ... FIM-SE.
Estrutura de Repetição Enquanto... Faça...	ENQUANTO (<condição>) FACA ... FIM-ENQUANTO.
Leitura de Valor	LEIA (<variável>);
Escrita de Valor	ESCREVA (<constante ou variável>);

Após a definição dos comandos a serem disponibilizados na aplicação, tornou-se possível elaborar um diagrama de classes para representar os tipos existentes no sistema de aprendizagem de algoritmos, conforme ilustrado na Figura 5.

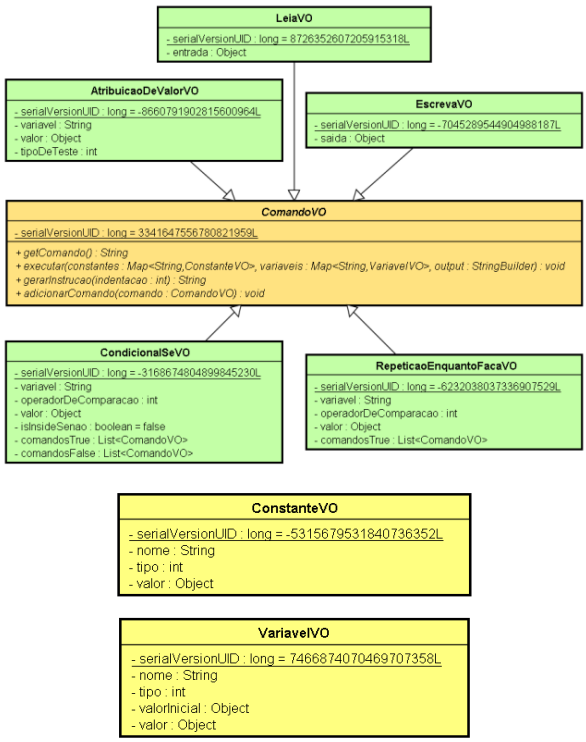


Figura 5. Diagrama de classes elaborado para representar os comandos disponíveis no sistema de aprendizagem de algoritmos

Concluída a etapa de análise e modelagem do projeto, iniciou-se a construção do sistema de aprendizagem de algoritmos em uma estrutura de projeto Java, implementando

os dispositivos adaptativos a partir dos conceitos estudados, com a finalidade de tornar a aplicação e o processo de aprendizagem mais dinâmico.

Em seguida, um projeto baseado na plataforma Google Android fora integrado aos componentes desenvolvidos anteriormente, sem quaisquer mudanças, comprovando a facilidade ao modificar e evoluir o software a partir da arquitetura utilizada.

B. Interação e fluxo de processos

A interação e o fluxo de processos do sistema de aprendizagem de algoritmos podem ser divididos em três etapas principais, cada qual com suas funcionalidades e responsabilidades, visando contribuir com a aquisição dos conhecimentos necessários durante o processo de ensino-aprendizagem.

Por meio da interface móvel, desenvolvida em um projeto Java baseado na plataforma Google Android, o usuário tem acesso aos recursos providos pela aplicação de ensino de algoritmos, auxiliando-o nas atividades de pensar, modelar e implementar soluções para os problemas gerados pelos dispositivos adaptativos.

O sistema de aprendizagem gerencia, pela camada de modelo do aluno, os níveis de conhecimento do aprendiz na disciplina de algoritmos, sendo possível dividi-los em:

- Nível 1: Implementação de algoritmos estruturados e sequenciais.
- Nível 2: Implementação de algoritmos estruturados e sequenciais, acrescentando conceitos de estrutura condicional (Se... Então... Senão...).
- Nível 3: Implementação de algoritmos estruturados e sequenciais, acrescentando conceitos de estrutura condicional e de repetição (Enquanto... Faça...).

A escolha da arquitetura e a aplicação de conceitos de componentização durante o desenvolvimento do projeto contribuem para uma melhor manutenibilidade, reusabilidade e novas implementações, já que, grande parte do sistema, é resultado da integração de diversos componentes fracamente acoplados.

Portanto, caso novos níveis ou módulos necessitem ser adequados ou acrescentados ao sistema de aprendizagem de algoritmos, basta realizar a sua implementação nas camadas de base de domínio e modelo pedagógico.

A primeira etapa, presente na interação e fluxo de processos, faz uso do primeiro dispositivo adaptativo implementado no sistema de aprendizagem e objetiva ser responsável pela geração dinâmica de enunciados de algoritmos. Por meio da recuperação de informações do modelo de aluno e com algumas configurações iniciais armazenadas em um arquivo de propriedades, o dispositivo adaptativo elabora um enunciado, solicitando, ao aluno, que implemente uma solução para determinado problema criado, estabelecendo-se a situação inicial do processo de ensino-aprendizagem.

Na segunda etapa, após a geração do enunciado, o aluno inicia a implementação do algoritmo, com o objetivo de fornecer uma solução por meio dos comandos disponíveis na aplicação. A interação entre o aluno e o sistema de aprendizagem resulta em uma fonte de estímulos utilizada para validar os comandos construídos pelo aprendiz, além de ser possível identificar erros durante cada passo do desenvolvimento do algoritmo.

A terceira e última etapa tem por finalidade realizar a validação completa do algoritmo implementado pelo aluno, comparando-o com o enunciado proposto no início do processo de ensino. Com o resultado da validação, fornecido pelo segundo dispositivo adaptativo e amparado por um conjunto de regras estabelecidas para o ensino e a construção de algoritmos, é possível obter informações sobre acertos e erros, além de etapas ou blocos pendentes de implementação.

A Figura 6 ilustra a interação e o fluxo de processos do sistema de aprendizagem, podendo-se notar, claramente, as três etapas citadas, além de permitir um rápido compreendimento de cada elemento e seus relacionamentos dentro do contexto de uma aplicação de ensino de algoritmos.

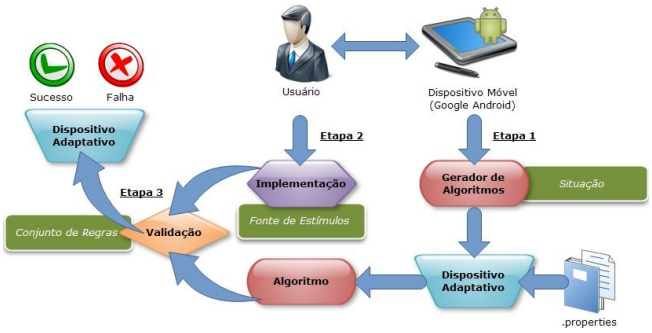


Figura 6. Interação e fluxo de processos do sistema de aprendizagem de algoritmos

C. Tecnologias e abordagens adotadas para a implementação

Para realizar a implementação do sistema de aprendizagem de algoritmos, com as funcionalidades e requisitos citados, um conjunto de tecnologias e abordagens foram pesquisadas e utilizadas durante o desenvolvimento da aplicação de ensino.

Iniciando pela escolha da linguagem de programação, optou-se pela tecnologia Java, por ser utilizada em todos os principais segmentos da indústria, estando presente em uma gama de dispositivos, computadores e redes. Com uma comunidade diversificada entre analistas, projetistas e desenvolvedores, Java é a tecnologia que tem sido a principal escolha do mercado de TI na área de desenvolvimento de aplicações Web, sistemas distribuídos, programação em redes de computadores e, mais recentemente, aplicações móveis.

O uso da moderna plataforma do Google Android, como tecnologia escolhida para o desenvolvimento deste projeto, se deve ao fato de que, por meio dela, torna-se possível a implementação de aplicações móveis que podem ser integradas a diversos recursos de maneira simplificada, utilizando a linguagem de programação Java e um ambiente de desenvolvimento de alto nível e produtividade. Além de que, cada vez mais, o uso de dispositivos móveis, como smartphones e tablets, vem crescendo, criando novas oportunidades de abordagem, entre elas, a integração das áreas de educação, aprendizado eletrônico e computação.

Devido à necessidade de que as instruções implementadas pelo usuário precisam ser validadas conforme os padrões de comandos estabelecidos para a criação de algoritmos, fez-se uso de Regular Expressions (RE) ou Expressões Regulares, um importante recurso que tem por finalidade realizar a validação e manipulação, de maneira precisa e flexível, de informações baseadas em cadeias de caracteres.

Portanto, para cada comando implementando, uma validação por meio de RE é realizada, autenticando com sucesso, ou não, a instrução elaborada pelo aluno. Assim que a instrução é validada e aceita, o comando é decomposto e uma instância do objeto Java é armazenada na lista de instruções do algoritmo que está sendo construído pelo aprendiz.

Além da portabilidade e integração de diversas tecnologias presentes no projeto, buscou-se, ainda, desenvolver um sistema de aprendizagem mais dinâmico a partir da geração de enunciados de algoritmos. Para isso, foram utilizados os conceitos da tecnologia adaptativa, possibilitando a criação de dispositivos adaptativos capazes de elaborar novos enunciados e situações para estudos de algoritmos e lógica de programação. Esses recursos automodificáveis também foram implementados no processo de validação e identificação de acertos e erros da solução proposta pelo aluno.

A construção de sistemas computacionais extensos, como ocorre no contexto de aplicações de ensino, tende a resultar em projetos maiores e mais complexos, dificultando alterações futuras e novas implementações. Para resolver esse e outros problemas recorrentes em projetos de software, novas abordagens surgiram no final da década de 1990 no ramo de Engenharia de Software, destacando-se a Component-Based Software Engineering (CBSE) ou Engenharia de Software Baseada em Componentes.

A engenharia de componentes foi adotada como metodologia de desenvolvimento do sistema proposto neste trabalho, sendo utilizada para definir um processo de análise, construção e integração de componentes independentes não firmemente acoplados. A CBSE tornou-se uma importante abordagem de desenvolvimento, contribuindo para a reusabilidade de diferentes partes já implementadas, com a finalidade de acelerar e garantir maior qualidade no processo de construção de software.

D. Imagens do sistema de aprendizagem de algoritmos

A Figura 7 ilustra a geração dinâmica, realizada pelo dispositivo adaptativo, de um enunciado de algoritmo, conforme definido na descrição da primeira etapa de interação com o sistema de aprendizagem.

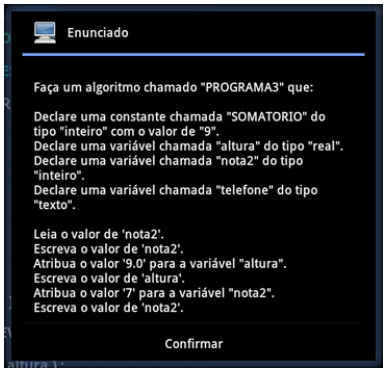


Figura 7. Enunciado de algoritmo gerado dinamicamente (primeira etapa) pelo dispositivo adaptativo

A Figura 8 ilustra a etapa de implementação de uma solução algorítmica pelo aluno, como possível resposta para atender o enunciado proposto.

Figura 8. Implementação da solução algorítmica (segunda etapa) realizada pelo aluno-aprendiz

A Figura 9 ilustra o resultado da validação realizada pelo dispositivo adaptativo, exibindo, nesse caso, uma mensagem de sucesso. Caso houvessem erros de implementação, o dispositivo adaptativo alertaria quais pontos estão em desacordo com o que fora solicitado no enunciado, cabendo, ao aprendiz, refatorar ou refazer a solução proposta.

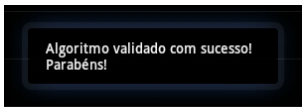


Figura 9. Resultado da validação (terceira etapa) realizada pelo dispositivo adaptativo

E. Experimentos acerca deste projeto

Com a finalidade de contribuir com o desenvolvimento deste projeto, assim como transmitir os conhecimentos gerados e acrescentar novos valores a este trabalho, uma atividade acadêmica fora realizada com grupos de alunos que frequentam a disciplina de Algoritmos e Estruturas de Dados I dos cursos de ADS e BCC.

Durante o evento, ocorrido nas dependências da instituição FEMA no ano de 2012, buscou-se apresentar os objetivos deste artigo, além de permitir o uso e, consequentemente, testes em um ambiente real de ensino com a aplicação totalmente desenvolvida, possibilitando o contato dos alunos com o sistema de aprendizagem de algoritmos.

Após uma introdução acerca dos conceitos envolvidos neste trabalho, assim como da utilização e testes do projeto implementado, alguns questionamentos foram avaliados pelos alunos presentes, com o objetivo de extrair indicadores sobre as opiniões de cada participante.

Os tópicos questionados abordam o uso de dispositivos móveis no processo de aprendizagem, destacando-se a qualidade de ensino, aumento do acesso à educação e a importância de sistemas de aprendizagem nos domínios social, acadêmico e profissional. Os questionamentos, no total de 14, são:

1. Facilidade ao utilizar o dispositivo móvel (smartphone e/ou tablet);
2. Facilidade ao utilizar o sistema de aprendizagem de algoritmos;

3. Ficou motivado a concluir as tarefas propostas pelo sistema de aprendizagem de algoritmos;
4. Utilizaria o sistema de aprendizagem de algoritmos com a finalidade de estudar para trabalhos e provas;
5. A geração dinâmica de problemas e novas situações contribuem para o desenvolvimento intelectual do aluno-aprendiz;
6. Facilidade ao interpretar o problema proposto;
7. Facilidade ao pensar e modelar uma solução para o problema proposto;
8. Facilidade ao solucionar o problema proposto;
9. O uso de dispositivos móveis aumenta a qualidade do processo de ensino-aprendizagem;
10. O uso de aplicações móveis para o ensino aumenta o acesso à educação e à formação profissional;
11. Utilizaria fora das instituições de ensino e/ou empresas um sistema de aprendizagem baseado em Mobile Learning;
12. Por meio de dispositivos e aplicações móveis, os objetivos do processo de ensino-aprendizagem podem ser alcançados;
13. Recomendaria Mobile Learning como uma metodologia de estudo para amigos ou colegas;
14. Professores, alunos, instituições de ensino e/ou empresas devem adotar soluções baseadas em Mobile Learning para a formação social, acadêmica e profissional.

Por meio dos questionamentos, apresentados a um grupo de 23 alunos do curso de ADS, foi possível extrair as estatísticas apresentadas na Tabela II.

TABELA II. ESTATÍSTICAS EXTRAÍDAS DOS QUESTIONAMENTOS AVALIADOS POR ALUNOS DO CURSO DE ADS

Questionamento	Concordam plenamente	Concordam	Não concordam	Incerto
1	17	6	0	0
2	17	6	0	0
3	18	5	0	0
4	18	5	0	0
5	18	5	0	0
6	15	8	0	0
7	11	12	0	0
8	12	11	0	0
9	17	5	0	1
10	17	6	0	0
11	16	7	0	0
12	13	10	0	0
13	17	6	0	0
14	15	8	0	0

Os mesmos questionamentos foram apresentados a um grupo de 27 alunos do curso de BCC, a partir dos quais foi possível extrair as estatísticas apresentadas na Tabela III.

TABELA III. ESTATISTICAS EXTRAIDAS DOS QUESTIONAMENTOS AVALIADOS POR ALUNOS DO CURSO DE BCC

Questionamento	Concordam plenamente	Concordam	Não concordam	Incerto
1	15	12	0	0
2	19	8	0	0
3	11	16	0	0
4	23	4	0	0
5	21	6	0	0
6	17	10	0	0
7	15	11	0	1
8	19	7	0	1
9	13	14	0	0
10	18	9	0	0
11	17	10	0	0
12	12	15	0	0
13	20	7	0	0
14	16	11	0	0

Ao visualizar as estatísticas extraídas das avaliações, pode-se notar o elevado nível de concordância com os questionamentos definidos, provando-se a real importância de trabalhos na área de sistemas de aprendizagem baseados em Mobile Learning.

Outro ponto que merece o devido destaque é a interação e motivação criada pelos alunos ao fazer uso do sistema de aprendizagem de algoritmos, sendo possível observar um grande desejo em continuar utilizando-o, como parte do processo e metodologia de ensino-aprendizagem fornecido pela instituição FEMA, para fornecer uma nova forma de estender e fomentar os estudos além dos limites das aulas presenciais.

A partir de tal atividade acadêmica, as estatísticas e produtos gerados também se integram a este trabalho, tornando-se parte do conjunto de resultados obtidos por meio da realização deste artigo e projeto.

VI. CONCLUSÃO

O aprendizado de algoritmos é um dos grandes desafios na área de ensino de computação, visto que o índice de dificuldades encontradas pelos alunos é bastante elevado. Portanto, torna-se importante o desenvolvimento de sistemas e ambientes que busquem promover o interesse do aluno, assim como facilitar a compreensão dos conceitos de programação.

A utilização de ambientes computacionais de ensino tem crescido e se tornado uma metodologia comum em universidades e centros de estudo, auxiliando alunos e professores no processo de aprendizagem. Nas áreas de computação e tecnologia, o aprendizado de algoritmos e lógica de programação é essencial para a carreira do estudante, contribuindo para que este consiga obter os fundamentos básicos, tornando-o apto a avaliar, pensar, modelar e desenvolver suas próprias soluções algorítmicas para problemas existentes no mundo atual.

Por meio do estudo e projeto realizados, conclui-se que o uso de ferramentas e recursos adequados ao ensino de determinado assunto é importante, permitindo que o aluno participe de experiências acadêmicas que nem sempre são possíveis em sala de aula, além de proporcionar diversas situações novas e dinâmicas a cada etapa do estudo.

Como produtos desta pesquisa, a proposta deste trabalho foi a de desenvolver os estudos teóricos com a finalidade de adquirir os conhecimentos necessários acerca da tecnologia adaptativa aplicados à área de educação, assim como realizar a

implementação de um sistema de aprendizagem de algoritmos baseado na plataforma Google Android.

A partir dessa abordagem inicial, buscou-se construir uma nova ferramenta de ensino, contribuindo com os estudos de alunos de disciplinas iniciais, como Algoritmos e Lógica de Programação, de cursos de computação, com o objetivo de reduzir o índice de dificuldades encontradas, além de incentivar os estudos após as aulas presenciais.

Espera-se que este trabalho venha a contribuir para a pesquisa na área de sistemas adaptativos, além de ampliar o interesse pelo seu uso no contexto de ambientes de aprendizagem, tornando-os dinâmicos e adaptáveis em relação ao processo de ensino-aprendizagem.

A. Trabalhos futuros

Como trabalhos futuros, almeja-se continuar as pesquisas e o desenvolvimento da aplicação, com a finalidade de melhorar e evoluir o sistema de aprendizagem de algoritmos apresentado neste projeto.

Entre alguns dos interessantes pontos a serem estudados, destaca-se a integração do sistema de aprendizagem com um ambiente acadêmico existente, para que as informações sobre a evolução do aluno possam ser enviadas a um servidor, fornecendo novas estatísticas para o professor acompanhar os níveis de aprendizado de uma maneira mais simplificada.

REFERÊNCIAS

[1] E. W. Dijkstra, "On the cruelty of really teaching computing science", Communications of ACM, Vol. 32, pp. 1398-1404, December 1989, in press.

[2] A. J. Gomes, "Ambiente de suporte à aprendizagem de conceitos básicos de programação", Dissertação (Mestrado), Centro de Informática e Sistemas da Universidade de Coimbra, Instituto Superior de Engenharia de Coimbra, Coimbra, Portugal, 2001.

[3] E. S. Almeida, E. B. Costa, J. D. H. Silva, K. S. Paes, A. A. M. Almeida, "AMBAP: Um ambiente de apoio ao aprendizado de programação", X Workshop sobre Educação em Computação, Florianópolis, Santa Catarina, Brasil, Anais do WEI 2002 / SBC 2002, 2002, in press.

[4] M. C. Rodrigues Jr., "Experiências positivas para o ensino de algoritmos", IV Escola Regional de Computação, Feira de Santana, EUFS, Bahia – Sergipe, Brasil, 2004, in press.

[5] D. N. Perkins, S. Schwartz, R. Simmons, "Instructional strategies for the problems of novice programmers", R. E. Mayer (ed.), Teaching and Learning Computer Programming, pp. 153-178, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988, in press.

[6] P. Byrne, G. Lyons, "The effect of student attributes on success in programming", Proceedings of the 6th Annual Conference on Inovation and Technology in Computer Science Education, ITiCSE, United Kingdom, pp. 49-52, 2001, in press.

[7] J. C. R. P. Júnior, C. E. Rapkiewicz, "O processo de ensino e aprendizagem de algoritmos e programação: uma visão crítica da literatura", III Workshop de Educação em Computação e Informática do Estado de Minas Gerais, WEIMIG 2004, Belo Horizonte, Minas Gerais, Brasil, 2004, in press.

[8] J. C. R. P. Júnior, C. E. Rapkiewicz, C. Delgado, J. A. M. Xexeo, "Ensino de algoritmos e programação: uma experiência ao nível médio", XIII Workshop de Educação em Computação, WEI 2005, São Leopoldo, Rio Grande do Sul, Brasil, 2005, in press.

[9] T. Jenkins, "On the difficulty of learning to program", Proceedings of 3rd Annual LTSN_ICS Conference, Loughborough University, United Kindgom, The Higher Education Academy, pp. 53-58, August 27-29, 2002, in press.

[10] J. J. Neto, M. E. S. Magalhães, "Um Gerador Automático de Reconhecedores Sintáticos para o SPD", VIII SEMISH – Seminário de Software e Hardware, pp. 213-228, Florianópolis, Santa Catarina, Brasil, 1981, in press.

[11] J. J. Neto, "Geração automática de analisadores sintáticos para o SPD: evolução e estado da arte", Anais do VI Congresso Nacional de Matemática Aplicada e Computação, ITA, São José dos Campos, São Paulo, Brasil, 1983, in press.

[12] J. J. Neto, "Uma Solução Adaptativa para Reconhecedores Sintáticos", Anais Escola Politécnica, Universidade de São Paulo – Engenharia de Eletricidade – Série B, vol. 1, pp. 645-657, São Paulo, Brasil, 1988, in press.

[13] J. R. Almeida, "STAD: Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos", Tese (Doutorado), Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 1995.

[14] J. M. N. Santos, "Um formalismo adaptativo com mecanismo de sincronização para aplicações concorrentes", Dissertação (Mestrado), Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 1997.

[15] A. R. Camolesi, J. J. Neto, "Modelagem AMBER-Adp de um ambiente para gerenciamento de ensino a distância", Anais do XIII Simpósio Brasileiro de Informática na Educação, SBIE 2002, pp. 401-409, São Leopoldo, Rio Grande do Sul, Brasil, 2002, in press.

[16] A. R. Camolesi, "Proposta de um gerador de ambientes para a modelagem de aplicações usando tecnologia adaptativa", Dissertação (Doutorado), Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, 2007.

[17] A. Yasmin, "Uso de computadores de mão no contexto do sub-projeto Ambiente de Execução direcionado à Pervarsive Computing – EXEHDA", Universidade Católica de Pelotas, Pelotas, Rio Grande do Sul, Brasil, April 2004.

[18] T. Georgiev, E. Georgieva, A. Smrikarov, "M-Learning: A New Stage of E-Learning", International Conference on Computer Systems and Technologies – CompSysTech 2004, Bulgaria, Rousse, June 2004, in press.

[19] L. M. R. Tarouco, M. C. J. M. Fabre, A. R. S. Grando, M. L. P. Konrath, "Objetos de Aprendizagem para M-Learning", Congresso Nacional de Tecnologia da Informação e Comunicação, Florianópolis, Santa Catarina, Brasil, 2004, in press.

Guilherme de Cleva Farto é graduado em Bacharelado em Ciência da Computação pela Fundação Educacional do Município de Assis (2010) e pós-graduado em Engenharia de Componentes utilizando Java pela TNT Educacional e Faculdades Integradas de Ourinhos (2011). Atualmente é analista e desenvolvedor de sistemas Java - Próxima - Software e Serviços, empresa do grupo TOTVS. Também é professor universitário do curso de Bacharelado em Ciência da Computação, junto ao Instituto Municipal de Ensino Superior de Assis (IMESA) e Fundação Educacional do Município de Assis (FEMA), em Assis/SP. Tem experiência na área de Ciência da Computação, com ênfase em desenvolvimento de sistemas e novas tecnologias, atuando principalmente nos seguintes temas: Java, Bancos de Dados, XML, Web Services, Computação em Nuvem, Computação Física com Arduino, Google Android, Google App Engine e integração de sistemas EAI.

Implementação de sistema de aprendizagem de algoritmos com uso de tecnologia adaptativa e plataforma Google Android

G. C. Farto

***Abstract* — The purpose of this article is to present the development of an algorithms learning system based on Google Android platform, in order to provide a new resource of teaching and learning for students of IT courses. Concepts of adaptive technology were used to make the application more dynamic.**

Keywords – sistema de aprendizagem; Mobile Learning; algoritmos; tecnologia adaptativa; Java; Google Android

I. INTRODUÇÃO

[1]As disciplinas de Algoritmos e Lógica de Programação, geralmente lecionadas nas primeiras etapas dos cursos de Tecnologia da Informação (TI), são consideradas desafiadoras por grande parte dos alunos, pois exige a formulação e desenvolvimento de soluções de problemas utilizando-se os conceitos base de matemática e lógica.

Por meio de um levantamento realizado em parceria com a Fundação Educacional do Município de Assis (FEMA) e Instituto Municipal de Ensino Superior de Assis (IMESA), obteve-se indicadores sobre a quantidade de alunos aprovados e reprovados de cursos de tecnologia de 2009, 2010 e 2011.

O gráfico ilustrado na Figura 1 apresenta as estatísticas de alunos aprovados, reprovados com exame, reprovados por frequência e reprovados sem exame que frequentavam a disciplina de Algoritmos e Estruturas de Dados I dos cursos de Análise e Desenvolvimento de Sistemas (ADS) e Tecnologia em Processamento de Dados (TPD) no período analisado.

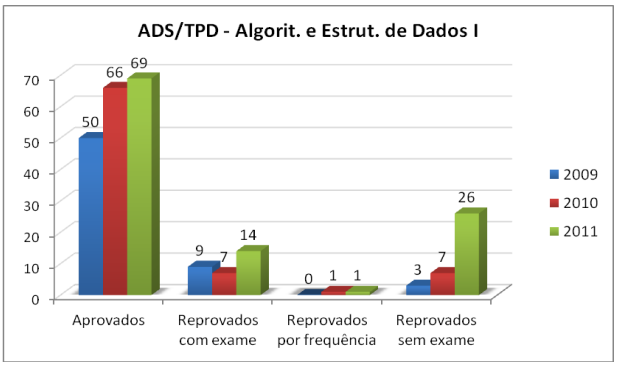


Figura 1. Gráfico de situações para os cursos de ADS e TPD

G. C. Farto, Fundação Educacional do Município de Assis (FEMA), Assis, São Paulo, Brasil, guilherme_computacao@yahoo.com.br

Apesar de ser em quantidade menor, o gráfico ilustrado na Figura 2 apresenta as estatísticas de alunos reprovados que frequentavam a disciplina de Algoritmos e Estruturas de Dados I do curso de Bacharelado em Ciências da Computação (BCC) no período analisado.

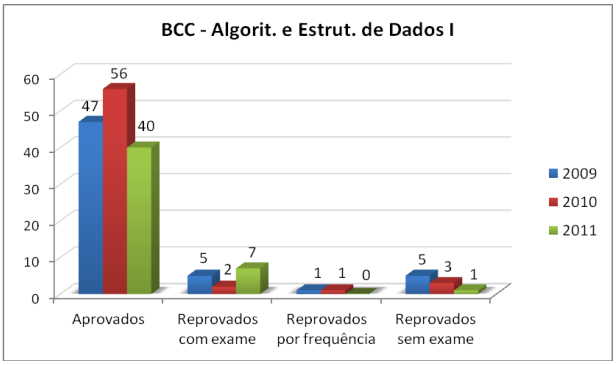


Figura 2. Gráfico de situações para o curso de BCC

Pode-se observar que o índice de reprovações é considerado alto, comparado a outras disciplinas do mesmo curso, entretanto reflete uma situação real da disciplina que é a tendência a dificuldades de aprendizado.

Na busca de soluções para esse problema recorrente nos cursos de tecnologia, diversas pesquisas enfocam tempo e recursos necessários em abstração, projeto e construção de ferramentas computacionais capazes de auxiliar o aluno nas fases iniciais de aprendizagem de algoritmos e lógica.

O principal desafio no processo de ensino de algoritmos e lógica se deve ao fato de que, na maioria das vezes, o conteúdo é aplicado a grupos heterogêneos de participantes, cada qual com seus talentos, modo de trabalhar e pensar, assim como métodos de aprendizagem diferentes. Uma alternativa para solucionar essa dificuldade está na possibilidade do próprio aluno gerir e conduzir boa parte da evolução de seu aprendizado.

Esse modo de aprender pode ser instituído por meio de um sistema de aprendizagem interativo, onde o participante, além de obter conteúdos e instruções de determinado assunto, é capaz de interagir com a aplicação, propondo soluções para resolver um determinado problema apresentado durante o processo de aprendizado.

A partir dessa ideia, os ambientes ou sistemas de aprendizagem interativos devem ser baseados em quatro princípios: o estudante deve construir conhecimento; o controle do sistema é feito, de forma mais significativa, pelo estudante; o sistema é individualizado para cada estudante; e o feedback é gerado em função da interação do estudante com o ambiente.

Atualmente, há uma grande intensificação no desenvolvimento de metodologias de Ensino a Distância (EAD), descrevendo a interação entre professor e aluno durante o processo de ensino-aprendizagem. Porém, em sua grande maioria, a proposta do modelo de ensino é apenas a de transmitir, ao aprendiz, informações e conteúdos definidos pelo tutor por meio de lições pré-estabelecidas.

Sabendo-se da necessidade de metodologias de ensino a distância baseadas na geração de conteúdos em um sistema de aprendizagem, uma alternativa é fazer uso de conceitos da tecnologia adaptativa para realizar a implementação de aplicações mais dinâmicas, contribuindo com o aprendizado do aluno em uma determinada área de conhecimento.

Os resultados proporcionados pelo uso de tecnologia adaptativa são caracterizados por apresentar uma estrutura dinâmica, objetivando a automodificação provocada por interações com o meio externo, sendo ele real ou virtual. Esta habilidade de mudança comportamental é essencial para a construção de máquinas e programas de computadores capazes de evoluir e gerar novas situações com a própria experiência.

O principal diferencial da tecnologia adaptativa é tornar possível, de maneira razoavelmente simples, o reaproveitamento e a ampliação das capacidades de teorias e técnicas existentes e consolidadas, principalmente nas áreas de Inteligência Artificial e Teoria da Computação.

Este artigo aborda o desenvolvimento de um sistema de aprendizagem de algoritmos baseado na plataforma Google Android, com a finalidade de ser uma ferramenta de estudo para alunos das disciplinas de Algoritmos e Lógica de Programação.

Objetivando tornar a aplicação mais dinâmica, fez-se uso de conceitos da tecnologia adaptativa para possibilitar a geração dinâmica de problemas que desafiam o aluno a criar, a cada enunciado proposto, novas soluções computacionais, sem a necessidade de tal situação estar pré-estabelecida.

II. AS DIFICULDADES DE APRENDIZAGEM DE ALGORITMOS

Presente na literatura há diversos trabalhos que identificam justificativas para a dificuldade, quase que intrínseca, na tarefa de aprender os conceitos de lógica e programação.

Entre os argumentos propostos, deve-se considerar que o aprendizado de programação, assim como em outras áreas de conhecimento, é obtido por meio um processo lento e gradual, onde novos fundamentos e ideias, até então desconhecidos, devem ser transformados em algo familiar, facilitando a sua assimilação [1].

Ressalta-se, também, que uma das principais dificuldades reside na técnica de compreender e, em particular, aplicar as noções básicas, como estruturas de controle, para a criação de algoritmos capazes de resolver situações reais [2].

No âmbito de aprendizado de programação, uma possível causa de dificuldade se deve ao fato de que muitos dos alunos não apresentam interesse neste tipo de disciplina. Tal desmotivação pode ser explicada pelo grande volume de conhecimentos abstratos relacionados à atividade de programar, da mesma forma que muitas linguagens e tecnologias estão cada vez mais sofisticadas [3].

Devido às dificuldades enfrentadas pelos alunos, as disciplinas de Algoritmos e Lógica de Programação têm apresentado altos índices de desistência e reprovação. A evasão, além de distanciar o aluno da formação intelectual e profissional, incita a desconfiança sobre a qualidade dos cursos superiores, contribuindo por impedir a entrada de novos alunos, atrasando, consequentemente, o crescimento da área de computação [4].

Para obter sucesso no aprendizado de programação, deve-se realizar um treino intensivo em resolução de problemas e exigir uma precisão e atenção a detalhes muito mais elevada do que a requerida por grande parte de outras disciplinas [1], [5].

Além dos argumentos apresentados anteriormente, alguns trabalhos referem-se que, ao invés de haver uma dificuldade inerente ao aprendizado de algoritmos, há alunos que não possuem as aptidões necessárias para a área de programação, especificamente na resolução de problemas envolvendo lógica e matemática [6], [7], [8].

Entre outras, há várias causas para o insucesso em disciplinas de computação, como a dificuldade de abstração e compreensão, a falta de competências necessárias para resolver problemas, o uso inadequado de metodologias didáticas se comparado ao modelo de aprendizagem dos alunos, além de que as linguagens e tecnologias são detentoras de sintaxes complexas para aprendizes com pouca ou nenhuma experiência [9].

Muitos dos temas relacionados ao processo de aprendizagem são bastante questionáveis, já que o método de ensino abrange tanto alunos quanto professores, assim como as metodologias aplicadas na sala de aula e o certo grau de dificuldade inerente da área de tecnologia.

Há diversos outros motivos, de natureza didática, que podem ser considerados a origem da dificuldade de aprendizado, como o grande número de alunos por turma, dificuldade de o professor compreender a lógica formulada pelo aluno, níveis diferentes de experiência e ritmo entre os alunos, assim como a ausência de bons materiais para disciplinas introdutórias e a própria dificuldade da escolha do curso superior.

Alcançando um novo patamar, também podem ser analisados problemas de naturezas cognitiva, como ausência de perfil necessário para a resolução de problemas, e afetiva, como problemas de ordem pessoal que impedem a concentração durante explanações de conteúdo.

Analisando as dificuldades encontradas por alunos dos cursos de tecnologia e a fundamental importância dessa base inicial de conhecimento e considerando que os modos de pensar e aprender são pessoais e que não é possível nem viável ao tutor adequar-se às necessidades de cada aluno, tornam-se justificáveis a abstração, modelagem e implementação de um sistema de aprendizagem de algoritmos.

III. ESTUDO DE CASO

O processo de ensino utilizando dispositivos computacionais teve início no final da década de 1950, quando o psicólogo americano B. F. Skinner, fundamentado na teoria comportamentalista, propusera uma metodologia de ensino, chamando-a de "máquina de ensinar". Nessa metodologia, um conjunto de conhecimentos a ser ensinado se divide em módulos sequenciais onde o aluno deve responder,

corretamente, as questões propostas para avançar de etapa durante o processo de ensino-aprendizagem.

Por meio desse método, foram desenvolvidas as primeiras aplicações de computador específicas para o ensino, sendo conhecidas como Computer Aided Instruction (CAI) ou Instrução Assistida ou Auxiliada por Computador.

Ao longo do tempo, das pesquisas realizadas e da constante evolução das tecnologias computacionais existentes, diversos sistemas de apoio à aprendizagem de programação foram desenvolvidos, como representações gráficas de algoritmos, sistemas tutores inteligentes, ambientes de aprendizagem a distância, companheiros de aprendizagem, entre outros.

A proposta e um dos objetivos principais para a realização deste trabalho é, além de contribuir de forma teórica para os ambientes computacionais especializados na área de educação, o de implementar um sistema de aprendizagem de algoritmos baseado na plataforma móvel Google Android, disponibilizando um novo recurso de ensino-aprendizagem a ser utilizado para auxiliar e motivar os alunos dentro e fora de instituições de ensino e empresas.

Portanto, com o uso dessa ferramenta, cria-se a possibilidade de os alunos adquirirem e compreenderem as diversas etapas de construção de um algoritmo, além de permitir, a partir de sua implementação, testes, validações e possíveis correções em suas próprias soluções algorítmicas para um determinado problema sugerido pelo sistema tutor.

A. Arquitetura do sistema de aprendizagem de algoritmos

Durante a etapa de análise e modelagem do sistema de aprendizagem de algoritmos, cinco componentes básicos, herdados dos métodos de construção de sistemas CAI, foram identificados:

- Interface: camada responsável por intercambiar ou trocar informações entre o sistema de aprendizagem e o aluno. É por meio da interface que ocorre a interação do aprendiz com os conhecimentos expostos pelo sistema;
- Controlador de eventos: camada responsável por gerenciar e efetuar a troca de informações entre os demais módulos da arquitetura;
- Modelo do aluno: camada responsável por gerenciar o conhecimento do aluno, registrando informações sobre seus acertos e erros, assim como quais conteúdos já foram assimilados;
- Base de domínio: camada responsável por conter e descrever os conhecimentos de um especialista na área de domínio do sistema, contribuindo para a evolução do modelo do aluno;
- Modelo pedagógico: camada responsável por gerenciar as instruções ou regras de ensino, assim como executar um diagnóstico baseado no conhecimento do aluno para decidir quais as estratégias de ensino serão adotadas durante o processo de aprendizagem.

A Figura 3 ilustra a arquitetura simplificada do sistema de aprendizagem de algoritmos proposto neste trabalho, destacando os cinco componentes básicos de uma aplicação de ensino: interface, controlador de eventos, modelo do aluno, base de domínio e modelo pedagógico.

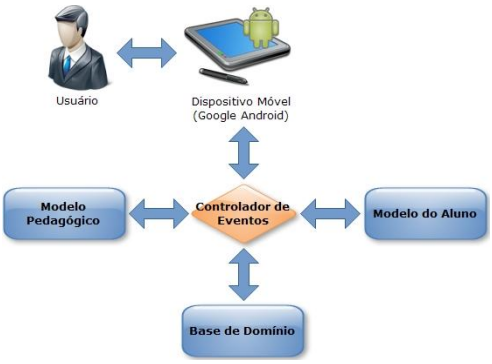


Figura 3. Arquitetura simplificada do sistema de aprendizagem de algoritmos

A Tabela I relaciona todas as instruções ou comandos existentes que podem ser utilizados para a criação de soluções algorítmicas no sistema de aprendizagem proposto neste trabalho. A tipagem de dados, utilizada na definição de constantes e variáveis, é restrita aos valores "booleano", "inteiro", "real" e "texto", enquanto os operadores condicionais são: igual a (==), diferente de (!=), menor que (<), maior que (>), menor ou igual a (<=) e maior ou igual a (>=).

TABELA I. COMANDOS DISPONÍVEIS NO SISTEMA DE APRENDIZAGEM DE ALGORITMOS

Comando	Padrão para utilização
Declaração de Algoritmo	ALGORITMO <nome do programa>
Início de Algoritmo	INICIO:
Fim de Algoritmo	FIM.
Declaração de Área de Constantes	CONSTANTES:
Declaração de Constante	<nome> = <valor> : <tipo>;
Declaração de Área de Variáveis	VARIAVEIS:
Declaração de Variáveis	<nome> : <tipo>;
Atribuição de Valor	<variável> = <valor>;
Estrutura Condicional Se... Então... Senão...	SE (<condição>) ENTAO ... SENAO ... FIM-SE.
Estrutura de Repetição Enquanto... Faça...	ENQUANTO (<condição>) FACA ... FIM-ENQUANTO.
Leitura de Valor	LEIA (<variável>);
Escrita de Valor	ESCREVA (<constante ou variável>);

Após a definição dos comandos a serem disponibilizados na aplicação, tornou-se possível elaborar um diagrama de classes para representar os tipos existentes no sistema de aprendizagem de algoritmos, conforme ilustrado na Figura 4.

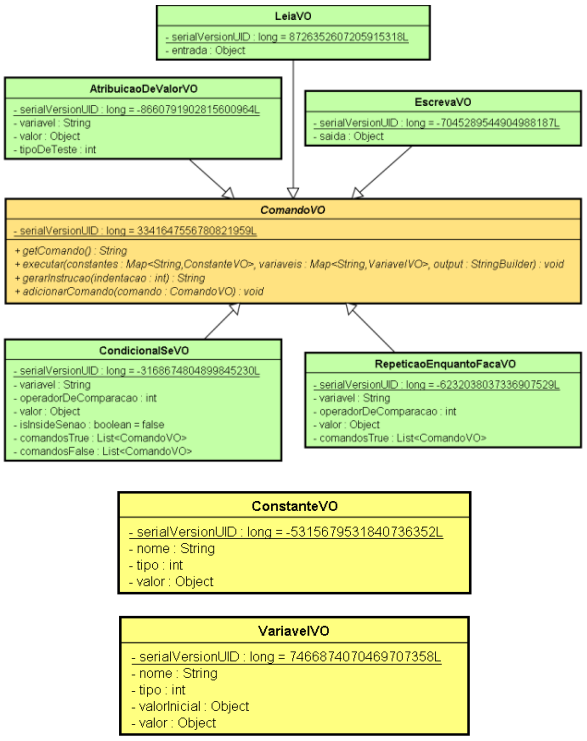


Figura 4. Diagrama de classes elaborado para representar os comandos disponíveis no sistema de aprendizagem de algoritmos

B. *Interação e fluxo de processos*

A interação e o fluxo de processos do sistema de aprendizagem de algoritmos podem ser divididos em três etapas principais, cada qual com suas funcionalidades e responsabilidades, visando contribuir com a aquisição dos conhecimentos necessários durante o processo de ensino-aprendizagem.

Por meio da interface móvel, desenvolvida em um projeto Java baseado na plataforma Google Android, o usuário tem acesso aos recursos providos pela aplicação de ensino de algoritmos, auxiliando-o nas atividades de pensar, modelar e implementar soluções para os problemas gerados pelos dispositivos adaptativos.

O sistema de aprendizagem gerencia, pela camada de modelo do aluno, os níveis de conhecimento do aprendiz na disciplina de algoritmos, sendo possível dividi-los em:

- Nível 1: Implementação de algoritmos estruturados e sequenciais.
- Nível 2: Implementação de algoritmos estruturados e sequenciais, acrescentando conceitos de estrutura condicional (Se... Então... Senão...).
- Nível 3: Implementação de algoritmos estruturados e sequenciais, acrescentando conceitos de estrutura condicional e de repetição (Enquanto... Faça...).

A primeira etapa, presente na interação e fluxo de processos, faz uso do primeiro dispositivo adaptativo implementado no sistema de aprendizagem e objetiva ser responsável pela geração dinâmica de enunciados de algoritmos. Por meio da recuperação de informações do modelo de aluno e com algumas configurações iniciais armazenadas em um arquivo de propriedades, o dispositivo adaptativo elabora um enunciado, solicitando, ao aluno, que implemente uma solução para determinado problema criado, estabelecendo-se a situação inicial do processo de ensino-aprendizagem.

Na segunda etapa, após a geração do enunciado, o aluno inicia a implementação do algoritmo, com o objetivo de fornecer uma solução por meio dos comandos disponíveis na aplicação. A interação entre o aluno e o sistema de aprendizagem resulta em uma fonte de estímulos utilizada para validar os comandos construídos pelo aprendiz, além de ser possível identificar erros durante cada passo do desenvolvimento do algoritmo.

A terceira e última etapa tem por finalidade realizar a validação completa do algoritmo implementado pelo aluno, comparando-o com o enunciado proposto no início do processo de ensino. Com o resultado da validação, fornecido pelo segundo dispositivo adaptativo e amparado por um conjunto de regras estabelecidas para o ensino e a construção de algoritmos, é possível obter informações sobre acertos e erros, além de etapas ou blocos pendentes de implementação.

A Figura 5 ilustra a interação e o fluxo de processos do sistema de aprendizagem, podendo-se notar, claramente, as três etapas citadas, além de permitir um rápido compreendimento de cada elemento e seus relacionamentos dentro do contexto de uma aplicação de ensino de algoritmos.

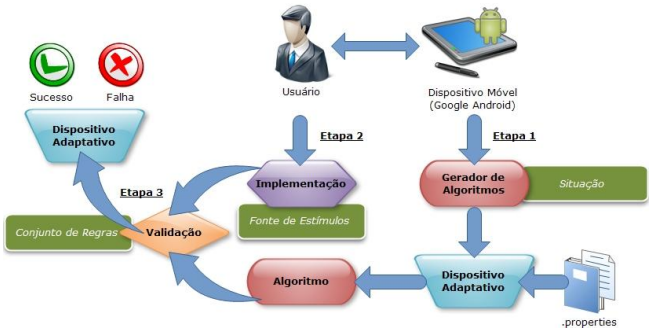


Figura 5. Interação e fluxo de processos do sistema de aprendizagem de algoritmos

C. *Imagens do sistema de aprendizagem de algoritmos*

A Figura 6 ilustra a geração dinâmica, realizada pelo dispositivo adaptativo, de um enunciado de algoritmo, conforme definido na descrição da primeira etapa de interação com o sistema de aprendizagem.

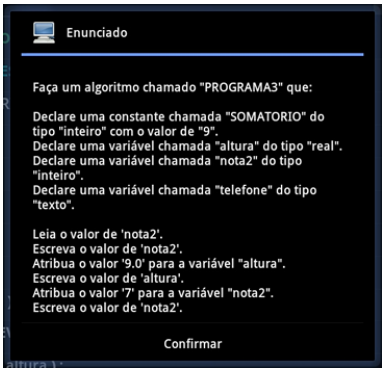


Figura 6. Enunciado de algoritmo gerado dinamicamente (primeira etapa) pelo dispositivo adaptativo

A Figura 7 ilustra a etapa de implementação de uma solução algorítmica pelo aluno, como possível resposta para atender o enunciado proposto.

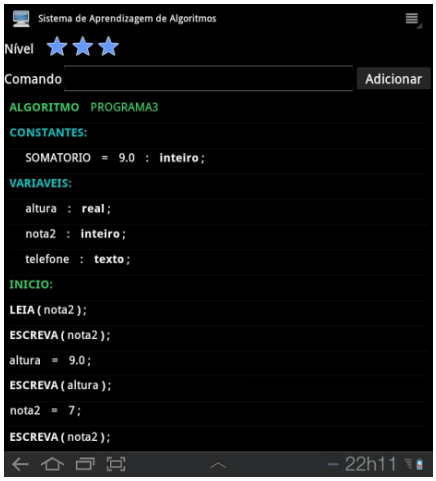


Figura 7. Implementação da solução algorítmica (segunda etapa) realizada pelo aluno-aprendiz

A Figura 8 ilustra o resultado da validação realizada pelo dispositivo adaptativo, exibindo, nesse caso, uma mensagem de sucesso. Caso houvessem erros de implementação, o dispositivo adaptativo alertaria quais pontos estão em desacordo com o que fora solicitado no enunciado, cabendo, ao aprendiz, refatorar ou refazer a solução proposta.

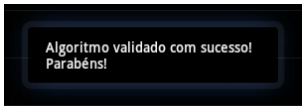


Figura 8. Resultado da validação (terceira etapa) realizada pelo dispositivo adaptativo

D. Experimentos acerca deste projeto

Com a finalidade de contribuir com o desenvolvimento deste projeto, assim como transmitir os conhecimentos gerados e acrescentar novos valores a este trabalho, uma atividade acadêmica, ocorrida nas dependências da instituição FEMA no ano de 2012, fora realizada com grupos de alunos que frequentam a disciplina de Algoritmos e Estruturas de Dados I dos cursos de ADS e BCC.

Os tópicos questionados abordam o uso de dispositivos móveis no processo de aprendizagem, destacando-se a qualidade de ensino, aumento do acesso à educação e a importância de sistemas de aprendizagem nos domínios social, acadêmico e profissional. Os questionamentos, no total de 14, são:

1. Facilidade ao utilizar o dispositivo móvel (smartphone e/ou tablet);
2. Facilidade ao utilizar o sistema de aprendizagem de algoritmos;
3. Ficou motivado a concluir as tarefas propostas pelo sistema de aprendizagem de algoritmos;
4. Utilizaria o sistema de aprendizagem de algoritmos com a finalidade de estudar para trabalhos e provas;
5. A geração dinâmica de problemas e novas situações contribuem para o desenvolvimento intelectual do aluno-aprendiz;
6. Facilidade ao interpretar o problema proposto;
7. Facilidade ao pensar e modelar uma solução para o problema proposto;
8. Facilidade ao solucionar o problema proposto;
9. O uso de dispositivos móveis aumenta a qualidade do processo de ensino-aprendizagem;
10. O uso de aplicações móveis para o ensino aumenta o acesso à educação e à formação profissional;
11. Utilizaria fora das instituições de ensino e/ou empresas um sistema de aprendizagem baseado em Mobile Learning;
12. Por meio de dispositivos e aplicações móveis, os objetivos do processo de ensino-aprendizagem podem ser alcançados;
13. Recomendaria Mobile Learning como uma metodologia de estudo para amigos ou colegas;
14. Professores, alunos, instituições de ensino e/ou empresas devem adotar soluções baseadas em Mobile Learning para a formação social, acadêmica e profissional.

Por meio dos questionamentos, apresentados a um grupo de 23 alunos do curso de ADS, foi possível extrair as estatísticas apresentadas na Tabela II.

TABELA II. ESTATÍSTICAS EXTRAÍDAS DOS QUESTIONAMENTOS AVALIADOS POR ALUNOS DO CURSO DE ADS

Questionamento	Concordam plenamente	Concordam	Não concordam	Incerto
1	17	6	0	0
2	17	6	0	0
3	18	5	0	0
4	18	5	0	0
5	18	5	0	0
6	15	8	0	0
7	11	12	0	0
8	12	11	0	0
9	17	5	0	1
10	17	6	0	0
11	16	7	0	0
12	13	10	0	0
13	17	6	0	0
14	15	8	0	0

Os mesmos questionamentos foram apresentados a um grupo de 27 alunos do curso de BCC, a partir dos quais foi possível extrair as estatísticas apresentadas na Tabela III.

TABELA III. ESTATÍSTICAS EXTRAÍDAS DOS QUESTIONAMENTOS AVALIADOS POR ALUNOS DO CURSO DE BCC

Questionamento	Concordam plenamente	Concordam	Não concordam	Incerto
1	15	12	0	0
2	19	8	0	0
3	11	16	0	0
4	23	4	0	0

5	21	6	0	0
6	17	10	0	0
7	15	11	0	1
8	19	7	0	1
9	13	14	0	0
10	18	9	0	0
11	17	10	0	0
12	12	15	0	0
13	20	7	0	0
14	16	11	0	0

Ao visualizar as estatísticas extraídas das avaliações, pode-se notar o elevado nível de concordância com os questionamentos definidos, provando-se a real importância de trabalhos na área de sistemas de aprendizagem baseados em Mobile Learning.

Outro ponto que merece o devido destaque é a interação e motivação criada pelos alunos ao fazer uso do sistema de aprendizagem de algoritmos, sendo possível observar um grande desejo em continuar utilizando-o, como parte do processo e metodologia de ensino-aprendizagem fornecido pela instituição FEMA, para fornecer uma nova forma de estender e fomentar os estudos além dos limites das aulas presenciais.

IV. CONCLUSÃO

O aprendizado de algoritmos é um dos grandes desafios na área de ensino de computação, visto que o índice de dificuldades encontradas pelos alunos é bastante elevado. Portanto, torna-se importante o desenvolvimento de sistemas e ambientes que busquem promover o interesse do aluno, assim como facilitar a compreensão dos conceitos de programação.

A utilização de ambientes computacionais de ensino tem crescido e se tornado uma metodologia comum em universidades e centros de estudo, auxiliando alunos e professores no processo de aprendizagem. Nas áreas de computação e tecnologia, o aprendizado de algoritmos e lógica de programação é essencial para a carreira do estudante, contribuindo para que este consiga obter os fundamentos básicos, tornando-o apto a avaliar, pensar, modelar e desenvolver suas próprias soluções algorítmicas para problemas existentes no mundo atual.

Por meio do estudo e projeto realizados, conclui-se que o uso de ferramentas e recursos adequados ao ensino de determinado assunto é importante, permitindo que o aluno participe de experiências acadêmicas que nem sempre são possíveis em sala de aula, além de proporcionar diversas situações novas e dinâmicas a cada etapa do estudo.

Como produtos desta pesquisa, a proposta deste trabalho foi a de desenvolver os estudos teóricos com a finalidade de adquirir os conhecimentos necessários acerca da tecnologia adaptativa aplicados à área de educação, assim como realizar a implementação de um sistema de aprendizagem de algoritmos baseado na plataforma Google Android.

A partir dessa abordagem inicial, buscou-se construir uma nova ferramenta de ensino, contribuindo com os estudos de alunos de disciplinas iniciais, como Algoritmos e Lógica de Programação, de cursos de computação, com o objetivo de reduzir o índice de dificuldades encontradas, além de incentivar os estudos após as aulas presenciais.

Espera-se que este trabalho venha a contribuir para a pesquisa na área de sistemas adaptativos, além de ampliar o interesse pelo seu uso no contexto de ambientes de aprendizagem, tornando-os dinâmicos e adaptáveis em relação ao processo de ensino-aprendizagem.

A. *Trabalhos futuros*

Como trabalhos futuros, almeja-se continuar as pesquisas e o desenvolvimento da aplicação, com a finalidade de melhorar e evoluir o sistema de aprendizagem de algoritmos apresentado neste projeto.

Entre alguns dos interessantes pontos a serem estudados, destaca-se a integração do sistema de aprendizagem com um ambiente acadêmico existente, para que as informações sobre a evolução do aluno possam ser enviadas a um servidor, fornecendo novas estatísticas para o professor acompanhar os níveis de aprendizado de uma maneira mais simplificada.

REFERÊNCIAS

[1] E. W. Dijkstra, “On the cruelty of really teaching computing science”, Communications of ACM, Vol. 32, pp. 1398-1404, December 1989, in press.

[2] A. J. Gomes, “Ambiente de suporte à aprendizagem de conceitos básicos de programação”, Dissertação (Mestrado), Centro de Informática e Sistemas da Universidade de Coimbra, Instituto Superior de Engenharia de Coimbra, Coimbra, Portugal, 2001.

[3] E. S. Almeida, E. B. Costa, J. D. H. Silva, K. S. Paes, A. A. M. Almeida, “AMBAP: Um ambiente de apoio ao aprendizado de programação”, X Workshop sobre Educação em Computação, Florianópolis, Santa Catarina, Brasil, Anais do WEI 2002 / SBC 2002, 2002, in press.

[4] M. C. Rodrigues Jr., “Experiências positivas para o ensino de algoritmos”, IV Escola Regional de Computação, Feira de Santana, EUFS, Bahia – Sergipe, Brasil, 2004, in press.

[5] D. N. Perkins, S. Schwartz, R. Simmons, “Instructional strategies for the problems of novice programmers”, R. E. Mayer (ed.), Teaching and Learning Computer Programming, pp. 153-178, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988, in press.

[6] P. Byrne, G. Lyons, “The effect of student attributes on success in programming”, Proceedings of the 6th Annual Conference on Inovation and Technology in Computer Science Education, ITiCSE, United Kingdom, pp. 49-52, 2001, in press.

[7] J. C. R. P. Júnior, C. E. Rapkiewicz, “O processo de ensino e aprendizagem de algoritmos e programação: uma visão crítica da literatura”, III Workshop de Educação em Computação e Informática do Estado de Minas Gerais, WEIMIG 2004, Belo Horizonte, Minas Gerais, Brasil, 2004, in press.

[8] J. C. R. P. Júnior, C. E. Rapkiewicz, C. Delgado, J. A. M. Xexeo, “Ensino de algoritmos e programação: uma experiência ao nível médio”, XIII Workshop de Educação em Computação, WEI 2005, São Leopoldo, Rio Grande do Sul, Brasil, 2005, in press.

[9] T. Jenkins, “On the difficulty of learning to program”, Proceedings of 3rd Annual LTSN_ICS Conference, Loughborough University, United Kindgom, The Higher Education Academy, pp. 53-58, August 27-29, 2002, in press.

Guilherme de Cleva Farto é graduado em Bacharelado em Ciência da Computação pela Fundação Educacional do Município de Assis (2010) e pós-graduado em Engenharia de Componentes utilizando Java pela TNT Educacional e Faculdades Integradas de Ourinhos (2011). Atualmente é analista e desenvolvedor de sistemas Java - Próxima - Software e Serviços, empresa do grupo TOTVS. Também é professor universitário do curso de Bacharelado em Ciência da Computação, junto ao Instituto Municipal de Ensino Superior de Assis (IMESA) e Fundação Educacional do Município de Assis (FEMA), em Assis/SP. Tem experiência na área de Ciência da Computação, com ênfase em desenvolvimento de sistemas e novas tecnologias, atuando principalmente nos seguintes temas: Java, Bancos de Dados, XML, Web Services, Computação em Nuvem, Computação Física com Arduino, Google Android, Google App Engine e integração de sistemas EAI.

Linguístico: Uma Proposta de Reconhecedor Gramatical Usando Tecnologia Adaptativa

Ana Contier, Djalma Padovani, João José Neto

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

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

Autômatos Adaptativos

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

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

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

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

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

Processamento da Linguagem Natural: Revisão da Literatura

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

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

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

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

Casa: verbo singular, 3a pessoa, presente indicativo, 1ª conjugação. Significado: contrair matrimônio

Dada uma determinada sentença, o analisador léxico-morfológico identifica os itens lexicais que a compõem e obtém, para cada um deles, as diferentes descrições correspondentes às entradas no léxico. A ambiguidade léxico-morfológica ocorre quando uma mesma palavra apresenta diversas categorias gramaticais. Neste caso existem duas formas de análise: a tradicional e a etiquetagem. Pela abordagem tradicional, todas as classificações devem ser apresentadas pelo analisador, deixando a resolução de ambiguidade para outras etapas do processamento. Já pela etiquetagem (POS *Tagging*), o analisador procura resolver as ambiguidades sem necessariamente passar por próximas etapas de processamento. Nesta abordagem, o analisador recebe uma cadeia de itens lexicais e um conjunto específico de etiquetas como entrada e produz um conjunto de itens lexicais com a melhor etiqueta associada a cada item. Os algoritmos para etiquetagem fundamentam-se em dois modelos mais conhecidos: os baseados em regras e os estocásticos. Os algoritmos baseados em regras usam uma base de regras para identificar a categoria de um item lexical, acrescentando novas regras à base à medida que novas situações de uso do item vão sendo encontradas. Os algoritmos baseados em métodos estocásticos costumam resolver as ambiguidades através de um corpus de treino marcado corretamente, calculando a probabilidade que uma palavra terá de receber uma etiqueta em um determinado contexto.

O passo seguinte é a análise sintática. Nesta etapa, o analisador verifica se uma sequência de palavras constitui uma frase válida da língua, reconhecendo-a ou não. O analisador sintático faz uso de um léxico e de uma gramática, que define as regras de combinação dos itens na formação das frases. A gramática adotada pode ser escrita por meio de diversos formalismos. Segundo [7] destacam-se as redes de transição, as gramáticas de constituintes imediatos (PSG ou phrase structure grammar), as gramáticas de constituintes imediatos generalizadas (GPSG) e as gramáticas de unificação funcional (PATR II e HPSG). As gramáticas de constituintes imediatos (PSG), livres de contexto, apresentam a estrutura sintática das frases em termos de seus constituintes. Por exemplo, uma frase (F) é formada pelos sintagmas nominal (SN) e verbal (SV). O sintagma nominal é um agrupamento de palavras que tem como núcleo um substantivo (Subst) e o sintagma verbal é um agrupamento de palavras que tem como núcleo um verbo. Substantivo e verbo representam classes gramaticais. O determinante (Det) compõe, junto com o substantivo, o sintagma nominal. O sintagma verbal é formado pelo verbo, seguido ou não de um sintagma nominal. O exemplo apresentado ilustra uma gramática capaz de reconhecer a frase: O menino usa o chapéu.

F → SN SV.
SN → Det Subst.
SV → Verbo SN.
Det → o
Subst → menino, chapéu
Verbo → usa

Considerando o processamento da direita para esquerda e de baixo para cima, a frase seria analisada da seguinda forma:

O menino usa o chapéu
1. chapéu = Subst: O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo: O menino verbo SN
5. verbo SN=SV: O menino SV
6. menino = Subst: O subst SV
7. O = Det: Det subst SV
8. Det subst= SN: SN SV
9. SN SV= F: F (Aceita)

No entanto, este formalismo não consegue identificar questões de concordância de gênero e número. Por exemplo, se fossem incluídos no léxico o plural e o feminino da palavra menino, frases como: "O meninos usa o chapéu." e "O menina usa o chapéu." seriam aceitas. Por exemplo:

O meninos usa o chapéu
1. chapéu = Subst: O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo: O menino verbo SN
5. verbo SN=SV: O menino SV
6. meninos = Subst: O subst SV
7. O = Det: Det subst SV
8. Det subst= SN: SN SV
9. SN SV= F: F (Aceita)

Para resolver este tipo de problema existem outros formalismos, tais como o PATR II:

F → SN, SV
  <SN numero> = <SV numero>
  <SN pessoa> = <SV pessoa>
SN → Det, Subst
  <Det numero> = <Subst numero>
  <Det genero > = <Subst genero>
SV → Verbo, SN
o
  <categoria> = determinante
  <genero> = masc
  <numero> = sing
menino
  <categoria> = substantivo
  <genero> = masc
  <numero> = sing
chapéu
  <categoria> = substantivo
  <genero> = masc
  <numero> = sing
usa
  <categoria> = verbo
  <tempo> = pres
  <numero> = sing
  <pessoa> = 3
  <argumento 1> = SN
  <argumento 2> = SN

Neste formalismo, a derivação leva em consideração outras propriedades do léxico, além da categoria gramatical, evitando os erros de reconhecimento apresentados anteriormente. Segundo [7], esse formalismo gramatical oferece poder gerativo e capacidade computacional, e tem sido usado com sucesso em ciência da computação, na especificação de linguagens de programação. Aplicando este formalismo ao exemplo acima, o erro de concordância seria identificado e a frase não seria aceita:

O meninos usa o chapéu

1. chapéu = Subst masc sing : O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo pres 3a. Pessoa sing: O menino verbo SN
5. verbo SN=SVsing: O menino SVsing
6. meninos = Subst masc plural: O subst SVsing
7. O = Det: Det subst SVsing
8. Det subst= SNplur: SNplur SVsing
9. SNplur SVsing = Não aceita

Certas aplicações necessitam lidar com a interpretação das frases bem formadas, não bastando o conhecimento da estrutura, mas sendo necessário o conhecimento do significado dessas construções. Por exemplo, quando é necessário que respostas sejam dadas a sentenças ou orações expressas em língua natural, as quais, por exemplo, provoquem um movimento no braço de um robô. Ou quando é necessário extrair conhecimentos sobre um determinado tema a partir de uma base de dados textuais. Nos casos nos quais há a necessidade de interpretar o significado de um texto, a análise léxico-morfológica e a análise sintática não são suficientes, sendo necessário realizar um novo tipo de operação, denominada análise semântica [7].

Na análise semântica procura-se mapear a estrutura sintática para o domínio da aplicação, fazendo com que a estrutura ganhe um significado [8]. O mapeamento é feito identificando as propriedades semânticas do léxico e o relacionamento semântico entre os itens que o compõe. Para representar as propriedades semânticas do léxico, pode ser usado o formalismo PATR II, já apresentado anteriormente. Para a representação das relações entre itens do léxico pode ser usado o formalismo baseado em predicados: cada proposição é representada como uma relação predicativa constituída de um predicado, seus argumentos e eventuais modificadores. Um exemplo do uso de predicados é apresentado para ilustrar o processo de interpretação da sentença "O menino viu o homem de binóculo". Trata-se de uma sentença ambígua da língua portuguesa, uma vez que pode ser interpretada como se (a) O menino estivesse com o binóculo, ou (b) O homem estivesse com o binóculo. Uma gramática para a análise do exemplo acima é dada pelas seguintes regras de produção:

F → SN SV
SN → Det Subst
SN → SN SP
SV → V SN
SV → V SN SP
SP → Prep Subst

Uma possível representação semântica para as interpretações da sentença seria:

1.Sentença de interpretação (a):
agente(ação(ver), menino)
objeto(ação(ver), homem)
instrumento(ação(ver), binóculo)
2.Sentença de interpretação (b):
agente(ação(ver), menino)
objeto(ação(ver), homem)
qualificador(objeto(homem), binóculo)

Existem casos em que é necessário obter o conteúdo não literal de uma sentença, ligando as frases entre si, de modo a construir um todo coerente, e interpretar a mensagem transmitida de acordo com a situação e com as condições do enunciado [7]. Por exemplo, para uma compreensão literal da sentença: "O professor disse que duas semanas são o tempo necessário", é possível recorrer aos mecanismos de representação expostos até aqui, porém para uma compreensão aprofundada, seria necessário saber a que problema se refere o professor, já que o problema deve ter sido a própria razão da formulação dessa sentença. Nestes casos, é necessária uma nova operação denominada análise pragmática.

A análise pragmática procura reinterpretar a estrutura que representa o que foi dito para determinar o que realmente se quis dizer [2]. Dois pontos focais da pragmática são: as relações entre frases e o contexto. À medida que vão sendo enunciadas, as sentenças criam um universo de referência, que se une ao já existente. A própria vizinhança das sentenças ou dos itens lexicais também constitui um elemento importante na sua interpretação. Assim, alguns novos fenômenos passam a ser estudados, como fenômenos pragmático-textuais. Inserem-se nessa categoria as relações anafóricas, co-referência, determinação, foco ou tema, dêiticos e elipse [7]. Por exemplo, nem sempre o caráter interrogativo de uma sentença expressa exatamente o caráter de solicitação de uma resposta. A sentença "Você sabe que horas são?" pode ser interpretada como uma solicitação para que as horas sejam informadas ou como uma repreensão por um atraso ocorrido. No primeiro caso, a pergunta informa ao ouvinte que o falante deseja obter uma informação e, portanto, expressa exatamente o caráter interrogativo. Entretanto, no segundo caso, o falante utiliza o artifício interrogativo como forma de impor sua autoridade. Diferenças de interpretação desse tipo claramente implicam interpretações distintas e, portanto, problemáticas, se não for considerado o contexto de ocorrência do discurso [9]. As questões relacionadas à análise pragmática são objetos de estudos de modo a prover mecanismos de representação e de inferência adequados, e raramente aparecem em processadores de linguagem natural [7].

Em [10] são apresentados trabalhos de pesquisas em processamento de linguagem natural para a Língua Portuguesa tais como o desenvolvido pelo Núcleo Interinstitucional de Linguística Aplicada (NILC) no desenvolvimento de ferramentas para processamento de linguagem natural; o projeto VISL – Visual Interactive Syntax Learning, sediado na Universidade do Sul da Dinamarca, que engloba o desenvolvimento de analisadores morfossintáticos para diversas línguas, entre as quais o português; e o trabalho de resolução de anáforas desenvolvido pela Universidade de Santa Catarina. A tecnologia adaptativa também tem contribuído com trabalhos em processamento da linguagem natural. Em [11], são apresentadas algumas das pesquisas desenvolvidas pelo Laboratório de Linguagens e Tecnologia

Adaptativa da Escola Politécnica da Universidade de São Paulo: um etiquetador morfológico, um estudo sobre processos de análise sintática, modelos para tratamento de não-determinismos e ambiguidades, e um tradutor texto voz baseado em autômatos adaptativos.

RECONHECEDOR ADAPTATIVO: SUPORTE TEÓRICO LINGUÍSTICO

A Moderna Gramática Brasileira de Celso Luft [12] foi escolhida como suporte teórico linguístico do reconhecedor aqui proposto. A escolha foi feita em função da forma clara e precisa com que Luft categoriza os diversos tipos de sentenças de língua portuguesa, se diferenciando das demais gramáticas que priorizam a descrição da língua em detrimento da análise estrutural da mesma.

Luft diz que a oração é moldada por padrões denominados frasais ou oracionais. Estes padrões são compostos por elementos denominados sintagmas. Sintagma é qualquer constituinte imediato da oração, podendo exercer papel de sujeito, complemento (objeto direto e indireto), predicativo e adjunto adverbial. É composto por uma ou mais palavras, sendo que uma é classificada como núcleo e as demais como dependentes. As palavras dependentes podem estar localizadas à esquerda ou à direita do núcleo. Luft utiliza os seguintes nomes e abreviaturas:

1. Sintagma substantivo (SS): núcleo é um substantivo;
2. Sintagma verbal (SV): núcleo é um verbo;
3. Sintagma adjetivo (Sadj): núcleo é um adjetivo;
4. Sintagma adverbial (Sadv): núcleo é um advérbio;
5. Sintagma preposicional (SP): é formado por uma preposição (Prep) mais um SS.
6. Vlig: verbo de ligação
7. Vi: verbo intransitivo
8. Vtd: verbo transitivo direto
9. Vti: verbo transitivo indireto
10. Vtdi: verbo transitivo direto e indireto
11. Vt-pred: verbo transitivo predicativo

A Tabela 1 apresenta os elementos formadores dos sintagmas, e a sequência em que aparecem, de acordo com Luft.

TABELA 1.
Elementos formadores de sintagmas [12]

Sintagmas	
Substantivo	Quantitativos+Pronomes Adjetivos+ Sintagma Adjetivo1+Substantivo+ Sintagma Adjetivo2+ Sintagma Preposicional+ Oração Adjetiva
Verbal	Pré-verbais+ Verbo Auxiliar+ Verbo Principal
Adjetivo	Advérbio de Intensidade+ Adjetivo+ Sintagma Preposicional
Adverbial	Advérbio de Intensidade+ Adverbio+ Sintagma Preposicional
Preposicional	Preposição+ Sintagma Substantivo

Um padrão oracional é determinado pelos tipos de sintagmas e pela sequência em que aparecem. Por exemplo, o padrão oracional SS Vlig SS, indica que a frase é composta por um sintagma substantivo, seguido de um verbo de ligação e de outro sintagma substantivo. A Tabela 2 apresenta a relação de todos os padrões oracionais propostos por Luft.

Os padrões são classificados em 5 tipos:

1. Padrões pessoais nominais: Neste caso, existe sujeito e o núcleo do predicado é um nome (substantivo, adjetivo, advérbio) ou um pronome (substantivo, adjetivo, advérbio). O verbo, nesses casos, é chamado de verbo de ligação (Vlig).

TABELA 2
Padrões oracionais de Luft [12]

Padrões Pessoais Nominais				
SS	Vlig	SS		
SS	Vlig	Sadj		
SS	Vlig	Sadv		
SS	Vlig	SP		
Padrões Pessoais Verbais				
SS	Vtd	SS		
SS	Vti	SP		
SS	Vti	Sadv		
SS	Vti	SP	SP	
SS	Vtdi	SS	SP	
SS	Vtdi	SS	Sadv	
SS	Vtdi	SS	SP	SP
SS	Vi			

2. Padrões pessoais verbais: São aqueles nos quais existe o sujeito e o núcleo do predicado é um verbo. O verbo pode ser transitivo direto (Vtd), transitivo indireto (Vti), transitivo direto e indireto (Vtdi), e intransitivo (Vi). Se o verbo for transitivo direto (Vtd), o complemento será um objeto direto; se o verbo for transitivo indireto (Vti), o complemento será um objeto indireto; se o verbo for transitivo direto e indireto (Vtdi), o complemento será um objeto direto e um indireto; se o verbo for intransitivo (Vi), não há complemento.

TABELA 2 - CONTINUAÇÃO

Padrões Pessoais Verbo-Nominais				
SS	Vtpred	SS	SS	
SS	Vtpred	SS	Sadj	
SS	Vtpred	SS	SP	
SS	Vtpred	SS	Sadv	
SS	Vtpred	SS		
SS	Vtpred	Sadj		
SS	Vtpred	SP		
Padrões Impessoais Nominais				
	Vlig	SS		
	Vlig	Sadj		
	Vlig	Sadv		
	Vlig	SP		

TABELA 2 - CONTINUAÇÃO

Padrões Impessoais Verbais				
	Vtd	SS		
	Vti	SP		
	Vi			

3. Padrões Pessoais Verbo-Nominais: Neste caso, existe o sujeito e o núcleo do predicado é um verbo transitivo predicativo (Vt-pred), cujo complemento é um objeto direto e um predicativo do objeto.

4. Padrões Impessoais Nominais: Ocorrem quando não existe sujeito e o núcleo do predicado é um nome (substantivo, adjetivo, advérbio) ou um pronome (substantivo, adjetivo, advérbio).

5. Padrões Impessoais Verbais: Neste caso, não existe sujeito e o núcleo do predicado é um verbo.

Luft apresenta uma gramática usada para análise sintática da Língua Portuguesa no modelo moderno, em que as frases são segmentadas o mais binariamente possível: Sujeito+Predicado; Verbo+Complemento; Substantivo+Adjetivo, etc. Neste modelo, a descrição explicita somente as classes analisadas; as funções ficam implícitas. Querendo explicar estas, Luft sugere que sejam escritas à direita das classes: SS:Sj (Sujeito), V:Núc (Núcleo do Predicado), PrA:NA (Adjunto Adnominal), etc.
A gramática proposta por Luft é a seguinte:

F → [Conec] [SS] SV [Conec]
Conec → F
SS → [Sadj] SS [Sadj | SP]
SS → [Quant | PrA] (Sc | Sp | PrPes)
SV→ [Neg] [Aux | PreV] (Vlig | Vtd| Vti | Vtdi| Vi)
[SS | Sadj | Sadv | SP] [SS| Sadj| Sadv| SP] [SP]
SP → Prep (SS | Sadj)
Sadj → Sadj [SP]
Sadj →[Adv] Adj
Sadv → Sadv [SP]
Sadv → [Adv] Adv
PrA → Ind | ArtDef | ArtInd| Dem| Pos
Sendo:
F – Frase
SS – Sintagma substantivo
SV – Sintagma verbal
SP – Sintagma preposicional
SN – Sintagma nominal
Sadv – Sintagma adverbial
Sadj – Sintagma adjetivo
Adv – adverbio
Adj – adjetivo
ArtDef – artigo definido
ArtInd – artigo indefinido
Aux – Partícula auxiliar (apassivadora ou pré-verbal)
Conec – Conector (conjunção ou pronome relativo)
Dem – pronome demonstrativo indefinido
Ind – pronome indefinido
Neg – partícula (negação)
PrA – pronome adjetivo
PrPes – pronome pessoal
Prep – preposição
Quant – numeral
Sc – substantivo comum
Sp – substantivo próprio

V– verbo
Vlig – verbo de ligação
Vi – verbo intransitivo
Vtd – verbo transitivo direto
Vti – verbo transitivo indireto
Vtdi – verbo transitivo direto e indireto

PROPOSTA DE UM RECONHECEDOR GRAMATICAL

O Linguístico é uma proposta de reconhecedor gramatical composto de 5 módulos sequenciais que realizam cada qual um processamento especializado, enviando o resultado obtido para o módulo seguinte, tal como ocorre em uma linha de produção, até que o texto esteja completamente analisado.

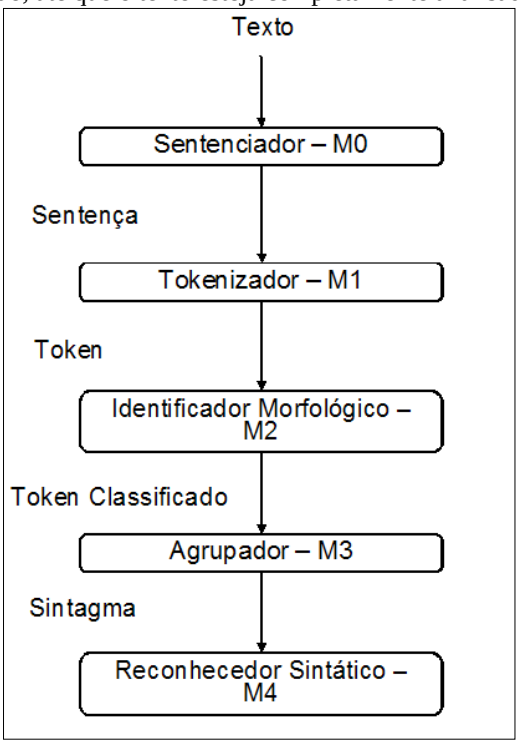


Figura 2. Estrutura do Linguístico

A Fig.2 ilustra a estrutura do Linguístico. O primeiro módulo, denominado Sentenciador, recebe um texto e realiza um pré-processamento, identificando os caracteres que possam indicar final de sentença, palavras abreviadas e palavras compostas, e eliminando aspas simples e duplas. Ao final, o Sentenciador divide o texto em supostas sentenças, para análise individual nas etapas seguintes.

O segundo módulo, denominado Tokenizador, recebe as sentenças identificadas na etapa anterior e as divide em *tokens*, considerando, neste processo, abreviaturas, valores monetários, horas e minutos, numerais arábicos e romanos, palavras compostas, nomes próprios, caracteres especiais e de pontuação final. Os *tokens* são armazenados em estruturas de dados (*arrays*) e enviados um a um para análise do módulo seguinte.

O terceiro módulo, denominado Identificador Morfológico, recebe os *tokens* da etapa anterior e os identifica morfologicamente, utilizando, como biblioteca de apoio, os

textos pré-anotados do corpus Bosque[13], os verbos, substantivos e adjetivos que fazem parte da base de dados do TeP2.0 – Thesauro Eletrônico para o Português do Brasil [14] e as conjunções, preposições e pronomes disponíveis no Portal São Francisco[15], cujas informações provém da Wikipedia [16]. O Bosque é um conjunto de frases anotadas morfossintaticamente (conhecido por *treebank)*, composto por 9368 frases retiradas dos primeiros 1000 extratos dos corpora CETEMPublico (Corpus de Extractos de Textos Electrónicos MCT/Público) e CETENFolha (Corpus de Extractos de Textos Electrónicos NILC/Folha de S. Paulo). A Fig. 3 apresenta um fragmento do Bosque.

```
#9363 CF997-3 Segundo declarações do próprio diretor, ele vive até hoje de forma angustiada:
(STA+fcl (ADVL+pp
                    (H+prp Segundo)
                    (P<+np
                      (H+n declarações)
                      (N<+pp
                        (H+prp de)
                        (P<+np
                          (>N+art o)
                          (>N+pron-det próprio)
                          (H+n diretor)))))
                    (,)
                    (SUBJ+pron-pers ele)
                    (P+v-fin vive)
                    (ADVL+advp
                      (>A+prp até)
                      (H+adv hoje))
                    (ADVL+pp
                      (H+prp de)
                      (P<+np
                        (H+n forma)
                        (N<+v-pcp angustiada)))
                    (.))
```

Figura 3. Exemplo de frase etiquetada do corpus Bosque [11].

O TeP2.0 é um dicionário eletrônico de sinônimos e antônimos para o português do Brasil, que armazena conjuntos de formas léxicas sinônimas e antônimas. É composto por 19.888 conjuntos de sinônimos, 44.678 unidades lexicais, e 4.276 relações de antonímia [14].

O Portal São Francisco apresenta um curso online da Língua Portuguesa e, entre seus módulos, encontra-se um sumário das classes morfológicas, no qual são encontrados exemplos de palavras e locuções mais comuns de cada classe [15].

Inicialmente, o Identificador Morfológico procura pela classificação morfológica dos tokens no léxico do Bosque, caso não a encontre, então ele procura na base de dados do TeP2.0 e no léxico do Portal São Francisco.

O padrão de etiquetas usado pelo Linguístico é o mesmo do Bosque, que apresenta, além da classificação morfológica, o papel sintático que o token exerce na sentença pré-anotada. Por exemplo, a etiqueta >N+art indica que o token é um artigo que está à esquerda de um substantivo (>N). A notação usa como gramática subjacente a Gramática Constritiva proposta por Fred Karlsson [17].

O léxico do Bosque foi organizado de modo a relacionar todas as classificações de um tokens ordenadas por frequência em que ocorrem no texto pre-anotado. Por exemplo, o token "acentuada" está classificado com as seguintes etiquetas: N<+v-pcp e P+v-pcp, o que significa que, no texto pré-anotado, ele foi classificado como verbo no particípio (+v-pcp) antecedido de um substantivo (N<) e como verbo no particípio no papel de predicador (P).

No caso de ambiguidade, o Identificador Morfológico assume a classificação mais frequente como inicial e verifica se a classificação mais frequente do token seguinte é consistente com o que indica a etiqueta do token analisado. Se for, vale a classificação mais frequente, senão o Identificador analisa a próxima classificação, repetindo o algoritmo. Caso o algoritmo não retorne uma classificação única, o Identificador passa todas as classificações encontradas para os módulos seguintes, para que ambiguidade seja resolvida pelas regras gramaticais do reconhecedor.

O quarto módulo, denominado Agrupador é composto de um autômato, responsável pela montagem dos sintagmas a partir de símbolos terminais da gramática e um bigrama, responsável pela montagem dos sintagmas a partir de não-terminais (Fig.4). Inicialmente, o Agrupador recebe do Identificador as classificações morfológicas dos *tokens* e as agrupa em sintagmas de acordo com a gramática proposta por Luft. Neste processo são identificados sintagmas nominais, verbais, preposicionais, adjetivos e adverbiais Para isso, o Agrupador utiliza um autômato adaptativo cuja configuração completa é definida da seguinte forma:

Estados = {1, 2, 3, 4, SS, SP, V, Sadj, Sadv, A},
Onde:
1,2,3 e 4 = Estados Intermediários
SS, SP, V Sadj, Sadv = Estados nos quais houve formação de sintagmas, sendo:
SS= Sintagma substantivo
SP = Sintagma preposicional
V = Verbo ou locução verbal
Sadj = Sintagma adjetivo
Sadv = Sintagma adverbial
A = Estado após o processamento de um ponto final
Tokens = {art, num, n, v, prp, pron, conj, adj, adv, rel, pFinal, sClass}, onde:
art = artigo, num = numeral
n = substantivo, v = verbo
prp = preposição, pron = pronome
conj = conjunção, adj = adjetivo
adv = advérbio, rel = pronome relativo
pFinal = ponto final, sClass = sem classificação
Estados de Aceitação = {SS, SP, V, Sadj, Sadv, A}
Estado Inicial = {1}
Função de Transição = {(Estado, *Token*)→Estado}, sendo:
{(1, art)→2, (2, art)→2, (3, art)→3
(1, num)→Sadv, (2, num)→2, (3, num)→3
(1, n)→SS, (2, n)→SS, (3, n)→SP
(1, v)→SV, (2, v)→ SV, (3, v)→SP
(1, prp)→3, (2, prp)→2, (3, prp)→3
(1, prop)→SS, (2, prop)→SS, (3, prop)→SP
(1, pron)→SS, (2, pron)→SS, (3, pron)→SP
(1, conj)→conj, (2, conj)→Ø, (3, conj)→ Ø
(1, adj)→Sadj, (2, adj)→Sadj, (3, adj)→3
(1, adv)→Sadv, (2, adv)→2, (3, adv)→3
(1, rel)→conj, (2, rel)→ Ø, (3, rel)→conj
(1, pFinal)→A, (2, pFinal)→ Ø, (3, pFinal)→ Ø}

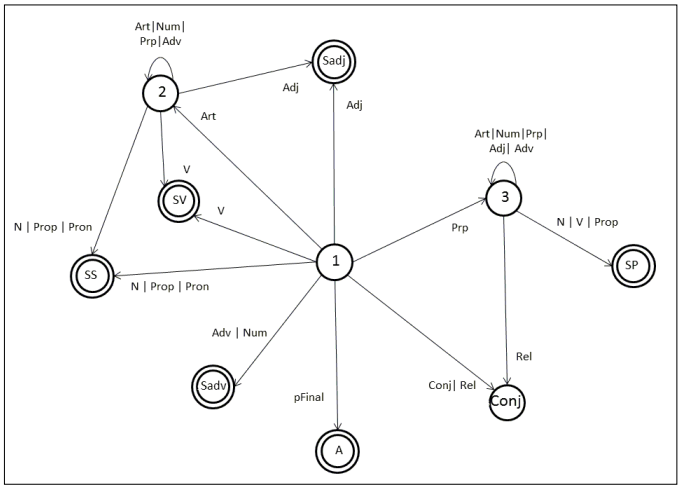


Figura 4. Configuração Completa do Autômato Construtor de Sintagmas.

Por exemplo, segundo a gramática de Luft, os sintagmas substantivos são obtidos através da seguinte regra:

SS → [Quant | PrA] (Sc | Sp | PrPes)

Pela regra acima, o conjunto de *tokens* "A" e "casa" formam um sintagma substantivo, da seguinte forma:

PrA = "A" (artigo definido)

Sc = "casa" (substantivo comum)

Da direita para esquerda, são realizadas as seguintes derivações:

PrA Sc → SS; A Sc → SS; A casa → SS

Já o Agrupador recebe o *token* "A", identificado pelo Tokenizador como artigo definido, e se movimenta do estado 1 para o estado 2. Ao receber o *token* "casa", identificado como substantivo comum, ele se movimenta do estado 2 para o estado SS, que é um estado de aceitação. Neste momento o Agrupador armazena a cadeia "A casa" e o símbolo "SS" em uma pilha e reinicializa o autômato preparando-o para um novo reconhecimento.

Em um passo seguinte, o Agrupador usa o bigrama para comparar um novo sintagma com o último sintagma formado, visando identificar elementos mais altos na hierarquia da gramática de Luft. Para isso ele usa a matriz apresentada na Tabela 3, construída a partir da gramática de Luft. A primeira coluna da matriz indica o último sintagma formado (US) e a primeira linha, o sintagma atual (SA). A célula resultante apresenta o novo nó na hierarquia da gramática.

TABELA 3
Matriz de agrupamento de sintagmas

US \ SA	SS	SP	V	Sadv	Sadj	Conj
SS	SS	SS	-	-	SS	-
SP	SP	-	-	-	-	-
V	-	-	V	-	-	-
Sadv	-	-	-	Sadv	Sadj	-
Sadj	SS	Sadj	-	-	Sadj	-
Conj	-	-	-	-	-	Conj

Esta técnica foi usada para tratar as regras gramaticais nas quais um sintagma é gerado a partir da combinação de outros, como é o caso da regra de formação de sintagmas substantivos: SS → [Sadj] SS [Sadj | SP]. Por esta regra, os sintagmas substantivos são formados por outros sintagmas substantivos precedidos de um sintagma adjetivo e seguidos de um sintagma adjetivo ou um sintagma preposicional. No exemplo anterior, supondo que os próximos 2 *tokens* fossem "de" e "madeira", após a passagem pelo autômato, o Agrupador formaria um sintagma SP. Considerando que na pilha ele tinha armazenado um SS, após a passagem pelo bigrama, e de acordo com a Tabela 3, o sintagma resultante seria um SS e o conteúdo que o compõe seria a combinação dos textos de cada sintagma que o originou. Caso não haja agrupamentos possíveis, o Agrupador envia o último sintagma formado para análise do Reconhecedor Sintático e movimenta o sintagma atual para a posição de último sintagma no bigrama, repetindo o processo com o próximo sintagma.

O quinto e último módulo, denominado Reconhecedor Sintático, recebe os sintagmas do módulo anterior e verifica se estão sintaticamente corretos de acordo com padrões gramaticais de Luft. O Reconhecedor Sintático utiliza um autômato adaptativo que faz chamadas recursivas sempre que recebe conjunções ou pronomes relativos, armazenando, em uma estrutura de pilha, o estado e a cadeia de sintagmas reconhecidos até o momento da chamada. Caso o Reconhecedor Sintático não consiga se movimentar a partir do sintagma recebido, ele gera um erro e retorna o ponteiro para o último sintagma reconhecido, finalizando a instância do autômato recursivo e retornando o processamento para aquela que a inicializou. Esta, por sua vez, retoma posição em que se encontrava antes da chamada e continua o processamento até o final da sentença ou até encontrar uma nova conjunção, situação na qual o processo se repete.

A configuração completa do autômato é definida da seguinte forma:

Estados = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27}
Tokens = {SS, SP, Vli, Vi, Vtd, Vti, Vtdi, Sadj, Sadv, Conj, A}
Estados de Aceitação = { 4, 5, 6, 9, 12, 13,14, 15, 17, 18, 19, 21, 22, 24, 25, 26, 27}
Estado Inicial = {1}

Função de Transição = {(Estado, *Token*)→Estado}, sendo:
{(1, SS)→2, (2, Vti)→3, (3, SP)→4, (4, SP)→4,
(3, Sadv)→5, (2, Vi)→6, (2, Vtdi)→7
(7, SS)→8, (8, SP)→9, (9, SP)→9, (8, Sadv)→10,
(2, Vlig)→11, (11, SP)→12, (11, Sadv)→13,
(11, Sadj)→ 14, (11, SS)→15, (2, Vtd)→16, (16, SS)→17,
(2, Vtpred)→18, (18, SP)→19, (18, Sadj)→20
(18, SS)→ 21, (21, SS)→22, (21, Sadj)→23, (21, Sadv)→24,
(21, SP)→25}

Pilha = {[Texto, Sintagma, Estado]}

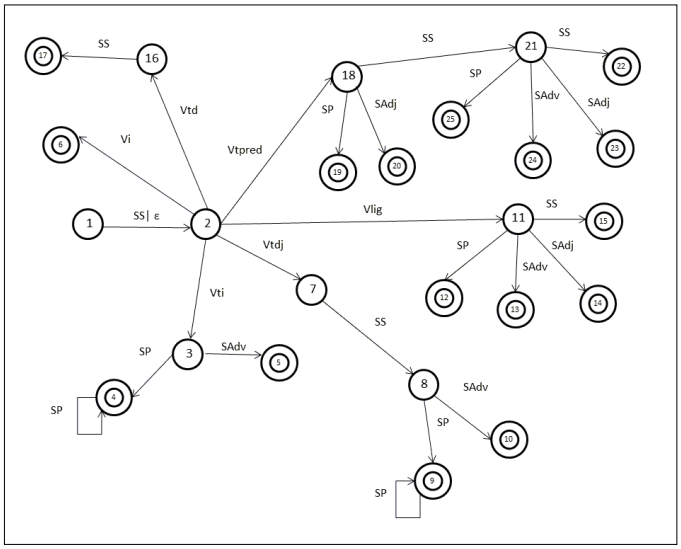


Figura 5.1. Configuração Completa do Reconhecedor Sintático.

No entanto, para que a análise sintática seja feita, não são necessárias todas as ramificações da configuração completa do autômato (Fig. 5.1). Por exemplo, quando se transita um verbo de ligação a partir do estado 2, o autômato vai para o estado 11 e todas as demais ramificações que partem deste estado para os estados 3, 7, 16 e 18, não são usadas. Com a tecnologia adaptativa, é possível criar dinamicamente os estados e transições do autômato em função dos tipos de verbos, evitando manter ramificações que não são usadas.

A Fig. 5.2 apresenta a configuração inicial do autômato adaptativo equivalente ao autômato de pilha apresentado anteriormente. No estado 1, o autômato recebe os *tokens* e transita para o estado 2 quando processa um sintagma substantivo (SS) ou quando transita em vazio. No estado 2, o autômato transita para si mesmo quando recebe qualquer tipo de verbo: Vi, Vtd, Vlig, Vtpred, Vtdi e Vti. Todas as outras ramificações são criadas por meio de funções adaptativas chamadas em função do tipo de verbo processado.

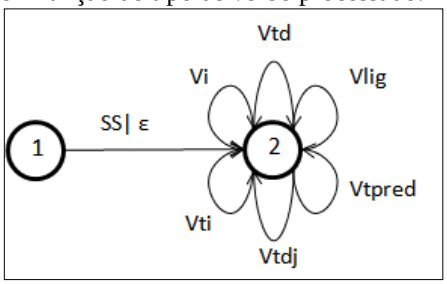


Figura 5.2. Configuração Inicial do Reconhecedor Gramatical.

Por exemplo, se o verbo é de ligação (Vlig), o autômato utiliza as funções adaptativas α (j) e β (o), definidas da seguinte forma:

α (j): { o* :
- [(j, Vlig)]
+[(j, Vlig) :→ o, β (o)]
}

β (o): { t*u*v*x*:
+ [(o, SP) :→ t]
+ [(o, Sadv) :→ u]
+ [(o, Sadj) :→ v]
+ [(o, SS) :→ x]
}

A função adaptativa α (j) é chamada pelo autômato antes de processar o *token*, criando o estado 11 e a produção que leva o autômato do estado 2 ao novo estado criado. Em seguida, o autômato chama a função β (o), criando os estados 12, 13, 14 e 15 e as produções que interligam o estado 11 aos novos estados. A Fig. 5.3 mostra a configuração do autômato após o processamento do verbo de ligação. Neste exemplo, o autômato criou apenas os estados 11, 12, 13, 14 e 15 e as respectivas transições, evitando alocar recursos que seriam necessários para criar o autômato completo, conforme apresentado na Fig. 5.1.

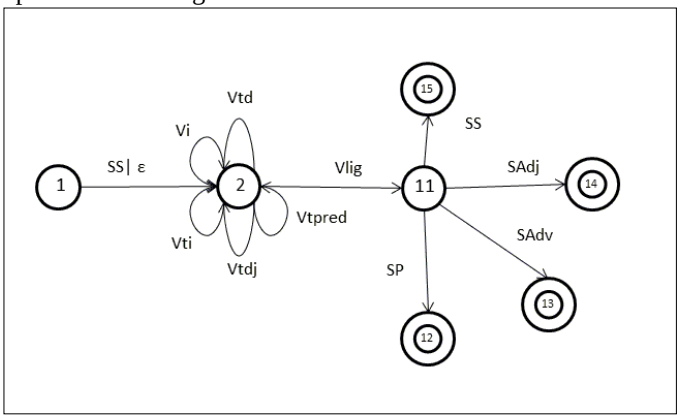


Figura 5.3. Configuração do autômato após o processamento do verbo de ligação.

Toda movimentação do autômato, assim como os sintagmas identificados em cada passagem e a classificação morfológica dos termos das sentenças, são armazenados em arquivos que podem ser acessados por um editor. Se o Linguístico não consegue reconhecer a sentença, ele registra os erros encontrados e grava uma mensagem alertando para o ocorrido.

Desenvolvimento do Linguístico – Preparação do Texto

O Linguístico está sendo desenvolvido na linguagem de programação Python[18], escolhida devido aos recursos nativos de processamento de expressões regulares, programação dinâmica e por oferecer flexibilidade de implementação tanto no paradigma procedural quanto na orientação a objetos. O primeiro componente desenvolvido, chamado Texto, cuida da preparação do texto que vai ser analisado. Ele é formado por uma classe chamada Texto que incorpora o Sentenciador e o Tokenizador do Linguístico. A classe Texto possui 3 métodos que são acionados em sequência, sempre que encontra um novo texto para ser analisado. O método Prepara_Texto se encarrega de eliminar aspas simples e duplas, e identificar abreviaturas e palavras compostas na base de dados formada pelos léxicos Bosque e TeP2.0. O método Divide_Texto cria uma coleção de sentenças através de expressões regulares que interpretam como caracteres limitadores de cada sentença o ponto final, de interrogação e de exclamação, seguidos de um espaço em branco. Ao final, o método Tokeniza_Sentenca cria uma coleção de tokens a partir das sentenças, usando como critérios de divisão, regras para identificação de tokens alfanuméricos, valores monetários, horas e minutos, numerais arábicos e romanos, percentuais, além das abreviaturas e palavras compostas identificadas na etapa de preparação.

Desenvolvimento do Linguístico- O Identificador Morfológico

O Identificador Morfológico encontra-se em fase de projeto e foi concebido para utilizar tecnologias adaptativas (Fig.6.1). O Identificador Morfológico é composto por um Autômato Mestre

e um conjunto de submáquinas especialistas. O Autômato Mestre é responsável pelo sequenciamento das chamadas às submáquinas, de acordo com um conjunto de regras cadastradas em base de dados.

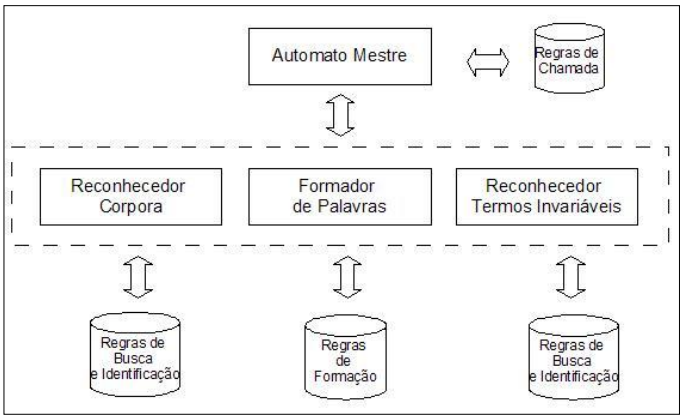


Figura 6.1. Arquitetura do Identificador Morfológico.

A prioridade é obter as classificações morfológicas do corpora (no caso, o Bosque, mas poderia ser outro, se houvesse necessidade); caso o termo procurado não exista no corpora, o Autômato Mestre procura por substantivos, adjetivos e verbos, no formato finito e infinito (flexionados e não flexionados), através de uma submáquina de formação e identificação de palavras, que usa, como base, o vocabulário do TeP2.0 (aqui também existe a possibilidade de usar outra base de dados, substituindo ou complementando o TeP2.0); por fim, o Autômato Mestre procura por termos invariáveis, ou seja, termos cuja classificação morfológica é considerada estável pelos linguistas, tais como, conjunções, preposições e pronomes, no caso, extraídos no léxico do Portal São Francisco. A Fig. 6.2 apresenta a estrutura adaptativa do Autômato Mestre.

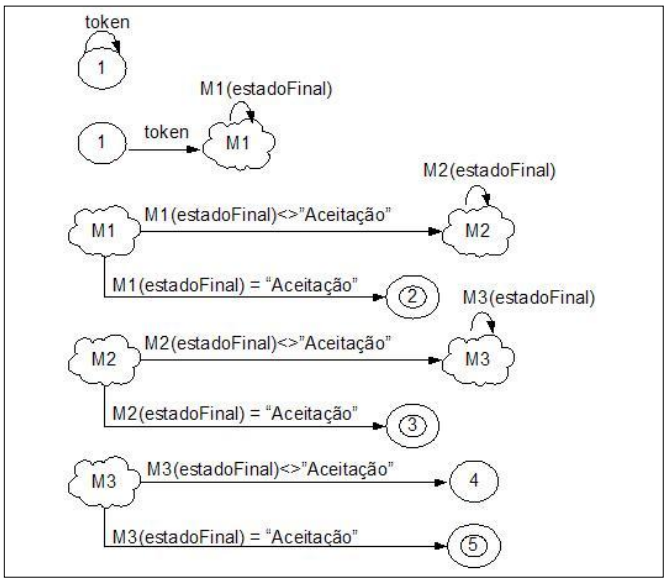


Figura 6.2. Estrutura Adaptativa do Autômato Mestre.

O Autômato Mestre inicia seu processamento recebendo o token da coleção de tokens criada pela classe Texto. Em seguida, antes de processá-lo, o autômato se modifica através de uma função adaptativa, criando uma submáquina de processamento (M1), uma transição entre o estado 1 e M1, e uma transição que aguarda o estado final da submáquina M1. O token é passado para M1 e armazenado em uma pilha. Quando M1 chega ao estado final, o autômato se modifica novamente, criando uma nova submáquina M2, o estado 2 e as transições correspondentes. Caso o estado final de M1 seja de aceitação, o processo é finalizado no estado 2, caso contrário a máquina M2 é chamada, passando o token armazenado na pilha. O processo se repete quando M2 chega ao final do processamento, com a criação da submáquina M3, do estado 3 e das transições correspondentes. Um novo ciclo se repete e se o estado final de M3 é de aceitação, o autômato transiciona para o estado 5, de aceitação, caso contrário, ele vai para o estado 4 de não aceitação. M1, M2 e M3 representam, respectivamente, as submáquinas do Reconhecedor de Corpora, do Formador de Palavras e do Reconhecedor de Termos Invariáveis. Portanto, caso o Autômato Mestre encontre a classificação morfológica ao final de M1, ele não chama M2; caso encontre em M2, não chama M3 e, caso também não encontre em M3, ele informa aos demais módulos do Linguístico que não há classificação morfológica para o termo analisado.

As submáquinas M1, M2 e M3 também foram projetadas de acordo com a tecnologia adaptativa. A Máquina M1 usa um autômato adaptativo que se automodifica de acordo com o tipo de token que está sendo analisado: palavras simples, palavras compostas, números, valores e símbolos. A Fig. 6.3 apresenta a estrutura adaptativa de M1.

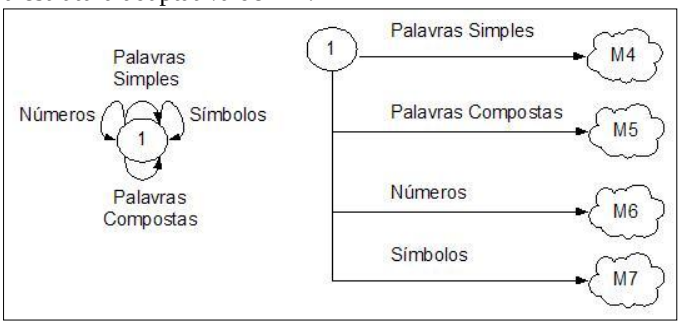


Figura 6.3. Estrutura Adaptativa do Reconhecedor de Corpora M1.

Inicialmente o autômato é composto por um único estado e por transições para ele mesmo (Fig.6.3, à esquerda). Ao identificar o tipo de token que será analisado (obtido no processo de tokenização), o autômato cria submáquinas e as transições correspondentes. As alternativas de configuração são apresentadas na Fig.6.3, à direita. As submáquinas M4, M5, M6 e M7 reconhecem os tokens através de outro tipo de autômato que processa os tokens byte a byte (Fig. 6.4).

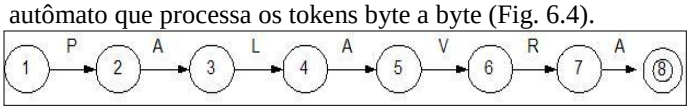


Figura 6.4. Reconhecedor de Palavras Simples.

No exemplo apresentado na Fig.6.4, o token "Palavra" é processado pela máquina M4; se o processamento terminar em um estado de aceitação, o token é reconhecido. As submáquinas M4, M5, M6 e M7 são criadas previamente por um programa que lê o corpora e o converte em autômatos finitos determinísticos. A estrutura de identificação morfológica é composta por um par [chave, valor], no qual a chave é o estado de aceitação do elemento lexical e, o valor, sua classificação morfológica. No exemplo apresentado, a chave do item lexical "palavra" seria o estado "8", e, o valor, a

classificação morfológica, H+n (substantivo, núcleo de sintagma nominal), proveniente do corpora Bosque.

O Formador de Palavras, submáquina M2, também é montado previamente por um programa construtor, levando em consideração as regras de formação de palavras do Português do Brasil, descritas por Margarida Basilio em [19]. Segundo a autora, o léxico é constituído por uma lista de formas já feitas e por um conjunto de padrões, os processos de formação de palavras, que determinam estruturas e funções tanto das formas já existentes quanto de formas ainda a serem construídas.

A submáquina M2 utiliza o vocabulário do TeP2.0 (formado por substantivos, adjetivos e verbos), como léxico de formas já feitas e o conjunto de regras de prefixação, sufixação e regressão verbal descrito pela autora para construir novas formas. No entanto, são necessários alguns cuidados para evitar a criação de estruturas que aceitem palavras inexistentes. A Fig. 6.5 apresenta um exemplo de autômato no qual são aplicados os prefixos "a" e "per", e os sufixos "ecer" e "ar" (derivação parassintética) ao radical "noit" do substantivo noite, que faz parte do TeP2.0.

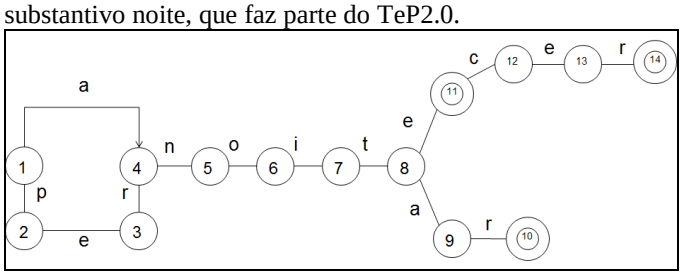


Figura 6.5. Autômato Formador de Palavras.

No exemplo apresentado, as palavras "anoitecer", "pernoitar e "anoitar" (sinônimo de "anoitecer") existem no léxico do português do Brasil. Já pernoitecer é uma combinação que não existe na língua portuguesa. Margarida Basílio diz que algumas combinações não são aceitas simplesmente porque já existem outras construções consagradas pelo uso [20]. No entanto, acrescenta, existem centenas de substantivos terminados em –ção que não admitem formação adjetiva com o sufixo –oso. Por exemplo, vocação/vocacioso; intenção/intencioso; atração/atracioso. Segundo a autora, as derivações mais prováveis são aquelas em que há generalidade das funções envolvidas no processo de formação. Noções como a negação, o grau e a nominalização de verbos são bastante comuns e de grande generalidade, consequentemente, mais prováveis de serem válidas.

Para reduzir o risco de aceitar derivações inexistentes, o processo de construção do autômato restringe as possíveis formações, utilizando apenas as regras que a autora destaca como sendo mais prováveis. É o caso de nominalização de verbos com o uso dos sufixos –ção , –mento e – da. Em [19], Basílio cita que o sufixo –ção é responsável por 60% das formações regulares, enquanto o sufixo –mento é responsável por 20% destas formações. Já o sufixo –da é, via de regra, usado em nominalizações de verbos de movimento, tais como, entrada, saída, partida, vinda, etc.

Outra característica do construtor do autômato é representar os prefixos e sufixos sempre pelo mesmo conjunto de estados e transições, evitando repetições que acarretariam o consumo desnecessário de recursos computacionais. A Fig. 6.6 apresenta exemplo de palavras formadas por reutilização de estados e transições na derivação das palavras rueira, novidadeira e noveleira. Os estados e transições usados para representar o sufixo "eira", usado para designar são os mesmos nas três derivações.

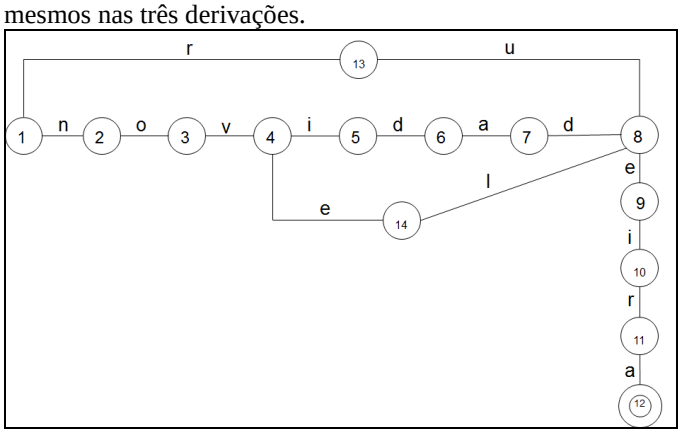


Figura 6.6. Autômato Formador de Palavras.

A submáquina M2 também reconhece palavras flexionadas, obtidas, no caso de substantivos e adjetivos, através da aplicação de sufixos indicativos de gênero, número e grau aos radicais do vocabulário TeP2.0. A Fig. 6.7 apresenta um exemplo de autômato usado para a formação das formas flexionadas do substantivo menino. Foram adicionados ao radical "menin" as flexões "o(s)", "a(s)", "ão", "ões", "onona(s)", "inho(s)" e"inha(s)".

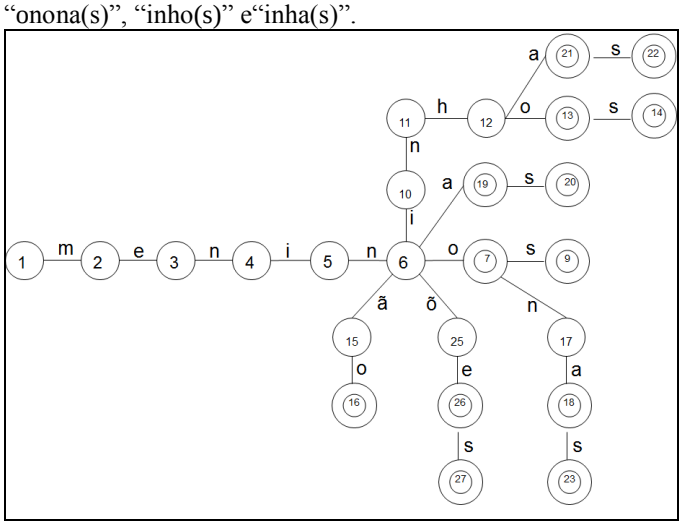


Figura 6.7. Autômato Formador de Flexões Nominais.

No caso de verbos, foi criada uma estrutura de estados e transições para representar as flexões de tempo, modo, voz e pessoa, obtendo-se, assim, as respectivas conjugações.

A Fig. 6.8 apresenta um exemplo de autômato usado para a formação das formas flexionadas do presente do indicativo do verbo andar. Foram adicionados ao radical "and" as flexões "o", "as", "a", "andamos", "ais" e "andam".

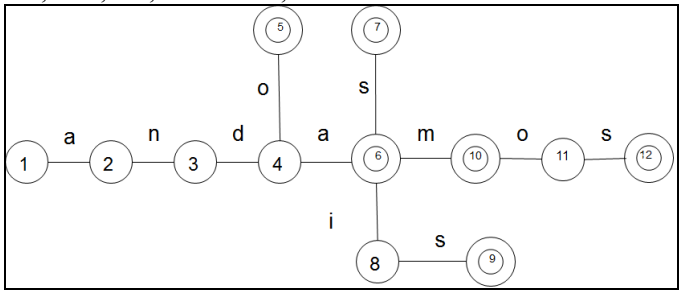


Figura 6.8. Autômato Formador de Flexões Verbais.

A estrutura de armazenamento da submáquina M2 também é composta por um par [chave, valor], no qual a chave é o estado de aceitação do elemento lexical e, o valor, sua classificação morfológica, acrescida da origem do termo, indicando que ele foi gerado pelo construtor. Por exemplo, o termo "novidadeira" seria classificado como H+n+C – substantivo, núcleo de sintagma nominal, gerado pelo construtor.

Já a submáquina M3 é um autômato que varia em função do tipo de termo (conjunções, preposições e pronomes) e utiliza uma estrutura arbórea similar a M4. A Fig. 6.9 apresenta um exemplo de autômato usado no reconhecimento de termos deste domínio.

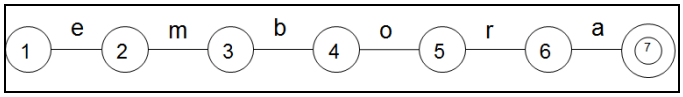


Figura 6.9. Autômato Reconhecedor de Conjunções, Preposições e Pronomes.

A estrutura de armazenamento da submáquina M3 é composta por um par [chave, valor], no qual a chave é o estado de aceitação do elemento lexical e, o valor, sua classificação morfológica, acrescida da origem do termo, indicando que ele faz parte da base de dados de apoio do Portal São Franscisco. Por exemplo, o termo "embora" seria classificado como cj+Ba –conjunção da base de dados de apoio.

Considerações Finais

Este artigo apresentou uma revisão dos conceitos de Tecnologia Adaptativa e de Processamento da Linguagem Natural. Em seguida, foi apresentado o Linguístico, uma proposta de reconhecedor gramatical que utiliza autômatos adaptativos como tecnologia subjacente.

O Linguístico encontra-se em fase de construção, dividida em etapas em função da estrutura do reconhecedor. A primeira versão do sentenciador e do tokenizador foram finalizadas. O trabalho encontra-se na fase de projeto do Identificador Morfológico, que foi totalmente concebido para utilizar tecnologia adaptativa.

Referências

[1] Neto, J.J. Apresentação LTA-Laboratório de Linguagens e Técnicas Adaptativas. Disponível em: http://www.pcs.usp.br/~lta. Acesso 01/11/2009.
[2] Taniwaki, C. Formalismos adaptativos na análise sintática de Linguagem natural . Dissertação de Mestrado, EPUSP, São Paulo, 2001.
[3] Menezes, C. E. Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos. Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo, 2000.
[4] Padovani, D. Uma proposta de autômato adaptativo para reconhecimento de anáforas pronominais segundo algoritmo de Mitkov. Workshop de Tecnologias Adaptativas – WTA 2009, 2009.
[5] Moraes, M. de Alguns aspectos de tratamento sintático de dependência de contexto em linguagem natural empregando tecnologia adaptativa, Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, 2006.
[6] Rich, E.; Knight, K. Inteligência Artificial, 2. Ed. São Paulo: Makron Books, 1993.
[7] Vieira, R.; Lima, V Linguística computacional: princípios e aplicações. IX Escola de Informática da SBC-Sul, 2001.
[8] Fuchs, C., Le Goffic, P. Les Linguistiques Contemporaines.
[9] Nunes, M. G. V. et al. Introdução ao Processamento das Línguas Naturais. Notas didáticas do ICMC Nº 38, São Carlos, 88p, 1999. Paris, Hachette, 1992. 158p.
[10] Sardinha, T. B. A Língua Portuguesa no Computador. 295p. Mercado de Letras, 2005.
[11] Rocha, R.L.A. Tecnologia Adaptativa Aplicada ao Processamento Computacional de Língua Natural. Workshop de Tecnologias Adaptativas – WTA 2007, 2007.
[12] Luft, C. Moderna Gramática Brasileira. 2ª. Edição Revista e Atualizada. 265p. Editora Globo, 2002.
[13] Linguateca : http://www.linguateca.pt/
[14]Tep2. Thesauro Eletrônico para o Português do Brasil. Disponível em: <http://www.nilc.icmc.usp.br/tep2/>
[15] Portal São Francisco. Materiais de Língua Portuguesa. Disponível em: <http://www.portalsaofrancisco.com.br/alfa/materias/index-lingua- portuguesa.php/>
[16] Wikipedia. Disponível em: <http://pt.wikipedia.org/wiki/P%C3%A1gina_principal/>
[17] Karlsson, F. Constraint Grammar as a Framework for Parsing Running Text. 13o. International Conference on Computational Linguistics, Helsinki (Vol.3, pp.168-173).
[18] Python. Python Programming Language Official Web Site. Disponível em: <http://www.python.org/>
[19] Basilio, M. Formação e Classes de Palavras no Português do Brasil. Ed.Contexto, 2004
[20] Basilio, M. Teoria Lexical. Ed.Atica, 1987

Ana Teresa Contier: formada em Letras-Português pela Universidade de São Paulo (2001) e em publicidade pela PUC-SP (2002). Em 2007 obteve o título de mestre pela Poli-USP com a dissertação: "Um modelo de extração de propriedades de textos usando pensamento narrativo e paradigmático".

Djalma Padovani nasceu em São Paulo em 1964. Cursou bacharelado em Física pelo Instituto de Física da Universidade de São Paulo, formou-se em administração de empresas pela Faculdade de Economia e Administração da Universidade de São Paulo, em 1987 e obteve o mestrado em engenharia de software pelo Instituto de Pesquisas Tecnológicas de São Paulo - IPT, em 2008. Trabalhou em diversas empresas nas áreas de desenvolvimento de software e tecnologia de informação e atualmente é responsável pela arquitetura tecnológica da Serasa S/A, empresa do grupo Experían.

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

Aplicação de metodologias de auto-organização em futebol de robôs

J.R.Ribeiro Junior, A. R. Hirakawa and C. A. Sakurai

***Abstract*— This paper presents a research based on what are the self-organization methods in the current literature, and how it can be modified to fit in a different scenario, in which, different of what happen in the common self-organization scenario, as in a robot swarm, the cells don't need to have an uniform behavior and each cell can have individual and different actions. Another pursuit is the homogeneity and simplicity of the cells, generating more generalized and amplifying the available scenarios. To test this methodology the robot soccer scenario was the choice, as it have all the desired features, such as simple cells, individual and collective actions by those cells, becoming an ideal scenario to be evaluated and validate the method.**

***Keywords*— Self-Organization, Digital Hormone System, Robot Soccer, Cellular automaton.**

I. INTRODUÇÃO

Os sistemas que se auto-organizam e as metodologias de organização são objeto de estudo de cientistas de diversas áreas até os dias atuais, pois a sua aplicabilidade não se restringe apenas a um tipo de sistema, podendo esse ser matemático, financeiro, robótico e até biológico, mas não sendo restrito apenas a esses tipos sistemas.

Mesmo havendo essa diversidade de áreas que estudam a auto-organização, no presente trabalho serão expostos os estudos de métodos de auto-organização para sistemas robóticos, em especial, enxames de robôs, mas com o intuito de realizar tarefas coletivas ou individuais, dependendo da atual situação do sistema.

O estudo e a aplicação dessas metodologias visam à criação de métodos para auto-organização de sistemas, nos quais as células (essas podem ser qualquer ente do sistema como, por exemplo, micro ou macro robôs, sensores em uma rede ou nós de uma rede ethernet) sejam as mais homogêneas possíveis, ou seja, saíram da mesma linha de montagem sendo de fabricação idêntica, ou tenham, pelo menos, tenham igual algoritmo de organização e funções muito próximas. Outro quesito para essas células é a busca pela simplicidade, quanto menos funcionalidades específicas ela possuir, melhor. A forma ideal seria que as funcionalidades devem estar presentes de forma emergente no conjunto das células e não necessariamente em cada célula individual.

Quando se atinge a simplicidade e a homogeneidade das células, fica mais fácil e barato de se produzir uma grande quantidade dessas células, aumentando assim a aplicabilidade em diferentes sistemas.

Atingindo-se essa simplicidade celular e com o correto método de auto-organização, o qual esse trabalho busca, muitos sistemas passarão a ter autonomia própria e a capacidade de tomar decisões de forma individual ou coletiva, para melhor se adaptar ao cenário apresentado, sem nenhuma ação de um ente centralizador, ou do ser humano.

Nesse cenário, o presente trabalho busca estudar e aplicar um método de auto-organização em um cenário bem definido, no qual, diferente de metodologias já existentes, os entes do sistema devem ser capazes de tomar tanto ações individuais e diferentes dos outros entes como ações coletivas, dependendo do estado do sistema, mas sempre buscando o objetivo geral do sistema.

II. MÉTODOS DE AUTO-ORGANIZAÇÃO

Para atingir o objetivo esperado, é necessário estudar as metodologias de auto-organização de sistemas já existentes. Buscando-se na literatura, alguns métodos são encontrados, mas com aplicações restritas como, por exemplo, para métodos para organização para monitoramento de redes [1].

Mas no presente trabalho busca-se uma metodologia que tenha uma aplicação mais ampla, buscando aplicações mais generalistas e de simples implementação.

Aliada a essa ideia, veem a busca pela auto-organização de enxames de robôs. Nesse tipo de sistema, uma grande quantidade de células robótica busca realizar uma série de tarefas de forma coletiva e descentralizada.

Métodos para esse tipo de organização podem ser encontrados em trabalhos como o de Wei-Min Shen [2] o qual se baseia em hormônios digital, como descrito em um trabalho prévio do mesmo autor [3], no qual ele propõe uma metodologia de auto-organização baseada no modo que as células biológicas se comunicam através da secreção de hormônios.

Com uma linha de pensamento parecido, Yaochu Jin [4], propõe uma metodologia, também baseada no que ocorre naturalmente na natureza, mas com foco na morfogênese para controlar um grupo de robôs e os alinharem em uma forma pré-definida através de uma superfície NURBS.

Dentre esses métodos foi escolhido para início dos estudos, aqui apresentados, o método proposto por Wei-Min Shen em seus trabalhos, primeiramente pela simplicidade necessária para as células se organizarem e por não estar com o foco apenas em uma estrutura fixa, como no trabalho apresentado por Yaochu Jin [4].

J.R.Ribeiro Junior, Universidade de São Paulo (USP), Escola Politécnica, São Paulo, Brasil, jose.junior@usp.br

A. R. Hirakawa, Universidade de São Paulo (USP), Escola Politécnica, São Paulo, Brasil, andre.hirakawa@poli.usp.br

C. A. Sakurai, Universidade de São Paulo (USP), Escola Politécnica, São Paulo, Brasil, akiocs@usp.br

III. HORMÔNIOS DIGITAIS

Como exposto por Wei-Min Shen em seu primeiro trabalho sobre auto-organização através da secreção de hormônios digitais [3], esse método é baseado em dois outros métodos prévios de auto-organização: O autômato celular [6] e o modelo de reação-difusão de Turing, [5].

O autômato celular é uma metodologia de auto-organização, no qual as células são dispostas de forma com que o seu estado atual dependa dos estados de suas células vizinhas e de suas regras. Um importante ponto sobre essa metodologia, para esse presente trabalho, reside no fato de que no autômato celular, todas as células são idênticas, especializando-se conforme a necessidade do ambiente em que elas se encontram, comunicando-se apenas com a suas vizinhas. Pode-se colocar como um exemplo clássico de autômato celular o Jogo da Vida criado e idealizado pelo britânico John Horton Conway[7] .

O modelo de hormônio digital fica completo, quando ao autômato celular coloca-se o modelo de difusão-reação de Turing,. Ao invés das células se comunicarem apenas com as células tangentes, elas irão se comunicar através da difusão de hormônios pelo espaço sendo que as células escolherão o seu novo estado através da concentração de hormônio por ela captada.

Segundo o modelo de Turing,, os hormônios são secretados em pares, sendo um ativador e outro inibidor e devem seguir as seguintes regras de dispersão:

$$CA(x,y) = \frac{a_A}{2\pi\sigma^2} e^{-\frac{(x-a)^2+(y-b)^2}{2\sigma^2}} + R \quad (1)$$

$$CI(x,y) = -\frac{a_I}{2\pi\rho^2} e^{-\frac{(x-a)^2+(y-b)^2}{2\rho^2}} + R \quad (2)$$

Nas quais a_A, a_I, ρ, σ, R, são constantes, sendo que $\sigma < \rho$ para satisfazer as condições de estabilidade de Turing,e CA e CI indicam as concentrações dos hormônios de ativação e de inibição em uma dada posição x,y do espaço.

No trabalho presente, para facilitar a simulação, as coordenadas deixam de ser cartesianas e passam a ser polares ficando com:

$$CA(r) = \frac{a_A}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}} + R \quad (3)$$

$$CI(r) = \frac{a_I}{2\pi\rho^2} e^{-\frac{r^2}{2\rho^2}} + R \quad (4)$$

Desse modo simula-se melhor a difusão através de uma antena transmissora, de forma que a concentração dependa apenas da distância entre das células e não de uma posição fixa em uma grade.

Fazendo-se uma análise dessa metodologia e a confrontando com o que se deseja atingir com esse presente trabalho, ela mostra-se um excelente ponto de partida já que se mostra um método fácil de aplicar para as células simples, a integração com o autômato celular faz com que as células sejam homogenias e o modelo de difusão consegue transmitir informações às células próximas de maneira simples e sem necessidade de uma coordenada geográfica global.

IV. APLICAÇÃO EM SISTEMAS DE CÉLULAS HOMOGÊNEAS

Como já exposto, um sistema de células homogêneas é aquele em que os componentes são idênticos, ou pelo menos, possuem a mesma estrutura básica como o algoritmo de controle e decisão da célula.

Em sistemas como esses, cada célula pode ser trocada por qualquer outra célula do sistema ou até pela inserção de uma nova sem que haja perda da funcionalidade total do sistema.

Dessa forma, imagine o sistema de monitoramento de uma usina em que todos os sensores são fabricados de forma igual e que eles possuem as informações necessárias para que haja o monitoramento de toda a usina. Assim, eles podem ser simplesmente colocados em todos os locais que necessitem o monitoramento, e os próprios sensores saberão qual é a função que eles devem exercer.

Se por algum motivo um desses sensores quebrarem, o sistema deverá, por si próprio, saber qual a criticidade do monitoramento exercido pela peça e decidir se é possível ou necessário para um outro sensor fazer o papel do problemático.

Outro exemplo, e o que está sendo utilizado nesse trabalho para verificar, é um jogo de futebol de robôs, no qual todos os jogadores são idênticos (apenas sendo diferentes os times que participam) e devem por si próprios se organizarem para atingirem as metas necessárias (fazer um gol, roubar a bola do adversário, passar a bola e defender a sua própria meta).

Essas funções devem ser exercidas dependendo da situação em que cada robô se encontra, por exemplo, se o robô está sozinho perto bola ele deve ir atrás da mesma, por outro lado, se ele está longe da bola e percebe que há companheiros próximos dela, é preferível ficar na defesa ou apenas acompanhar os companheiros do time.

Nesse cenário cada uma das células do sistema secretará um tipo de hormônio, de acordo com a ação esperada que aquela célula deseje que o sistema tome e receberá a soma de todos os hormônios que as outras células estejam secretando, irá verificar em seu banco de dados ou em seu algoritmo a que ação aquela quantidade de hormônio corresponde, tomará a decisão e realizará a ação e por fim, excretará novamente os hormônios adequados.

V. APLICAÇÃO NO FUTEBOL DE ROBÔS

Para testar o método de auto-organização foi criado um simulador de futebol de robôs, com o campo segundo as regras em [9] e cinco robôs em cada time e uma bola, também seguindo as medidas em [9].

O simulador foi criado utilizando-se o framework d Microsoft XNA [8] com a linguagem C#. Nele todos os robôs, assim como a bola, são criados a partir da mesma classe que representa uma célula, e possui a possibilidade de se especializar em dois times diferentes de robôs ou na bola.

Essa classe ainda tem o poder de criar os hormônios, segundo a metodologia em [2] e [3] e exposto aqui pelas equações (3) e (4) , além de conseguir medir a quantidade de hormônios presentes na região onde essa célula se encontra.

Diferentemente do exposto em [3], a grade do autômato celular não é discreta, mas sim continua, para ficar mais próximo da realidade, sendo assim os hormônios produzem áreas circulares em volta dos robôs, simulando-se antenas transmissoras e podem ser facilmente captadas por antenas receptoras.

Uma das grandes diferenças da metodologia, aqui proposta, consiste no fato de que as células não sabem onde elas estão geograficamente no espaço, isso simplifica as células, as quais não precisam de um sistema de posicionamento (GPS, ou visual) o qual complicaria a célula e poderia limitar a utilização, por exemplo, uma célula dentro de um ambiente fechado seria muito difícil utilizar um GPS.

Assim as células se organizam apenas de forma relativa e se orientam exclusivamente pelos hormônios por elas gerados. Isso é de fundamental importância a metodologia que é aqui buscada, pois existem muitos ambientes ou sistemas nos quais a localização absoluta é impossível, e que mesmo assim, necessite que seja conhecida a localização da célula, desse modo, a busca por um método genérico de localização relativa entre as células e que dela consiga-se extrair uma boa localização absoluta (ou pelo menos de forma relativa a localização do sistema) é fundamental.

VI. Resultados

Como pode ser observado na Fig. 1, os robôs são colocados dentro de campo inicialmente de forma aleatória, sendo que o time de robôs roxos fica sempre à esquerda e o de vermelhos à direita.

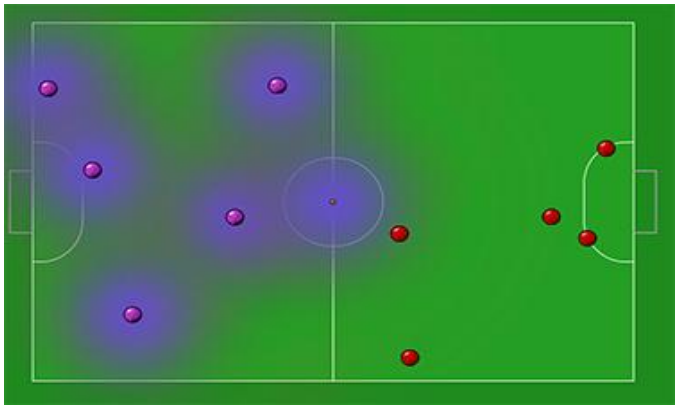


Figura 1 - Disposição inicial dos Robôs.

As manchas escuras que podem ser observadas nos robôs da esquerda é a representação da área de influência dos hormônios gerados por cada robô. Ao centro vê-se a bola, e a área de influência do hormônio por ela gerado. Os robôs do time da direita não estão com as áreas dos hormônios por não ser necessária essa visualização para essa simulação.

A. Primeira rodada de testes

Em um primeiro momento, o simulador foi concebido apenas com o método do sistema de hormônios digitais, no qual apenas a bola excretava o hormônio de localização da mesma. Dessa forma esse hormônio acabou por realizar o papel muito mais parecido com o de um feromônio, o qual fazia todas as células presentes no local irem em direção à ela.

Dessa forma, as células que estavam suficientemente próximas à bola, ou seja, na área de influencia positiva de seu hormônio foram atraídas para a bola.

Mesmo as células não possuindo nenhum tipo de localizador e serem completamente cegas ao ambiente, elas conseguiram ser atraídas e chegar até o ponto onde a bola se encontrava, graças à percepção do gradiente da concentração do hormônio. Como cada célula sabe como o hormônio é produzido, a partir do gradiente da concentração do hormônio entre dois tempos diferentes, a célula é capaz de se orientar até a chegada definitiva na bola.

Existe um agravante ao cálculo, como a bola não é um ente estático, ou seja, ela se movimenta durante o jogo, as células tem que calcular a posição da bola a cada novo passo que ela faz no sistema, se a bola fosse fixa, só seria necessário um passo para saber a posição real da bola.

Assim, é possível observar na Fig. 2, que o robô que estava sobre a influencia positivava do hormônio gerado pela bola, foi ao se encontro, tirando-a da marcação inicial, enquanto os outros robôs, que estavam sobre influência negativa ou zero do hormônio da bola, continuaram em seus lugares.

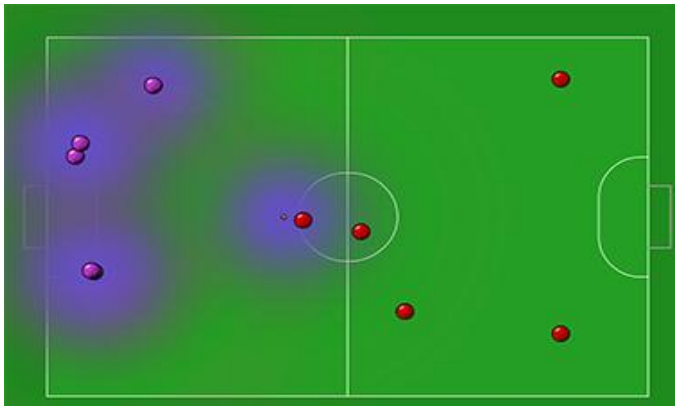


Figura 2 - Robô vermelho vai à direção da bola e a tira da marca inicial.

O problema dessa primeira rodada fica no fato da falta de coletividade dos robôs, não á interação entre eles ficando inexistente o jogo coletivo, não havendo uma função do sistema, o que não serve para validar a metodologia para esse tipo de sistema.

B. Segunda rodada

Para a segunda rodada de testes, foi introduzido um segundo tipo de hormônio para serem excretados pelas células. Esse hormônio tem como função mostrar aos companheiros ou até aos robôs adversários a sua posição relativa dentro do campo.

Com a introdução desse segundo hormônio, um dos problemas que havia na primeira rodada desaparece. Agora como os robôs sabem da existência dos outros, eles são capazes de tomarem decisões mais complexas, como ficar na defesa, ir ao ataque, acompanhar o companheiro com a bola, ou tentar toma-la de seus adversários.

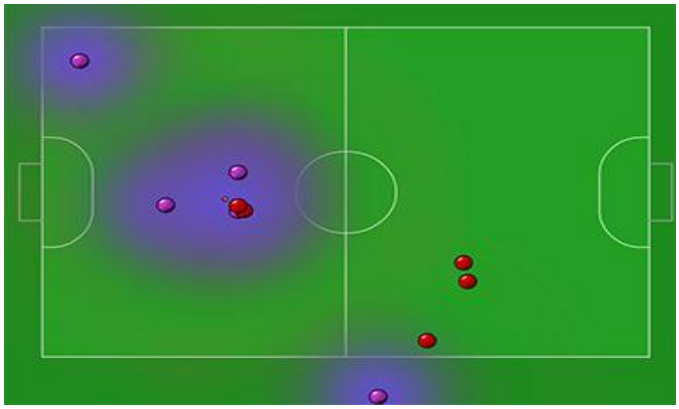


Figura 3 - Resultado da segunda rodada de testes

A Fig. 3 mostra alguns dos novos resultados gerados pela inserção do novo hormônio no sistema. No movimento de ataque do time vermelho, três robôs roxos decidem marcar o jogar vermelho que está no ataque, enquanto um jogador roxo vai à marcação dos outros jogadores vermelhos, enquanto o quinto roxo se mantem na defesa.

Mas a Fig. 3 também mostra dois problemas no atual cenário da simulação.

O primeiro de menor importância é o problema de colisões, como o sistema de posicionamento é contínuo, diferentemente do que acontece no autômato celular padrão e os robôs não se comportam como pontos materiais, o seu tamanho é relevante ao sistema, existem essas sobreposições como é possível ver nos jogadores perto da bola, mas isso é algo simples de ser corrigido, apenas colocando-se o tamanho dos robôs em seu algoritmo de organização.

O segundo problema e de maior relevância para o método é a presença de um robô fora da área destinada ao jogo. Isso ocorre pelo fato dos robôs não possuírem noção espacial do campo, como as coordenadas são relativas entre eles e os robôs não tem um sistema de posicionamento global, eles não possuem um meio de saberem onde o campo acaba.

Isso também é agravado pelo fato dos robôs não saberem onde estão os gols.

VII. Trabalho Futuro

Em vista aos problemas levantados pela segunda rodada de experimentos, alguns trabalhos futuros ficam bem claros, sendo que a principal preocupação fica por parte do posicionamento dos robôs dentro de campo.

Para resolver esse problema, no momento da escrita desse trabalho, está sendo pesquisada uma forma de colocar dentro do algoritmo dos robôs as dimensões e os principais pontos do campo, e através de uma condição inicial, ser possível, para cada robô saber a sua posição. Algo muito simples quando se tem uma bola inicialmente sempre no mesmo local. O único agravante e que deverá ser pensado, reside no fato que não há garantia de que pelo menos um robô, de cada time, esteja perto da bola no inicio do jogo, devido à aleatoriedade da distribuição dos robôs no inicio do jogo. Se isso for possível, mesmo que de maneira forçada, quando um dos robôs souber a sua posição dentro do campo, ele pode passa-la para os seus outros companheiros e assim, todos ficarem com as dimensões e pontos estratégicos bem definidos.

Depois de solucionado o problema do posicionamento e a metodologia for verificada e validade para esse tipo de sistema, ela será testada e avaliada em outros sistemas para melhor refinamento do método, buscando um generalização maior do mesmo.

VIII. Conclusão

Tomando-se os primeiros resultados obtidos nas duas primeiras rodadas da aplicação do hormônio digital dentro do futebol de robôs, o método mostrou-se promissor e mesmo com os problemas encontrados na aplicação direta do algoritmo nas células, isso se torna um interessante objeto de pesquisa para a continuação do trabalho e irá agregar novas funcionalidades ao método.

Dessa forma o objetivo inicial de verificação das metodologias e da sua aplicação dentro do futebol de robôs foi alcançado, gerando o ponto de partida para a adaptação do método para que os robôs sejam capazes de realizar todas as tarefas necessárias para o jogo de forma coletiva ou individual.

IX. Referências

[1] T. Fan, L. Xu, L.Yin, H. Wang, X. Li, “A Self-organization Algorithm Based on 1D Cellular Automata for Networking Monitoring” in The 1st International Conference on Information Science and Engineering, ICISE2009

[2] W. Shen, P. Will, A. Galstyan, “Hormone-Inspired Self-Organization and Distributed Control, in Autonomous Robots of Robotic Swarms”, nº 17, pp. 93–105, 2004

[3] W.Shen, C.Chuong, and P. Will, “Digital Hormone Models for Self-Organization” in Articial Life VIII. pp. 116-120, 2002

[4] Y. Jin, Y. Meng, and H. Guo “A Morphogenetic Self-Organization Algorithm for Swarm Robotic Systems using Relative Position Information” in Computational Intelligence (UKCI), UK Workshop, 2010

[5] A. Turing, “The chemical basis of morphogenesis.” in Philos. Trans. R. Soc. London B 237, pp.37-72, 1952.

[6] H. Gutowitz, “Cellular Automata | Theory and Experiment.” Cambridge, MA: MIT Press, 1991.

[7] E. R.Berlekamp, J. H. Conway, R.K. Guy, “Winning Ways for your Mathematical Plays” in A K Peters ed. 2, 2001.

[8] http://msdn.microsoft.com/en-us/centrum-xna.aspx

[9] http://www.aishack.in/2010/07/robocup-soccer-rules-for-the-small-sized-league-ssl/

Mecanização da Aprendizagem com Dispositivos Adaptativos: Conceitos e Aplicação

R. L. Stange e J. J. Neto

***Resumo*— A aprendizagem incremental requer que o mecanismo de aprendizagem seja baseado no acúmulo dinâmico da informação extraída das experiências realizadas. A palavra adaptatividade sugere a capacidade de modificação do conjunto de regras aprendidas em resposta a eventos que podem ocorrer durante o processo de aprendizagem, ou então autoajustes no conjunto de parâmetros. O objetivo deste trabalho é investigar questões relacionadas à utilização da adaptatividade no processo de aprendizagem de máquina, tais como mecanização da aprendizagem, representação do conhecimento, inferência e tomada de decisão. Para isso, propõe-se aqui a utilização de dispositivos adaptativos para representar o conhecimento adquirido através da aprendizagem incremental.**

***Palavras-chaves*— Adaptatividade, Dispositivos Adaptativos, Aprendizagem de Máquina, Mecanismo de Aprendizagem.**

I. Introdução

O TERMO aprendizagem de máquina (ML, no original em inglês *Machine Learning*), refere-se ao funcionamento de sistemas computacionais capazes de aprender e modificar o seu comportamento em resposta a estímulos externos, ou através de experiências acumuladas durante sua operação [1].

Aprendizagem de Máquina é uma área de pesquisa que estuda métodos, técnicas e ferramentas computacionais relacionadas à aquisição de novos conhecimentos e, novas habilidades para melhorar o desempenho de algoritmos por meio da experiência [8][1].

A adaptatividade é uma característica atribuída ao comportamento automodificável de sistemas computacionais. Este comportamento autônomo ocorre em resposta a estímulos de entrada e ao histórico de operação desses sistemas [12]. As pesquisas em adaptatividade investigam soluções para diversos problemas complexos de teoria da computação [11], de aprendizagem de máquina [18], de tomada de decisão [27] e de engenharia da computação [16], entre outros.

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

[14] apresentam o mecanismo de inferência ativo nos autômatos adaptativos. Como exemplo ilustrativo, o autômato adaptativo é utilizado para o aprendizado supervisionado de linguagens regulares. Um conjunto de amostras positivas[1] e negativas[2] da linguagem é submetido ao autômato, que deve inferir as sentenças aceitas ou rejeitadas.

[17] propõem um algoritmo de indução de árvores de decisão utilizando técnicas adaptativas, que combina estratégias sintáticas e estatísticas, chamado *AdapTree*.

Outras experiências bem-sucedidas em aprendizagem de máquina utilizando dispositivos adaptativos incluem: aprendizagem de modelos para distribuição de espécies [15][25], classificação de padrões geométricos [5], decodificação do alfabeto de LIBRAS [3], localização de padrões em imagens [19], identificação de diagnósticos médicos [4]e mineração de dados [28], entre outras.

O processo de aprendizagem de máquina traz a tomada de decisão de forma crucial. A tomada de decisão exige um processo de raciocínio em que as informações já adquiridas e as novas informações, quando comparadas entre si, possam levar a novas informações e, com isso, influenciar o processo [26]. Esse processo de raciocínio é muitas vezes complexo e dinâmico, uma vez que as decisões devem ser flexíveis, pois eventualmente dependem de vários fatores e prioridades que nem sempre são fáceis de identificar antes de iniciar o processo de aprendizagem.

De acordo com [11], a resolução de problemas complexos e de natureza dinâmica utilizando a tecnologia adaptativa pode ser mais expressiva do que a utilização de métodos tradicionais, em alguns casos.

Em aprendizagem de máquina, por exemplo, uma das dificuldades está relacionada à representação do conhecimento humano em uma linguagem simbólica que tenha grande poder de expressividade. Métodos tradicionais para essa representação incluem o uso de regras de produção, árvores de decisão e redes Bayesianas, entre outros. O uso de dispositivos adaptativos pode agregar expressividade à representação do conhecimento e contribuir para o crescimento dessa área. Os autômatos adaptativos, por exemplo, além de possuírem o mesmo poder de expressão da Máquina de Turing [23], são eficientes e de fácil visualização, pois são baseados em modelos de autômatos finitos [10].

Contudo, a exploração de questões referentes à aprendizagem de máquina utilizando a tecnologia adaptativa se aplica, de maneira abrangente, ao tratamento de problemas de tomada de decisão.

A proposta deste trabalho tem como objetivo investigar questões relacionadas à utilização da adaptatividade no processo de aprendizagem de máquina, tais como mecanização da aprendizagem, representação do conhecimento, inferência e tomada de decisão.

Para atingir este objetivo, propõe-se aqui a utilização de dispositivos adaptativos para representar o conhecimento adquirido através da aprendizagem incremental. Além disso, é realizado um estudo de caso que combina aprendizagem de máquina com técnicas adaptativas para implementar um esquema de aprendizagem autônoma de estratégias, com o objetivo de vencer uma particular instância do jogo que é apresentado.

A utilização de dispositivos adaptativos em problemas reais e aplicar técnicas adaptativas para extração de regras

[1] Conjunto de sentenças que pertence à linguagem.
[2] Conjunto de sentenças que não pertence à linguagem.

desses dispositivos. A aplicação escolhida, apesar de simples, tem sido utilizada na aplicação de diferentes técnicas, tais como redes neurais artificiais [30] e distância de Hamilton [22]. Isso permite a fácil associação entre a utilização de técnicas adaptativas e as diferentes técnicas de aprendizado de máquina.

II. Adaptatividade em Aprendizagem de Máquina

A adaptatividade possui uma ampla aplicação em aprendizagem de máquina devido às suas principais características, tais como a capacidade de reter regras, representar conhecimento, entre outras.

Em particular, os dispositivos adaptativos são adequados para modelar o processo de aprendizagem devido à maneira que operam. A inferência, aquisição e representação de conhecimento são notadamente favorecidas com utilização de técnicas adaptativas [13].

Os dispositivos adaptativos na mecanização da aprendizagem, por exemplo, permitem a inserção de um artifício de adaptação nas regras enquanto são aprendidas e isso proporciona ao algoritmo de aprendizagem a capacidade de automodificar o conjunto de regras aprendidas apenas em função das instancias de entrada do algoritmo. Outra característica da aplicação de técnicas adaptativas em aprendizagem de máquina é que a utilização de dispositivos adaptativos na representação do conhecimento proporciona um bom entendimento dos resultados obtidos com a aprendizagem. Eventualmente, a capacidade de entender as regras pode ser útil se comparado com outras técnicas de aprendizagem que fornecem bons resultados, mas possuem limitações para explicar como chegaram ao resultado fornecido.

Para representar o processo de mecanização da aprendizagem utilizando um dispositivo adaptativo um modelo para aprendizagem de um jogo foi definido. Este modelo chamado de mecanismo de aprendizado de um jogo é divido nas seguintes etapas: Interface, Inferência e Memória.

O termo mecanismo de aprendizado se refere a essas três etapas.

A *Interface* estabelece a comunicação entre as informações externas e o mecanismo de aprendizado. Entende-se por informações externas qualquer estímulo de entrada provocado por um agente inteligente, seja ele humano (externo) ou a própria máquina (realimentação). De modo geral, a função da Interface é receber as informações e alimentar o mecanismo de aprendizado.

A *Inferência* está em constante ciclo de aprendizagem e é responsável por analisar as informações capturadas e agregar conhecimento sobre elas, para melhorar o desempenho na tomada de decisão. O conhecimento adquirido é representado por um conjunto de regras, que pode ser alterado a cada ciclo de aprendizagem. A situação corrente do conjunto de regras é representada por dispositivos adaptativos.

A *Memória* representa o conjunto de regras. Tudo que o dispositivo aprende deve ser armazenado em uma base de regras. Posteriormente, a base de regras é utilizada na tomada de decisão e, depois é possível inferir novas regras a partir dos dados da base. Nos dispositivos adaptativos a memória está acoplada ao próprio dispositivo, que é uma das vantagens de sua utilização.

A Figura 1 mostra o mecanismo de aprendizado refinado para o aprendizado de estratégias para um jogo dinâmico com informação completa.

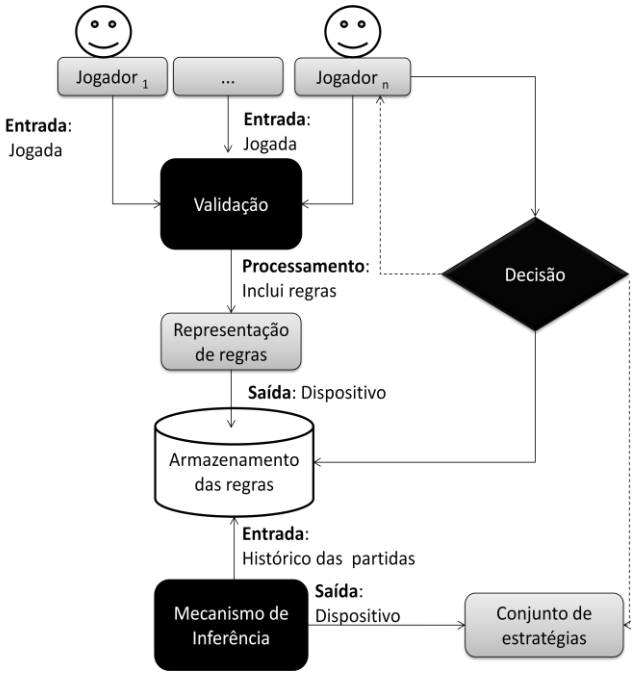


Figura 1. Mecanismo de aprendizagem de um jogo.

O mecanismo de aprendizado de um jogo mostra as relações entre os componentes que compõem o mecanismo de aprendizado, a saber: *Jogadores* ($Jogador_1$, $Jogador_2$,..., $Jogador_n$), *Validação*, *Representação de Regras*, *Armazenamento de Regras*, *Mecanismo de Inferência*, *Conjunto de Estratégias* e *Decisão*.

A componente *Jogadores* representa o canal de comunicação entre o mecanismo de aprendizado e o ambiente externo (Interface). Cada jogador deve fornecer uma *entrada* de dados para o funcionamento do mecanismo de aprendizado.

A *Validação* é a componente que recebe os estímulos de entrada.

No aprendizado de uma partida, cada jogada é uma regra a ser validada e deve ser capturada e posteriormente representada por um dispositivo apropriado.

Na *Representação de Regras*, um dispositivo adaptativo pode ser uma solução alternativa se considerarmos que, idealmente, o mecanismo de aprendizagem deve ser capaz de incorporar novas regras de forma autônoma.

Segundo [20], sistemas com características autônomas favorecem a utilização de modelos formais em sua especificação. As Máquinas de Estados Finitos, por exemplo, são modelos formais que favorecem a Engenharia de Software na especificação do comportamento de sistemas reativos, que se torna mais precisa e menos sujeita a ambiguidades. Esses modelos facilitam o entendimento do sistema.

Particularmente, os autômatos adaptativos podem evoluir gradualmente, modificando sua topologia inicial e, sucessivamente, inserir ou excluir transições de seu próprio conjunto de transições, como resultado da execução de ações adaptativas [14].

Neste trabalho, na *Representação das Regras*, a proposta é utilizar um autômato adaptativo para representar o conjunto de regras aprendidas em cada partida disputada. Porém, sua representação possui aplicações práticas restritas, se for considerado como dispositivo subjacente o autômato finito, que possui a informação de saída limitada à lógica binária aceita/rejeita [6]. Assim, com a finalidade de extrair cada movimento do autômato no decorrer de seu aprendizado, aplica-se uma extensão de autômatos conhecida como *Máquina de Mealy*, que possui saídas associadas às transições. A Máquina de *Mealy* é um autômato finito modificado capaz de gerar uma palavra de saída para cada transição da máquina, a qual inclusive pode ser vazia [6].

Na definição da Máquina de *Mealy* é incorporado um alfabeto de símbolos de saída Δ, que pode ser o mesmo alfabeto de entrada. A função de transição pode ser representada como um diagrama, assim como nos autômatos finitos, adicionando a cada transição a saída associada, quando diferente da palavra vazia.

A representação gráfica deste autômato é mostrada na Figura 2.

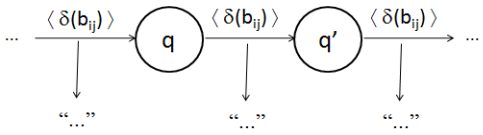


Figura 2 – A Máquina de *Mealy*.

A seguir a simbologia utilizada no diagrama de transição da Máquina de *Mealy*.

⟨...⟩ entrada fornecida pelo usuário (ex.: por "Jogador").

"..." saída gerada pela transição (ex.: jogada realizada).

A entrada representa o conjunto (finito) de todos os possíveis símbolos que são estímulos de entrada válidos do autômato. Os estados q e q' representam os estados de origem e destino de uma transição.

Para fins de apresentação, as saídas da máquina serão ocultadas e será definido como representação equivalente o autômato adaptativo, apresentado na Figura 3. Assim a transição $(q, \langle ... \rangle) \rightarrow q'$ da Máquina de *Mealy* é equivalente $(q, \Theta) \rangle\rangle q$ do autômato adaptativo.

As saídas do autômato marcam as escolhas efetuadas por "Jogador" e "Oponente", a cada jogada e a cada partida. O caminho que determina o sucesso ou o insucesso na partida é mapeado através das saídas geradas pelo autômato.

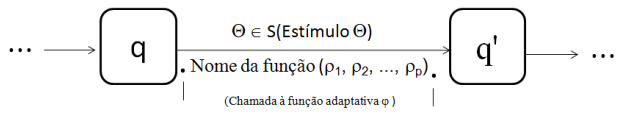


Figura 3 – Autômato Adaptativo com saída oculta.

Ao se pensar no jogo como uma máquina de estados que recebe uma entrada, sendo a entrada uma jogada $\delta(b_{ij})$, o tabuleiro começa no estado vazio e os estados mudam a cada vez que a máquina recebe uma entrada. Alguns estados são especiais quando são atingidos, no caso do jogo da velha, quando a partida acaba um estado final é atingido. Alguns estados finais determinam uma vitória para "Jogador" e outros, uma vitória para "Oponente", enquanto os demais estados significam que o jogo terminou com um empate.

No *Armazenamento de Regras*, as jogadas capturadas pelos autômatos são armazenadas na memória do mecanismo de aprendizado e representam o histórico das partidas. A memória também é chamada de base de regras e as informações contidas nesta base podem ser utilizadas futuramente para melhorar o modelo de aprendizagem através da experiência. Em dispositivos adaptativos, as regras ficam armazenadas na estrutura do próprio dispositivo. Nos autômatos adaptativos, as regras ficam armazenadas através dos estados e transições, nas tabelas de decisão adaptativas, são representadas pelos valores de condições e ações e, nas árvores de decisão adaptativas, são representadas através dos caminhos que guiam uma decisão da raiz até as folhas.

Para exercitar a utilização da adaptatividade em aprendizagem de máquina foi escolhido um tipo particular de jogo, classificado como jogos dinâmicos. Esse é um tipo de jogo em que o processo de interação estratégica se desenvolve em etapas sucessivas. Nestes jogos, a decisão tomada por um jogador considera as decisões tomadas pelos demais jogadores. Assim, os jogadores fazem escolhas a partir do que os outros jogadores decidiram no passado. Além disso, nesse tipo de interação, as escolhas presentes exigem considerar as consequências futuras, uma vez que o oponente pode retaliar em etapas posteriores do jogo [29].

O matemático e filósofo alemão Ernst Friedrich Ferdinand Zermelo (1817-1953) provou um teorema que conduz à ideia de que jogos dinâmicos de duas pessoas, com ações sequenciais e informação completa, tais como os jogos da velha ou de xadrez, são perfeitamente determinados, ou seja, possuem solução definida. Neste caso, se um jogador souber aplicar a estratégia correta, dificilmente perderá o jogo.

Em linhas gerais, o processo de aprendizagem de uma estratégia para vencer uma instância de um jogo provoca a adaptação do comportamento dos jogadores para melhorar o desempenho na tomada de decisão. Cada tomador de decisão em um jogo é denominado jogador. Esse adota estratégias, que conduz às decisões para atingir seus objetivos, para cada informação que um jogador possa ter e em cada momento que ele é chamado a realizar uma ação.

A seguir é o mecanismo de aprendizado para um jogo é exemplificado através do processo de aprendizagem de uma estratégia para vencer o jogo da velha.

III. Estudo de Caso

Este estudo de caso é realizado com um jogo de tabuleiro popularmente conhecido como Jogo da Velha. Os jogos de tabuleiros são casos de estudos interessantes em aprendizagem de máquina, se consideramos que o resultado dessa

experiência representa uma habilidade humana [29]. A capacidade de jogar, neste caso, representa muitas vezes um desafio aos jogadores que devem ser criativos para criar planos de ação, combinar estratégias e aprender com a experiência.

O nível de dificuldade de um jogo de tabuleiro varia de um simples jogo como o do jogo da velha ou um quebra-cabeça até os jogos que necessitam de uma estratégia mais sofisticada, como as aplicadas nos jogos de dama ou xadrez.

No jogo da velha participam dois jogadores, onde cada jogador tem a sua vez para marcar com o seu próprio símbolo (ex.: 'X' ou 'O') uma posição em um tabuleiro de tamanho 3×3.

As posições do tabuleiro do jogo da velha são representadas por b_{ij}, onde i é linha da matriz 3×3 e j é a coluna, representadas na Figura 4.

b_{11}	b_{12}	b_{13}
b_{21}	b_{22}	b_{23}
b_{31}	b_{32}	b_{33}

Figura 4. Posições do tabuleiro do jogo da velha.

As regras do jogo da velha são as seguintes [2]:

a) O jogo começa com o tabuleiro vazio;
b) Um jogador assume a vez marcando o seu símbolo em uma das posições vazias do tabuleiro;
c) O jogador que forma uma linha reta completa (de ponta a ponta), ou seja, uma linha vertical, horizontal ou diagonal primeiro é o vencedor;
d) O jogo empata se nenhuma linha reta é formada e não existem posições vazias no tabuleiro.

O estado inicial da partida é o tabuleiro vazio. Em cada etapa da partida, que será chamada de jogada, um jogador faz um movimento que consiste em colocar um 'X' em uma das posições vazias do tabuleiro e o outro jogador faz um movimento que consiste em colocar um 'O' em uma das posições vazias.

A Figura 5 mostra uma possível configuração de três tabuleiros de jogos.

(a)

(b)

X	X	X
	O	O
O		X

(c)

X	O	X
X	O	O
O	X	X

Figura 5. Três diferentes configurações para o tabuleiro do jogo: (a) Configuração inicial, (b) Jogador 'X' vence o jogo, (c) Jogo empatado.

Com a finalidade de obter um sistema de aprendizado capaz de representar computacionalmente um determinado problema, bem como a sua solução, é necessário descrever objetos, processos e situações que fazem parte do seu domínio [9]. Existem diferentes linguagens de descrição com diferentes complexidades capazes de descrever exemplos (casos observados), hipóteses e conhecimento de domínio (ex.: lógica de atributos, lógica proposicional, lógica relacional, funções matemáticas, etc.).

Neste trabalho é adotado um tipo de linguagem para descrição de exemplos baseada na lógica de atributos, amplamente adotada em algoritmos de aprendizagem. Neste tipo de linguagem, um exemplo é descrito por um conjunto de atributos ou características que assumem diversos valores. Cada exemplo é formado pela conjunção do par atributo e valor que representa um caso observado, ou uma instância do problema (ex.: dor = sim $\wedge$ febre=sim $\wedge$ classe=doente). A classe é um atributo especial definido em alguns exemplos, que representa a saída do algoritmo para aquela instância. No caso do jogo da velha, cada partida é um exemplo de treinamento, as jogadas são atributos e a classe é o resultado obtido em cada partida, sendo os valores possíveis "P", "N" ou "E" respectivamente para resultados positivos, negativos ou empate.

Assim utilizando uma notação de aprendizagem de máquina, o jogo é representado da seguinte forma:

- A = $\{\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8, \alpha_9\}$ é o conjunto de atributos, onde: α_0 = "1ª jogada", α_1 = "2ª jogada"; α_2 = "3ª jogada"; α_3 = "4ª jogada"; α_4 = "5ª jogada"; α_5 = "6ª jogada"; α_6= "7ª jogada"; α_7 = "8ª jogada"; α_8 = "9ª jogada"; α_9= "Resultado da partida".

- $\delta(b_{ij})$ é um possível valor para cada atributo de A, onde δ = {X,O} e b_{ij} a posição escolhida do tabuleiro. Assim, cada atributo de α_0 a α_8 pode conter os seguintes valores $v_{atributos}$ = {X(b_{11}), X(b_{12}), X(b_{13}), X(b_{21}), X(b_{22}), X(b_{23}), X(b_{31}), X(b_{32}), X(b_{33}), O(b_{11}), O(b_{12}), o(b_{13}), O(b_{21}), O(b_{22}), O(b_{23}), O(b_{31}), O(b_{32}), O(b_{33}), θ }, onde θ significa posição do tabuleiro vazia e $\delta(b_{ij})$ a jogada escolhida.

- $\Omega = \{\omega_1, \omega_2, \omega_3\}$ são os possíveis valores de α_9, onde: ω_1 = "P"; ω_2 = "N"; e ω_3 = "E", representando respectivamente os resultados: "Jogador" venceu a partida (Positivo), "Jogador" perdeu a partida (Negativo) e "Jogador" empatou a partida.

- T é o conjunto de exemplos treinamento composto por i partidas, onde i = {0, 1, 2, 3,...}. Cada partida é representada pelo conjunto ρ_i.

Considere o conjunto de dados de treinamento *T*, apresentados na Tabela 18.

TABELA I
DADOS DE TREINAMENTO REFERENTES A SEIS PARTIDAS DE TTT.

	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
α_0	$X(b_{22})$	$X(b_{11})$	$X(b_{12})$	$X(b_{31})$	$X(b_{22})$	$X(b_{22})$
α_1	$O(b_{11})$	$O(b_{33})$	$O(b_{31})$	$O(b_{22})$	$O(b_{13})$	$O(b_{11})$
α_2	$X(b_{33})$	$X(b_{31})$	$X(b_{33})$	$X(b_{11})$	$X(b_{33})$	$X(b_{13})$
α_3	$O(b_{12})$	$O(b_{21})$	$O(b_{11})$	$O(b_{21})$	$O(b_{11})$	$O(b_{31})$
α_4	$X(b_{13})$	$X(b_{13})$	$X(b_{13})$	$X(b_{23})$	$X(b_{21})$	$X(b_{21})$
α_5	$O(b_{31})$	$O(b_{22})$	$O(b_{31})$	$O(b_{12})$	$O(b_{12})$	$O(b_{23})$
α_6	$X(b_{23})$	$X(b_{12})$	θ	$X(b_{31})$	θ	$X(b_{12})$
α_7	θ	θ	θ	$O(b_{33})$	θ	$O(b_{31})$
α_8	θ	θ	θ	$X(b_{13})$	θ	$X(b_{33})$
α_9	P	P	N	E	N	E

Baseado no mecanismo de aprendizagem de um jogo proposto e apresentado na Figura 1. O estudo de caso é descrito a seguir.

A representação de *Jogadores*, neste caso, é dada por dois jogadores, sendo um deles a própria aplicação, a partir deste momento chamada de "Jogador", e o outro um humano, que será chamado de "Oponente". A jogada de "Jogador" é representada pelo símbolo 'X' e a jogada de "Oponente" pelo símbolo 'O'. Cada jogada é uma *entrada*, representada por $\delta(b_{ij})$. Para escolher uma jogada (*Decisão*, Figura 1), "Jogador" pode consultar a base de regras (*Armazenamento de regras*, Figura 1) e/ou adotar uma estratégia (*Conjunto de Estratégias*, Figura 1), caso estas já possuam regras.

Após Jogador ou Oponente escolher uma jogada, esta deve ser validada. A *Validação* recebe uma jogada $\delta(b_{ij})$ e determina se esta entrada é válida. Por exemplo, se um jogador escolhe a posição b_{23} do tabuleiro a componente deve verificar se a posição existe e está vazia, caso contrário, o jogador deve escolher uma nova posição. Após validação, a jogada passa a ser uma nova regra e deve ser representada adequadamente (*Representação de regras*, Figura 1).

Na fase de *Representação de regras*, cada regra ou jogada deve ser capturada por um autômato distinto, seja de "Jogador" ou de "Oponente". Isso porque as jogadas de "Oponente" são importantes para que "Jogador" aprenda com a derrota. Assim, "Jogador" pode fazer escolhas a partir do que "Oponente" decidiu no passado. Cada exemplo ρ_i é capturado por um autômato.

A seguir, é descrito a técnica utilizada para a construção do autômato baseada em [14], que representa o comportamento dos jogadores.

Passo 1: Iniciar com um autômato conhecido;
Passo 2: Modificar o autômato incrementalmente para aceitar jogadas sucessivas na qual o jogador vença a partida;
Passo 3: Modificar o autômato incrementalmente para rejeitar jogadas sucessivas na qual o jogador perca a partida;
Passo 4: Modificar o autômato incrementalmente para aceitar jogadas sucessivas na qual jogador empate a partida;
Passo 5: Repetir as ações até que o treinamento seja considerado aceitável.

Antes de cada jogada "Jogador" consulta a base de regras (*Decisão*, Figura 1, *Armazenamento de regras*, Figura 1). Para isso executa o algoritmo abaixo:

Passo 1: Para i:= 0 até n faça
Passo 2: Pesquisar no conjunto de transições do autômato se existe alguma transição $?(q_i, \sigma_?) \rangle\rangle q_?$, onde q_i é conhecido. Para todas as transições consumindo σ com estado inicial q_i e estado destino $q_?$, retornar os valores $\sigma_?$ e $q_?$;
a) Se encontrar apenas uma transição: retorna os valores de $\sigma_?$ e $q_?$, aplique a regra;
b) Se não encontrar nenhuma transição: Criar uma nova transição consumindo um símbolo $A \neq \varepsilon$ com estado inicial q_i e estado destino $q_{\#j}$, para j:= i+1. O símbolo # é utilizado para indicar que um novo estado foi gerado;
i) Remova a transição $(q_i, \sigma_?) \rangle\rangle q_?$;
ii) Insira a transição $(q_i, A) \rangle\rangle q_{\#j}$ onde A é uma posição vazia do tabuleiro escolhida e j:= i+1;
iii) Insira a transição: $(q_i, \varepsilon[.\varphi_i]) \rangle\rangle q_{\#j}$;
iv) Inserir a transição: $(q_{\#j}, \varepsilon[.\varphi_j]) \rangle\rangle q_f$;
Passo 3: Escolher a posição σ do tabuleiro.

A SEGUIR, A EVOLUÇÃO DO AUTÔMATO QUE REPRESENTA AS JOGADAS DE "JOGADOR" PARA A PRIMEIRA JOGADA α_0 DE ρ_1 CONSIDERANDO O CONJUNTO T DA

Tabela I.

Assim, para i:= 0, dado o autômato inicial representado na Figura 6, aplica-se a função a ação elementar de inspeção $\mathbf{?(q_i, \sigma_?) \rangle\rangle q_?}$.

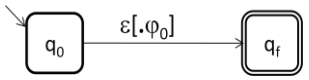


Figura 6 - Configuração Inicial do Autômato.

Ação elementar de Inspeção 1: Se encontrar uma transição retorna os valores de σ? e q?. Aplique a regra.

O resultado da execução da ação elementar de inspeção para a Figura 6 retorna $\mathbf{\sigma = \varepsilon[.\varphi_0]}$ e $\mathbf{q = q_f}$.

Ação elementar de Remoção 1: Remova a transição $(q_i, \sigma_?) \rangle\rangle q_?$

Na sequência aplica-se a função a ação elementar de remoção **-(q_i, ε[.φ_i]) ⟩⟩ q_f** . O resultado da execução da ação elementar de remoção **-(q_0, ε [.φ_0]) ⟩⟩ q_f)** é apresentado na Figura 7.

Figura 7 - Resultado da execução da Ação elementar de Remoção

Ação Elementar de Inserção 1: Insira a transição (q_i,A) ⟩⟩ $q_{\#j}$ onde A é uma posição vazia do tabuleiro escolhida e j:= i+1.

A próxima ação elementar a ser executada é a de inserção **+(q_i, A) ⟩⟩ q_j** . Que neste momento é representada por (q_0, X(b_{22}) ⟩⟩ $q_{\#1}$.

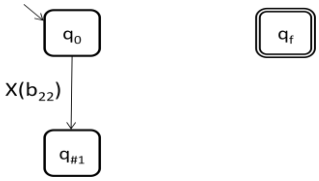


Figura 8 - Resultado da execução da Ação elementar de Inserção **+** (q_i, A) ⟩⟩ q_j

Ação Elementar de Inserção 2: Insira a transição: +(q_i,ε[.φ_i])⟩⟩$q_{\#j}$

Outra ação elementar de inserção é executada na sequência: **+(q_i,ε[.φ_i]) ⟩⟩ q_f** , neste caso **+(q_0 , ε[.φ_0]) ⟩⟩ q_f**.

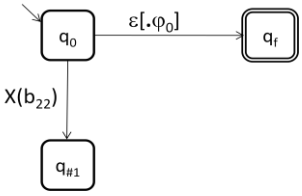


Figura 9 - Resultado da execução da Ação elementar de Inserção +(q_i,ε[.φ_i]) ⟩⟩ q_f

Ação Elementar de Inserção 3: Insira a transição: **+(q_j ,ε[.φ_j]) ⟩⟩ q_f** . Neste caso, a execução de ($q_{\#1}$, ε[.φ_1]) ⟩⟩ q_f resulta no autômato da Figura 10.

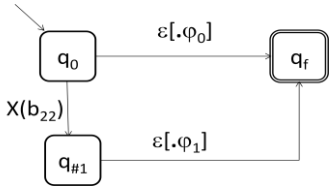


Figura 10 - Resultado da execução da Ação elementar de Inserção (q#1, ε[.φ1]) ⟩⟩ qf

O AUTÔMATO É RESULTADO DA PRIMEIRA JOGADA, CONFORME EXEMPLO DE TREINAMENTO ρ_1 DA Tabela I. No final da partida, uma possível configuração final do autômato adaptativo inferido, após quatro jogadas de "Jogador", é representada pela Figura 11.

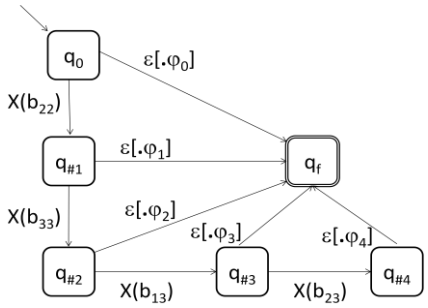


Figura 11 – Autômato Coletor de jogadas de "Jogador".

As funções adaptativas do autômato impõem uma ordenação, pré-determinada, ao conjunto de atributos. É essa ordenação que determina em que jogada da partida irão ocorrer os diversos atributos. Na medida em que exemplos vão sendo submetidos ao dispositivo, a estrutura vai se modificando, na forma de um autômato.

O MESMO ALGORITMO CAPTURA AS JOGADAS DE "OPONENTE", QUE CONSIDERANDO O EXEMPLO DE TREINAMENTO ρ_1 DA Tabela I, apresenta na Figura 12 uma possível configuração final do autômato adaptativo inferido.

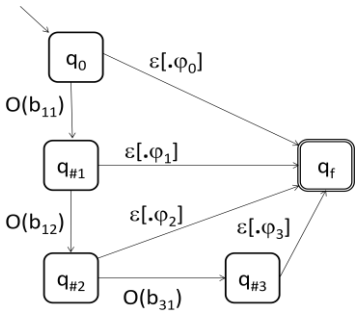


Figura 12 – Autômato Coletor de jogadas de "Oponente".

A primeira partida tem a função de calibrar o mecanismo de aprendizado, "Jogador" realiza suas jogadas sem auxílio do tomador de decisão (*Decisão*, Figura 1), pois ainda não existe um histórico de operações (partidas) que possa ser consultado (*Armazenamento de Regras*, Figura 1).

O mecanismo de aprendizado captura em diferentes autômatos tanto as jogadas de "Jogador" quanto às de "Oponente" e, dependendo do resultado da partida (venceu, perdeu ou empatou), repete as jogadas nas partidas posteriores.

Neste trabalho, se considerarmos que o objetivo final é obter um conjunto de regras para auxílio à tomada de decisão em um jogo de tabuleiro, é conveniente à utilização da Tabela de decisão Adaptativa (TDA). [13] considera que sistemas que envolvem tomadas de decisão podem utilizar Tabelas de Decisão Adaptativas para a construção de sistemas inteligentes que recebem dados de treinamento, aprendem a classificá-los e, futuramente, classificam novos dados.

O conjunto de transições (jogadas) resultante de cada

partida é inserido em uma Tabela de Decisão Adaptativa, porém é relevante destacar que o próprio autômato adaptativo pode representar o conjunto de regras adquiridas. Apesar disso, diferentes dispositivos adaptativos podem apresentar uma solução mais aderente a um problema em particular, assim a TDA representa a memória do mecanismo de aprendizagem.

Outra motivação para a representação utilizando uma TDA está fundamentada em [12] (*apud* [28]), que articula "[...] os dispositivos adaptativos dirigidos por regras podem dar maior flexibilidade às tabelas de decisão, permitindo, não somente a consulta às regras, como também a inclusão e a exclusão de regras durante a operação do dispositivo, transformando, assim, a tabela de decisão numa ferramenta mais poderosa".

A seguir é descrito o algoritmo para construção da TDA.

Passo 1: Iniciar uma TDA conhecida;
Passo 2: Buscar no conjunto de regras R uma ou mais regras que satisfaçam as condições de entrada P_k;
- **a)** Caso encontre uma ou mais regras aplicáveis, inclua todas elas em um conjunto de regras aplicáveis;
- **b)** Caso contrário, prepara o dispositivo para uma nova entrada, aplicando o **Passo 4**;

Passo 3: Extrair as regras aplicáveis do conjunto de regras:
- **a)** Caso apenas uma regra seja aplicável, o dispositivo executa a ação determinada pela regra; essa situação define uma operação determinística;
- **b)** Caso mais de uma regra seja aplicável, uma situação de não determinismo é detectada. As regras são executadas em paralelo;

Passo 4: Decodificar as ações.
- **a)** No caso de uma regra não adaptativa, extrair as ações associadas às regras;
- **b)** No caso de uma regra adaptativa, extrair as ações adaptativas associadas às regras.

Passo 5: Executar as ações:
- **a)** No caso de uma regra não adaptativa, executar as rotinas semânticas associadas a cada ação.
- **b)** No caso de uma regra adaptativa:
  i) Para uma função adaptativa posterior, executar o **Passo 5.a** e na sequência o **Passo 5.c**;
- **c)** Executar as ações adaptativas.

A configuração inicial da TDA está representada na Tabela II. Na configuração inicial apenas uma regra adaptativa, P_a, é inserida na TDA. No decorrer de sua execução, novas regras vão sendo inseridas.

As jogadas capturadas pelo autômato são representadas por um conjunto de regras do tipo "Se c*ondições então Ação"*, onde as condições são as jogadas válidas que levam o "Jogador" a um resultado. A ação é o resultado obtido na partida. O conjunto de regras é utilizado para jogar novas partidas.

A saída do algoritmo é um reconhecedor de partidas τ, neste caso representado por um dispositivo do tipo tabela de decisão adaptativa. Dado uma nova partida χ, o reconhecedor deve verificar se a partida já foi realizada anteriormente, caso já tenha sido realizada aplica a regra aprendida na partida. Caso contrário, aprende uma nova regra.

TABELA II
CONFIGURAÇÃO INICIAL DA TABELA DE DECISÃO ADAPTATIVA.

Cabeçalhos (Tags)			H	?	+	P_a
Condições	Condição$_1$			p_0	v_0	-
	Condição$_2$			p_1	v_1	-
	Condição$_3$			p_2	v_2	-
	Condição$_4$			p_3	v_3	-
	Condição$_5$			p_4	v_4	-
	Condição$_6$			p_5	v_5	-
	Condição$_7$			p_6	v_6	-
	Condição$_7$			p_7	v_7	-
	Condição$_8$			p_8	v_8	-
Ações	Ação$_1$			p_9	v_9	?
Funções Adaptativas	Anterior	f_i	B	√	√	√
	Posterior					
	Parâmetros	p_0	P			p_0
		...	...			...
		p_9	P			p_9
	Variáveis	v_0	V			
		...	...			
		v_9	V			
	Geradores					

A partir da segunda partida, representada pelo conjunto ρ_2 da Tabela I, já existem regras na base de regras (*Memória*) e o "Jogador" pode decidir as jogadas de acordo com as jogadas passadas. Porém, alguns movimentos podem ser aleatórios, sem qualquer auxílio do tomador de decisões, considerando que a posição do tabuleiro está ocupada. A *Decisão* é baseada nas regras armazenadas na Tabela de Decisão Adaptativa, caso nenhuma regra seja aplicável, deve-se decidir aleatoriamente e aprender uma nova regra.

O RESULTADO DA EXECUÇÃO DA TDA PARA O CONJUNTO DE JOGADAS CAPTURADAS PELOS AUTÔMATOS DE FIGURA 11E FIGURA 12 ATRAVÉS DO CONJUNTO DE TREINAMENTO DA

Tabela I é o conjunto de partidas P = {P_1, P_2,..., P_k}, onde k é o número de partidas, conforme Tabela III. As condições representam a ordem das jogadas capturadas pelo autômato, tanto de "Jogador" quanto de "Oponente".

O exemplo a seguir ilustra a operação da TDA para representar o conhecimento adquirido pelo autômato.

Considere o exemplo de uma partida χ = {X(b_{22}), O(b_{11}), X(b_{13}), O(b_{31}), X(b_{21}), O(b_{23}), X(b_{12}), O(b_{31}), X(b_{33}), Empate}.

Neste caso, não existe uma regra aplicável, ou seja, uma partida já realizada com a mesma configuração. Porém, é sabido que χ é um exemplo de partida que "Jogador" venceu e deve ser adicionado à TDA. Na TDA a regra adaptativa P_a é satisfeita toda vez que nenhuma regra não adaptativa for aplicável. A regra P_a possui a seguinte função adaptativa

associada:

TABELA III
TABELA DE DECISÃO ADAPTATIVA APÓS 6 MODIFICAÇÕES.

Cabeçalhos (Tags)			H	?	+	P1	P2	P3	P4	P5	P6	Pa
Condições	Jogada_1_x			p0	v0	X(b22)	X(b11)	X(b12)	X(b31)	X(b22)	X(b22)	θ
	Jogada_1_o			p1	v1	O(b11)	O(b33)	O(b31)	O(b22)	O(b13)	O(b11)	θ
	Jogada_2_x			p2	v2	X(b33)	X(b31)	X(b33)	X(b11)	X(b33)	X(b13)	θ
	Jogada_2_o			p3	v3	O(b12)	O(b21)	O(b11)	O(b21)	O(b11)	O(b31)	θ
	Jogada_3_x			p4	v4	X(b13)	X(b13)	X(b13)	X(b23)	X(b21)	X(b21)	θ
	Jogada_3_o			p5	v5	O(b31)	O(b22)	O(b31)	O(b12)	O(b12)	O(b23)	θ
	Jogada_4_x			p6	v6	X(b23)	X(b12)	θ	X(b31)	θ	X(b12)	θ
	Jogada_4_o			p7	v7	θ	θ	θ	O(b33)	θ	O(b31)	θ
	Jogada_5_x			p8	v8	θ	θ	θ	X(b13)	θ	X(b33)	θ
Ações	Resultado			p9	v9	P	P	N	E	N	E	?
Funções Adaptativas	Anterior	f1	B	√	√							√
	Posterior											
	Parâmetros	p0	P									p0
		...	...									...
		p9	P									p9
	Variáveis	v0	V									
		...	...									
		v9	V									

Exemplo do pseudocódigo para declaração de uma função adaptativa.

```
Função Adaptativa f1(Parâmetros: p1, p2, p3, p4, p5, p6, p7, p8, p9){
   Variáveis: v1, v2, v3, v4, v5, v6, v7, v8, v9
   Geradores: necessários para gerar novas regras
   Ações elementares Δ {
       δ1:?[v1, v2, v3, v4, v5, v6, v7, v8, v9]
       (Existe uma regra que satisfaça as condições χ).
       δ2:+[v1,v2,v3,v4,v5,v6,v7, v8, v9](Inserir uma regra Pk).
   }
}
```

No caso de aplicação da regra P_a em χ, os valores dos atributos são atribuídos aos parâmetros da função f_1. A função recebe os valores como parâmetro e atribui os valores dos parâmetros para as variáveis. Na sequência, a ação elementar de consulta pesquisa se existe uma regra em R que satisfaça χ e a ação elementar de inserção inclui uma regra P_k, para n= 1, 2,... k-1, onde n é o número de regras atuais da TDA.

No domínio do Jogo da velha, cada jogador enfrenta um desafio básico: como criar uma estratégia flexível o suficiente para lidar com certas situações de mudança a cada jogada. A estratégia é formada por um conjunto de regras que permite aumentar as chances de vitória de um jogador. O mecanismo de aprendizagem deve aprender não somente a aplicar jogadas válidas, mas também como aplicar uma jogada que garanta uma vitória ou, no pior caso, leve a partida a um empate.

Após várias partidas, a base de regras se torna uma rica fonte de dados que representa o comportamento dos jogadores e estimula a execução de uma próxima etapa do aprendizado, que é a descoberta de estratégias de jogo.

O objetivo da descoberta de uma estratégia para jogar o Jogo da velha considera que os jogadores devem usar os meios mais adequados para atingir seus objetivos, seja vencer ou não perder. Assim, após diversas partidas o jogador acumula experiências e essa experiência pode conduzir os jogadores a um comportamento estratégico. Cada jogador toma decisões considerando que elas terão efeitos sobre os outros jogadores, bem como as decisões dos outros jogadores terão efeitos sobre suas decisões. A estratégia tenta minimizar o efeito negativo de cada decisão.

Esta etapa de aprendizagem ou descoberta de uma estratégia é representada pelo *Mecanismo de Inferência*.

O *Mecanismo de Inferência* atua sobre o conjunto de regras aprendidas, armazenado na base de regras, e infere um conjunto de estratégias que poderão ser consultadas na tomada de decisão dos jogadores.

Neste trabalho, a aprendizagem de estratégias é baseada nas medidas de ganho de informação e entropia, similar ao ID3[21].

A entropia, no contexto da teoria de informação, pode ser considerada como uma medida da quantidade de informação que uma pessoa necessita para organizar seus conhecimentos e descobrir uma regra. Analogamente, será adotada como a medida para organizar o conhecimento adquirido por um jogador e descobrir um conjunto de regras para realizar as jogadas de forma estratégica.

Quanto mais alternativas uma sistema de tomada de decisão possui (ex.: mais jogadas possíveis), mais informações são necessárias para aprender a tomá-las (maior entropia). Se um sistema de tomada de decisão não tem alternativas, não é necessária nenhuma informação (a entropia é 0).

A seguir é descrito um algoritmo para inferência de regras utilizando ganho de informação.

1: Para cada jogada K de "Jogador" faça:
   a. Crie um estado que representa a jogada K.
   b. Calcule o ganho de informação de cada jogada (valor de atributo).

$$Ganho(S,A) = Entropia(S) - \sum_{v \in V} \frac{|S_v|}{|S|}.Entropia(S_v)$$

   Onde:

$$\text{Entropia (S)} = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

   c. Estender a árvore adicionando um ramo para cada valor do atributo.
   d. Dividir o conjunto de exemplos P (tendo em conta o valor do atributo escolhido) e passe os subconjuntos para as folhas da árvore.
   e. Repetir os passos para cada novo nó gerado.

Seja *P* o conjunto de exemplos de treinamento armazenado na base de regras do mecanismo de aprendizagem. A base de

regras é composta por 958 exemplos (626 exemplos de vitória e 332 exemplos de derrota) extraídos do repositório de dados UCI *Machine Learning Repository*. Estes dados foram tratados de forma que a ordem das jogadas fossem representados no conjunto. Seja 9 o total de atributos, cada atributo corresponde a um uma posição do tabuleiro do Jogo da Velha.

Aplicando a fórmula ao subconjunto S temos:

$$\text{Entropia}\,(S) = -\frac{626}{958}\log_2\frac{626}{958} - \frac{332}{958}\log_2\frac{332}{958} = 0.93$$

Considere os resultados obtidos com o calculo do ganho de informação: $\text{Ganho}(S, b_{11})$ = 0.93- 0.91 = 0.02, $\text{Ganho}(S, b_{12})$ = 0.93 - 0.92 = 0.01, $\text{Ganho}(S, b_{13})$ = 0.93 - 0.91 = 0.02, $\text{Ganho}(S, b_{21})$ = 0.93 - 0.92 = 0.01, $\text{Ganho}(S, b_{22})$ = 0.93 - 0.84 = 0.09, $\text{Ganho}(S, b_{23})$ = 0.93 - 0.96 = -0.03, $\text{Ganho}(S, b_{31})$ = 0.93 - 0.91 = 0.02, $\text{Ganho}(S, b_{32})$ = 0.93 - 0.92 = 0.01, $\text{Ganho}(S, b_{33})$ = 0.93 - 0.91 = 0.01.

Na primeira iteração do algoritmo, é possível concluir que a jogada com maior ganho de informação é escolher a posição central do tabuleiro, ou seja, $\delta(b_{22})$. A partir da segunda jogada, o ganho de informação é calculado baseado na jogada de "Oponente".

A Figura 13 mostra a possibilidade de jogadas e seus valores correspondentes.

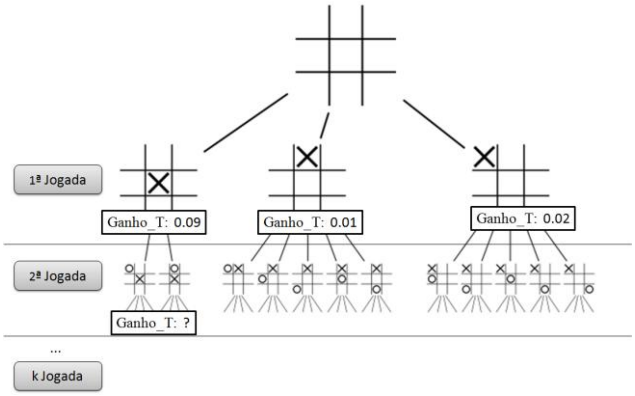


Figura 13 – Representação das possibilidades de jogadas com ganho de informação.

Na sequência, os cálculos de ganho de informação são refeitos, considerando cada possível jogada de "Oponente", e assim sucessivamente até obter uma árvore que seja capaz de auxiliar o jogador na tomada de decisão aplicando uma estratégia.

A seguir é descrito como analisar o comportamento dos jogadores a partir das regras geradas do conjunto de estratégias. As estratégias de "Jogador" consideram que este deve iniciar a partida, porém é perfeitamente possível extrair uma estratégia que permita a "Oponente" iniciar a partida.

A *Decisão* das jogadas utilizando uma estratégia é baseada no *Conjunto de estratégias*.

Existem diversos algoritmos disponíveis para decidir as jogadas de "Jogador" em resposta aos movimentos do "Oponente" durante o jogo. A seguir é apresentada uma abordagem heurística simples para automatizar os movimentos do computador, para o nível iniciante.

Passo 1: Verifique se existe um movimento que o computador pode fazer de modo que obterá maior ganho de informação.
- **a)** Em caso afirmativo, preencher o quadrado relevante.
- **b)** Senão, verifique se há um movimento que irá bloquear uma vitória para o outro jogador e preencha o quadrado correspondente.
- **c)** Senão, preencha o quadrado que fica na linha/coluna com o número máximo de células livres de marcas.

O Jogo da Velha com estratégias deve permitir que, para uma particular instância do jogo, "Jogador" vença ou pelo menos empate a partida. As estratégias inferidas a partir da base de regras são as seguintes:

1. *Se* Jodada_1_x *então* escolha X(b_{22}): "Jogador" deve fazer a jogada de abertura, neste caso a melhor jogada é escolher a posição central do tabuleiro com maior ganho de informação (Ganho_T: 0.09).

Na sequência, o adversário pode marcar uma lateral ou um canto. Por exemplo, caso o "Oponente" marque um dos cantos, é observado que "Jogador" deve escolher o canto oposto, formando uma linha diagonal, conforme a Figura 14.

2. Se Jodada_2_o = O(b_{33}) *então* escolha X(b_{11}): Dependendo da opção escolhida pelo "Oponente", "Jogador" deve escolher aquela com maior ganho de informação. Para a jogada O(b_{33}), O(b_{11}), O(b_{13}) e O(b_{31}) a melhor opção é o canto oposto, caso ele esteja vazio. Caso contrário, escolhe-se a próxima opção com melhor ganho de informação, e assim sucessivamente.

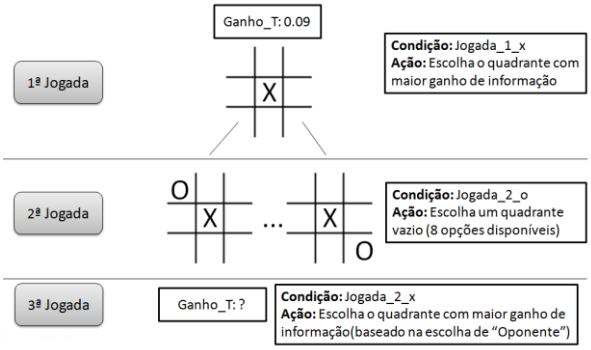


Figura 14 – "Jogador" escolhe a 3ª jogada baseada na escolha do oponente.

Se o próximo movimento de "Oponente" for adjacente da marca anterior, "Jogador" vai ter boas chances de vencer, como mostrado Figura 15.

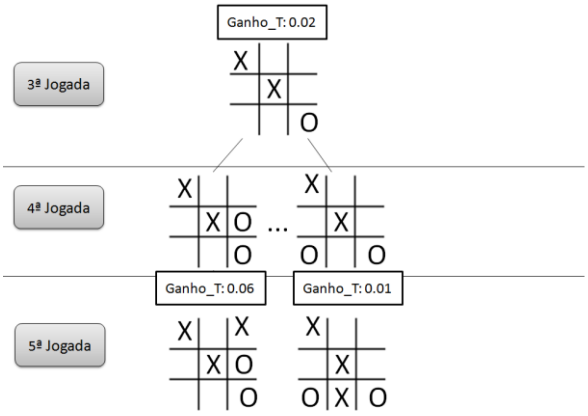


Figura 15 – "Jogador" escolhe a 5ª jogada baseada na escolha do oponente.

3. Se Jodada_2_o = O(b_{23}) *então* escolha X(b_{13}): Neste caso, o "Jogador" deve retaliar o "Oponente". O termo retaliar significa que "Jogador" deve bloquear todas as tentativas de vencer do "Oponente".

Considerando que "Jogador" tome as decisões baseadas nas regras que definem uma estratégia, o clássico Jogo da Velha pode ser jogado de modo que "Jogador" é direcionado para uma vitória ou um empate.

IV. Discussão dos resultados

A aprendizagem de máquina usando adaptatividade considera a integração de técnicas de aprendizagem de máquina, determinísticas ou estatísticas, e técnicas adaptativas para a construção de sistemas de aprendizado.

Na aprendizagem tradicional, ou seja, sem a utilização de técnicas adaptativas, os ajustes para a melhoria no processo de aprendizagem, em geral, ocorrem na fase de treinamento. Porém, muitas informações relevantes podem ser capturadas na fase de utilização do sistema (ex.: na fase de classificação).

Em geral, isso ocorre quando o problema é de natureza dinâmica. Com a utilização de técnicas adaptativas, através dos dispositivos adaptativos, é permitido descrever de forma natural os aspectos dinâmicos dos problemas de aprendizagem.

A aprendizagem adaptativa é uma forma de adaptar gradualmente o modelo aprendido. A adaptatividade pode ser útil para detectar ajustes no conjunto de regras e modificar o modelo, de maneira incremental. Embora existam maneiras de refazer a estrutura de dispositivos convencionais, tais como árvores e tabelas de decisão, após a obtenção de novos dados, refazer o conjunto de regras pode ser ineficiente, podendo levar à perda parcial ou total de informações que haviam sido anteriormente aprendidas.

A grande vantagem da adaptatividade sobre outras técnicas correntemente utilizadas para a formulação de modelos de representação e de manipulação do conhecimento reside no fato de que [24]:

(a) a informação total, encerrada no dispositivo adaptativo, está representada integralmente no conjunto de regras. A memória é representada pelo próprio dispositivo;

(b) a aprendizagem angariada em cada passo adaptativo do dispositivo encontra-se integralmente confinada à variação sofrida pelo conjunto de regras. É possível determinar o que foi aprendido pelo dispositivo adaptativo analisando as alterações de configuração durante o processo de aprendizagem;

Em segundo plano, a adaptatividade permite que as diversas características inerentes aos problemas reais de aprendizagem, tais como comportamento dinâmico e estocástico, possam ser tratadas de forma transparente. A interação do especialista no mecanismo de aprendizagem é simplificada e não apresenta complexidades técnicas, pois estas devem ficar a cargo do mecanismo de aprendizagem. De fato, o especialista deve manter o controle da definição do problema, incluindo o conhecimento sobre o domínio, restrições ou preferências através da definição das funções adaptativas. Esta é a parte que, dependendo do nível da adaptatividade, não pode ser automatizada.

Uma vantagem comparada a outros mecanismos de representação, tais como naïve Bayes, é que o autômato armazena a sequência das jogadas, assim é possível saber a ordem em que as jogadas foram efetuadas no tabuleiro. Essa relação de ordem das jogadas fica incorporada na estrutura do autômato.

Contudo, os métodos que utilizam dispositivos simbólicos para representar o que foi aprendido são adequados quando é desejável obter um conjunto de regras mais compreensível por especialistas humanos. Considerando que esses dispositivos podem ser representados por um conjunto de regras do tipo "Se...então", os dispositivos adaptativos se enquadram na representação simbólica das regras aprendidas. Tal formato utilizado em dispositivos adaptativos fornece uma perspectiva unificada sob a qual as regras de um mecanismo de aprendizado podem ser convertidas e analisadas.

Outro aspecto percebido, similar ao observado em dispositivos baseados em regras, tais como árvores de decisão, expressões lógicas, regras de produção e tabelas de decisão, é a naturalidade que se tem em modelar os problemas de aprendizagem de máquina utilizando diferentes dispositivos adaptativos. Os autômatos adaptativos, as tabelas de decisão adaptativas e as árvores de decisão são equivalentes e facilmente convertidos.

Em particular, os dispositivos adaptativos, tais como autômatos adaptativos, árvores ou tabelas de decisão adaptativas, permitem a representação estrutural das regras. Segundo [7], a criação de estruturas simbólicas que sejam compreensíveis e utilizadas por modelos mentais na aprendizagem é mais interessante do que os modelos estatísticos.

A combinação equilibrada entre a obtenção de modelos de aprendizagem compreensíveis e expressivos pode ser adquirida com a utilização de dispositivos adaptativos. Tanto os autômatos adaptativos quanto as tabelas de decisão adaptativas, e intuitivamente, outros dispositivos adaptativos utilizados para representar o conhecimento adquirido em sistemas de aprendizagem, facilitam o entendimento das regras. A estrutura simples e compreensível dos dispositivos

adaptativos são ferramentas úteis para a descoberta do conhecimento.

V. CONSIDERAÇÕES FINAIS

A aprendizagem de máquina utilizando tecnologia adaptativa pode ser considerada uma técnica inspirada na aprendizagem indutiva supervisionada com o objetivo de melhorar o entendimento das regras, mas principalmente cumprir exigências de eficiência. Na técnica, uma base de regras é utilizada para representar o comportamento inteligente de um jogador em função de experiências passadas.

As regras (ex.: uma jogada é uma regra) são extraídas da memória e aplicadas durante as partidas podendo gerar novas regras que alteram o comportamento dos jogadores ou a base de regras (inclusão ou exclusão de regras). A probabilidade de aplicação de uma regra que leva o jogador a uma vitória é proporcional ao ganho de informação associada a esta regra (ex.: jogadas com maior ganho de informação têm maior probabilidade de vencer a partida).

A aplicação da medida estatística baseada no ganho de informação pode ser considerada uma forma de avaliar os dispositivos adaptativos. A avaliação é interessante para testar a eficácia e eficiência dos dispositivos adaptativos. Neste caso, não parece ser vantajoso considerar que os dispositivos adaptativos apenas melhoram a expressividade e a compreensibilidade da solução, comparados a dispositivos não adaptativos. Porém, ainda é necessário avançar em formas de avaliar os dispositivos para analisar o funcionamento das funções adaptativas e determinar se as ações correspondentes estão atingindo os resultados esperados.

REFERÊNCIAS

[1] ALPAYDIN, E.: "Introduction to Machine Learning". MIT Press, 2ª Edição, 2010. ISBN-10: 0-262-01243-X

[2] BECK, J; "Combinatorial Games: Tic-Tac-Toe Theory", Cambridge University Press, 2008.

[3] DIAS, J. B.; SOUZA, K. P. de; PISTORI, H.: Conjunto de Treinamento para Algoritmos de Reconhecimento de LIBRAS. II Workshop de Visão Computacional, São Carlos, Outubro 16-18, 2006.

[4] GANZELI, H. S.; BOTTESINI, J. G.; PAZ, L. O.; RIBEIRO, M. F. S.: Skan-Skin Scanner: software para o reconhecimento de câncer de pele utilizando técnicas adaptativas. Memórias do WTA 2010 – IV Workshop de Tecnologia Adaptativa, São Paulo, 2010.

[5] HIRAKAWA, A. R.; SARAIVA, A. M.; CUGNASCA, C. E.: Autômatos Adaptativos Aplicados em Automação e Robótica. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 539-543)

[6] MENEZES, P. B.: "Linguagens Formais e Autômatos". 6ª Edição. Bookman, 2011. ISBN: 9788577807659

[7] MICHALSKI, R. S.: "A theory and methodology of inductive learning". In Machine Learning L R. S. Michalski, J. Carbonelli, and T. Mitchell, eds. Palo Alto, CA: Tioga Publishing, 1983.

[8] MITCHELL, T. M.: "Machine Learning". 1ª Edição. McGraw-Hill, 1997. ISBN: 0070428077.

[9] MONARD, M. C.; BARANAUKAS, J. A.: "Aplicações de Inteligência Artificial: Uma Visão Geral". São Carlos: Instituto de Ciências Matemáticas e de Computação de São Carlos, 2000.

[10] NETO, J. J.: "Contribuição à metodologia de construção de compiladores". São Paulo, 272p. Tese (Livre-Docência), Escola Politécnica, Universidade de São Paulo, 1993.

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

[12] NETO, J. J. "Adaptive Rule-Driven Devices – General Formulation and Case Study". Revista de Engenharia de Computação e Sistemas Digitais, São Paulo, v.1, n.1, 2001.

[13] NETO, J. J. "Adaptive Technology and Its Applications". Encyclopedia of Artificial Intelligence. New York, v.1, p.: 37-44, 2009.

[14] NETO, J. J., IWAI, M. K.: "Adaptive Automata for Syntax Learning". In Anais da XXIV Conferência Latino-americana de Informática - CLEI 98, pg. 135–149, Quito, Equador, 1998.

[15] PARIENTE, C. B.; NETO, J. J.; SANTANA, F. S.: Towards an Adaptive Implementation of Genetic Algorithms. I Taller Latinoamericano de Informática para la Biodiversidad (INBI) - CLEI 2007, San José, Costa Rica, 9-12 Octubre, 2007.

[16] PISTORI, H. "Tecnologia Adaptativa em Engenharia de Computação: estado da arte e aplicações". Tese (Doutorado), Escola Politécnica da USP, 2003.

[17] PISTORI, H.; NETO, J. J.: AdapTree: Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas. Anais Conferência Latino Americana de Informática - CLEI 2002. Montevidéu, Uruguai, Novembro, 2002.

[18] PISTORI, H.; NETO, J. J.: "Decision Tree Induction using Adaptive FSA". CLEI Eletronic Journal. Volume 6, Number 1, 2003.

[19] PISTORI, H.; NETO, J. J.: "An Experiment on Handshape Sign Recognition using Adaptive Technology: Preliminary Results". XVII Brazilian Symposium on Artificial Intelligence - SBIA 04. São Luis, September 29 - October 1, 2004

[20] PRESSMAN, R. S.: "Engenharia de Software". 6ª Edição. Rio de Janeiro: McGraw-Hill, 2006. ISBN: 8586804576.

[21] QUINLAN, J. R.: "Induction of decision trees". Machine Learning, 1, 81-106, 1986.

[22] RAJANI, N.F. DAR, G. BISWAS, R. RAMESHA, C.K.: Solution to the Tic-Tac-Toe Problem Using Hamming Distance Approach in a Neural Network. In: Second International Conference on Intelligent Systems, Modelling and Simulation - ISMS 2011, Cambodia, 2011.

[23] ROCHA, R.L.; NETO, J.J.: "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, p. 33-48, 2000.

[24] STANGE, R. L.; NETO, J. J.: Aprendizagem Incremental Usando Tabelas De Decisão Adaptativas. Memórias do WTA 2011 – Workshop De Tecnologia Adaptativa, EPUSP, São Paulo, 2011.

[25] STANGE, R. L.; GIANNINI, T. C.; SANTANA, F. S.; JOSE, J.; MAURO SARAIVA, A.: Evaluation of Adaptive Genetic Algorithm to Environmental Modeling of Peponapis and Cucurbita. Revista IEEE América Latina, v. 9, p. 171-177, 2011.

[26] TCHEMRA, A. H.: Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão. I WTA – Workshop sobre Tecnologia Adaptativa, São Paulo, 2007.

[27] TCHEMRA, A. H.: "Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério". Tese (Doutorado), EPUSP, São Paulo, 2009.

[28] TCHEMRA, A. H.; CAMARGO, R.: Descoberta de padrões em bases de dados utilizando Técnicas Adaptativas. III WTA – Workshop sobre Tecnologia Adaptativa, São Paulo, 2009.

[29] WIDYANTORO, D. H.; VEMBRINA, Y. G: Learning to play tic-tac-toe. In: International Conference on Electrical Engineering and Informatics, 2009.

[30] YAU Y.J., TEO J. AND ANTHONY P.: Evolution and Co-Evolution in Cognitive Neural Agents Synthesis for Tic-Tac-Toe. IEEE Symposium on Computational Intelligence and Games (CIG 2007), pages 304-311, Hawaii, USA, 2007.

Sobre o uso de formalismos adaptativos no gerenciamento de diálogos em robôs sociáveis

D.A. Alfenas, M. R. Pereira-Barretto, R. S. Paixão

***Abstract*— Conversation analysis, ethnomethodology and many other fields of research have improved how much the humanity knows about daily conversation. In this work, we use that knowledge to evaluate existent adaptive formalisms on how fit to sociable robots dialog management they are.**

***Keywords*— Sociable robots, dialog management, adaptive technology, conversation analysis.**

I. Introdução

Desde o artigo publicado no VI Workshop de Tecnologia Adaptativa [1], em que foi proposta uma arquitetura de robôs sociáveis capazes de dialogar com seres humanos de forma simples e sobre assuntos gerais, o grupo do Laboratório de Robôs Sociáveis (LRS) trabalhou em diversas frentes, como memória, detecção de emoções, ASR (reconhecimento automáticos de voz) e gerenciamento do diálogo. Este último, dentro da arquitetura proposta, é a central consciente do robô, responsável por todas as ações com propósito comunicativo, como falar algo ou olhar em alguma direção específica quando solicitado. Os demais módulos o alimentam com as diversas informações sobre o mundo externo, como a localização e as falas do interlocutor, e com o estado interno do robô, como memória, motivações e emoções.

Este artigo discute a utilização de dispositivos adaptativos na modelagem do gerenciamento do diálogo, buscando uma representação mais natural e parecida com a de um ser humano. Para entender melhor como a conversação acontece, é necessário buscar estudos relevantes também fora da computação e da engenharia, como na psicologia, na linguística e na sociologia. Duas correntes desta última possuem grande relação com este trabalho: a etnometodologia e a análise da conversação. A primeira surgiu na década de 60 e tornou-se mais conhecida a partir da publicação de *Studies in Ethnomedhodology*, de Garfinkel[2]. Essa corrente de estudos é voltada a análise das propriedades da ação e do raciocínio prático implicados nas situações ordinárias da vida cotidiana [3]. A análise da conversação tem inspiração etnometodológica e ficou conhecida através do trabalho de Harvey Sacks na década de 70. Ela abrange temas como a abertura e o encerramento das conversações, revezamento de turnos e o funcionamento dos pares adjacentes. Além dos aspectos verbais da conversação, consideraremos também os aspectos não verbais, tais como gestos e prosódia [4], cuja importância não pode ser ignorada, conforme evidenciado em diversos trabalhos, nas áreas mencionadas e em outras correlacionadas[5].

Para modelar tais aspectos no gerenciador de diálogos é necessária a utilização de modelos adaptáveis, que é característica obrigatória de um dispositivo adaptativo. A adaptatividade é um termo que se refere à capacidade de um sistema de, sem a interferência de qualquer agente externo, tomar a decisão de modificar seu próprio comportamento, em resposta ao seu histórico de operação e aos dados de entrada [6].

O restante deste artigo está estruturado da seguinte maneira: no tópico seguinte, apresenta-se com mais detalhes o robô e o próprio gerenciador de diálogo. No tópico III. é apresentada a revisão de diversos textos das áreas mencionadas buscando pontos que devem ser considerados na modelagem. Em IV. especifica-se brevemente as entradas, as saídas e o modelo interno do gerenciador. Em V, é feita a avaliação de alguns dispositivos adaptativos já existentes para uso interno no gerenciador. O último capítulo contém a conclusão sobre o trabalho.

II. A Minerva e o Gerenciamento do diálogo

A figura de um gerenciador de conversação não é nova em arquiteturas de robôs sociáveis e outros agentes conversacionais. Ela é, na verdade, bem comum [1]. Entretanto, existem vários aspectos específicos do nosso GD que enunciamos a seguir.

Como sugerido no próprio nome do nosso gerenciador, ele é responsável apenas por comunicações relacionadas ao diálogo, isto é, quando uma única pessoa conversa com o robô. A partir disso, removemos do projeto algumas situações específicas de conversas com vários participantes, como a do *tropo* comunicacional [7] (que ocorre quando o destinatário real de uma fala não é o destinatário aparente, mas um terceiro) ou o cisma (*schism*) da conversa com mais de três pessoas [8].

Outro motivo que torna nosso GD específico é o robô para o qual ele está sendo feito, a Minerva. Uma foto dela está na Figura 1. Como é possível observar, a Minerva não tem corpo: apenas a cabeça. Desta forma ela não se locomove ou gesticula. Além disso, o robô não tem olfato e, por hora, nem tato. Todas as demais capacidades estão, direta ou indiretamente, relacionadas ao GD. São elas:

D. A. Alfenas, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, d.alfenas@fdte.org.br.

M. R. Pereira-Barretto, Escola Politécnica da USP (EPUSP), São Paulo, Brasil, marcos.barretto@poli.usp.br.

R. S. Paixão, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, r.paixao@fdte.org.br.

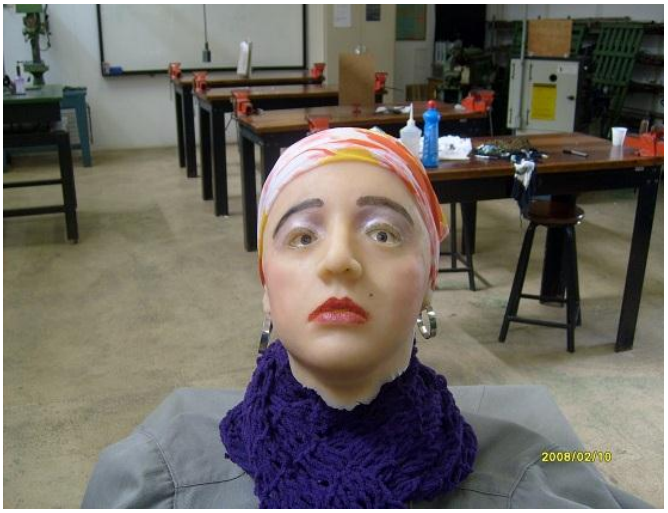


Figura 1 Foto do rosto antigo (2010) da Minerva, o robô em que o GD deve rodar.

- Escutar. Diretamente relacionada;
- Falar. Diretamente relacionada;
- Enxergar e detectar movimento. Indiretamente relacionados através de eventos específicos processados por outros módulos possíveis, como aproximação, detecção de gestos, reconhecimento facial e reconhecimento de emoções faciais;
- Movimentação do pescoço e dos olhos. Na maior parte do tempo, o relacionamento é indireto. O controle destes membros é feito por seus próprios módulos, que se preocupam em torna-los consistentes com as ações tomadas pelo GD. Entretanto, se alguém diz para o robô olhar para um objeto específico, por exemplo, a cognição de tal fala passa obrigatoriamente pelo GD, que deve gerar ele mesmo um comando de direcionamento – resultando em controle consciente de tais membros;
- Mover os músculos da face para demonstrar emoções. Indiretamente relacionado. O controle destes músculos mecânicos é feito por seus próprios módulos, que, assim como os anteriores, também devem se preocupar com a consistência com as ações tomadas pelo GD.

A partir destas capacidades, é possível determinar uma série de eventos relevantes para o GD, cada um com suas propriedades específicas. Esses eventos precisam ser avaliados quanto a seu significado e sua função comunicativa, de tal forma que o robô possa, estando consciente de suas capacidades, escolher a melhor ação para atingir seus objetivos.

Ainda se está em debate quais são os geradores internos de eventos que devem ser considerados. O robô pode simular necessidades internas do ser humano, tanto fisiológicas como o cansaço e a doença quanto psicológicas como mudanças de humor ou vontade de ver algo [9]. Pode ainda estar ciente de necessidades reais do sistema robótico, como energia próxima do fim ou rotinas de *house-keeping* (manutenção do sistema decorrente do uso diário, como atualizações ou reorganização da memória permanente). Até a escrita deste artigo, nenhum desses geradores internos está sendo considerado.

III. Contribuições de outras áreas do conhecimento

A. *Multimodalidade na Comunicação*

É bastante evidente que um robô sociável que pretenda se comunicar na forma mais similar possível a de um ser humano deve considerar um modelo multimodal de comunicação que inclua gestos, movimento da cabeça, expressão facial e mesmo orientação corporal e direcionamento dos olhos [10], além da voz em suas partes verbais e prosódicas.

O trabalho de Goodwin [11], relacionando análise da conversa e teoria da ação, foi, talvez, o marco inicial a favor da ideia de que a análise da ação humana deveria considerar o uso simultâneo de múltiplos recursos semióticos pelos participantes de uma interação conversacional. Argumentou contra a ideia, bastante comum até então, de que tal análise deveria tratar a comunicação verbal falada como primária e o resto como contexto. Ele introduz o conceito de *campo semiótico*, que pode ser entendido como uma fonte distinta de sinais que colabora simultaneamente com outros campos para que as ações humanas possam ser compreendidas; podemos citar diversos exemplos de campos semióticos, tais como a parte verbal da fala de uma pessoa, a prosódia de tal fala e até mesmo a posição dos participantes de um jogo de futebol em relação à quadra em que partida é jogada. Um conjunto particular de campos semióticos aos quais os participantes de uma ação se orientam em um determinado momento é chamado de *configuração contextual*.

Lógicas similares de análise têm levado, na última década, a análises da conversa cada vez mais complexas e sofisticadas, como, por exemplo, em [12]. Obviamente, os campos semióticos considerados em um trabalho de um robô sociável são restringidos tanto pela capacidade técnica e pelo esforço necessário para construir um sistema computacional tão complexo quanto pela própria falta de conhecimento de como o ser humano interpreta e sincroniza esses campos. Entretanto, uma boa arquitetura para o GD deveria considerar a adição futura de *plug-ins* (componentes de software que adicionam ou alteram características de um software maior, e que podem ser atualizados ou desativados separadamente daquele em que se acopla) para permitir a captação de campos semióticos originalmente não considerados.

B. *Comunicações básicas e motivações sociais*

Tomasello [13] descreve três tipos básicos de comunicação: solicitar, informar e compartilhar. Elas podem ser utilizadas sem depender de um canal verbal. Destes três, podemos inferir vários outros subtipos, como reforçar, negar, agradecer, etc. Quando trazemos a linguagem e as motivações sociais do ser humano moderno, esta lista de comunicações básicas aumenta. Uma das mais discutidas estruturas da conversão até hoje, os pares adjacentes [14] e suas diversas expansões trazem uma série de tipos básicos de comunicações que precisam ser atendidos pelo gerenciador de diálogos, e que tem forte relação com as intervenções e expectativas. Como exemplos de pares adjacentes, podemos citar pergunta e resposta, elogio e agradecimento e solicitação de saudação e resposta de saudação.

Nossa análise em cima dessas estruturas de conversação e exemplos práticos nos levou a uma extensa série de intervenções possíveis, baseadas principalmente no padrão para anotações de diálogo ISO 24617-2 [15]. Podemos citar alguns exemplos:

- Responder uma pergunta. O robô pode saber ou não saber a resposta ou, ainda, não estar certo de que a resposta que ele possui está certa. Dependendo de uma série de fatores, o robô pode (i) responder sinceramente, (ii) evitar a resposta ou (iii) mentir.
- Perguntar algo relacionado a seus objetivos e planos. Por exemplo, ao iniciar uma conversa com um desconhecido, o objetivo inicial do robô seria conhecer melhor o interlocutor, associando uma série de objetivos de hierarquia menor (descobrir nome, descobrir idade, descobrir gostos, etc.) que resultarão em um plano de execução composto dessas perguntas.
- Perguntar algo relacionado a uma situação social. Por exemplo, se alguém perguntar ao robô sua idade, não é só esperado que ele responda, mas também que pergunte também ao interlocutor sua idade, mesmo que tal pergunta não estivesse nos planos e objetivos originais do robô.
- Lembrar o interlocutor de alguma pendência. O assunto pode ter sido claramente solicitado pelo interlocutor previamente, como em "Lembre-me do aniversário de minha mãe um dia antes", ou consequência de uma conversa anterior, como alguma pergunta feita ao robô cuja resposta ele não sabia.
- Reforçar ou se opor a alguma ação prévia do interlocutor. Isso pode acontecer porque o robô "gostou" (alinhamento com seus planos, objetivos, afetos e motivações) ou porque não é verdade.

C. Livre de contexto e ao mesmo tempo dependente de contexto

A conversação tem características estruturais livre de contexto e, ao mesmo tempo, é extraordinariamente capaz de sensibilidade de contexto, como já observado por outros autores [8]. Em parte é livre de contexto porque não é possível limitar as ações do interlocutor ou os imprevistos que possam acontecer no ambiente: um robô sociável, em um modelo definitivo, deveria estar preparado para reagir a qualquer situação. Segue que a ação do robô é sempre situada, do qual o planejamento é apenas uma parte [16], e, portanto, há sensibilidade ao contexto. Mesmo em relação aos fatores fora do controle do robô existem expectativas fortemente vinculadas ao contexto.

O modelo do GD deve refletir essas características.

D. Estrutura da conversa

Além dos pares adjacentes, existem outras estruturas da conversa relevantes para o diálogo. Entretanto, na Análise da Conversa, é possível diferenciar pelo menos duas abordagens distintas para esta estruturação: a linguística pragmática e a semântica-ideacional. Na primeira, a comunicação verbal direciona toda a análise, como em [7], que embora mencione a importância do não verbal, estrutura o diálogo através de sequência de trocas de fala e divide cada intervenção em atos de fala. Na segunda, o foco é nas ideias transmitidas, quer seja através de gestos e expressões faciais ou verbalmente, como no trabalho de Calbris [17], que utiliza sequência de unidades ideacionais. Nesta última linha, que é mais complexa e abrangente, a arquitetura do robô deveria suportar um grande sincronismo entre as detecções visual, de movimentos e de voz, o que é condizente com uma proposta multimodal de arquitetura.

Ainda assim, é possível definir algumas estruturas hierárquicas importantes na Análise da Conversa:

- Interação: encontro, conversação, etc. Algumas vezes é difícil delimitar o início e o fim de uma interação, e como elas estão ligadas entre si. No robô, que se restringe ao diálogo, ela pode ser delimitada pela chegada e partida dos interlocutores.
- Sequência: bloco de trocas conversacionais ligadas por um forte grau de coerência semântica ou pragmática, isto é, que trata de um mesmo assunto ou tarefa. Em uma interação, o início é marcado por uma sequência de abertura, com cumprimentos, e o fim por uma sequência de encerramento, com despedidas.
- Troca: a menor unidade dialogal. Os pares adjacentes são tipos de trocas típicas. Entretanto, as trocas podem não ser verbais: um aperto de mão silencioso pode substituir a saudação inicial entre humanos. De fato, as trocas que o robô pode fazer são limitadas por suas capacidades, como já discutidas previamente.
- Intervenção: bastante óbvia e já bastante discutida através do texto.

A menor unidade conversacional a ser utilizada no robô e no gerenciador de diálogo é uma *assertion* (afirmação, sentença) que pode ser equiparada a uma unidade ideacional e pode substituir tanto uma fala quanto um gesto ou movimento de cabeça. A diferença é que ela é construída de forma estruturada, para permitir a utilização de ferramentas computacionais tradicionais de *reasoning*, baseadas em lógica descritiva ou de primeira ordem.

E. Common Ground, Senso Comum

Se utilizarmos o princípio de que o robô busca eficiência, naturalidade e sucesso na comunicação, a relação entre o interlocutor e o robô restringem fortemente as intervenções que podem ser feitas. Embora exista o senso comum, isto é, um grupo de informações que o robô assume que toda pessoa conheça e que não possui qualquer problema de confidencialidade, boa parte das informações são restritas a um interlocutor ou ao próprio espaço de memória do robô.

Além da própria questão de confidencialidade, há também a questão do *common ground* [2] [13], um conjunto de informações que se assume que o interlocutor sabe ou que ele sabe que o robô sabe, e que são básicas para o bom relacionamento nas coisas do dia-a-dia. Para o robô, isto reflete do fato de que, mesmo se estiver conversando sobre um assunto já trabalhado com o interlocutor em uma interação anterior, perguntas e informações já dadas não são feitas novamente por padrão. Até o idioma utilizado faz parte do

common ground, incluindo expressões ou palavras inventadas durante as interações com um interlocutor específico.

O modelo do GD deve refletir essas diferenças, como já proposto em relação à memória no nosso trabalho anterior [1].

F. Tomada de Turnos e Projetabilidade

O sistema de tomada de turnos introduzido em [8] tornou-se o modelo padrão para análises de conversação. Embora o modelo original de Sacks, Schegloff e Jefferson considere apenas o canal de voz (foi baseado em gravações de conversas telefônicas), ele tem sido discutido intensivamente e sua validade tem se mantido mesmo quando se considera a multimodalidade da comunicação, ao se fazer as adaptações adequadas [4]. Em um breve resumo, tal sistema relata que, em uma conversa eficiente, as pessoas falam uma de cada vez, há poucas sobreposições de fala e pouca latência entre os turnos. As sobreposições, embora em geral não desejadas, são naturais e tendem a acontecer nos assim chamados pontos de relevância para transição, ou TRP (*transitional-relevance place*). Um TRP é definido a partir da análise de diversos campos semióticos, como gestos, prosódia, léxico, etc. Associada a esta questão aparece o conceito de projetabilidade, que está relacionado à capacidade tanto do falante quanto do receptor de projetar uma intervenção até seu final a partir de certo ponto durante a própria intervenção. Desde este momento, aquele que projeta já é capaz de planejar uma reação, mesmo que seja essa fazer nada, e reflexos deste planejamento já podem ser observados antes mesmo de sua execução falada, como através dos gestos [18].

Estes fatos tem uma relação mais forte com a arquitetura como um todo do robô. Mesmo que, por limitações técnicas, se utilize um modelo que tente limitar a fala a apenas um ator por vez, sem sobreposições, deve-se considerar a possibilidade, dentro do GD, de que o interlocutor tenha voltado a falar entre a recepção de um estímulo de fala que tenha sido considerado como final e o início da execução de uma intervenção pelo robô. Da mesma forma, pode-se considerar um evento especial em que o interlocutor começa a falar enquanto o robô está falando. A definição atual é, entretanto, de que o não se deve interromper a fala interlocutor, que tem a preferência. Não está certo, ainda, se gestos deveriam interromper uma intervenção do robô.

G. Outros assuntos

Além dos assuntos listados previamente, ainda há outros assuntos relevantes para os processos internos do GD como emoção e aprendizado. O primeiro assunto divide-se em duas partes, relevância das emoções do interlocutor e das próprias, simuladas. Por hora e por simplicidade, dado a complexidade do assunto, podemos considerar ambos como campos semióticos adicionais que servem como critério adicional para seleção das regras de um modelo interno, como parte de um *plug-in* que será adicionado futuramente.

O segundo assunto refere-se ao aprendizado de próprias técnicas de comunicação. Por exemplo, como alguém poderia ensinar ao robô uma técnica de entrevista de emprego através da própria conversa, o que obrigatoriamente envolveria relacionar objetivos internos com intervenções e expectativas?

Tais assuntos, entre outros, serão deixado de lado no projeto aqui relatado por conveniência.

IV. Discussão das Entradas, Saídas e Modelo Interno

A primeira coisa que o GD deve fazer ao receber um estímulo ao qual deve reagir é sincronizar todas as entradas necessárias para a tomada de decisão. Isto serve para dois propósitos: (i) tornar todo o processamento posterior coerente, sobre o mesmo conjunto de dados e (ii) verificar se já não existe uma intervenção em andamento que seja conflitante com a entrada, como no exemplo já citado para evitar sobreposição de turnos de fala.

Em seguida, ele deve avaliar a entrada de acordo com seus planos e objetivos. Se há indicadores de que a entrada não atende as expectativas do plano, pode ser necessário adaptá-lo ou escolher um novo plano, dependendo da relação com seus objetivos e com as normas sociais. Este ponto em especial, e mesmo toda a discussão anterior relativa às outras áreas do conhecimento, sugere fortemente a utilização de modelos que se adaptam automaticamente, o que é exatamente o foco da adaptatividade. A intervenção a ser executada pelo robô é decorrente do plano definido pelo GD.

A. Entradas

Uma das maneiras de classificar os dados de entradas decorre da comparação da duração e do término das mesmas em relação ao momento em que elas são recebidas pelo GD. Existem três categorias nesta classificação:

- *Eventos que terminam antes do recebimento*. Este é o caso da maioria dos eventos comunicativos, que são, de certa forma, equivalentes aos turnos dos sistemas conversacionais de tomada de turnos;
- *Eventos em andamento durante o recebimento*. O recebimento de tais eventos pelo GD é atípico. Normalmente caracteriza urgência, como nos casos em que o robô não pode aguardar o término de um evento para agir e evitar uma situação indesejada. Por exemplo, se enquanto o robô está falando o interlocutor iniciar seu turno, o primeiro deve interromper para que o outro possa utilizar o canal de comunicação mais confortavelmente. O GD deve continuar recebendo atualizações deste evento até que ele termine, permitindo reparar ações erroneamente cognadas;
- *Objetos passivos*. Normalmente eles não geram eventos que requerem respostas, mas estão disponíveis de forma contínua e podem ser consultados sob demanda em outros módulos, como, por exemplo, emoções do interlocutor, motivações internas e mesmo a utilização do canal de voz – antes de começar a falar, deve-se verificar se o interlocutor já começou a falar para o robô ou mesmo com outra pessoa, evitando desta forma turnos simultâneos.

O formato de entrada de eventos é baseado no ISO 24617-2 [15]. Uma das características mais importantes deste padrão é que ele já foi feito tendo como requisito o suporte a multimodalidade. Cada intervenção precisa já estar dividida em *assertions* e com o conteúdo semântico já estruturado e

ligado à memória do robô. Também precisam estar anotados os atos dialogais, que, conforme definido na norma ISO, é uma "operação de atualização no estado informativo, em particular as atualizações que o emissor intenciona ocorrer no estado informativo do diálogo de um participante que entenda o seu comportamento comunicativo". Cada ato é composto de uma função comunicativa, uma dimensão e um conjunto de dependências funcionais (e de *feedback*) com intervenções prévias. De forma simplificada, podemos dizer que a dimensão do ato dialogal especifica qual o tipo de informação que ele atualiza [19]; podemos citar a dimensão das funções de gerenciamento das obrigações sociais e a de gerenciamento dos turnos. Uma única *assertion* pode ter mais de um ato dialogal.

Além disso, também é necessário fazer uma avaliação psicológica do evento recebido. Ao propor uma arquitetura para lidar com emoção em agentes computacionais, chamada de *component process model* (ou CPM) [20], Scherer descreve um módulo de avaliação de objetos e eventos que interpreta e avalia as entradas sob quatro perspectivas: (i) relevância, (ii) implicações e consequências, (iii) capacidade de enfrentamento (ou *coping*) do agente e (iv) importância para autoimagem e normas sociais. É possível que existam diferenças entre as verificações e os objetivos utilizados pelo Scherer e os de um robô para diálogos (mesmo que este robô tenha emoções, primordialmente o objetivo dele é servir ao diálogo), mas um modelo similar de cognição também pode ser aplicado ao robô, em que vários módulos interpretam e avaliam aspectos distintos das entradas de forma paralela, as relacionando às memórias episódica e semântica e definindo indicadores de relevância, urgência e intenção, entre outros. Desta forma, podemos enxergar estes últimos como entradas para o GD que acompanham o estímulo original.

B. Saídas

Existem os tipos de saída direto e indireto.

Das saídas diretas, as principais são as intervenções, visíveis ao mundo externo e dos quais podemos citar a fala ou o direcionamento do olhar para algo específico. A outra são as saídas internas: os planos, os objetivos e as expectativas. As intervenções estão associadas com objetivos, como aprofundar o conhecimento sobre um objeto ou lembrar o interlocutor de um assunto pendente. As expectativas, ou situações esperadas, são utilizadas para averiguar se as intervenções tiveram o efeito desejado. Os planos guardam quais ações se pretendia realizar mediante as expectativas para alcançar determinado objetivo. Todos os demais módulos precisam sincronizar seus funcionamentos com tais saídas para manter o sistema consistente entre si.

Indiretamente, ambas as memórias semântica e episódica são alteradas. As saídas diretas precisam ser memorizadas de forma a manter coerência através do tempo. Além disso, a própria forma como um ser humano avalia inconscientemente os eventos de entrada é alterada pelas decisões conscientemente tomadas, através de mecanismos como a atenção, a classificação e a qualificação. Por exemplo, se os processos internos para a tomada de decisão da ação envolveram classificar uma pessoa como má intencionada, uma próxima fala que de outra forma poderia ter sido interpretada como elogio pode ser agora interpretada como uma ironia – tal decisão interna precisaria ser memorizada e acessada por outros módulos posteriormente. Além disso, há, novamente, o problema da coerência, pois não é nem natural nem prático raciocinar sobre a classificação de uma pessoa a cada ação da mesma. Se algo sair errado em relações às expectativas, é sempre possível reclassificar as pessoas, pois, a princípio, todas as propriedades contextuais que levaram à decisão anterior ainda estão na memória e serão consideradas em um novo raciocínio, mesmo que seus valores tenham sido alterados.

C. Modelo Interno

O GD precisa, então, resolver problemas de planejamento e de tomada de decisão utilizando-se modelos computacionais apropriados. São eles:

- Representação do plano, relacionando as intervenções do robô às expectativas em relação ao interlocutor. As expectativas possuem probabilidades de ocorrência e uma série de indicadores sobre o quanto desejável é que ela aconteça;
- Escolha dos objetivos. Eles precisam ser coerentes com a motivação do robô, como, por exemplo, com a sua função social. Existem objetivos globais da interação, locais em sequencias e pontuais nas intervenções, e pode existir uma relação entre eles. Para um modelo definitivo, é necessária uma listagem de tais objetivos a partir de exemplos e a uma análise da dependência entre eles e com outras estruturas para determinar como o processo de escolha deve acontecer;
- Geração dos planos. Uma vez definido os objetivos, planos devem ser gerados. Sempre que acontece algo que não está nos planos ou um objetivo é atingido, é preciso analisar o impacto no mesmo, se ele precisa ser adaptado ou refeito;
- Escolha do plano a ser seguido. Sempre que um ou mais planos são gerados para atender aos objetivos, é necessária uma maneira de compará-los, conforme critérios como probabilidade de se atingir os objetivos, número de intervenções até atingi-los, alterações no relacionamento com o interlocutor e outras funções de custo.

V. AVALIAÇÃO DOS DISPOSITIVOS ADAPTATIVOS

O termo dispositivo, a que se refere o texto deste artigo, corresponde a alguma abstração formal, na qual o comportamento do dispositivo é regido por um conjunto finito explícito de regras que especificam, para cada situação em que se encontre o dispositivo, sua nova situação e também as correspondentes alterações esperadas no conjunto de regras que o definem [6]. Um dispositivo adaptativo é um em que se podem separar claramente dois componentes: um dispositivo subjacente, tipicamente não adaptativo, e um mecanismo adaptativo, responsável pela incorporação da adaptatividade.

Os formalismos subjacentes dos dispositivos adaptativos podem ser classificados em variadas categorias, conforme sua

forma de operação, algumas mais relevantes que outras para o gerenciamento de diálogos. Entre outras, têm-se as seguintes:

- Dispositivos de reconhecimento, da classe dos autômatos, baseados na sucessão de mudanças de estados;
- Dispositivos de geração, da classe das gramáticas;
- Dispositivos para a representação de sistemas assíncronos, tais como *statecharts*;
- Dispositivos estocásticos, como as cadeias de Markov, capazes de representar fenômenos aleatórios;
- Dispositivos de auxílio à tomada de decisão, como tabelas e árvores de decisão;
- Dispositivos de processamento, como as linguagens de programação adaptativa.

Exceto pelo último grupo, cujo domínio de problemas relacionados não é relevante para o GD, os demais grupos tem ao menos um representante que deve ser considerado. A seguir faremos uma análise de alguns desses dispositivos.

A. Autômato adaptativo

Os autômatos adaptativos [21] possuem como dispositivo subjacente um autômato finito de estados. Ao lado da sua variação transdutora, tem sido utilizado em tarefas como, reconhecimento de padrões, processamento de texto em linguagem natural e síntese de voz.

Dependendo da relação entre os objetivos, um autômato poderia ser o dispositivo de escolha. Suponha uma conversa com um desconhecido e os seguintes objetivos locais de sequencia: saudar, apresentar-se e obter conhecimento sobre o interlocutor, descobrir um assunto a ser explorado, explorar assunto descoberto, perguntar sobre alguma ajuda e encerrar a conversa. Os objetivos poderiam ser estados e as transições seriam causadas por eventos como objetivo alcançado, objetivo impossibilitado de ser alcançado ou desvio do objetivo atual pelo interlocutor. Em uma conversa, assuntos referenciados pelo interlocutor podem ser guardados para serem trabalhados mais detalhadamente assim que o assunto atual estiver resolvido, através da execução de uma ação adaptativa.

Entretanto, seriam necessárias algumas alterações no modelo, pois em cada um desses estados da sequencia, há objetivos menores associados que precisam ser trabalhados no plano, e, portanto, precisam ser disponibilizados para consulta e utilizados para se definir os eventos de transição.

B. Sistemas de Markov adaptativos

Os sistemas de Markov adaptativos [22] [23], ou SMA, são cadeias de Markov com transições especiais associadas a ações adaptativas que alteram as probabilidades de seleção das transições e criam e removem estados do dispositivo. Além disso, o sistema permite a existência simultânea de várias máquinas, que funcionam paralelamente umas às outras. Ações adaptativas permitem uma relação hierárquica entre as máquinas, de forma que uma possa alterar o conjunto de regras da outra.

Essa natureza aleatória de execução permite dar alguma imprevisibilidade a cada execução de um SMA. Essa imprevisibilidade é importante para que o robô seja mais natural e não reaja sempre da mesma maneira aos estímulos de entrada em contextos muito similares. Entretanto, esse dispositivo não pode ser utilizado para a representação do plano, uma vez que o acionamento das transições não é probabilístico: é definido pela ocorrência de um evento externo que pode inclusive não estar contido entre as expectativas definidas pelo plano; a probabilidade associada à expectativa é apenas uma medida de quanto o robô acredita na ocorrência de determinado evento.

Esse modelo poderia ser utilizado na criação dos planos, gerando variações inesperadas e que possam surpreender o interlocutor. Mesmo assim seriam necessárias alterações no dispositivo original, que é puramente aleatório e não é sensível a dados de entrada, neste caso, os objetivos a atender, as intervenções possíveis para atender esses objetivos, as expectativas associadas com cada um e informações em memória, entre outros.

C. Tabela de decisão adaptativa

As tabelas de decisão são naturalmente apropriadas para situações em que há independência de contexto, pois todas as situações são possíveis a cada instante. As tabelas de decisão adaptativas multi-critério [24] adicionam alguma dependência de contexto e parecem ser uma opção para o suporte da escolha e geração dos planos frente aos eventos de entrada. Seja um modelo em que todas as situações estão mapeadas a uma das seguintes situações: manter o plano atual, adaptar o plano atual ou utilizar um gerador de planos distinto para criar um novo. Por exemplo, se o robô verifica entre suas pendências que precisa dar uma notícia ruim (objetivo) para alguém que o robô conhece e é bastante sensível (situação), opta-se por um gerador X de planos para dar notícias ruins de forma suave, embora o gerador Y de planos para dar notícias de forma geral também possa ser utilizado. Entretanto, se na mesma situação o interlocutor já iniciou outro assunto logo após os cumprimentos, existem dois objetivos paralelos (conversar sobre assunto motivado pelo interlocutor e dar uma notícia ruim) e um terceiro gerador de planos Z pode ser utilizado.

A parte adaptativa é importante no sentido do aprendizado, de duas formas. A primeira é que a seleção de algumas regras deve diminuir ou aumentar a chance de selecionar outra regra posteriormente. Por exemplo, cada vez que um determinado gerador é utilizado dentro de uma mesma conversa ou mesmo interlocutor, reduz-se a avaliação do mesmo frente ao critério novidade, favorecendo assim a diversidade. Ou então se escolhemos um gerador de plano para tentar melhorar o humor de uma pessoa, aumenta-se o peso do critério “melhoria de humor” na seleção dos geradores, favorecendo assim a coerência.

A segunda forma de aprendizado é referente ao reforço. Se ao utilizar um plano para alcançar determinado objetivo o custo em algum critério foi muito alto, a avaliação do gerador de tal plano é reduzida em tal critério; um bom desempenho deveria surtir o efeito oposto, interferindo diretamente na chance dele ser selecionado em uma próxima vez.

As propostas aqui descritas não podem ser resolvidas diretamente pelo dispositivo proposto por Tchemra [24], chamada de *Tabela de Decisão Adaptativa Estendida*, ou *TDAE*. Primeiro, porque a escolha por um gerador não seria apenas baseada em uma função utilidade, mas também seria probabilística, causando a desejada imprevisibilidade. Segundo porque há tipos distintos de alternativas que não se misturam, sugerindo uma divisão adicional do modelo: reavaliações de planos após o abandono dos mesmos levam apenas a ações adaptativas que alterem o peso de algum critério ou avaliação de uma regra segundo um critério; eventos de entrada do GD levam também à manutenção, adaptação ou geração de um plano. Isto sugere duas tabelas distintas, com uma relação adaptativa hierárquica entre elas, se isso for possível.

D. *Outros dispositivos, adaptativos e não adaptativos*

Outros dispositivos adaptativos ainda não avaliados devem ser considerados na continuação deste trabalho. As árvores de decisão adaptativas, por exemplo, podem ser avaliadas como alternativa a tomada de decisão, e as gramáticas adaptativas consideradas como alternativa para a geração de planos.

Entretanto, existem alguns problemas relacionados aos processos internos do GD que os dispositivos adaptativos analisados não conseguem resolver e talvez seja preciso considerar modelos não adaptativos como solução do problema. A primeira é a representação do plano. Como ele mistura expectativas e previsão de tomadas de decisão, talvez até com uma definição de política de escolhas, uma opção a ser analisada é o *Markov Decion Process*. Um segundo problema é a modelagem das intenções e objetivos do interlocutor, que até agora não foi discutido. Tais tipos de problema, em que não é possível visualizar a intenção real, mas apenas fazer observações sobre elas, sugerem modelos como *Hidden Markov Model*. Dependendo da necessidade, podem-se propor variações a estes dispositivos, inclusive adaptativas.

VI. CONCLUSÃO

A tecnologia adaptativa possui boas opções de modelos computacionais para utilização em um gerenciador de diálogos para robôs adaptativos. Para chegar a uma conclusão definitiva, é necessário, além de analisar outras opções de modelo computacional:

- detalhar um ou mais modelos de implementação;
- escolher ao menos um deles e construir um gerenciador de diálogos, mesmo que com um número bem limitado de dados em memória, intervenções, objetivos e planos;
- definir cenários de teste para levantamento de resultados;
- comparar os resultados obtidos frente aos requisitos aqui listados e comparativamente a um ser humano.

Agradecemos ao professor João José Neto, da Escola Politécnica da USP pela colaboração tanto com muitas ideias quanto com a pesquisa de adaptatividade; e aos professores Leland McCleary, da Faculdade de Filosofia, Letras e Ciências Humanas da USP, e Aílton A. Silva, do Instituto de Psicologia da USP, pelas excelentes disciplinas de pós-graduação relacionadas à conversação ministradas no ano de 2012 e pela indicação de leitura de diversos textos importantes, sem os quais este artigo não teria se tornado realidade.

REFERÊNCIAS

[1] D. A. Alfenas and M. R. Pereira-Barretto, “Adaptatividade em Robôs Sociáveis : uma Proposta de um Gerenciador de Diálogos,” in *MEMÓRIAS DO WTA 2012 SEXTO WORKSHOP DE TECNOLOGIA ADAPTATIVA*, 2012.

[2] H. Garfinkel, *Studies in Ethnomethodology*. Englewood Cliffs, NJ: Prentice-Hall, 1967.

[3] R. Almeida, “ANÁLISE DA CONVERSAÇÃO COMO METODOLOGIA,” in *I Encontro dos Programas de Pós-graduação em Comunicação de Minas Gerais*, 2008.

[4] T. Leite, “A segmentação da língua de sinais brasileira (libras): Um estudo lingüístico descritivo a partir da conversação espontânea entre surdos,” USP, 2008.

[5] M. L. Knapp, *Comunicação não-verbal na interação humana*. Judith A. Hall, 2006.

[6] J. J. Neto, “Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa,” *IEEE Latin America Transactions*, vol. 5, no. 7, 2007.

[7] C. Kerbrat-Orecchioni, *Análise da Conversação. Princípios e Métodos.* São Paulo, Brasil: Parábola Editorial, 2006.

[8] H. Sacks, E. A. Schegloff, and G. Jefferson, “A Simplest Systematics for the Organization of Turn-Taking for Conversation,” *Language*, vol. 50, no. 4, pp. 696–735, 1974.

[9] J. Reeve, *Understanding Motivation and Emotion*, 5th ed. John Wiley & Sons, 2009.

[10] E. A. Schegloff, “Body Torque *,” *Social Research*, vol. 65, no. 3, pp. 535–596, 1991.

[11] C. Goodwin, “Action and embodiment within situated human interaction,” *Journal of Pragmatics*, vol. 32, no. 10, pp. 1489–1522, Sep. 2000.

[12] L. Mccleary and T. D. A. Leite, “Turn-taking in Brazilian Sign Language : Evidence from overlap,” *Journal of Interactional Research in Communication Disorders*, 2012.

[13] Tomasello, “Human Cooperative Comunication,” in *Origins of Human Communicatio*, Cambridge, MA: MIT Press, 2008, pp. 57–108.

[14] E. A. Schegloff and H. Sacks, “Opening up Closings *,” *Semiotica*, vol. 8, pp. 289–327, Aug. 1973.

[15] H. Bunt, J. Alexandersson, J. Choe, A. C. Fang, K. Hasida, V. Petukhova, A. Popescu-belis, and D. Traum, “ISO 24617-2 : A semantically-based standard for dialogue annotation,” in *Proceedings of LREC 2012*, 2012, pp. 430–437.

[16] L. A. Suchman, *Human-machine reconfigurations: plan and situated actions*, 2nd ed. Cambridge University Press, 2007.

[17] G. Calbris, *Elements of meaning in gesture*. John Benjamins Publishing Company, 2011.

[18] E. A. Schegloff, "On some gestures ' relation to talk," in *Structures of Social Action: Studies in Conversation Analysis*, no. June, J. M. Atkinson and J. Heritage, Eds. Cambridge: Cambridge University Press, 1984, pp. 266–296.

[19] H. Bunt, "Dimensions in Dialogue Act Annotation," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, 2006, pp. 919–924.

[20] K. R. Sherer, T. Bänzinger, and E. B. Roesch, Eds., *Blueprint for Affective Computing: A sourcebook*. Oxford University Press, 2010.

[21] J. J. Neto, "CONTRIBUIÇÕES À METODOLOGIA DE CONSTRUÇÃO DE COMPILADORES," USP, São Paulo, Brasil, 1993.

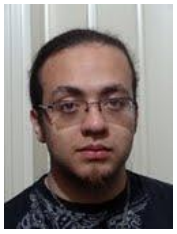
[22] B. A. Basseto, "Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos," USP, São Paulo, Brasil, 2000.

[23] D. A. Alfenas, D. P. Shibata, J. J. Neto, and M. R. Pereira-Barretto, "Sistemas de Markov Adaptativos : Formulação e Plataforma de Desenvolvimento," in *MEMÓRIAS DO WTA 2012 SEXTO WORKSHOP DE TECNOLOGIA ADAPTATIVA*, 2012.

[24] A. H. U. M. Tchemra, "TABELA DE DECISÃO ADAPTATIVA NA TOMADA DE DECISÃO MULTICRITÉRIO," USP, 2009.

Daniel Assis Alfenas formou-se em Engenharia da Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP) em 2004. Trabalhou, nos últimos dez anos, nas diversas áreas do desenvolvimento de sistemas, desde a definição da demanda até a implantação do sistema. Recentemente tem trabalhado como coordenador técnico no desenvolvimento de sistemas complexos que envolvem simulação, otimização e interfaces ricas. Atualmente, é mestrando no Laboratório de Técnicas Adaptativas da EPUSP e a área de pesquisa é a aplicação da tecnologia adaptativa no diálogo em linguagem natural entre usuários e sistemas.

Marcos Ribeiro Pereira-Barretto é graduado em Engenharia Elétrica (1983), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Mecânica (1993) pela Escola Politécnica da Universidade de São Paulo. É professor da Escola Politécnica da USP desde 1986. Atua nas seguintes áreas de pesquisa: robôs sociáveis, computação afetiva e arquitetura de sistemas para aplicações críticas como Automação Industrial.



Ricardo dos Santos Paixão formou-se em Engenharia da Computação pelo Centro Universitário Faculdade Instituto de Ensino para Osasco (UniFIEO) em 2010. Desde então tem trabalhado com desenvolvimento de sistemas de engenharia e auxílio a tomada de decisão, atuando principalmente na análise e na implementação. Atualmente é mestrando em Engenharia de Controle e Automação Mecânica e tem como linha de pesquisa a modelagem de memória episódica para robótica sociável.

Use of Extended Adaptive Decision Tables on Reconfigurable Operating Systems

S. M. Martin, G. E. De Luca, and N. B. Casas

***Abstract*— Throughout the last decades, many reconfigurable operating systems have been developed in order to let users and programmers make decisions upon the configuration of some of the innermost kernel's aspects. These decisions, however, require an advanced level of skill in order to obtain some performance advantage. Furthermore, they can be detrimental to the system's performance if they are not careful taken. Letting the kernel itself do the decision-making upon the implementation of its aspects is a safe and powerful way to manage re-configurability that does not require interaction with the user nor the need of advanced skills to take advantage of. In this article, we propose the usage of Extended Adaptive Decision Tables as a mean for the kernel to achieve the capability of taking intelligent decisions based solely on the users' process creation behavior.**

***Keywords*— Decision Tables, Operating Systems, Adaptive Device**

I. Introduction

The operating systems have been developed and used for decades to abstract the complexity of the underlying hardware for both end-users, and application programmers. Through them, the computer and its components can be seen as administrable resources that users and their applications can request and use. The list of resources that are administrated by the system kernel includes, but is not limited to: processor usage, memory distribution, permission for input/output operations on actual or virtual devices, among others.

The first operating systems, such as MS-DOS, only allowed the execution of a single process at a time. For that reason, its kernel's only functions were limited to booting, providing a command line, and some simple services. The executing process would take the complete control over the processor, the memory, and the I/O devices, and the system kernel had no administrative responsibilities whatsoever.

With the development of the more advanced UNIX-based operating systems that allowed the execution of more than one process at a time, and the logging of multiple users concurrently, new challenges arose. The issue of which processes or users should have more processor usage time, or greater free memory chunks available, was addressed differently among the different kernel developers. Now, the diversity of resource administration policies among different operating systems provides users and companies a range of problem-specific products to serve their own business necessities.

S. M. Martin, G. E. De Luca, N. B. Casas, Departamento de Ingeniería e Investigaciones Tecnológicas, Universidad Nacional de La Matanza Buenos Aires, Argentina. {smartin, gdeluca, ncasas}@unlam.edu.ar

Most of the now available public operating systems are non re-configurable. This means that their resource administration policies are built-in and cannot be changed by the users.

We will use the nomenclature used in our previous work [1] in which the processor usage, the memory distribution or any other resource management are said to be *aspects* of the operating system. Each one of these aspects can be administrated using a policy or *mode.* An operating system is said to be *reconfigurable* when one or more of its aspects can change their mode during runtime without rebooting.

Although almost all of the commercial and home requirements for an operating system can be satisfied with a non-reconfigurable conventional kernel, there is a potential for the adaptable capabilities of a reconfigurable kernel.

Some examples of reconfigurable operating systems, such as Kea [2] and Synthetix [3], have shown better results than their commercial counterparts on tests driven under diverse and changing execution conditions and process behaviors. Others, such as SPIN [4], provided an interface for extensibility where the programmer himself could develop new modes for the kernel's aspects.

All of the reconfigurable operating systems available – ready to use, source code, or just in academic bibliography – depend upon the user/programmer to decide both which changes make to the kernel configuration and when to make them. In the majority of the cases, this is achieved by providing an object-model based kernel-process interface [5] that presents the functionality for a certain service, and allows the programmer to define which object instance gets to execute them.

In spite of the great flexibility and adaptive potential that can be obtained with the interface approach, its limitations can be often enough to prevent users, programmers, or even kernel designers, from using it.

At first, it takes an application programmer to know at least something about the kernel's intricacies in order to obtain any advantage. Some programmers may even have to investigate about them before knowing where and what to change.

On the other hand, legacy, standard, or reused programs wouldn't have the opportunity to harness its benefits. They might have to be re-engineered before being able to use the interfaces properly. End-users with programs of their own wouldn't stand a chance on harvesting the potential of the underlying re-configurability.

In this article, we present a different viewpoint for kernel reconfiguration. Since letting the users have the decision-taking responsibility offers great potential for performance improvement, but lacks the portability and demands high skills, we thought that the kernel itself may be able to take

charge of the issue. The three main arguments supporting this idea are:

First, the kernel developers already have all the skills and knowledge about its aspects and modes' inner complexities. They can design better and faster mechanisms for mode exchanging so no skills or specific knowledge should be demanded to their users and programmers whatsoever.

Second, all the applications to be run can be kernel-agnostic and still harness the benefits of reconfiguration.

Lastly, the kernel can take decisions upon several more indicators than a programmer could. Some of them may be inaccessible for a process at runtime, and some other may be too complex or too kernel-specific for a programmer to take into account.

We will use Extended Adaptive Decision Tables – from now on, abbreviated as EADTs – as the device that will allow us, as designers, give the kernel the mechanism for decision making based on the behavior of its users.

All the analysis and examples presented in this article were conducted on SODIUM[1], a project for an academic reconfigurable operating system [6] [7] and, more specifically, its reconfigurable process scheduler aspect.

The rest of this article is organized as follows: Section 2 introduces the reconfigurable kernel design methodology used to analyze each aspect. Section 3 introduces the SODIUM's reconfigurable process administration aspect in more detail. In section 4, that aspect will be analyzed in order to obtain the decision taking criteria necessary for the construction of the initial decision table that will be presented in Section 5. In section 6 explain what adaptive elements were used in order to obtain the adaptive decision table. In section 7, we analyze the extended mechanism that allows for multi-criteria decision taking in order to generate the final extended adaptive decision table. Finally, section 8 discusses the future work and tests to be conducted, and the conclusions of this investigation.

II. Reconfigurable design methodology

The SODIUM project was started back in 2005 with the aim to give the operating system class' alumni the opportunity, not only to grasp the theory concepts, but also to let them involucrate actively in the development of a kernel. At the end of each year, all the practices were tested, and the best ones were integrated into the kernel for the next year's alumni to use.

One of the first dilemmas of this methodology emerged when, predictably, many different modes were programmed for the same particular aspect. For example, having a basic fixed partition memory administrator as a base, a practice asked to develop another one based on paging, forced the student to replace the existing one. This forcibly implicated a loss in didactic value since, even though the paging approach was, in overall, better, the former mode was still useful for teaching purposes.

Given this situation, the professorship agreed on implementing a mechanism by which SODIUM could handle multiple modes for any aspect, and that those modes could be exchanged while still on runtime. The efforts on developing a design pattern for this mechanism ended up in the methodology proposed in [1].

In that methodology, four steps had to be conducted upon each aspect to be designed for re-configurability:

1. The current situation must be analyzed for each possible transition between modes – if any –, the kernel mechanism that should be developed, and the timing level.
2. All the currently available modes should be enumerated along with a graph showing new ones, in order to uncover possible missing transitions.
3. Design the reconfigurable aspect indicating implementation details, timing levels, and element to be verified for possible data loss for each transition.
4. Design, document, and publish the kernel/user interfaces for the reconfiguration mechanism.

Although this methodology is useful to design reconfigurable schemes for existing aspects and modes, it does not contemplate the decision-taking process. In fact, it ends at step 4, where it indicates that a kernel/user interface should be designed. In our proposal, we seek to vary this methodology to let the kernel reconfigure itself. In order to do so, we shall replace the former step with a new one:

4. Establish the criteria and events determining the need for mode changes, and the mechanisms to allow them.

The timing levels are also important because they indicate which conditions within the system determine whether a transition can be executed or not. Four timing levels are defined. Level 1, when the transition can be only be executed by recompiling the kernel; level 2, when the transition can only be executed by rebooting the system; level 3, when the transition can be executed in runtime, but globally for all processes; and level 4, when the transition can be executed in runtime, and in particular for each process.

In the next section, we will use the presented methodology in order to analyze one of the main reconfigurable aspects of SODIUM's kernel.

III. SODIUM's Reconfigurable Process Scheduler

Even though this article presents a general proposal for kernel decision-taking on re-configurability using EADTs, it is necessary and much easier to explain its implementation steps by using an existing reconfigurable aspect as base.

No other reconfigurable aspect has been more refined and researched upon in SODIUM than the process scheduler. It counts with 6 different modes available: Round-Robin (RR), Round-Robin with Priority Queues (RRPR), Round-Robin with Variable Quantum (RRQV), First-Come-First-Serve (FCFS), Shortest-Finishing-Job-First (SJFS), and Best-Time-Of-Service (BTS). Many of these modes specification are

[1] Sistema Operativo del Departamento de Ingeniería de la Universidad Nacional de La Matanza. Web: http://www.so-unlam.com.ar/

standards and can be found in specialized operating system's books. We used a general specification found in [8] as a base for almost all of them.

This aspect has the particularity that all of its 6 modes can transition to all of the others without any limitation. This means that it counts with 30 different transitions to evaluate. Also, the timing level for all transitions is 3. This is because any changes in the process scheduler will forcibly affect all the process at the same time. No changes can be made individually for any process.

Since straightforwardly programming 30 different functions –one for each transition– could result in a bloated and difficult to understand kernel code, we decided to analyze, at depth, which actions were to be shared between the different transitions' procedures. We identified 8 common actions that could be normalized and reutilized between all the procedures. Briefly, these actions are: *setbasealg(alg)*, to set the basic scheduling algorithm; *initPriorities()*, to initialize priorities for RRPR; *initQTime()* to initialize quantum times for RRQV; *setNonPreemptive()* and *setPreemptive()* to set a preemptive or non-preemptive scheduler behavior, respectively; *queueMode(mode)* to set a unique or level based ready process queue; and *quantumMode(mode)*, to set a unique or priority based quantum evaluation. In Table 3.1, different groups of actions are set and identified with an alphabet letter codification.

modes would be so complex that it would go against its main benefits.

		Next Mode					
		RR	PR	QV	FC	SJ	BT
Current Mode	RR	0	a	b	c	d	e
	PR	f	0	g	h	i	j
	QV	k	l	0	m	n	ñ
	FC	o	p	q	0	r	s
	SJ	t	p	q	u	0	s
	BT	v	w	x	y	z	0

Table 3.2 – Mode transitions and their set of actions

Although this transition-actions relation allow the kernel to execute transitions by itself, it is still not enough to let it take decisions. Having this relation between transitions and actions to be performed is the first step in order to generate a conventional decision table –from now on, abbreviated as cTD–. The cTD will hold the first trivial condition-action mechanism that will indicate the kernel when and what transitions execute.

		a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Actions	setbasealg(BTS)					x					x					x					x							
	setbasealg(RR)																							x	x	x	x	x
	initPriorities()	x	x															x	x						x	x		
	initQTime()		x					x											x							x		
	initReorder(CTIME)			x					x					x									x				x	
	initReorder(PTIME)				x					x					x					x								x
	setNonPreemptive()			x	x					x	x			x	x												x	x
	setPreemptive()																x	x	x		x	x						
	queueMode(UN)						x	x	x	x	x																	
	queueMode(PR)	x											x					x							x			
	quantumMode(UN)											x	x	x	x	x												
	quantumMode(PR)		x					x											x							x		
Codification																												

Table 3.1 – Set of actions and their codification for mode transitions on SODUIUM's process scheduler

These groups of actions represent functions that allow the execution of all the analyzed transitions. The map of actions per transition can be seen in Table 3.2. The *'0'* action means that no procedure should be executed.

In Table 3.2, the SODIUM kernel can query which actions to execute – all of them are commutative – in order to change the scheduling mode to another while still on runtime. This is a key step for an autonomously reconfigurable kernel, and must be designed by the systems developers. This table will be used later again in Section 7, when new rules of transition will have to be created to contemplate new executing conditions.

One of the visible limitations of this approach is that it is not extensible. All the transactions and groups of actions must be determined before the programming of the kernel. Any mechanism that would allow a programmer to enable new

IV. Decision-Taking Criteria

In order to let a transition-mode-actions table, as the one obtained before, allow the kernel to have decision-taking capabilities, we need to define two new key elements: conditions, which will indicate which rules –as a generalization of what a transition is, in the context of this article– must execute in a given moment; and events, that indicate when the conditions must be evaluated.

Events by themselves can be also considered as conditions, for if they do not trigger, no rules associated to them will be executed. However, from an operating system's view, the distinction is important. Events are codified as trigger functions set in different parts of the kernel while conditions are a part of the cTD to be developed. In the case of our

process scheduler, event triggers will be set every time a process is created, killed, interrupted, or released. This includes hardware/software interruptions, syscalls, I/O requests and responses, and process intercommunication routines.

Defining conditions as such will be a little more difficult. In order to define in which executing scenarios we will have to evaluate three different edges of the process scheduling: scheduling metrics, application categories, and algorithm-metrics relations.

A. Scheduling metrics

SODIUM's process scheduler counts with a set of 7 metrics to evaluate and report the performance of each scheduling algorithm. Most of them are based on the metrics list present in [8]. Here is a brief enumeration and explanation for each one:

- Processor Usage ($\%_{cpu}$) indicates the ratio between the time the processor spends actually executing processes and the time that it spends idle or in overhead costly procedures.
- Throughput ($\#_f$) indicates the amount of processes finished given a finite differential of time. SODIUM updates this metric every 10 minutes.
- Turnaround Time (t_{cv}) indicates the time that took a process to completely executes, from when it is created until it is terminated. It includes the time spent waiting for enough free memory, to be executed, and I/O operations.
- Waiting Time (t_w) indicates the total waiting time of a process during all its execution from the moment it is created. I/O operations or syscalls are not included in this metric since they represent actual requested operations from the process.
- Response Time ($\overline{t_r}$) indicates the average time, for each process, after which, the first response, such as writing a character on the screen, is produced after a user request.
- Effective Time of Service (t_s) indicates the sum of time that the process has spent in actual usage of the processor.
- Overhead (O_v) indicates the time that the processor spends executing system maintenance or managing routines.

B. Application Categories

Since it is not possible to know completely what actions and services will an application request before it is completely executed, the only way to estimate its future behavior is to profile it into well-known categories. We focused that profiling using a technique presented in [9] based on frequencies of system calls, and maintaining a per-process profiling information structure such as the ones specified for the Solaris operating system in [10].

Executing processes fall, after a short period of profiling, into one of the following application categories defined by us: Interaction Intensive Applications (II), such as games or command-line consoles; Multimedia Applications (M), such as video and audio editing tools; I/O Intensive Applications (ES), such as DVD burners or data transfer programs; Internet Applications (WEB), such as web browsers or network programs; Processing Intensive Applications (P), such as compilers or scientific programs; and System Applications (S), such as services or maintenance programs.

After conducting several tests on sample programs that were successfully profiled, we could establish which metrics are more important for each application category. In the Table 4.1 we show the most important metrics for each category that resulted from the tests.

Categories	Metrics
Interaction Intensive Applications (II)	Response Time Waiting Time
Multimedia Applications (M)	Waiting Time Processor Usage
I/O Intensive Applications (ES)	Throughput Response Time
Internet Applications (WEB)	Effective Time of Service Response Time
Processing Intensive Applications (P)	Turnaround Time Throughput
System Applications (S)	Overhead

Table 4.1 – High priority metrics for each category

For simplicity reasons, all the other metrics that are not considered as high priority will be considered as indifferent for that given category.

C. Algorithm-Metrics Relations

Since we now count with the possibility of profiling a process into a category, keep scheduling metrics up to date, and know which metrics favor each category, we only need to determine which scheduling algorithms (modes) improve those metrics.

We found some of these algorithm-metric relations already documented [8] in the bibliography. However, we conducted additional tests to confirm them, and also figure out those that were lacking. From these tests, we obtained the results shown in the Table 4.2. In this table, the beneficial relations in which the algorithm improves the measures from each metric are

marked with a (+); the neutral relations in which the algorithm doesn't affect the metric are marked with a (0); the negative relations are marked with a (-), and the (x) marks indicate that the algorithm is extremely negative to the metric.

	Metric						
Algorithm	%cpu	$\#_f$	t_{cv}	t_w	t_r	t_s	o_v
RR	+	0	-	0	0	0	0
RRPR	+	0	0	+	0	x	0
RRQV	0	0	0	0	+	x	0
FCFS	-	0	+	x	x	x	+
SJFS	-	+	+	x	x	x	-
BTS	-	0	-	-	0	+	-

Table 4.2 – Relations between metrics and algorithms

We can now combine Tables 4.1 and 4.2 in order to obtain an algorithm-category rating in which the score will indicate how much each algorithm benefits/handicaps each application category. The amount of (+) marks per each high priority metric that the algorithm beneficiates will score 1 positive point for that category; (-) will rest 1 point; (0) will produce no effect; and (x) marks will completely disqualify the algorithm. The results from the combination are presented in the Table 4.3.

	Algorithm					
Category	RR	RRPR	RRQV	FCFS	SJFS	BTS
(II)	0	1	1	X	X	0
(M)	1	2	0	X	X	0
(ES)	0	0	1	X	X	0
(WEB)	0	X	X	X	X	1
(P)	-1	0	0	1	2	-1
(S)	0	0	0	1	-1	-1

Table 4.3 – Rating relation table between algorithms and application categories.

V. Conventional Decision Table

The rating information obtained in the Table 4.3 is enough to decide which scheduling algorithm to use when the processes in execution pertain to the same application category. However, this scenario doesn't cover all the execution possibilities.

It could happen that only one process is in the ready queue. This case may be important for batch processing systems. Using a overhead costly algorithm in these cases is not convenient. Therefore, we can establish that in case of mono-processing (*mono*), the algorithm used will be FCFS.

Also, a more common scenario is that when different processes of different categories try to execute concurrently. In this case, using the rating from Table 4.3 can be misleading, because we are using simple heuristic and empiric information on complex cases. In fact, there are 720 different combinational scenarios of mixed categories. This complexity cannot be addressed a priori, by analyzing each case in particular. This case of multi-category (multi) will be resolved using an adaptive decision table –from now on, abbreviated as ADT– to be developed in the next section.

Rules			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Conditions	Current Mode		RR	RR	RR	RR	RR	PR	PR	PR	PR	PR	QV	QV	QV	QV	QV	FC	FC	FC	FC	FC	SJ	SJ	SJ	SJ	SJ	BT	BT	BT	BT	BT
	Current Scenario	Mono			x					x					x											x					x	
		II		x					x											x					x					x		
		M	x											x					x					x					x			
		ES		x					x											x					x					x		
		WEB					x					x					x					x					x					
		P				x					x					x					x										x
		S			x					x					x										x					x	
		Multi						x					x					x					x					x				
Actions		func()	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	p	q	u	s	v	w	x	y	z

Table 5.1 – Conventional Decision Table for SODIUM Process Scheduler

By now, we can contemplate the general case of multiple categories with the most generally acceptable of the scheduling algorithms: RR.

The *mono*, *multi*, and the pure categories processes can be interpreted as the conditions that, without combining with each other, determine the whole universe of possible execution scenarios. Also, we know which algorithm to use for each condition, we can establish the transitions –and their actions– to execute in each case by combining Tables 4.3 and 3.2.

We now have all the elements to elaborate the first cDT for the SODIUM kernel to decide which algorithm use at each moment. It will consist in a set of 30 rules combining conditions –current mode and current scenario– and their transition actions. The cDT for the SODIUM process scheduler is shown as the Table 5.1.

VI. Adaptive Decision Table

A. Normalized Base cDT

The condition evaluation in the cDT of the Table 5.1 obeys to that of an inclusive OR. This means that, it takes only one

condition to be true in order to execute a given rule. For example, rule 2 will execute if a *Mono* scenario is detected and also if it detects an *ES* scenario. However, the formal definition of the cDT, that we must use as a base for developing the more complex ADTs in order to contemplate all the possible scenarios from the generalized *Multi*, requires the conditions to be evaluated with an AND logic. For this, it will be necessary to add duplicated rules that contemplate one of the conditions that will be eliminated from the original one. Also, we need to eliminate *mono* scenario corresponding rules because they can be directly programmed into the scheduler; and those corresponding the *multi* scenario, because the ADT will allow us to contemplate all the particular rules of combining categories scenarios.

Another modification to the original cDT is the addition of the not mark (-) indicating that that condition must be false in order to execute that rule. The resulting fixed cDT is shown in the Table 6.1 (some of the rules have been bypassed in the representation for format purposes).

	1	2	3	4	5	6	7	8	…	24	25	26	27	28	29	30
Current Mode	RR	RR	RR	RR	PR	PR	PR	QV	…	PR	QV	FC	SJ	SJ	BT	BT
II	-	x	-	-	x	-	-	-	…	-	-	-	-	-	-	-
M	x	-	-	-	-	-	-	x	…	-	-	-	-	-	-	-
ES	-	-	-	-	-	-	-	-	…	-	-	x	x	-	x	-
WEB	-	-	-	x	-	-	x	-	…	-	-	-	-	-	-	-
P	-	-	x	-	-	x	-	-	…	-	-	-	-	-	-	-
S	-	-	-	-	-	-	-	-	…	x	x	-	-	x	-	x
func()	a	b	d	e	g	i	j	l	…	h	m	q	q	u	x	y

Table 6.1 – Normalized cDT

simultaneously, we will have to add adaptive mechanisms to our static cDT in order to turn it into a ADT. ADTs formal definition [11] [12] requires the specification of a normalized base cDT such as the one in Table 6.1, and also the addition of adaptive functions to perform rule query, elimination, and insertion actions upon it.

In this investigation, we used a simple adaptive mechanism that consists in the usage of two different adaptive functions. One is used to verify the existence of rules contemplating the current execution scenario. The other is used to add a new rule in case that no such rules were detected.

These adaptive functions called Ad1 and Ad2, execute after and before their calling rules, respectively. Ad1 only sets the value of the *state* variable to D (determined) when an existing rule can manage the current state of conditions. When no rule can handle the current conditions, a new rule (31) is in charge to set the *state* variable to ND (non-determined). This variable is set back by the execution of Ad1, in case that a rule was found. On the next step, if the state is equal to ND, another new rule (32) executes the Ad2 adaptive function.

The Ad2 adaptive function is in charge to add 6 new rules. These new rules will transition to the current mode from any other mode given the current conditions. The result for this is that, whenever in the future, when the same conditions repeat, they will transition to the mode in which those conditions were found initially. The rationale behind this is that, when a new set of conditions is found, is probably because only one new different category process was created, where there is a whole group of processes of an existing category already running. By doing this, we try to maintain the benefits of the current mode for the existing processes.

		Adaptive Functions Declarations									Rules												
		Ad1		Ad2							0	1	2	3	…	26	27	28	29	30	31	32	33
		H	?	H	+	+	+	+	+	+	S	R	R	R	R	R	R	R	R	R	R	R	E
Conditions	Current Mode				RR	PR	QV	FC	SJ	BT		RR	RR	RR	…	FC	SJ	SJ	BT	BT			
	State														…							N	
	II				C_1	C_1	C_1	C_1	C_1	C_1		-	x	-	…	-	-	-	-	-			
	M				C_2	C_2	C_2	C_2	C_2	C_2		x	-	-	…	-	-	-	-	-			
	ES				C_3	C_3	C_3	C_3	C_3	C_3		-	-	-	…	x	x	-	x	-			
	WEB				C_4	C_4	C_4	C_4	C_4	C_4		-	-	-	…	-	-	-	-	-			
	P				C_5	C_5	C_5	C_5	C_5	C_5		-	-	x	…	-	-	-	-	-			
	S				C_6	C_6	C_6	C_6	C_6	C_6		-	-	-	…	-	-	x	-	x			
Actions	func()				$\$_m$	$\$_m$	$\$_m$	$\$_m$	$\$_m$	$\$_m$	0	a	b	d	…	q	q	u	x	y			
	State=		D												…						N		
Adaptive Functions	Ad1	A			x	x	x	x	x	x		x	x	x	x	x	x	x	x	x		x	
	Ad2			B																		x	

Table 6.2 – Adaptive Decision Table for SODIUM Process Scheduler

B. Adaptive mechanism

In order to contemplate new rules for complex scenario conditions, such as II and M category processes running

The implementation of this mechanism onto the subjacent cDT in order to generate the first SODIUM process scheduler ADT is shown as the Table 6.2.

The action to be performed by every new rule is set by the $\$_m$ function that takes the rule's own starting mode and the current mode –as destination mode– as parameters to obtain the specific set of actions for that transition. The result of executing $\$_m$ are exactly those found in the Table 3.2.

An example of its adaptive functions can be illustrated as when a new scenario is detected in which the (M) and the (S) conditions are detected simultaneously. Six new rules are created to handle those conditions in combination with the six possible current modes, and are added to the subjacent cDT. The Table 6.3 shows the effect of their execution, highlighting the new rules in shaded green.

We will recur to the formal definition of the EADT from [13] and [14] in which multiple criteria can be defined in order to determine an alternative.

The original formulation consists on a base ADT such as the one obtained in the Table 6.2 and the addition of auxiliary functions (FM) that execute prior any other action and define values for variables that those actions will use.

In our case, we want to define the destination mode parameter for the function $\$_m$ using a multi-criteria method. The required steps for defining the method consist in three modules:

	Adaptive Functions Declarations									Rules																				
	Ad1		Ad2							0	1	2	3	…	26	27	28	29	30	31	32	33	34	35	36	37	38	39
	H	?	H	+	+	+	+	+	+	S	R	R	R	R	R	R	R	R	R	R	E	R	R	R	R	R	R
Current Mode				RR	PR	QV	FC	SJ	BT		RR	RR	RR	…	FC	SJ	SJ	BT	BT				RR	PR	QV	FC	SJ	BT
State														…							N								
II				C_1	C_1	C_1	C_1	C_1	C_1		-	x	-	…	-	-	-	-	-				-	-	-	-	-	-
M				C_2	C_2	C_2	C_2	C_2	C_2		x	-	-	…	-	-	-	-	-				x	x	x	x	x	x
ES				C_3	C_3	C_3	C_3	C_3	C_3		-	-	-	…	x	x	-	x	-				-	-	-	-	-	-
WEB				C_4	C_4	C_4	C_4	C_4	C_4		-	-	-	…	-	-	-	-	-				-	-	-	-	-	-
P				C_5	C_5	C_5	C_5	C_5	C_5		-	-	x	…	-	-	-	-	-				-	-	-	-	-	-
S				C_6	C_6	C_6	C_6	C_6	C_6		-	-	-	…	-	-	x	-	x				x	x	x	x	x	x
func()				$\$_m$	$\$_m$	$\$_m$	$\$_m$	$\$_m$	$\$_m$	0	a	b	d	…	q	q	u	x	y				c	h	m	0	u	y
State=		D												…						N									
Ad1	A			x	x	x	x	x	x		x	x	x	x	x	x	x	x	x		x		x	x	x	x	x	x
Ad2			B																		x								

Table 6.3 – Example of an ADT creating new rules for a formerly non-contemplated (M) and (S) Scenario

However basic, this mechanism actually learns from the users' behavior, and will converge into a complete 720 rules cDT differently for each user, or each run. On the other hand, it shows some limitations on the fact that, once created, the new rules can't be modified, even if the reaching scenario should indicate a new transition for that rule. Also, it doesn't provide the ability to contemplate multiple criteria for the decision making process. These problems are addressed by using the extensions shown in the following section.

VII. Extended Adaptive Decision Table

The ADT obtained in the previous section allows the creation of new rules that contemplate conditions that were not included in the original cDT. However, the only criterion used for determining the transition actions was that of maintaining the original current mode. This criterion, doesn't contemplate the usage of direct indicators of performance such as the metrics, nor a mechanism to alter the already created rules. Therefore, it is necessary to extend the definition of our ADT in order to include specific functions that could decide which mode to utilize based in the relation between the processes categories and the maintained metrics.

Module I consists in the identification of the different criteria –metrics, in our case– and alternatives –modes– for the decision problem. Their quantitative relation will define, in each case what alternative will be better for each scenario taking the metrics as reference. In our case we define the a C set of conditions, and an A set of alternatives as the following:

- $C = \{ \%cpu, \#_f, t_{cv}, t_w, t_r, t_s, o_v \}$
- $A = \{RR, RRPR, RRQV, FCFS, SJFS, BTS\}$

	Criterion Preference						
Category	%cpu	$\#_f$	t_{cv}	t_w	t_r	t_s	o_v
(II)	1	1	1	3	3	1	1
(M)	3	1	1	3	1	1	1
(ES)	1	3	1	1	3	1	1
(WEB)	1	1	1	1	3	3	1
(P)	1	3	3	1	1	1	1
(S)	1	1	1	1	1	1	3

Table 7.1 – Criterion preference by category

Module II consists in obtaining a Z matrix of performance for each combination of criteria and alternatives. Using the Saaty fundamental scale for comparing the relative importance of each criterion with each category we could elaborate the Table 7.1.

On the other hand, we define the importance of each criterion pair will vary regarding the amount of processes of each category that is ready to execute multiplied by the criterion preference shown in the Table 7.1. For example, for an scenario where 2 (II) category processes, and 1 (S) category process are ready, the importance of the t_w and t_r metrics will be equal to 6, and that of the o_v will be 3.

With those values in mind, a criteria pair preference can be elaborated for the example as the one shown in the Table 7.2.

	Criterion Preference						
Criteria	%cpu	$\#_f$	t_{cv}	t_w	t_r	t_s	o_v
%cpu	1	1	1	6	6	1	3
$\#_f$	1	1	1	6	6	1	3
t_{cv}	1	1	1	6	6	1	3
t_w	1/6	1/6	1/6	1	1	1/6	3
t_r	1/6	1/6	1/6	1	1	1/6	3
t_s	1	1	1	6	6	1	3
o_v	1/3	1/3	1/3	1/3	1/3	1/3	1

Table 7.2 – Relative preferences between criterion pairs

	Criterion Preference							Total Preference
Category	%cpu	$\#_f$	t_{cv}	t_w	t_r	t_s	o_v	
Multiplier	x 1	x 1	x 1	x 6	x 6	x 1	x 3	
RR	0,35	0,13	0,05	0,18	0,16	0,23	0,15	0,17
RRPR	0,35	0,13	0,15	0,54	0,16	0,03	0,15	0,28
RRQV	0,12	0,13	0,30	0,02	0,02	0,03	0,45	0,25
FCFS	0,06	0,13	0,30	0,02	0,02	0,03	0,45	0,11
SJFS	0,06	0,38	0,30	0,02	0,02	0,03	0,05	0,06
BTS	0,06	0,13	0,05	0,06	0,16	0,68	0,05	0,13

Table 7.3 – Matrix Z of alternatives preference

It can be seen on Table 7.3 that, for the example with two (II) processes, and one (S) process, results in a better preference for the RRPR mode, just above that of the RRQV mode. The Z matrix will be regenerated completely based in the amount of ready processes per category each time that an adaptive function calls to a *z_gen()* named function. Another *z_get()* named function will be used to obtain the most preferred scheduling mode from the current newest Z matrix.

Module III consists in the development of the functions for the insertion of new rules for the non-contemplated scenarios in the moment that they are detected. This was already achieved in the TDA presented in the Table 6.2.

It will only take to add calls to the *z_gen()* and *z_get()* functions along with a new variable *m* to hold their obtained value. In the Table 7.4 the final form of the EADT for the SODIUM process scheduler is shown.

		Adaptive Functions Declarations									Rules													
		Ad1		Ad2							0	1	2	3	…	26	27	28	29	30	31	32	33
		H	?	H	+	+	+	+	+	+	S	R	R	R	R	R	R	R	R	R	R	R	E
Conditions	Modo Actual				RR	PR	QV	FC	SJ	BT		RR	RR	RR	…	FC	SJ	SJ	BT	BT			
	Estado														…							N	
	II				C_1	C_1	C_1	C_1	C_1	C_1		-	x	-	…	-	-	-	-	-			
	M				C_2	C_2	C_2	C_2	C_2	C_2		x	-	-	…	-	-	-	-	-			
	ES				C_3	C_3	C_3	C_3	C_3	C_3		-	-	-	…	x	x	-	x	-			
	WEB				C_4	C_4	C_4	C_4	C_4	C_4		-	-	-	…	-	-	-	-	-			
	P				C_5	C_5	C_5	C_5	C_5	C_5		-	-	x	…	-	-	-	-	-			
	S				C_6	C_6	C_6	C_6	C_6	C_6		-	-	-	…	-	-	x	-	x			
Extended Adaptive Actions	z_gen()																					x	
	z_get()																					m	
Actions	func()				$m	$m	$m	$m	$m	$m	0	a	b	d	…	q	q	u	x	y			
	Estado=		D												…						N		
Adaptive Functions	Ad1	A			x	x	x	x	x	x		x	x	x	x	x	x	x	x	x		x	
	Ad2			B																		x	

Table 7.4 – Final form of the EADT for the SODIUM process scheduler

The values of the Table 7.2, obtained for this particular example, will vary depending on the current conditions scenario and the amount of processes per category. Nevertheless, continuing with the example, a normalized final Z matrix of performance can be obtained as the one shown in the Table 7.3.

Until now only brief tests and simulations for testing the EADT performance regarding different simple scenarios has been conducted due to the initial complexity of the implementation. Their results were satisfactory although yet not sufficient to determine its full potential. We estimate that in the next few months, new developments will support the usage of this powerful tool.

VIII. Conclusions and further work

With the usage of EADTs we were able to figure out alternatives for execution conditions of an aspect of an operating system that were too complex or inaccessible to figure out a priori. It also provided the kernel with the capability of changing the existing rules based on the new process usage behavior of each user. While these features may be possible to attain otherwise, none of the other existing adaptive devices provide such an intuitive mechanism for specifying rules and adaptive functions.

Although we are still lacking actual results from tests conducted on a variety of complex scenarios, the preliminary results on simple executions show that the decision tables converged into ideal solutions in each case, and that the kernel was actually learning from the process scheduler events. This actually serves as a demonstration that, so far, it is possible to create an automatic adaptive reconfiguration mechanism for a kernel without the supervision or explicit interactions with the users and their application.

There is still much potential to be harnessed from the EADTs. For example, we are not yet using the measures from the different metrics to evaluate each algorithm benefits from the actual execution. Doing this would converge and replace the initial heuristics on the algorithm-metrics relation tables.

Regarding SODIUM, there are other aspects of its design that are yet to be analyzed and converted into adaptively reconfigurable. That work should be done in the following months, during which we would still testing the results of the adaptive process scheduler.

Perspectives on the usage of adaptive mechanisms for automatic non-interactive reconfiguration are promising. These could be applied on any other home or enterprise operating systems in the market without the need of re-engineering their existing applications. An initial cost should be paid, nonetheless; mechanisms for automatic reconfiguration must be developed and provided as inputs for the EADTs, conditions must be analyzed, and events must be set.

References

[1] S. Martin, N. Casas, G. De Luca, “Diseño de un sistema operativo reconfigurable para fines didácticos y prácticos”. 6° Workshop de Tecnologia Adaptativa. San Pablo, Brasil, 2012.

[2] A. C. Veitch, N. C. Hutchinson, “Kea – a dynamically extensible and configurable operating system kernel”, 3rd International Conference on Configurable Distributed Systems. Vancouver, Canada, 1996.

[3] C. Cowan, T. C. Autrey, C. Krasic, C. Pu, J. Walpole, “Fast concurrent dynamic linking for an adaptive operating system”. 3rd International Conference on Configurable Distributed Systems. Vancouver, Canada, 1996.

[4] B. N. Bershad, S. Savage, P. Pardyak, E. G. Sirer, M. E. Fiuczynski, D. Becker, C. Chambers, S. Eggers, “Extensibility: Safety and Performance in the SPIN Operating System”. 5th Symposium on Operating Systems Principles. ACM, New York, United States, 1995.

[5] R. Lea, Y. Yokote, J. Itoh. “Adaptive operating system design using reflection”. Object-Based Parallel and Distributed Computation. Volume 1107, Springer Berlin, 1996.

[6] N. Casas, G. De Luca, M. Cortina, G. Puyo, W. Valiente, “Implementación de distintos tipos de memoria en un sistema operativo didáctico”. XIV Congreso Argentino de Ciencia de la Computación. La Rioja, Argentina, 2008.

[7] H. Ryckeboer, N. Casas, G. De Luca, “Construcción de un Sistema Operativo Didáctico”. X Workshop de Investigadores en Ciencias de la Computación. La Pampa, Argentina, 2008.

[8] A. Silberschatz, P. B. Galvin, G. Gagne. “Operating System Concepts”. 8va Edición. John Wiley & Sons, New Jersey, United States, 2012.

[9] S. M. Varghese, K. P. Jacob, “Process Profiling Using Frequencies of System Calls”. The Second International Conference on Availability, Reliability and Security (ARES'07). Viena, Austria, 2007.

[10] R. McDougall, J. Mauro, “Solaris Internals”. Second Edition. Prentice-Hall. California, United States, 2007.

[11] J. J. Neto, “Adaptative rule-driven devices - general formulation and a case study”. Sixth International Conference on Implementation and Application of Automata. Pretoria, South Africa, 2001.

[12] T. Pedrazzi, A. Tchemra, R. Rocha, “Adaptive Decision Tables A Case Study of their Application to Decision-Taking Problems”. Adaptive and Natural Computing Algorithms, Springer. Vienna, Austria, 2005.

[13] A. H. Tchemra, “Tabela de decisão adaptativa na tomada de decisões multicritério”. Phd Thesis. Escola Politécnica, USP, San Pablo, Brasil, 2009

[14] A. H. Tchemra, “Adaptatividade na Tomada de Decisão Multicritério”. 4° Workshop de Tecnologia Adaptativa. Escola Politécnica, USP, San Pablo, Brasil, 2010.

[15] T. L. Saaty, “Método de Análise Hierárquica”. McGraw-Hill. San Pablo, Brasil, 1991.

Sergio Miguel Martin is a Software Engineer from Universidad Nacional de La Matanza (UNLaM), Buenos Aires, Argentina since 2010. He performs as a teaching assistant on the operating systems class since 2010; and on the automata and formal languages class since 2011. He is currently finishing his master thesis on Software Engineering on the same university. His main investigation fields are operating systems and high-performance computing.

Graciela Elisabeth De Luca Is a Systems Analyst from the Universidad Tecnológica Nacional, and Bachelor of Computer Science from the Universidad Católica de Salta. Since 2005, belongs to the SODIUM research group of the Universidad Nacional de La Matanza. Also performs as professor for the Operating Systems Class.

Nicanor Blas Casas, Is a Software Engineer from the Universidad the Universidad Católica de Salta. Since 2005, belongs to the SODIUM research group of the Universidad Nacional de La Matanza. Also performs as the associate professor for the Operating Systems Class.

Comparativo entre duas estratégias adaptativas para definição dinâmica do intervalo de amostragem de dados em rede de sensores

I. M. Santos e C. E. Cugnasca

***Resumo*— As redes de sensores sem fio têm potencial para diferentes aplicações, dentre elas o monitoramento de ambientes para agricultura de precisão. Com objetivo de minimizar o consumo de energia dos nós da rede de sensores, são apresentadas duas estratégias baseadas em autômatos adaptativos, que determinam dinamicamente o intervalo (tempo) entre os registros e envios de dados pelos nós sensores. Essas estratégias consistem basicamente em assumir automaticamente intervalos longos enquanto os sensores estiverem coletando informações dentro de um conceito de normalidade, ou assumir intervalos curtos caso os nós sensores registrem informações consideradas atípicas ou fenômenos especiais, que demandam o registro mais detalhado de informações. Com a utilização dessa técnica há uma economia no consumo de energia nos nós sensores, sem que a rede perca eficiência no monitoramento dos fenômenos.**

***Palavras-chave*— redes de sensores sem fio, autômato adaptativo, otimização do consumo de energia, agricultura de precisão.**

I. Introdução

As Redes de Sensores Sem Fio (RSSF) são um dos principais exemplos de aplicação da computação pervasiva. Ela é uma tecnologia emergente, que promete revolucionar e ampliar o potencial de aplicações de monitoramento, instrumentação e controle [1].

Uma RSSF é um tipo especial de rede *ad hoc* com capacidade de coletar e processar informações de maneira autônoma [2, 3, 4]. Uma RSSF requer pouca ou nenhuma infraestrutura para a sua operação, consiste em um grande número de sensores que trabalham de forma colaborativa, monitorando um ambiente ou região para obter informações de interesse. Essas redes têm características e propriedades específicas, que as diferem das redes de computadores tradicionais, principalmente quanto aos recursos disponíveis [5]. Um dos principais desafios na aplicação das RSSF é a limitação de energia dos nós sensores, de modo que o uso racional e otimizado desse recurso é fundamental, pois determina o tempo de vida útil da rede.

Esse tipo de rede pode ser utilizado em muitas aplicações, como no monitoramento ambiental, agropecuária, processos industriais, sistemas embarcados em automóveis e aeronaves, sistemas de segurança, sistemas de suporte pessoal, mobilidade, controle logístico, entre outros [2, 3, 6, 7, 8, 9].

Dentre essas diversas aplicações, o foco deste trabalho é o monitoramento agrícola, que envolve o acompanhamento e a observação contínua de uma área de plantio, com o objetivo de avaliar as mudanças ocorridas nesse ambiente. Esse monitoramento é importante no processo de tomada de decisão e auxilia na solução de problemas, como ataques de pragas e doenças, correção do solo, aplicação de insumos e mudanças climáticas que podem prejudicar a produtividade da plantação.

Em aplicações nas quais os dados se modificam lentamente, como as relacionadas com a agricultura de precisão, pode-se esperar que os nós sensores utilizem intervalos dinâmicos de amostragem de dados, que podem ser longos, em situações de normalidade (com baixa variação da grandeza monitorada), e curtos, caso sejam observados dados que representam fenômenos que merecem maior atenção (situações de frio ou calor excessivo, ou variações repentinas nos dados monitorados).

A utilização de estratégias que consideram intervalos de amostragem de dados dinâmicos pode proporcionar economia de energia na RSSF, já que o número de medições de dados e o envio de pacotes pelos nós sensores da rede é reduzido. Cabe destacar que a comunicação é responsável pela maior parte do consumo de energia de um nó sensor. No entanto, esse procedimento não deve significar perda de eficiência da RSSF no processo de monitoramento do ambiente.

Este trabalho compara duas propostas adaptativas para o ajuste dinâmico do intervalo de amostragem de dados nos nós de uma RSSF. A primeira considera um "padrão de normalidade" previamente definido, impondo aos nós da rede de sensores a utilização de intervalos de amostragem mais longos para a coleta de dados quando os dados coletados estiverem dentro do padrão de normalidade e intervalos gradativamente mais curtos à medida que os dados registrados se distanciam do padrão de normalidade. A segunda proposta altera o intervalo de amostragem por meio da variação dos dados. Nesse caso, ocorrendo baixa variação nos registros dos dados monitorados, os nós da RSSF passam a considerar gradativamente intervalos mais longos, enquanto que caso ocorra alta variação dos dados, são adotados intervalos mais curtos.

Para controlar a dinamicidade dos intervalos de registro de dados em ambas as estratégias são propostos dois Autômatos Adaptativos (AA) [10]. Cada nó da RSSF irá executar seu AA

I. M. Santos, Universidade de São Paulo (USP), São Paulo, São Paulo, Brasil, ivairton@usp.br

C. E. Cugnasca, Universidade de São Paulo (USP), São Paulo, São Paulo, Brasil, carlos.cugnasca@poli.usp.br

de modo a garantir a autonomia individual na decisão do intervalo de amostragem de dados, em função dos fenômenos que estão sendo monitorados pontualmente.

Este trabalho apresenta na Seção 2 uma breve contextualização do monitoramento agrícola, as potencialidades de se utilizar RSSF nesse contexto e descreve as estratégias adaptativas que serão utilizadas pelos AA. A Seção 3 discute e apresenta os AA para o problema de definição dinâmica do intervalo de amostragem de dados em RSSF. Na Seção 4 são apresentados e discutidos os resultados obtidos por meio das simulações computacionais e na Seção 5 são apresentadas as considerações finais.

II. Monitoramento de ambientes agrícolas com RSSF

A Agricultura de Precisão (AP) consiste em um manejo específico da área cultivada, tratando de modo diferenciado as parcelas de terreno de acordo com suas necessidades. Com uso desse modelo, algumas vantagens podem ser obtidas, com destaque para o aspecto econômico, com maior produção a menor custo, além do menor impacto ambiental, ocorrendo menor desgaste ambiental com o manjo (por exemplo, com a irrigação e empobrecimento do solo) e menor índice de contaminação com defensivos agrícolas [11]. A aplicação das RSSF na AP é uma alternativa promissora, possibilitando um monitoramento da cultura e das propriedades do ambiente de cultivo mais detalhado e contínuo [12].

No monitoramento agrícola, geralmente os dados (fenômenos) variam lentamente. Nesse contexto uma RSSF não precisa monitorar o ambiente com alta frequência de registro de dados, pois isso representa desperdício de energia e registro redundante de dados.

Há diferentes maneiras de implementar a variação do intervalo de amostragem de dados. Uma possível estratégia é considerar um "padrão de normalidade" (valores esperados) fazendo com que a rede de sensores adote intervalos de amostragem de dados longos enquanto as informações obtidas corresponderem ao padrão. Quando ocorrer a leitura de dados fora do padrão de normalidade, então os nós da rede devem reduzir o intervalo de amostragem, de modo a registrar os dados mais frequentemente. Essa estratégia será denominada de "Estratégia 1".

Entretanto, dependendo do contexto nem sempre é possível definir um padrão de normalidade, ou esse padrão pode ser dinâmico durante o monitoramento. Dessa forma, pode-se considerar outra estratégia que permite ao nó da rede decidir de maneira autônoma quando e quanto alterar o intervalo de amostragem de dados, com base na variação dos dados monitorados. Quando ocorrer variações consistentes nos dados o intervalo será reduzido, aumentando a freqüência do registro de dados, enquanto que baixas variações determinam intervalos maiores. A grandeza da variação dos dados irá influenciar na determinação do intervalo de amostragem. Essa estratégia será denominada de "Estratégia 2".

A Fig. 1 ilustra as duas estratégias por meio de gráficos. O eixo *x* corresponde a um intervalo de tempo, enquanto que o eixo *y* ao valor do dado monitorado pela rede de sensores. Os pontos representam um registro efetuado por um nó da rede e a linha corresponde ao comportamento da grandeza monitorada. O gráfico da Fig. 1(a) ilustra a Estratégia 1 que considera um padrão de normalidade representado pelo intervalo a, b. Observa-se que enquanto os dados obtidos estão dentro do intervalo de normalidade a frequência entre um registro e outro é menor do que quando ocorre uma medição fora do padrão. O gráfico da Fig. 1(b) ilustra a Estratégia 2 que considera a estratégia baseada na variação dos dados. Nota-se que enquanto ocorre uma baixa variação entre os dados registrados pelo sensor há um intervalo maior entre um registro e outro. Ao ocorrer uma variação acentuada nos dados aumenta-se a frequência das amostras.

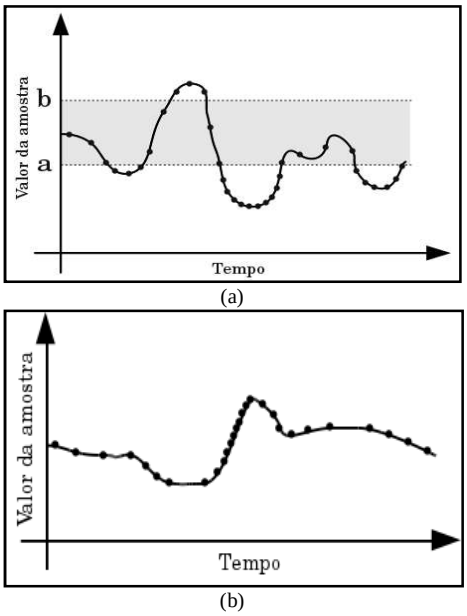


Figura 1. Exemplo de sequências de medidas realizadas por um nó sensor ao longo do tempo, empregando duas estratégias distintas. Em (a), quando ocorrem medidas fora do padrão de normalidade (limite a, b) o intervalo entre as amostras é reduzido. Em (b), quando há variação acentuada entre os dados registrados o intervalo entre as amostras é reduzido.

Com essas estratégias espera-se economizar energia dos nós sensores enquanto o ambiente apresenta uma situação de normalidade ou baixa variação de dados e garantir a eficiência do monitoramento do ambiente em situações de maior interesse.

III. Descrição do modelo adaptativo

Um AA é uma máquina de estados a qual são impostas sucessivas alterações, resultantes da aplicação de ações adaptativas associadas às regras de transições executadas pelo autômato [10]. Dessa forma, estados e transições podem ser eliminados ou incorporados ao autômato em decorrência de cada um dos passos executados durante a análise da entrada. Pode-se dizer que um AA é formado por um dispositivo convencional (não-adaptativo) e um conjunto de mecanismos adaptativos responsáveis pela auto-modificação do sistema.

Para determinar a mudança nos valores dos intervalos de amostragem de dados nos nós da RSSF é proposto um AA para cada estratégia, onde cada estado do autômato corresponde a um valor para o intervalo de amostragem de

dados e cada nó da rede executa seu autômato. Para cada estratégia proposta é preciso considerar alguns aspectos e requisitos específicos a serem discutidos a seguir.

A. Autômato Adaptativo para definição do intervalo de amostragem de dados com base num padrão de normalidade

Para determinar a mudança nos intervalos de amostragem de dados nos sensores com um padrão de normalidade é proposto um AA que assume as seguintes condições:

- Se o dado coletado pelo nó sensor pertence ao padrão de normalidade, então o AA transita para o estado correspondente ao maior intervalo de amostragem de dados;
- Se o dado coletado mantém-se estável, então o AA permanece no estado corrente;
- Se o dado coletado pelo nó sensor extrapola os limites do padrão de normalidade, então o AA transita para um estado que corresponda a um intervalo de amostragem de dados menor, por meio de uma função adaptativa;
- Se o valor do dado coletado retorna ao padrão de normalidade, então o AA transita de volta para o estado que corresponde ao maior intervalo de amostragem de dados.

O AA correspondente à Estratégia 1 é apresentado na Fig. 2. No autômato o símbolo "<" corresponde a uma leitura pelo sensor de uma amostra com valor abaixo do padrão de normalidade e com variação (em relação à leitura anterior) superior a um limite mínimo. Aqui este limite será referenciado como *k*. O símbolo ">" representa uma leitura com valor superior ao padrão de normalidade e variação maior que *k*. Portanto, na ocorrência de uma entrada com um desses símbolos, irá ocorrer uma ação adaptativa gerando um novo estado no autômato que corresponde a um intervalo de amostragem de dados mais curto e a adição da transição para esse novo estado. O símbolo "=" corresponde a um registro dentro do padrão de normalidade ou com variação menor que *k*.

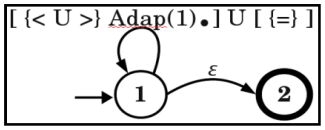


Figura 2. AA para determinação do intervalo de amostragem de dados com base em um padrão de normalidade.

A Fig. 3 ilustra um possível exemplo de execução do AA apresentado. Na Fig. 3(a) o AA encontra-se em seu estado inicial. Considerando uma suposta leitura pelo nó sensor de um valor abaixo do padrão de normalidade e com variação superior a *k*, ocorre uma ação adaptativa, criando um novo estado (#1). A nova configuração do AA pode ser observada na Fig. 3(b). O novo estado corresponde a um intervalo de amostragem de dados menor do que o intervalo representado pelo estado 1. Nessa nova configuração, caso o nó sensor volte a executar uma nova leitura e a obter um valor ainda menor (<), irá ocorrer uma nova ação adaptativa, criando um novo estado com intervalo de amostragem de dados ainda menor que em #1. Caso ocorra uma amostragem "melhor", então o autômato transita para um estado com intervalo de amostragem de dados maior, no caso do exemplo, retornando para o estado 1.

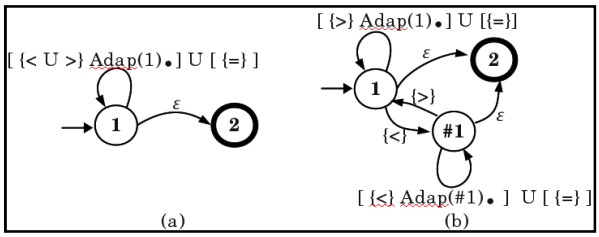


Figura 3. Exemplo de execução de uma ação adaptativa no AA após a leitura de um valor abaixo do padrão de normalidade.

O comportamento do AA é análogo para o caso de registros de dados acima do padrão de normalidade.

Tabela de decisão adaptativa: Durante a execução de uma transição adaptativa o AA sofre alguma mudança em sua configuração (eliminando, acrescentando, ou simplesmente modificando estados e transições). Isto faz com que uma nova máquina de estados apareça, no lugar da anterior, caracterizando a execução de um passo adicional.

Uma função adaptativa é formada por um conjunto de ações executadas em sequência na ocasião da aplicação da função. Há três ações elementares, que podem ser usadas para compor as funções adaptativas: consulta (representada por "?"), eliminação (representada por "-") e adição (representada por "+") de transições ao autômato. Uma Tabela de Decisão Adaptativa (TDA) permite representar as regras de uma função adaptativa e consequentemente representar toda a lógica do AA [10, 13].

TABELA I

TDA do AA para determinação do intervalo de amostragem de dados considerando um padrão de normalidade.

	Label	Tag	Condições		Ações			Funções Adaptativas		Parâmetros				Geradores	
			Estado =	Entrada =	Estado ←	Consome	Aceita ←	R1	R2	P1	P2	P3	P4	G1	G2
1		S			1	√									
2		R	1	=	1	√									
3		R	1				x								
4		R	1	ε	2										
5		R	1	<	1			√		1	1				
6		R	1	>	1				√			1	1		
7		R	2				√								
8		E													
		H						√		√	√			√	
		-	P1	<	P2			√		P1	P2				
		+	P1	<	G1	√									
		+	G1	>	P1	√									
		+	G1	=	G1	√									
		+	G1	<	G1			√		G1	G1				
		+	G1	ε	2										
		H							√			√	√		√
		-	P3	>	P4				√			P3	P4		
		+	P3	>	G2	√									
		+	G2	<	P3	√									
		+	G2	=	G2	√									
		+	G2	>	G2				√			G2	G2		
		+	G2	ε	2										

Considerando o diagrama do AA proposto na Fig. 2 a Tabela I apresenta sua respectiva TDA.

Validação da TDA: Visando verificar a TDA da Tabela I, foi utilizado o software AdapTools [14], que permite representar, implementar e simular o comportamento de um AA a partir da sua TDA.

As simulações executadas demonstraram o comportamento esperado do AA, portanto a correta construção da TDA. A Fig. 4 corresponde à representação da TDA da Tabela I no *software* AdapTools e a Fig. 5 à representação gráfica do AA em um determinado instante da simulação.

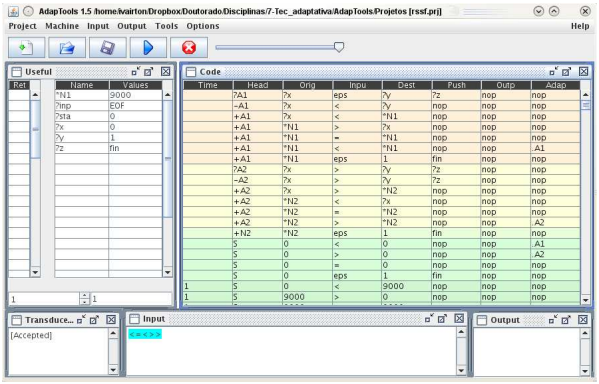


Figura 4. Representação do AA no *software* AdapTools.

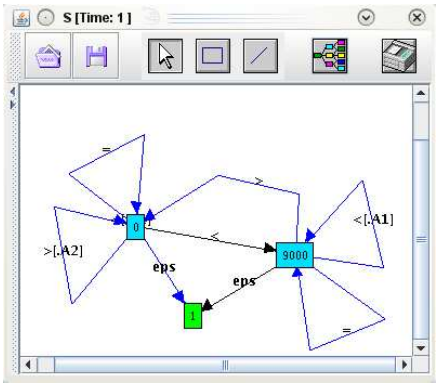


Figura 5. Representação gráfica do AA durante a simulação computacional.

B. Autômato Adaptativo para definição do intervalo de amostragem de dados com base na variação dos dados

Para determinar a mudança nos intervalos de amostragem de dados nos nós sensores da RSSF a partir da variação dos dados das amostras é proposto um AA que assume as seguintes condições:

- O estado inicial do AA corresponde ao intervalo de amostragem de dados mais curto;
- Se o dado coletado pelo nó sensor mantém-se estável, ou com baixa variação, então o AA altera seu estado por meio de uma função adaptativa, transitando para um estado com intervalo de amostragem de dados maior, definido dinamicamente;
- Se o dado coletado pelo nó sensor varia acentuadamente, acima de um limite aqui denominado de *w*, então o AA retorna para o estado inicial, para o menor intervalo de amostragem de dados disponível.

É importante destacar que deve ser determinado um limite máximo para o valor do intervalo de amostragem de dados, isso evitar uma possível definição de um intervalo excessivamente longo, o que poderia causar inconsistência no registro de informações pela rede de sensores.

A Fig. 6 apresenta o AA correspondente. O símbolo "=" corresponde ao conjunto de amostras com baixa variação (inferior a *w*), enquanto que o símbolo "+" corresponde ao conjunto de amostras com alta variação (superior a *w*). Nota-se que quando ocorre uma leitura de um dado com baixa variação é executada uma ação adaptativa que cria um novo

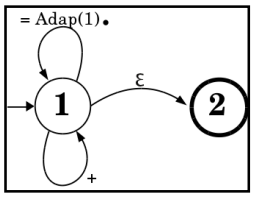


Figura 6. AA para determinação do intervalo de amostragem de dados baseado na variação dos dados registrados pelo nó sensor.

estado no autômato, com valor de intervalo de amostragem de dados maior, e adiciona-se a transição para este novo estado.

A determinação do valor do intervalo de amostragem de dados para o novo estado do AA será inversamente proporcional à variação do dado. Portanto, quanto menor a variação, maior será o incremento no intervalo. Diferentes funções podem ser adotadas nesse processo, aqui é proposta a Equação 1 para esse cálculo:

$$t' = t + t_{min} \times (1 - \Delta v/w) \times z \qquad (1)$$

onde:

t é o valor do intervalo de amostragem de dados do estado atual;

t_{min} corresponde ao valor do intervalo de amostragem de dados mínimo;

Δv é a variação entre os dados registrados pelo sensor;

w corresponde à variação máxima permitida;

z é um fator multiplicador adimensional para ajustes.

A Fig. 7 ilustra um exemplo de execução do AA. Considere uma aplicação em que a RSSF está monitorando a temperatura ambiente de uma cultura. Assuma os seguintes parâmetros para o exemplo: o intervalo de amostragem de dados inicial e mínimo (t_{min}) é de 5 minutos, a variação máxima (w) é de 0,5 ºC e o fator multiplicador (z) é igual a 1. Na Fig. 7(a) o AA encontra-se em seu estado inicial, o estado 1 corresponde ao intervalo de amostragem de dados de 5 minutos. Supondo um

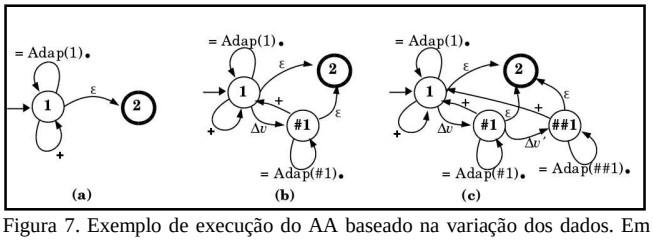


Figura 7. Exemplo de execução do AA baseado na variação dos dados. Em (a) o autômato está em seu estado inicial. Em (b) ocorre uma ação adaptativa no estado 1. Em (c) ocorre uma nova ação adaptativa no estado #1.

novo registro do nó sensor com variação menor que *w* (0,1ºC por exemplo), é executada a função adaptativa que cria um novo estado no autômato (#1). Esse novo estado é alcançado com a variação Δ*v* conforme demonstra a Fig. 7(b). Considerando os dados do exemplo e a Equação 1, o valor do intervalo de amostragem de dados do estado #1 será de 9 minutos. Ao imaginar novamente um novo registro do sensor, supondo uma variação de 0,4 ºC, a função adaptativa é executada criando um novo estado no autômato (##1), conforme ilustra a Fig. 7(c). Considerando a variação dos dados (0,4 ºC) e o estado atual do autômato, o novo estado (##1) terá o intervalo de amostragem de dados igual a 10 minutos (incremento em 1 minuto).

Em qualquer estado do AA, se for registrado um novo dado cuja variação seja superior a *w* o autômato retorna ao seu estado de entrada (estado 1).

Tabela de decisão adaptativa: Considerando o diagrama do AA apresentado na Fig. 6, a Tabela II apresenta a TDA que descreve as suas respectivas ações adaptativas. Nesta TDA ocorrem somente ações adaptativas de adição (+).

Na execução da função adaptativa, a variação Δ*v* obtida pelo nó sensor deve ser retirada do conjunto representado por "=", se tornando a entrada que corresponde à transição para o novo estado criado. Isso é importante para evita ambiguidade

TABELA II
TDA referente ao AA para determinação do intervalo de amostragem baseado na variação dos dados.

Label	Tag	Condições		Ações			Funções Adaptativas	Parâmetros	Geradores
		Estado =	Entrada =	Estado ↓	Consome	Aceita ↓	R1	P1	G1
1	S			1	√				
2	R	1	+	1	√				
3	R	1				x			
4	R	1	ε	2					
5	R	1	+	1			√	1	
6	R								
7	R	2				√			
8	E								
	H						√	√	√
	+	P1	=	G1	√				
	+	G1	+	P1	√				
	+	G1	=	G1			√	G1	
	+	G1	ε	2					

no autômato.

Validação da TDA: Para verificar a TDA apresentada na Tabela II, também foi utilizado o software AdapTools. As simulações executadas demonstraram o comportamento esperado do AA, portanto, a correta construção da TDA. A Fig. 8 corresponde à representação da TDA no AdapTools e a Fig. 9 ilustra a representação gráfica do AA em um determinado instante da simulação.

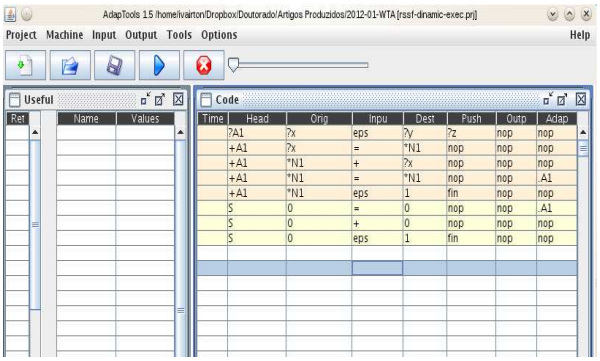


Figura 8. Representação do AA no *software* AdapTools.

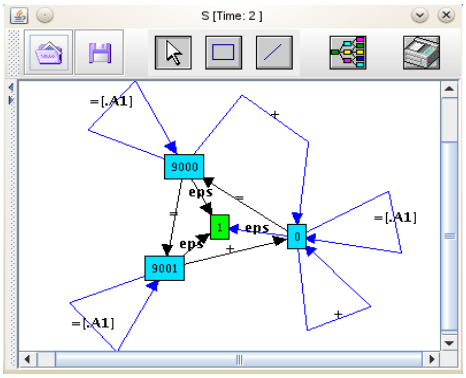


Figura 9. Representação gráfica do AA durante a simulação computacional.

IV Resultados

Buscando avaliar e comparar a economia de energia em uma RSSF proporcionada pela utilização dos AA propostos, foi desenvolvido um sistema computacional que simula seu funcionamento. O sistema foi implementado em linguagem de programação Java e simula o consumo de energia dos nós sensores quando registram dados, enviam pacotes de dados e quando estão em modo de espera (*stand-by*). O sistema não implementa algoritmos de roteamento de dados, limita-se a simular o funcionamento dos sensores (registro, envio/recebimento de dados e modo de espera), o consumo de energia de suas baterias e o reenvio de pacotes de dados por razão de falha (de acordo com parâmetros definidos).

O sistema de simulação considera uma aplicação de monitoramento de temperatura. Como referência, foi utilizada uma série histórica real de temperatura registrada pelo CPTEC INPE (http://www.cptec.inpe.br/) da região do município de Bento Gonçalves/RG. A escolha dessa área se deu pelo número relevante de parreirais presentes nessa região e da importância de se monitorar a temperatura nessas culturas, especialmente em épocas frias por conta da ocorrência de geadas.

Para simular a temperatura ambiente, adotou-se uma temperatura simulada "global" (gerada por meio de métodos pseudo randômicos). Os dados gerados neste processo são tendenciosos, seguindo como parâmetro os registros de temperatura determinados pela série histórica. Cada nó da rede "registra" (simula o registro por meio da geração de dados

psudo randômicos) temperaturas que correspondem a variações da temperatura "global". Dessa forma mantém-se a tendência em seguir a temperatura global e permite simular diferentes microrregiões na área onde a rede de sensores está instalada. O funcionamento dos dois AA apresentados foram simulados nesse ambiente, empregando os mesmos parâmetros.

Em todas as simulações foi considerada a mesma configuração para os nós da rede, assumindo como limite mínimo do intervalo de amostragem de dados o valor de 3 minutos e o máximo de 30 minutos. Para o AA correspondente à Estratégia 1 foi determinado o padrão de normalidade com intervalo entre 20 e 28 ºC. O valor para o limite de variação mínimo (*k*) foi de 1% em relação à temperatura corrente registrada pelo sensor.

As simulações da Estratégia 1 objetivaram verificar o desempenho energético com a variação no número de intervalos de amostragem de dados disponíveis para os sensores. Os parâmetros utilizados são apresentados na Tabela III.

TABELA III
Tabela das configurações utilizadas nas simulações do AA da Estratégia 1.

Nº de intervalos	Duração dos intervalos (em minutos)
2	{30,3}
4	{30,20,10, 3}
6	{30,24,18,13, 8, 3}
8	{30,26,22,18,14,10, 6, 3}
10	{30,26,23,20,17,14,11, 9, 6, 3}
12	{30,27,24,21,19,17,15,13,11, 9, 6, 3}
14	{30,27,25,23,21,19,17,15,13,11, 9, 7, 5, 3}

Nas simulações da Estratégia 2 o objetivo foi verificar o desempenho energético da rede investigando o valor para a variação máxima permitida (*w*). As simulações consideraram valores para *w* de 1,0%, 1,5%, 2,0%, 2,5% e 3,0% do valor da temperatura registrada pelo nó sensor.

O gráfico na Fig. 10 demonstra os resultados obtidos em um conjunto de simulações. O eixo *x* corresponde à sequência de simulações, enquanto que o eixo *y* à quantidade de dias que a rede funcionou (segundo a simulação). As simulações que empregaram a Estratégia 1 estão representadas por linhas tracejadas, enquanto que as simulações baseadas na Estratégia 2 estão em linhas contínuas. Observa-se no gráfico que no caso da Estratégia 1 quanto maior o número de intervalos pré-definidos melhor o desempenho da rede (maior duração em dias). Entretanto os resultados obtidos por essa estratégia apresentaram variação acentuada, tendo ocorrido resultados superiores a 30 dias de diferença, em um mesmo contexto. Já os resultados obtidos com a Estratégia 2 demonstraram um comportamento mais estável, sendo a simulação com valor para a variação máxima (*w*) igual a 2% a de maior eficiência.

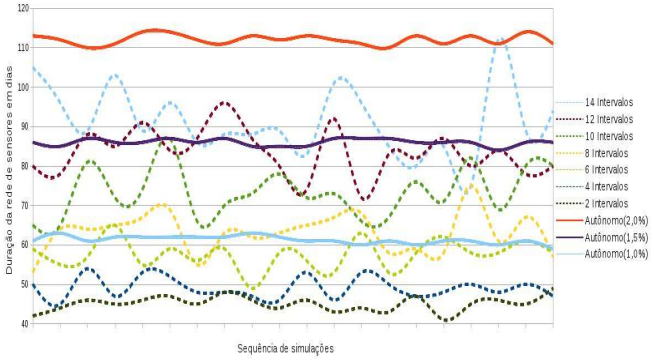


Figura 10. Resultados das simulações dos AA baseados na Estratégia 1 (linhas tracejadas) e na Estratégia 2 (linhas contínuas).

Os resultados das simulações para a Estratégia 2 com valor de *w* igual a 2,5% e 3,0% não aparecem no gráfico da Fig. 10 em razão das simulações com esses parâmetros terem apresentado um comportamento indesejado. Pelo fato do limitante da variação estar alto, os nós apresentaram tendência em manter o intervalo de amostragem de dados em seu valor máximo. O gráfico apresentado na Fig. 11 demonstra essa tendência nas simulações com *w* igual a 2,0%, 2,5% e 3,0%. O eixo *x* corresponde aos intervalos de amostragem de dados adotados pelo nó sensor durante a simulação, variando de 3 a 30 minutos. O eixo *y* corresponde ao número de vezes que o intervalo ocorreu durante a simulação. Em razão do comportamento tendencioso, as simulações com valor de *w* acima de 2,0% foram descartadas.

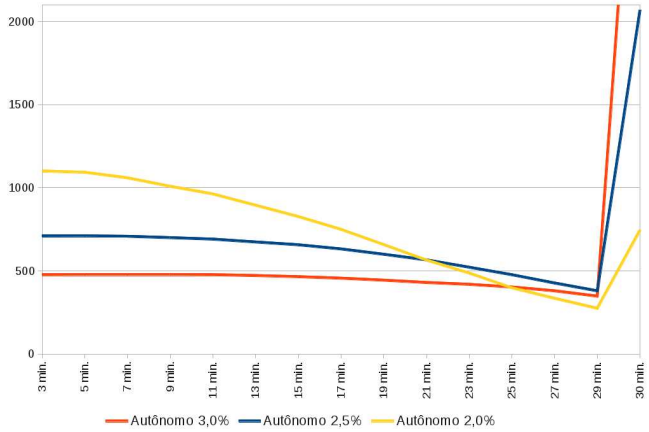


Figura 11. Tendência em manter o intervalo de amostragem de dados em seu valor máximo nas simulações da Estratégia 2 com valor de *w* superior a 2% do valor da temperatura.

A variação nos resultados obtidos pela Estratégia 1 é melhor visualizada na Fig. 12. O exemplo que melhor

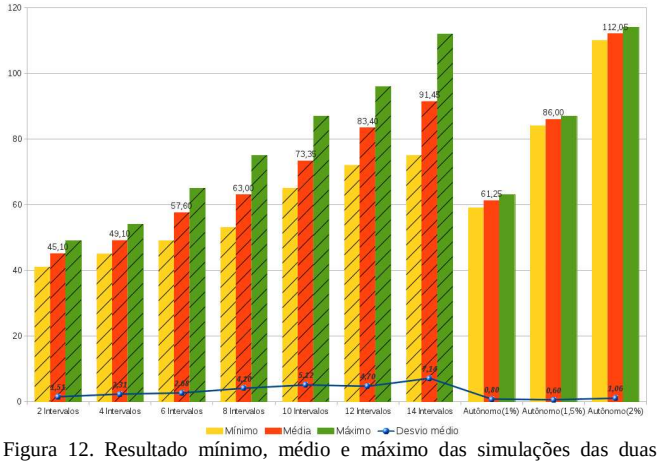


Figura 12. Resultado mínimo, médio e máximo das simulações das duas estratégias com suas respectivas configurações e desvio médio obtido em cada contexto.

demonstra esse comportamento é a simulação que adota 14 intervalos pré-definidos. Nesse caso os resultados obtidos em diferentes simulações variaram entre 75 e 112 dias, sendo o desvio médio de 7,14. Em contra partida, a Estratégia 2 apresentou uma baixa variação em seus resultados, apresentando no máximo um desvio médio de 1,06 para as simulações que consideraram *w* = 2%. Esse comportamento pode estar associado à função que determina dinamicamente o valor do intervalo de amostragem de dados, o que faz crer que uma função que propicie maior amplitude nos resultados obtidos pela estratégia venha a interferir nesse comportamento.

A Fig. 12 também demonstra o resultado médio das simulações das duas estratégias. Como esperado, na Estratégia 1 a utilização de 14 intervalos pré-definidos propiciou maior tempo de vida da RSSF. Isso ocorre em razão da utilização de intervalos de amostragem de dados com valores maiores, o que não ocorre (ou em menor quantidade) em outros contextos que possuem poucas opções de intervalos de amostragem.

Nas simulações com a Estratégia 2, o melhor resultado ocorreu com o parâmetro *w* = 2%. Os resultados obtidos por esta estratégia estão diretamente associados ao valor de *w*. Um valor muito baixo determina uma inflexibilidade ao autômato e portanto um constante retorno ao estado inicial. O que corresponde ao uso freqüente do intervalo de amostragem de dados mais curto. Em contra partida, conforme tendência notada na Fig. 11, um valor de *w* muito alto leva o autômato a atingir facilmente o valor máximo para o intervalo de amostragem de dados, prejudicando e eficiência do monitoramento executado pela rede de sensores. Esse contexto revela a necessidade de uma calibração adequada para o valor de *w*.

O comportamento estável do AA em diferentes simulações na Estratégia 2 leva à outra análise quanto a função de cálculo do valor para o intervalo de amostragem. Acredita-se que há margem para melhorias nesta função, de modo a ampliar as combinações de valores para os intervalos e dar melhor oportunidade ao AA de gerar resultados melhores.

O gráfico na Fig. 13 demonstra a quantidade de vezes que um intervalo de amostragem de dados foi determinado durante a simulação, por meio de curvas de tendência logarítmica.

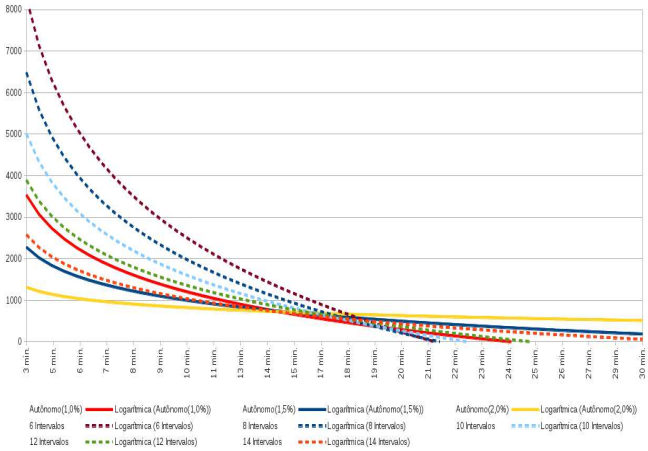


Figura 13. Linha de tendência de utilização dos intervalos de amostragem durante a simulação.

Com esse gráfico é possível notar a distribuição da ocorrência de cada intervalo de amostragem de dados durante a simulação. Inicialmente observa-se que em todas as simulações os intervalos de amostragem mais curtos são mais frequentes, enquanto que aqueles intervalos mais longos ocorrem com menor frequência. Isso demonstra que a rede mantém um monitoramento mais detalhado a maior parte do tempo. Ainda assim, durante as simulações não foram feitas análises quanto a possíveis falhas no monitoramento de eventos considerados importantes, o que pode ser explorado em trabalhos futuros.

No gráfico observa-se que em algumas simulações há uma utilização deficiente dos intervalos disponíveis, com baixa utilização ou mesmo nula de alguns intervalos, como no caso das simulações da Estratégia 1 com 2, 4, 6, 8, 10 e 12 intervalos pré-definidos e na Estratégia 2 com *w* igual a 1%. Como o intervalo de amostragem de dados poderia assumir valores entre 3 e 30 minutos, o gráfico da Fig. 13 demonstra que em algumas simulações esse intervalo não foi plenamente utilizado. Isso mostra que o AA pode ser melhor ajustado de modo a explorar de maneira mais plena o intervalo disponível.

As deficiências encontradas com a análise dos resultados demonstram que as pesquisas podem evoluir nos trabalhos futuros. É importante destacar o potencial dos AA que, em simulação computacional, permitiram uma RSSF comum com duração média de 40 dias pudesse alcançar uma duração até 112 dias (Estratégia 1 com 14 intervalos) ou 114 dias (Estratégia 2 com *w* = 2%), o que equivale dizer que praticamente triplicou sua capacidade de duração de funcionamento sem necessariamente perder eficiência no monitoramento de variáveis ambientais de interesse da aplicação.

IV. Conclusões

A AP é um exemplo entre as diversas áreas que pode tirar proveito do potencial das RSSF para aprimorar ou desenvolver novos processos de aplicação.

As RSSF têm como um dos seus principais desafios a necessidade de economizar energia dos seus nós sensores, de modo a prolongar ao máximo o tempo de duração da rede. Na AP não há a demanda por um monitoramento com alta frequência de coleta de dados, pois eles se alteram lentamente. Pode-se empregar uma estratégia que adota intervalos de amostragem de dados dinâmicos, longos enquanto as informações obtidas estiverem em um contexto de normalidade e curtos para eventos que demandam um monitoramento mais detalhado. Esse processo pode ser baseado em dois aspectos, na adoção de um padrão de normalidade para os dados e na variação entre os dados monitorados.

Neste trabalho foram apresentados dois autômatos adaptativos para explorar essas estratégias por serem estruturas computacionais que podem ser implementadas individualmente nos nós da rede de sensores mantendo sua autonomia. Por meio de simulação computacional a aplicação dos AA demonstrou melhor desempenho energético da RSSF. Economia de energia que significa maior tempo de vida útil da

rede de sensores, podendo chegar a cerca de três vezes o tempo original.

A continuidade deste trabalho prevê a utilização de novos parâmetros para melhor ajuste dos autômatos, além da verificação de novas formulações para a definição do valor do intervalo de amostragem de dados na Estratégia 2. Além disso, espera-se verificar a eficiência do monitoramento de diferentes fenômenos pela rede de sensores e elaborar uma métrica que qualifique os valores gerados para os intervalos de amostragem de dados, de modo que o AA possa escolher por intervalos mais apropriados e "aprender" durante sua execução, segundo os padrões encontrados.

AGRADECIMENTOS

Os autores agradecem a Fundação de Amparo à Pesquisa do Estado de Mato Grosso – FAPEMAT – pelo apoio a este trabalho via projetos de pesquisa e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) via programa Doutorado Interinstitucional Escola Politécnica da USP e Universidade Federal de Mato Grosso (EPUSP-UFMT).

REFERÊNCIAS

[1] WEISER, M., The computer for the 21st century. Disponível em <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>, 1991. Acesso em: Agosto de 2009.

[2] AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., CAYIRCI, E., Wireless sensor networks: a survey. *Computer Networks*. 38, pp. 393-422, 2002.

[3] TUBAISHAT, M., MADRIA, S., Sensor networks: an overview, *IEEE Potentials*. 22 (2), 2003.

[4] GAJBHIYE, P., MAHAJAN, A., A survey of architecture and node deployment in Wireless Sensor Network. *Applications of Digital Information and Web Technologies*, pp. 426-430, 2008.

[5] ESTRIN, D., GIROD, L., POTTIE, G., SRIVASTAVA, M., Instrumenting the world with wireless sensor networks. *International Conference on Acoustics, Speech, and Signal Processing*, USA, 2001.

[6] POTTIE, G. J., KAISER, W. J., Wireless integrated network sensors. *Communications of the ACM*, 43, pp. 51–58, 2000.

[7] YICK, J., MUKHERJEE, B., GHOSAL, D., Wireless sensor network survey. *Computer Networks*, 52, (12), pp. 2292-2330, 2008.

[8] SOHRABY, K., MINOLI, D., ZNATI, T., *Wireless Sensor Networks – Technology, Protocols, and Applications*. Ed. Wiley, 2007.

[9] SIKKA, P., Wireless ad hoc sensor and actuator networks on the farm. *ACM*, pp. 492-499, 2006.

[10] NETO, J. J., Adaptive rule-driven devices – general formulation and case study, *International Conference on Implementation and Application of Automata*, pp. 234-250, ISBN 3-540-00400-9, 2002.

[11] MOLIN, J. P., Tendências da agricultura de precisão no Brasil, *Anais do Congresso Brasileiro de Agricultura de Precisão 2004*, Ribeirão Preto/SP, pp. 1-10, 2004.

[12] SANTOS, I. M., DOTA, M. A., CUGNASCA, C. E., Visão Geral da Aplicabilidade de Redes de Sensores Sem Fio no Monitoramento Agrícola no estado de Mato Grosso. *Anais do Congresso Brasileiro de Agricultura de Precisão 2010*, Ribeirão Preto/SP, 2010.

[13] ROCHA, R. L. A., NETO, J. J., Uma proposta de linguagem funcional com características adaptativas, *CACIC 2003*, 2003.

[14] JESUS, L. D., SANTOS, D. G. D., CASTRO, A. A. D., PISTORI, H., AdapTools 2.0: Implementation and Utilization Aspects, *IEEE Latin America Transactions*, 5, pp. 527-532, 2007.

Ivairton Monteiro Santos é graduado em Ciência da Computação pela Universidade Federal de Mato Grosso (2002) e mestre em Ciência da Computação pela Universidade Federal Fluminense (2005). Atualmente é aluno de doutorado da Escola Politécnica da USP, sob a orientação do Prof. Dr. Carlos Eduardo Cugnasca. É membro do Laboratório de Automação Agrícola da Escola Politécnica da USP e professor da Universidade Federal de Mato Grosso, no Campus Universitário do Araguaia. Tem experiência na área de Ciência da Computação e seus interesses em pesquisa concentram-se em Redes de Sensores Sem Fio e otimização combinatória.

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

Online Learning of Abstract Stochastic Policies with Monte Carlo

M. L. Koga, V. F. da Silva e A. H. R. Costa

***Abstract*— When an autonomous agent faces many different, but related tasks, obtaining abstract stochastic policies can be helpful because they can be used as transferred knowledge to accelerate learning in a new task throughout transfer learning from previously solved tasks. This paper presents a novel model-free algorithm for online building abstract stochastic policies for Relational Markov Decision Processes, within the reinforcement learning framework. This algorithm, AbsProb-RL, performs a search in the policy-space and uses Monte Carlo techniques to estimate the gradient of the value function of a policy. Results show that AbsProb-RL can effectively find optimal abstract policies, if given enough time.**

***Keywords*— Reinforcement Learning, Transfer learning, Relational Markov Decision Process, Monte Carlo, Abstraction.**

I. INTRODUÇÃO

Cada vez mais nos deparamos com máquinas autônomas na vida cotidiana e os esforços da área de Inteligência Artificial são justamente voltados a construir agentes (máquinas) inteligentes que possam resolver os mais diversos problemas. Nas tarefas de navegação robótica, o agente tem como objetivo achar o menor caminho para alcançar um local específico, sua meta. A capacidade de um agente de aprender e adaptar-se a diferentes situações através da interação com o ambiente é um dos desafios da área, que é atacado pelo aprendizado por reforço (AR) [1]. Nesse tipo de aprendizado, a agente aprendiz deve maximizar sua recompensa (numérica) descobrindo, através de uma estratégia de tentativa-e-erro com repetidas interações com o ambiente, quais são as melhores ações a serem tomadas em cada situação.

Um dos principais problemas do aprendizado por reforço é que o aprendizado pode ser lento e o agente pode demorar a encontrar uma política de ação ótima, pois como o aprendizado é baseado na interação do agente com o ambiente, dependendo da sua dimensão a exploração do ambiente todo pode levar muito tempo. Uma alternativa para se acelerar esse processo é usar o conhecimento adquirido na resolução de um problema anterior (fonte) e usá-lo no aprendizado de um outro problema similar (destino), ao invés de aprender a resolver o novo problema sem qualquer conhecimento prévio. A esse processo se dá o nome de transferência de conhecimento [2].

Para se realizar tal transferência, dois aspectos são importantes de serem considerados: qual o conhecimento a ser transferido e a representação utilizada para se descrever o problema. Quanto ao conhecimento a ser transferido, diversas opções têm sido exploradas, desde a construção de macroações (planos parciais) para transferência e reuso no aprendizado de novas políticas de atuação [3], a transferência baseada na função valor de cada estado [4] até mesmo construção de uma biblioteca de políticas, onde cada uma representa a solução de um problema anterior [5]. A transferência de políticas se mostra vantajosa por ter poucos pré-requisitos para poder ser transferida: apenas um mapeamento entre os estados das tarefas fonte e destino é necessário [6].

Já quanto à representação, em geral os modelos são simples e com pouca semântica associada, sendo muito atrelados ao problema em questão e tornando assim mais difícil transferir o conhecimento representado dessa maneira. Uma abordagem para tornar possível essa transferência é o uso de uma *representação relacional*, mais rica e generalizada [7]. Ela usa objetos e as relações entre eles para descrever o mundo, permitindo a*bstração*. Essa é uma abstração natural proveniente da representação escolhida e deve ser bem definida pelo projetista. Com ela, não só a descrição do problema como também sua solução pode ser generalizada, de modo a facilitar a sua aplicação em outras situações similares.

Nesse cenário, existem diversos trabalhos que exploram a *abstração de políticas* para transferência de conhecimento, que é o foco também deste trabalho. Um dos primeiros trabalhos a utilizar uma representação relacional com o aprendizado por reforço propôs o algoritmo TILDE [8] que, a partir de diversas experiências usando uma política ótima, induz uma árvore de decisão que representa uma política abstrata, usando os predicados para separar (e abstrair) os estados e apresentando a(s) ação(ões) mais frequentes em suas folhas. Matos *et al.* [9] estendeu esse algoritmo, para que fossem geradas políticas não-determinísticas, e usou essas políticas como conhecimento a ser transferido, mostrando que políticas não-determinísticas se comportam melhor na transferência (por serem mais flexíveis, já que o problema destino não é igual ao fonte). Já Beirigo *et al.* [10] explorou uma abordagem diferente para se construir uma política abstrata. Ao invés de induzí-la a partir de uma política ótima, construiu-a buscando direto nos espaços de estados e ações abstratos, usando uma adaptação do clássico algoritmo de AR, o Q-Learning [11]. Já o algoritmo AbsProb-PI [12] também constrói uma política realizando uma busca direta no espaço abstrato, mas no espaço de políticas e não de estados e ações. Resultados mostraram que essa abordagem é mais efetiva no aprendizado. Isso porque como estados abstratos podem agregar vários estados concretos, estados com diferentes valores são misturados. No entanto, o AbsProb-PI é um

M. L. Koga, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, mlk@usp.br

V. F. da Silva, EACH, Universidade de São Paulo, São Paulo, SP, Brasil, valdinei.freire@usp.br

A. H. R. Costa, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, anna.reali@poli.usp.br

algoritmo de planejamento, baseado em modelo, e não livre de modelo como são os de aprendizado por reforço.

O problema de navegação robótica pode ser descrito como um problema de decisão sequencial, pois é necessário que o agente tome uma série de decisões até que ele atinja seu objetivo (chegar a um lugar especificado). Além disso, é um problema probabilístico, pois o controle de um robô nunca é perfeito, então após executada uma ação, seu resultado é probabilístico. Esse tipo de problema pode ser modelado como um processo de decisão de Markov relacional (RMDP) [13].

O objetivo deste trabalho é o desenvolvimento do AbsProb-RL, um algoritmo de aprendizado por reforço, livre de modelo, para construção políticas estocásticas abstratas para a solução de RMDPs, políticas que generalizam o conhecimento adquirido. Esse algoritmo faz uma busca no espaço de políticas, ele começa com uma política arbitrária e a cada iteração ela é refinada. Além disso, para ser livre de modelo, ele usa técnicas Monte-Carlo para estimar *online* os valores necessários para o refinamento da política, ou seja, as estimativas são feitas a partir das repetidas *interações* do agente com o ambiente. Esse algoritmo é testado em domínios de robótica móvel. A principal motivação de se encontrar políticas no nível abstrato é possibilitar a transferência do conhecimento entre problemas similares, aproveitando-se da natural abstração da representação relacional, atuando apenas sobre classes de objetos e suas relações.

Este trabalho organiza-se da seguinte maneira: a seção II descreve os principais conceitos abordados e necessários para o entendimento do trabalho, enquanto a seção III descreve com mais detalhes as políticas estocásticas abstratas. A seção IV apresenta o algoritmo AbsProb-RL e a seção V mostra os experimentos realizados. Finalmente, na seção VI estão as conclusões finais.

II. Conceitos Fundamentais

Os conceitos básicos envolvidos nesse trabalho – Processos de Decisão de Markov, sua extensão Relacional e Aprendizado por Reforço – são apresentados a seguir.

A. Processo de decisão de Markov (MDP)

Uma abordagem tradicional de formalização para problemas estocásticos de decisão sequencial consiste no uso de Processos de Decisão de Markov (MDP – Markov Decision Process). Um MDP pode ser definido formalmente pela quádrupla $\langle S, A, T, R, G, b^0 \rangle$ [14], onde:

- S é um conjunto finito de estados do ambiente;
- A é um conjunto finito de ações que o agente pode realizar;
- T: $S \times A \times S \rightarrow [0, 1]$ é a função de transição de estado; $T(s_t, a_t, s_t+1)$ define a probabilidade de realizar a transição do estado s_t para o estado s_{t+1} quando se executa a ação a_t em s_t;
- R: $S \times A \times S \rightarrow \mathbb{R}$ é a função de recompensa, com $r_t = R(s_t, a_t, s_{t+1})$;
- $G \subset S$ é um conjunto de estados-meta;
- $b^0 : S \rightarrow [0, 1]$ é a distribuição de probabilidades do estado inicial, indicando a probabilidade do agente iniciar um episódio em cada estado.

O conjunto de ações aplicáveis no estado $s \in S$ é denotado por $A(s)$. Dessa forma, uma transição do estado $i \in S$ para $j \in S$, decorrente da execução de alguma ação $a \in A(i)$, ocorre com probabilidade $T(i, a, j)$, e então uma recompensa $R(i, a, j)$ é recebida. A função recompensa muitas vezes não depende do estado destino, podendo ser descrita como $R(s_t, a_t)$ ou simplesmente depender apenas do estado no qual o agente se encontra, reduzindo-se para $R(s)$. Estamos interessados em MDPs *episódicos*, i.e., quando o agente alcança um estado $s \in G$, o episódio termina e um novo se inicia em algum estado inicial de acordo com b^0.

Assim, no ciclo percepção-ação, o agente aprendiz observa, a cada passo de iteração, o estado corrente s_t do ambiente e escolhe a ação a_t para realizar. Ao executar esta ação a_t, o agente recebe um reforço r_t, penalização ou recompensa, que indica quão desejável é o estado resultante s_{t+1}. A função de transição deve seguir a condição de Markov, i.e., as probabilidades de transição de um estado ao outro dependem exclusivamente do estado atual, e não do histórico de estados passados pelo agente.

Resolver um problema modelado por um MDP consiste em computar uma política π que especifica quais ações $a \in A$ devem ser executadas quando o agente está no estado $s \in S$. Se a política for determinística, tem-se que $\pi: S \rightarrow A$, ou seja, a política π é determinística se leva cada estado em S a uma única ação em A. Dada a política π e um fator de desconto $\gamma \in [0, 1]$ que desconta recompensas futuras, a função de valor de estado V^π: $S \rightarrow \mathbb{R}$, onde:

$$V^\pi(s) = E\left[\sum_{t=0}^{\infty} (\gamma^t r_t | \pi, s_0 = s)\right] \forall s \in S \qquad (1)$$

representa o valor de estar em um estado seguindo a política π, isto é, representa as recompensas esperadas. A política ótima π^* é a política que satisfaz a condição: $V^{\pi^*} \geq V^{\pi'}$, $\forall s \in S$ e $\forall \pi'$. A função valor ótima é denotada por V^*. Assim, o interesse consiste em computar uma política que melhor aproxime, ou, idealmente, iguale a política ótima π^*. Uma forma possível de resolver um MDP é utilizando um algoritmo de aprendizado por reforço. Para um MDP, sabe-se que uma política determinística é suficiente para otimalidade [14].

A descrição do MDP dada prevê que o agente possui observabilidade total, ou seja, ele sempre sabe em que estado se encontra e portanto, de acordo com a condição de Markov, a política ótima depende apenas do estado atual. Entretanto, existem muitos casos nos quais o ambiente é parcialmente observável, nos quais o agente não tem certeza de qual estado ele se encontra. Podemos definir problemas desse tipo como POMDPs (Processos de Decisão de Markov Parcialmente Observáveis). Os POMDPs são definidos tais quais os MDPs com o acréscimo de uma função O(s,o) que indica a probabilidade de se perceber a observação o num estado s. Assim sendo, num POMDP, ao realizar uma obervação, ao

invés do agente determinar qual o seu estado corrente, ele determina um estado de crença, que é uma distribuição de probabilidades sobre todos os estados possíveis. Na sua interação com o ambiente o agente percebe uma observação o_t, atualiza seu estado de crença indicando a probabilidade dele estar de fato em s_1, s_2, etc., executa então uma ação a_t (a transição $T(s_t,a_t,s_{t+1})$ ocorre de acordo com a função T inerente ao problema, ainda que o agente desconheça s_t e s_{t+1}) e uma uma recompensa r_t e uma nova observação o_{t+1} são percebidas pelo agente. Em POMDPs complexos pode ser difícil de se encontrar políticas ótimas aproximadas, na realidade trata-se de um problema PSPACE-difícil [15].

Pode-se utilizar diversas formas de representação dos estados e ações em um (PO)MDP. A representação relacional, descrita a seguir, é uma delas e é bastante efetiva por possibilitar abstrações do problema.

B. Processo de decisão de Markov Relacional (RMDP)

MDPs são essencialmente proposicionais, uma vez que cada estado tem que ser representado usando uma proposição separada, limitando sua expressividade por não conseguir representar adequadamente a estrutura embutida no problema e, assim, dificultando a generalização de políticas para diversos domínios com propriedades similares. Para aumentar a expressividade da representação proposicional, pode-se usar uma representação relacional, que permite evidenciar relações entre objetos e usar variáveis.

Um vocabulário relacional Σ é um conjunto de constantes e predicados. Na representação relacional, expressões da forma $p(t_1, t_2,..., t_m)$ são átomos, com p sendo um predicado de relação entre os termos t_i. Um termo pode consistir em uma variável (iniciando com letra maiúscula) ou uma constante (iniciando com letra minúscula, como c_1). Um conjunto de átomos forma uma conjunção. Átomos ou conjunções que não possuem variáveis são ditos *concretos.* Toda conjunção é considerada existencialmente quantificada. Uma base de Herbrand HB_Σ é o conjunto de todos os possíveis átomos concretos que podem ser formados com os elementos de Σ.

Com base neste conceito, foi proposto o MDP relacional, RMDP [13,[16]], definido pela tupla $\langle \Sigma, B, T, R, G, b^0 \rangle$, onde:

- $\Sigma = C \cup P_S \cup P_A$ é um conjunto de constantes C que representam os objetos do ambiente; predicados P_S usados para descrever as propriedades e relações entre objetos; e predicados P_A usados para descrever as ações.
- B é um conjunto de sentenças de lógica de primeira-ordem que representam uma base de conhecimento do domínio, usadas para restringir os estados (e ações) possíveis;
- Um conjunto de estados S é definido como o subconjunto do conjunto de todas as possíveis conjunções sobre os átomos da $HB_{P_S \cup C}$ que satisfaz as restrições de B;
- O conjunto de ações A um o subconjunto de $HB_{P_A \cup C}$, novamente satisfazendo B;
- T, R, G e b^0 são definidos como anteriormente, no MDP.

Conjunções de átomos lógicos representam estados. Assim, um exemplo de estado $s \in S$ seria:

$$s \equiv inRoom(r1) \wedge seeAdjRoom(r2)$$

indicando que o robô está na sala *r1*, a qual é conectada a sala *r2*, pois consegue enxergá-la. Estamos assumindo a condição de mundo fechado, i.e., todos os predicados omitidos da conjunção são falsos.

Da mesma forma, define-se o conceito de estados abstratos como conjunções de átomos lógicos que contêm variáveis no lugar de todas as constantes. Por exemplo, o estado abstrato

$$\sigma \equiv inRoom(R) \wedge seeAdjRoom(R')$$

representa estados nos quais o robô está na sala R, a qual é conectado a uma outra sala R'. Neste caso, o estado s seria uma instância do estado abstrato σ. Uma substituição θ atribui termos às variáveis de forma que s seja um dos possíveis estados concretos descritos por σ. Assim, para o exemplo acima, a instância s seria alcançada com a substituição $\theta = \{R/r1, R'/r2\}$. Um estado é denominado concreto se não contiver variáveis. Note que um estado abstrato pode representar mais de um estado concreto. Outro estado concreto s_2 alcançaria σ com a substituição $\theta_2 = \{R/r3, R'/r4\}$. Ao conjunto de todos os estados concretos possíveis de serem alcançados a partir do estado abstrato σ de S_σ.

Analogamente, ações abstratas podem ser representadas por átomos tendo variáveis como argumentos. Assim, $goToRoom(R)$ é uma ação abstrata que indica que o robô navega para uma sala R, e $goToRoom(r1)$ é uma ação concreta com a substituição $\theta = \{R/r1\}$. A_α é o conjunto de ações concretas abstraidas por uma ação abstrata α.

Dadas essas definições de estados e ações abstratos, define-se como S_{ab} o conjunto de todos os estados abstratos e A_{ab} o conjunto de todas as ações abstratas.

A tarefa de um agente para resolver um RMDP também é encontrar uma política, tal qual no MDP. No RMDP, pode-se também considerar apenas os estados e ações abstratas (S_{ab} e A_{ab}) para a solução. Nesse caso, diz-se que resolvemos um RMDP abstrato, e a sua solução será uma *política abstrata*. Além disso, ao invés de políticas abstratas determinísticas, aqui usamos políticas abstratas estocásticas. O uso de políticas abstratas estocásticas é explicado com mais detalhes na seção III.

C. Aprendizado por Reforço

Aprendizado por reforço é aprender o que fazer – i.e., mapear estados a ações – maximizando uma função reforço numérica. O agente não é ensinado sobre quais ações tomar, como a maioria das formas de aprendizado supervisionado, mas deve descobrir quais ações resultam em maiores somas de recompensas através de tentativas. Nos casos mais interessantes, as ações não afetam apenas a recompensa imediata, mas também os estados futuros e consequentemente as recompensas futuras. Essas duas características: tentativa-e-erro e recompensa futura são as mais importantes do aprendizado por reforço [1].

O aprendizado por reforço é, portanto, uma das maneiras de se encontrar a política ótima de um MDP, sem a necessidade de possuir o modelo completo dele, necessitando

apenas interagir com o ambiente e receber seus reforços mediante as ações tomadas. Por modelo completo, entende-se que o agente conhece *S, A, T, R, G* e b^0. No aprendizado por reforço, o agente pode desconhecer *T* e/ou *R*.

No aprendizado por reforço relacional, as técnicas de aprendizado são modificadas para tratar de RMDPs ao invés de MDPs. A principal diferença é que cada estado (inclusive abstrato) não é mais atômico e sim uma estrutura relacional, que permite o uso de variáveis e, assim, um estado abstrato pode representar diversos estados concretos [13].

III. Políticas Abstratas Estocásticas

Um RMDP é definido, portanto, com descrição relacional de estados e ações. Seja S_{ab} o conjunto de todos os estados abstratos e A_{ab} o conjunto de todas as ações abstratas, define-se uma política abstrata estocástica como $\pi_{ab}: S_{ab} \times A_{ab} \rightarrow [0,1]$, com $p(\alpha_i|\sigma) = \pi(\sigma, \alpha_i)$, $\sigma \in S_{ab}$, $\alpha_i \in A_{ab}^{\sigma}$, $A_{ab}^{\sigma} \subseteq A_{ab}$, e A_{ab}^{σ} é o conjunto de ações abstratas permitidas no estado abstrato σ. Ou seja, numa política abstrata estocástica, cada par estado abstrato-ação abstrata possui uma probabilidade associada e dado um estado abstrato, a soma das probabilidades de cada ação deve ser 1. Vale ressaltar que aqui estamos sempre falando de políticas sem memória, ou seja, somente o estado atual determina a ação a ser tomada, e não uma história.

Construir uma política abstrata para resolver um RMDP abstrato assemelha-se a se resolver um POMDP, pois em ambos os casos queremos encontrar as melhores ações para se executar em cada *agregação* de estados. Um estado abstrato é uma agregação de vários estados concretos, assim como uma observação parcial pode refletir vários estados também.

Levando em conta essa semelhança, políticas estocásticas são mais adequadas que determinísticas. As políticas determinísticas são um caso particular das estocásticas, onde para cada estado uma das ações possui probabilidade 1 e todas as outras 0. O fato é que uma política estocástica pode ser arbitrariamente melhor que uma determinística [17].

Para ilustrar essa afirmação, vamos analisar um simples exemplo. Considere um ambiente discreto com seis células dispostas em linha, como mostra a Fig. 1, com a meta na célula 4. Cada célula representa um estado e o agentepode realizar as seguintes ações: ir para a direita, para a esquerda ou ficar parado. A política determinística ótima para esse problema seria: se o agente estiver nas células 1,2 ou 3, ir para direita; se estiver na célula 4, parar; e se estiver nas células 5 ou 6, ir para a esquerda.

Considere que o agente possui apenas uma observação parcial do ambiente, estando apto apenas a perceber o que está imediatamente ao seu redor. Ou seja, ele tem as seguintes percepções:

- *Percepção na célula 1* = [parede acima, parede a esquerda, porta a direita, parede abaixo, não está na meta];
- *Percepção na célula 2* = [parede acima, porta a esquerda, porta a direita, parede abaixo, não está na meta];
- *Percepção na célula 3* = [parede acima, porta a esquerda, porta a direita, parede abaixo, não está na meta];
- *Percepção na célula 4* = [parede acima, porta a esquerda, porta a direita, parede abaixo, está na meta];
- *Percepção na célula 5* = [parede acima, porta a esquerda, porta a direita, parede abaixo, não está na meta];
- *Percepção na célula 6* = [parede acima, porta a esquerda, parede a a direita, parede abaixo, não está na meta].

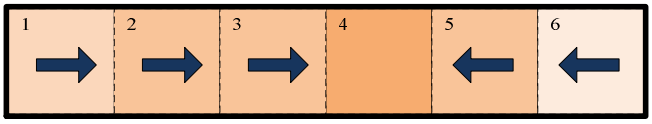


Figura 1 - Um ambiente discreto simples.

Com essas percepções, o agente não consegue distinguir os estados 2,3 e 5. Na realidade, o agente percebe apenas quatro situações distintas:

- *Situação 1* = σ_1 = [parede acima, parede a esquerda, porta a direita, parede abaixo, não está na meta];
- *Situação 2* = σ_2 = [parede acima, porta a esquerda, porta a direita, parede abaixo, não está na meta];
- *Situação 3* = σ_3 = [parede acima, porta a esquerda, porta a direita, parede abaixo, está na meta];
- *Situação 4* = σ_4 = [parede acima, porta a esquerda, parede a direita, parede abaixo, não está na meta].

Essas situações poderiam ser também quatro estados abstratos, dada a abstração adequada. Dadas essas quatro situações, qual política determinística resulveria (mesmo que subotimamente) o problema? Se o agente se encontrar na situação 2, nem ir para a direita nem ir para a esquerda resolve o problema para qualquer estado inicial. Nesse caso, uma política estocástica pode resolver, mesmo que subotimamente. Uma política estocástica abstrata possível seria:

- Se em σ_1, ir para direita com probabilidade 1;
- Se em σ_2, ir para direita com probabilidade 0.66 e ir para esquerda com probabilidade 0.33;
- Se em σ_3, parar com probabilidade 1;
- Se em σ_4, ir para esquerda com probabilidade 1.

Assim, essa política estocástica apresenta um valor médio de cada estado (considerando todos os estados igualmente prováveis) maior que qualquer política determinística.

Definida a política estocástica abstrata, a pergunta que surge é: como aplicar uma política estocástica abstrata π_{ab} a um problema concreto? O método utilizado por este trabalho é descrito a seguir. Quando o agente observa um estado $s \in S$, encontra-se o estado abstrato σ tal que $s \in S_{\sigma}$. Então, a política π_{ab} fornece um conjunto de ações abstratas, com uma probabilidade associada a cada ação. Uma ação abstrata α é escolhida probabilisticamente desse conjunto. Note que uma ação abstrata também pode ser mapeada para uma série de ações concretas. Uma ação concreta a é então escolhida aleatóriamente (com distribuição uniforme) do conjunto A_{α}, numa operação chamada de $concretizacao(\pi_{ab}(\sigma, \alpha))$. Essa

ação *a* é aplicada, um novo estado *s'* é observado e o processo se repete.

Um algoritmo para se construir uma política estocástica abstrata é detalhado na próxima seção.

IV. Proposta de Algoritmo

Para se construir uma política abstrata, que atua sobre uma abstração de estados e ações, uma busca direta no espaço de políticas é mais adequada [18]. Um dos mais famosos algoritmos com busca no espaço de políticas é o *Policy Iteration*, algoritmo de programação dinâmica para se resolver MDPs quando se possui o modelo completo do problema. O Policy Iteration possui basicamente dois passos, que se alternam iterativamente: avaliação e melhoramento da política atual. A avaliação da política é o cálculo do valor de cada estado (Eq. 1) e o melhoramento é a escolha da ação que vai maximizar o valor de cada estado (Eq. 2).

$$\pi_{i+1}(s) = \underset{a \in A}{argmax} \left\{ r(s) + \gamma \sum_{s' \in S} T(s,a,s') V^{\pi_i}(s') \right\} \forall s \in S \qquad (2)$$

O AbsProb-PI [12] é a adaptação desse algoritmo para o nível abstrato, sendo necessárias, obviamente, algumas mudanças. Ele deve aprender uma política abstrata probabilística π_{ab}. Para o passo de melhoramento da política, foi necessário redefinir a função de transição de estados (Eq. 3), definindo-a em dependência da política em análise:

$$T^{\pi_{ab}}(s,s') = \sum_{\alpha \in A_{ab}} \pi_{ab}(\alpha|\sigma_s) \sum_{a \in A_\alpha} \frac{1}{|A_\alpha|} T(s,a,s') \qquad (3)$$

onde A_{ab} é o conjunto de ações abstratas, A_α é o conjunto de ações concretas abstraídas pela ação abstrata α e σ_s é o estado abstrato que contém o estado concreto *s* e $\pi_{ab}(\alpha|\sigma_s)$ é o mesmo que $\pi_{ab}(\sigma_s,\alpha)$. Note que a função $T^{\pi_{ab}}$ apresenta as probabilidades de transição de cada estado s para cada estado s' quando se executa a política π_{ab}, i.e., a probabilidade de cada ação α ser executada pela política é multiplicada pela média das probabilidades de transição possiveis com ações subsumidas por α e então as probabilidades de cada ação são somadas.

Dadas essas definições, o algoritmo AbsProb-PI calcula o gradiente da função valor *V*, que indica a melhor direção (ação abstrata) para se mudar a política [19]. Inicialmente, uma política abstrata πab é arbitrariamente inicializada. Então o algoritmo refina iterativamente essa política, até alcançar algum critério de parada. Uma iteração do algoritmo, cujos cálculos estão na forma matricial e portanto usa representações vetoriais das funções, está detalhada no Algoritmo I, na qual: $\boldsymbol{I}$ é a matriz identidade; $\boldsymbol{T}^{\pi_{ab}}$ é a representação vetorial de $T^{\pi_{ab}}(s,s')$; $\boldsymbol{b^0}$ é a representação vetorial de $b^0(s)$, distribuição inicial de probabilidades de ocorrência de cada estado; $\boldsymbol{R}$ é a representação vetorial da função recompensa *R(s)*; S_{ab} é o conjunto de estados abstratos e $\boldsymbol{T}^{\pi_\alpha}$ é uma matriz de transições definida similarmente a $\boldsymbol{T}^{\pi_{ab}}$, mas cuja política escolhe uma ação abstrata α sempre.

Algoritmo I
Uma iteração do AbsProb-PI

1. Calcule a função valor $\boldsymbol{V}^{\pi_{ab}} = (\boldsymbol{I} - \gamma \boldsymbol{T}^{\pi_{ab}})^{-1} \boldsymbol{R}$;
2. Calcule o produto $\boldsymbol{C} = \gamma \boldsymbol{b}^0 (\boldsymbol{I} - \gamma \boldsymbol{T}^{\pi_{ab}})^{-1}$;
3. Para cada ação $\alpha \in A_{ab}$, calcule:
$$\Delta^{\alpha,\pi_{ab}} = (\boldsymbol{T}^{\pi_\alpha} - \boldsymbol{T}^{\pi_{ab}}) \boldsymbol{V}^{\pi_{ab}};$$
4. Para cada $\sigma \in S_{ab}$ e $\alpha \in A_{ab}$, calcule:
$$G(\sigma,\alpha) = \sum_{s \in S_\sigma} C(s) \Delta^{\alpha,\pi_{ab}}(s);$$
5. Para cada $\sigma \in S_{ab}$ ache a melhor direção
$$\alpha_\sigma^* = arg\ max_{\alpha \in A_{ab}} G(\sigma,\alpha)$$
6. Escolha um tamanho de passo δ e atualize a política:
$$\pi_{ab}(\sigma,\alpha) \leftarrow \begin{cases} (1-\delta)\pi_{ab}(\sigma,\alpha) + \delta\left(\frac{\varepsilon}{|A_{ab}|}(1-\varepsilon)\right), se\ \alpha \neq \alpha_\sigma^* \\ (1-\delta)\pi_{ab}(\sigma,\alpha) + \delta\frac{\varepsilon}{|A_{ab}|}, se\ \alpha = \alpha_\sigma^* \end{cases}$$

O passo 1 corresponde ao passo de avaliação da política abstrata atual, calculando seu valor. Ele é o equivalente do AbsProb-PI à equação 1 do Policy Iteration. O produto $\boldsymbol{C}$ (passo 2) contém a probabilidade de ocorrência de cada estado, considerando a política π_{ab} e a distribuição inicial, pois a política será avaliada baseada na distribuição inicial dos estados. Em seguida, calcula-se a diferença no valor de cada estado que uma política que usa principalmente a ação α causaria, e isso é feito para cada ação abstrata existente (passo 3). No passo 4, calcula-se o *G* para cada par estado-ação abstratos, somando o produto de *C(s)* e $\Delta^{\alpha,\pi_{ab}}(s)$ somente dos estados subsumidos pelo σ da vez.

O valor *G*, calculado no passo 4, representa justamente o gradiente da função valor, cujo valor máximo indica qual é a ação abstrata cuja probabilidade de ocorrência deve ser aumentada na política para a obtenção de um melhor resultado (passo 5). A política é então atualizada nessa direção, usando um tamanho de passo δ (passo 6), que pode variar a cada iteração.

Esse algoritmo é baseado em modelo, necessitando possuir um modelo completo do problema para funcionar. A contribuição deste trabalho é apresentar o AbsProb-RL, uma versão livre de modelo para o AbsProb-PI. Para deixar esse algoritmo livre de modelo, é necessário avaliar a política (passo 1) e calcular *G* sem o conhecimento da função de transição *T* e reforço *R* completas (passos 2, 3 e 4). Para que isso seja possível, o AbsProb-RL usa técnicas *Monte-Carlo* para estimar os valores desconhecidos, semelhante ao que já foi feito para se encontrar políticas em POMDPs [20], [21].

O uso de Monte-Carlo para se estimar implica que o agente deve realizar diversas interações com o ambiente e o conjunto de experiências resultante dessas interações permite que ele estime os valores necessários. Isso significa que, para cada iteração do AbsProb-PI (passos de avaliação e melhoramento), o agente deve realizar diversos episódios com a política atual, um número suficiente para que ele possa estimar o valor dela para então sim poder realizar um passo de melhoramento.

O AbsProb-RL encontra-se descrito no Algoritmo 2. A ideia básica é a mesma do Policy Iteration: ele começa com uma política π_{ab} (passo 1) e a cada iteração deve avaliá-la e

melhorá-la. Para poder avaliá-la e calcular o gradiente do valor, um número $H(i)$ de episódios deve ser observado. Este número pode ser variável de acordo com a iteração e diversas funções H são testadas na seção V. Repare que, diferentemente do AbsProb-PI, este algoritmo é baseado na interação do agente com o ambiente (passos 5 a 7). Ao final de cada episódio, a estimativa $\hat{G}$ é atualizada (passo 9) com o valor G_e calculado em cada episódio (passo 8), seguindo uma taxa de aprendizado $\mu(i,h)$. Essa taxa pode ser fixa ou variável e diversos valores para a função μ também são explorados na seção V.

Após a execução de $H(i)$ episódios, a política é atualizada (passos 10 e 11) tal qual no AbsProb-PI, porém usando a estimativa $\hat{G}$ do gradiente. E então uma nova iteração se inicia, repetindo-se o processo.

ALGORITMO II
ABSPROB-RL

1. Inicialize arbitrariamente π_{ab}
2. **Para** cada iteração *i*
3. **Para** cada episódio $h \in H(i)$
4. **Para** cada passo *t* do episódio *h*
5. Observe *s* e encontre σ tal que $s \in S_\sigma$
6. Escolha ação $a = concretizacao(\pi_{ab}(\sigma,\alpha))$
7. Execute *a*, receba *r* e observe *s'*
8. G_e = *AtualizaEstimativaG*
9. $\hat{G} = \hat{G} + \mu(i,h)(G_e - \hat{G})$
10. Para cada $\sigma \in S_{ab}$ ache a melhor direção
$$\alpha_\sigma^* = arg\ max_{\alpha \in A_{ab}} \hat{G}(\sigma,\alpha)$$
11. Atualize a política:
$$\pi_{ab}(\sigma,\alpha) \leftarrow \begin{cases} (1-\delta)\pi_{ab}(\sigma,\alpha) + \delta\left(\frac{\varepsilon}{|A_{ab}|}(1-\varepsilon)\right), se\ \alpha \neq \alpha_\sigma^* \\ (1-\delta)\pi_{ab}(\sigma,\alpha) + \delta\frac{\varepsilon}{|A_{ab}|}, se\ \alpha = \alpha_\sigma^* \end{cases}$$

Falta definir o procedimento *AtualizaEstimativaG* (passo 8), que contém justamente como Monte-Carlo é usado para se estimar o valor do gradiente *G*. Para o cálculo de *G*, são necessárias estimativas de $\boldsymbol{V}^{\pi_{ab}}$, $\boldsymbol{T}^{\pi_{ab}}$, $\boldsymbol{T}^{\pi_\alpha}$, para toda ação $\alpha \in A_{ab}$, e $\boldsymbol{C}$. Basicamente, Monte Carlo usa todas as experiências do agente para estimar cada um desses valores. Uma experiência no passo *t* consiste dos valores $\langle s_t, a_t, r_t, s_{t+1}\rangle$ e um conjunto desses valores gera as estimativas necessárias.

$\boldsymbol{V}^{\pi_{ab}}$ é estimado conforme a Eq. 1, só que com tempo finito ao invés de infinito.

$$\boldsymbol{T}^{\pi_{ab}}(s,s') = p(s_{t+1} = s' | s_t = s, \pi_{ab}), \quad (4)$$

ou seja, é a probabilidade de transição entre estados concretos dado que a política π_{ab} é seguida. No caso das experiências do agente, a política π_{ab} é sempre a política seguida. Então essas probabilidades são estimadas através da contagem simples de todas as transições que o agente experimentar. Analogamente, para cada ação abstrata $\alpha \in A_{ab}$, temos

$$\boldsymbol{T}^{\pi_\alpha}(s,s') = p(s_{t+1} = s' | s_t = s, a_t \in A_\alpha), \quad (5)$$

e a cada passo, uma das funções $\boldsymbol{T}^{\pi_\alpha}$ tem sua estimativa atualizada, dependendo da ação que foi realizada naquele passo. E finalmente, o produto $\boldsymbol{C}$, que tem sua definição reproduzida novamente aqui:

$$\boldsymbol{C} = \gamma \boldsymbol{b}^0 (\boldsymbol{I} - \gamma \boldsymbol{T}^{\pi_{ab}})^{-1}. \quad (6)$$

Sabemos que:

$$(\boldsymbol{I} - \gamma \boldsymbol{T}^{\pi_{ab}})^{-1} = I + \gamma \boldsymbol{T}^{\pi_{ab}} + \gamma^2 (\boldsymbol{T}^{\pi_{ab}})^2 + \cdots = \sum_{t=0}^{\infty} \gamma^t (\boldsymbol{T}^{\pi_{ab}})^t \quad (7)$$

E também podemos dizer que a distribuição de probabilidades iniciais de cada estado multiplicada pela função de transição *t* vezes representa a probabilidade de ocorrência de cada estado no passo *t:*

$$\boldsymbol{b}^0 (\boldsymbol{T}^{\pi_{ab}})^t (s) = p(s_t = s | \pi_{ab}) \quad (8)$$

Então, podemos definir $C(s)$ como a probabilidade de ocorrência de cada estado *s* dada a política π_{ab}, descontados pelo fator de desconto γ. E essa probabilidade é facilmente estimada através do conjunto de experiências do agente

$$C(s) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s | \pi_{ab}) \quad (9)$$

Assim sendo, o procedimento *AtualizaEstimativaG* atualiza as estimativas de todas as probabilidades descritas, gerando ao final de cada episódio uma estimativa :

$$G_e(\sigma,\alpha) = \sum_{s \in S_\sigma} C(s) \sum_{s' \in S} \left(T^{\pi_\alpha}(s,s') - T^{\pi_{ab}}(s,s')\right) V^{\pi_{ab}}(s') \quad (10)$$

Esse é o algoritmo AbsProb-RL, que usa Monte-Carlo para transformar o AbsProb-PI num algoritmo livre de modelo e com aprendizado *online.* A sua efetividade é testada na seção V, na qual ele é submetido a uma série de experimentos.

V. EXPERIMENTOS

Nesta seção os experimentos realizados são descritos. Todos os experimentos foram feitos num ambiente simulado de navegação robótica.

A. Domínio da navegação robótica

Modelamos o ambiente de navegação robótica dividindo-o em regiões discretas que são descritas unicamente com um conjunto de predicados e um conjunto de objetos. Cada uma dessas regiões representa um estado. Elas são divididas em salas e corredores e os predicados $inRoom(ri)$ e $inCorridor(ci)$ indicam se o agente se encontra numa sala ou corredor, respectivamente. Com um alcance de visão de duas células, o agente pode também ver: salas adjacentes, corredores adjacentes, portas, marcadores (perto ou longe) e espaços vazios; e os predicados $seeAdjRoom(ri)$, $seeAdjCorridor(ci)$, $seeDoor(di)$, $seeMarkerNear(mi)$,

$seeMarkerFar(mi)$ e $seeEmptySpace$ indicam, respectivamente, se o que o agente pode ver numa região. O agente observa salas ou corredores adjacentes somente se está ao lado de uma porta.

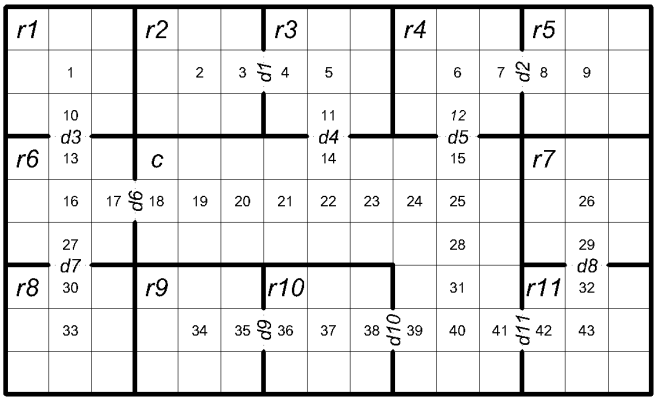


Figura 2 - Ambiente de navegação robótica. Linhas grossas representam paredes e as células numeradas são estados.

Finalmente, o agente também possui a habilidade de sentir a direção e distância para a meta com três predicados: $nearGoal$, $appGoal(X)$, $awayGoal(X)$. O primeiro indica se o agente está a uma distância Manhattan de no máximo 5 da meta; os outros indicam se o objeto X que o agente está vendo está mais perto da meta ($appGoal$) ou mais longe ($awayGoal$) relativamente ao próprio agente. Os objetos podem ser salas(ri), corredores(ci), portas(di) e marcadores(mi), $i \in \mathbb{N}$.

Todos os experimentos usam um mapa com 43 regiões (estados concretos). A Fig. 2 mostra esse mapa, e o estado-meta pode ser escolhido entre qualquer estado dentro de uma sala. Por exemplo, considere a meta no estado 16. O estado 12 seria descrito como:

$$s_{12} = inRoom(r4) \wedge seeAdjCorridor(c) \wedge appGoal(c) \wedge seeEmptySpace.$$

O estado abstrato correspondente,

$$\sigma_1 = inRoom(X) \wedge seeAdjCorridor(Y) \wedge appGoal(Y) \wedge seeEmptySpace,$$

também engloba os estados 11 e 42, i.e., $S_{\sigma_1} = \{11,12,42\}$. Dependendo da posição da meta, o conjunto de predicados descreve diferentemente cada estado, gerando abstrações diferentes. Os experimentos foram rodados considerando as seguintes metas: 5, 16, 34 e 43.

As ações são descritas pelos predicados:

$P_A = \{gotoDoorAppGoal(di), gotoRoomAppGoal(ri), goToCorridorAppGoal(ci), goToMarkerAppGoal(mi), goToEmptyAppGoal, gotoDoorAwayGoal(di), gotoRoomAwayGoal(ri), goToCorridorAwayGoal(ci), goToMarkerAwayGoal(mi), goToEmptyAwayGoal\}$

e seus nomes são autoexplicativos, indicando que o agente deve andar em direção ao objeto especificado.

A distribuição de probabilidades para estado inicial é uniforme para todos os estados.

B. Resultados

Para averiguar o funcionamento do AbsProb-RL, realizamos experimentos no ambiente de navegação robótica, sempre rodando 100000 episódios, sendo que cada episódio possui um número máximo de 500 passos. O tamanho de passo δ usado foi fixo, com $\delta = 0{,}05$. Avaliamos então o valor da política π resultante a cada episódio, sendo o valor da política V^{π} definido como na Eq. 11.

$$V^{\pi} = \sum_{s \in S} b^0(s) \left\{ E\left[\sum_{t=0}^{\infty} (\gamma^t r_t | \pi, s_0 = s) \right] \right\} \quad (11)$$

No primeiro experimento, usamos uma taxa de aprendizado $\mu = 1/h$. Isso significa que a estimativa $\hat{G}$ é a média dos $H(i)$ episódios de cada iteração i do AbsProb-RL e que ela é reinicializada a cada iteração. Isso porque o valor de G depende da política atual, então a cada vez que essa política é mudada (ou seja, a cada iteração) a estimativa deve ser refeita. A função $H(i) = 20i$ indica que a política é atualizada a cada n episódios e esse número aumenta de 20 em 20. Ou seja, a primeira atualização ocorre após 20 episódios, a segunda após mais 40 (novos) episódios, e assim por diante. Esse número é crescente para que o agente possa ter estimativas melhores dos valores com o passar do tempo, pois com o passar do tempo é cada vez mais difícil também encontrar a direção para melhorar a política. Três valores diferentes foram testados para a função H: incremento de 20, 50 e 100.

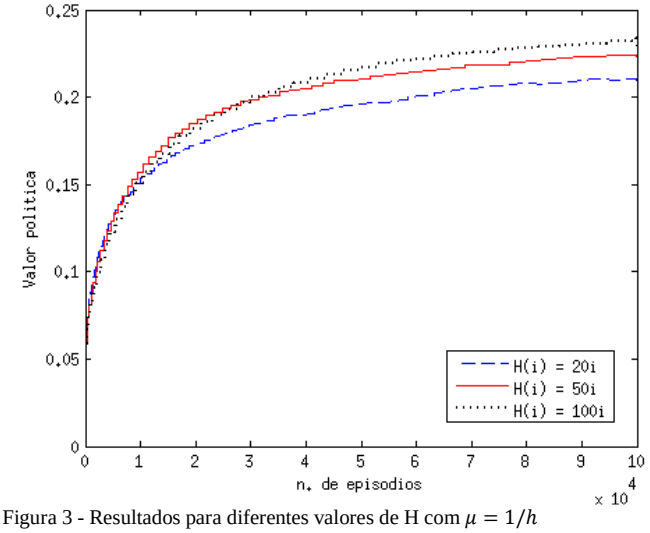


Figura 3 - Resultados para diferentes valores de H com $\mu = 1/h$

A Fig. 3 ilustra os resultados do primeiro experimento. Foram consideradas quatro diferentes metas e a média de 10 execuções para cada uma é mostrada. Percebe-se que valores menores no incremento do número de episódios apresentam um aprendizado ligeiramente mais rápido no início, mas que acaba sendo mais lento após certo ponto. H_3 tem início mais lento que H_1, mas após 10000 episódios o valor de sua política cresce mais rápido. H_2 se encontra entre essas duas curvas. Entretanto, nenhuma das curvas chegou ao valor ótimo de $V^{\pi*} = 0.30$, calculado com o AbsProb-PI após os 100000 episódios.

Para tentarmos melhorar o desempenho do AbsProb-RL, o segundo experimento investigou diferentes formas para a taxa de aprendizado μ. A primeira forma é a que foi usada no experimento anterior, a *média por iteração*. Uma alternativa que mostrou ser bem mais eficiente, é usar uma *média global*, e não por iteração. Nesse caso, $\mu = \frac{1}{ih}$. Isso significa que a estimativa dos valores das políticas das iterações passadas não é esquecida; apesar de cada iteração possuir uma política diferente, as políticas são similares entre si, então uma estimativa já inicializada se mostra ser melhor que uma inicialização em zero.

A Fig. 4 ilustra os resultados deste segundo experimento, desta vez com a média de 10 execuções para apenas uma única meta (estado 16) e com $H(i) = 50i$. A média por iteração apresenta resultado similar ao experimento anterior, não chegando ao valor ótimo. Já a média global teve um desempenho muito melhor, alcançando eventualmente valores muito próximos ao valor ótimo $V^{\pi*} = 0.362$.

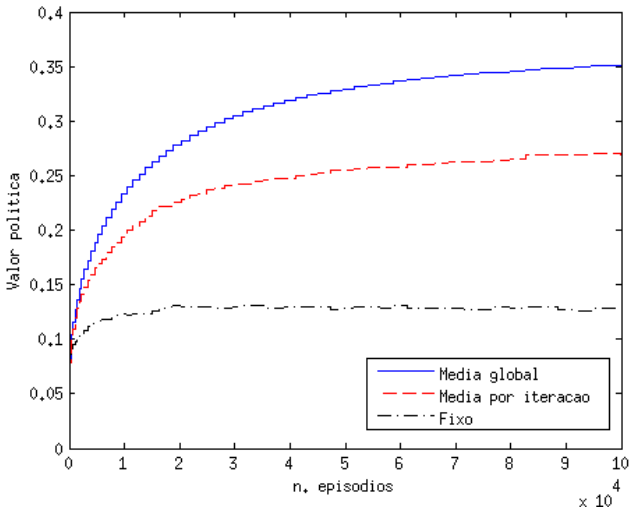


Figura 4 - Resultados para diferentes valores da taxa de aprendizado.

Visto que $\mu = 1/ih$ apresentou melhor resultado, o terceiro experimento fixa essa taxa de aprendizado para novamente avaliar a variação do tamanho da janela de episódios *H*. Além das funções H já definidas anteriormente, como a média global já carrega algum conhecimento prévio, foram testados também valores fixos para H: a política é sempre atualizada a cada 20 ou 50 episódios. As médias de 20 execuções para a tarefa de chegar na meta 16 são mostradas na Fig. 5.

Todas as curvas convergem para o mesmo ponto após os 100000 episódios, mas aqui estamos mostrando apenas até o episódio 50000 para uma melhor visualização. Nota-se que as curvas que possuem *H* crescente sempre crescem também. Isso porque a cada atualização da política, mais episódios são usados para a estimativa do gradiente e portanto ela é mais apurada, garantindo que o passo de melhoramento da política sempre tenha a direção correta (como sempre acontece no algoritmo baseado em modelo).

Já com *H* fixo, a estimativa do gradiente é sempre feita com o mesmo número de episódios, não ficando mais apurada com o tempo. Por essa razão, as curvas com *H* fixo são bastante ruidosas, pois nem sempre a direção correta é a escolhida. Apesar disso, pelo fato de *H* não aumentar, muito mais passos de atualização são feitos e no geral vemos que as curvas apresentaram sempre um valor superior as versões incrementais. Vale notar também que a curva $H(i) = 50$ possui menos quedas que a curva $H(i) = 20$, pois faz uma estimativa melhor do gradiente.

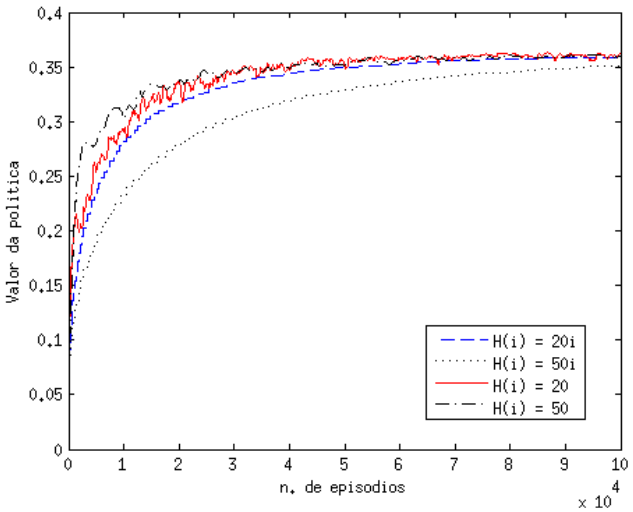


Figura 5 - Resultados para diferentes valores de H com $\mu = 1/ih$

VI. CONCLUSÃO

Diante do exposto, pode-se concluir que é possível construir uma política estocástica abstrata de modo *online*, livre de modelo. Este trabalho apresentou um novo algoritmo, AbsProb-RL, que permite a construção de uma política nesses moldes através de técnicas Monte Carlo para o cálculo do gradiente da função valor de uma política. Resultados mostram que o AbsProb-RL efetivamente alcança resultados similares ao algoritmo equivalente dependente de modelo, ainda que o processo possa ser lento. Conclui-se também que para se estimar os valores do gradiente, é mais efetivo usar valores similares já conhecidos para acelerar essa estimativa.

Uma vantagem de se obter uma política abstrata, já demonstrada em trabalhos anteriores, é acelerar o aprendizado de uma nova tarefa, através da transferência de conhecimento. Trabalhos futuros devem integrar o AbsProb-RL com a transferência de conhecimento, possibilitando assim um aprendizado *online* e simultâneo de ambas políticas concreta e abstrata.

AGRADECIMENTOS

Agradecemos o apoio da FAPESP (proc. n. 12/02190-9, proc. n. 11/19280-8) e da CNPq (proc. n. 311058/2011-6).

REFERÊNCIAS

[1] R.S. Sutton and A.G. Barto.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA. 1998.

[2] M. E. Taylor and P. Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research* 10, p. 1633-1685, 2009.

[3] L. Torrey, J. Shavlik, T. Waker and R. Maclin. *Relational macros for transfer in reinforcement learning*. 2008. In: *Proc. of the 17th Int. Conf. on Inductive Logic Programming (ILP '07)*, 2008.

[4] M. E. Taylor, P. Stone and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, vol. 8, n. 1, p. 2125-2167, 2007.

[5] F. Fernández, F. and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proc. of the 5th international joint conference on Autonomous agents and multiagent systems (AAMAS 2006)*, p.720-727, 2006.

[6] F. Fernández, F. and M. Veloso. Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems*, vol. 58, n. 7, p. 866-871. Elsevier, 2010.

[7] E. F. Morales. Scaling up reinforcement learning with a relational representation. In *Proc. of the Workshop on Adaptability in Multi-agent systems (AORC 2003)*, p. 15-26, 2003.

[8] H. Blockeel and L. D. Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, vol. 101, n. 1-2, p. 285-297, 1998.

[9] T. Matos, Y. Bergamo, V. F. da Silva and A. H. R. Costa. Stochastic Abstract Policies for Knowledge Transfer in Robotic Navigation Tasks. In: *MICAI 2011 – Mexican International Conference on Artificial Intelligence*, 2011.

[10] R. L. Beirigo, F. A. Pereira, M. L. Koga, T. Matos, V. F. da Silva and A. H. R. Costa. Avaliação de Políticas Abstratas na Transferência de Conhecimento em Navegação Robótica. In *Proc. of VI Workshop de Tecnologias Adaptativas*, 2012.

[11] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[12] V. F. da Silva, F. A. Pereira and A. H. R. Costa. Finding memoryless probabilistic relational policies for inter-task reuse. In *Proc. of 14th Int. Conf. on Information Processing and Management of Uncertainty in Knowlegde-Based Systems (IPMU 2012)*, 2012.

[13] M. van Otterlo. *The Logic of Adaptative Behaviour*, IOS Press, Amsterdam, 2009.

[14] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[15] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, Pearson Education, Inc., 2nd. edition, 2003.

[16] K. Kesting, M. van Otterlo, and L. D. Raedt. Bellman goes relational. In *Proc. of 21th Int. Conf. on Machine Learning*, p. 465–472, 2004.

[17] S. P. Singh, T. Jaakkola and M. I. Jordan. Learning without state-estimation in partially observable Markovian decision processes. In *Proc. of the eleventh international conference on machine learning (ICML 2011)*, vol. 31, p. 37, 1994.

[18] L. Li, T. J. Walsh and M. L. Littman. Towards a unified theory of state abstraction for MDPs. In: *9th Int. Symp. Artificial Int. and Mathematics*. p. 531-539, 2006.

[19] X. R. Cao. A sensitivity view of markov decision processes and reinforcement learning. In: *Modeling, Control and Optimization of Complex Systems*, 2002.

[20] T. Jaakkola, S. P. Singh and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. *Advances in Neural Information Processing Systems 7: Proceedings of the 1994 Conference*, vol. 7, p. 345. MIT Press, 1995.

[21] J. K. Willians and S. Singh. Experimental results on learning stochastic memoryless policies for partially observable markov decision processes. In *Proc. of the 1998 conference on Advances in neural information processing systems II*, p. 1073-1079, 1999.

Marcelo Li Koga graduou-se em Engenharia Elétrica com ênfase em Computação (2010) e atualmente é estudante de Mestrado em Inteligência Artificial, ambos pela Escola Politécnica da Universidade de São Paulo (EPUSP). Atualmente seus interesses de pesquisa se concentram nas áreas de Aprendizado por Reforço e Transferência de Conhecimento.

Valdinei Freire da Silva é Professor Doutor do curso de Sistemas de Informação da Escola de Artes, Ciências e Humanidades da Universidade de São Paulo (USP, 2011). Possui Doutorado em Engenharia Elétrica (USP em co-tutela com IST-UTL, 2009) e é formado em Engenharia da Computação (USP, 2002). Seus interesses em pesquisa abrangem vários aspectos de Aprendizado de Máquina e Tomadas de Decisões Sequenciais em ambientes probabilísticos.

Anna Helena Reali Costa é Professora Titular do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo (USP, 2011). Possui Doutorado em Engenharia Elétrica (USP, 1994) e Mestrado em Engenharia (FEI, 1989). É formada em Engenharia Elétrica (FEI, 1983). De 1985 a 1988 e de 1991 a 1992 foi pesquisadora na Universidade de Karlsruhe, Alemanha. Em 1998 e 1999 foi pesquisadora visitante na Carnegie Mellon University, nos Estados Unidos. Seus interesses em pesquisa abrangem um grande espectro da Inteligência Artificial, incluindo particularmente o Aprendizado de Máquinas, Robótica Inteligente e Visão Computacional.

Sobre a Representação de Estruturas Coalizionais como um Dispositivo Adaptativo - Estudo Preliminar

Frank Cara e Marcio Lobo Netto

***Resumo*—Nesse artigo apresentamos o estudo preliminar de um disposito adaptativo para implementação do processo de formação de coalizões em sistemas multiagentes. O dispositvo representa a estruturas coalizional resultante da montagem de coalizões, juntamente com as regras para a transformação dessa estrutura.**

Definiremos o dispositivo adaptativo e utilizaremos um exemplo didático para mostrar que, utilizando essa ferramenta, a definição do processo de formação de coalizão fica mais concisa, separando detalhes de implementação das regras de formação.

***Index Terms*—Formação de Coalizões, Dispositivos Adaptativos, Colaboração e Cooperação, Sistemas Multi-agentes**

I. Introdução

Agentes podem possuir habilidades sociais, permitindo que eles busquem lucros através de associações. Quando estas associações obedecem normas específicas, formam organizações [1]. Uma categoria de organização é a coalizão, que é uma associação transitória montada para alcançar fins específicos. Agentes coligados tem acesso a mais recursos e, possivelmente, executam tarefas que não poderiam fazer sozinhos [2]. Dessa forma, coalizões tem recebido atenção da comunidade de pesquisa em sistemas multiagentes [3], [4].

O estudo teórico das coalizões é objeto da *teoria dos jogos coalizionais*, que é uma área de pesquisa altamente desenvolvida mas, tradicionalmente, focada em conceitos específicos de solução, como o núcleo do jogo, ou o valor de Shapley de um agente. A comunidade de sistemas multiagentes, por outro lado, está interessada em algoritmos práticos e eficientes para montar coalizões e que, no entanto, incorporem os resultados apresentados pela *teoria dos jogos*.

No processo de formação de coalizões os agentes decidem racionalmente com quem se associar. O conjunto completo das coalizões resultantes dessas associações é denominado *estrutura coalizional*. Nos sistemas multiagentes, o processo de formação de coalizões é um processo iterativo, em que os agentes avaliam sucessivamente as possibilidades de associações, decidem qual associação é mais vantajosa e constituem coalizões.

Acreditamos que esses algoritmos apresentam uma estrutura comum, e que a utilização de técnicas adaptativas pode fornecer um modelo conciso para representá-los. Nesse trabalho apresentaremos as primeiras concepções da representação do processo de formação coalizional como um dispositivo adaptativo. Esse é um trabalho preliminar que pretende mostrar, através de um exemplo didático, a possibilidade da utilização de um dispositivo adaptativo como forma de modelar o problema da formação de coalizões.

A estrutura coalizional é o objetivo final do processo de formação de coalizões. Apresentaremos a noção de *estrutura coalizional adaptativa*, que incorpora o carater iterativo do processo de montagem coalizões através de um dispositivo inspirado nos dispositivos adaptativos apresentados em [5] e [6].

Testaremos o modelo proposto através de um algoritmo simplificado para compartilhamento de recursos. Usaremos a noção de *recursos* como representação de habilidades dos agentes, ou posse de objetos do ambiente. Consideramos uma definição concisa de agentes autônomos, como aqueles que tem capacidade de construir planos de ações para alcançar seus objetivos específicos. Por outro lado, os planos podem possuir carência de recursos, estimulando a formação de coalizões pelas quais podem obter melhor performance.

II. Jogos Coalizionais

Tabela I
Símbolos Utilizados

N	Conjunto de agentes
S	Coalizão
$v(S)$	Função característica da coalizão
E	Estados do ambiente
Ψ	Regras de associação dos agentes
Φ_1	Função de ativação de agentes
Φ_2	Função adaptativa
rc_i e rc_i^{Π}	Vetores de recursos do agente. Respectivamente, os recursos próprios e demandados
$al_i(S)$	Alocação de recursos para o agente na coalizão S
$u_i(S)$	Função utilidade do agente na coalizão S

A teoria dos jogos coalizionais tem o ganho coletivo como seu objeto de estudo [7]. O conceito de *jogo coalizional* é formalizado como o par $<N, v>$, onde N é um conjunto de jogadores ou agentes $\{1, \ldots, n\}$ com uma função característica $v(S)$ que associa um valor real a cada subconjunto $S \subset N$, denominado coalizão. Jogos de utilidade transferível são jogos coalizionais em que a utilidade da coalizão (o ganho auferido pelo trabalho conjunto dos agentes) pode ser distribuído livremente entre os agentes, seguindo normas internas à coalizão.

A Teoria dos Jogos Coalizionais analisa um jogo através de vários *conceitos de solução*. Os conceitos de solução estão divididos naqueles que enfocam a *estabilidade da coalizão (core, nucleous)* ou a *divisão dos recursos dentro da coalizão (valor de Shapley)*.

Um exemplo de jogo coalizional é o *jogo ponderado de votação* que representa a votação majoritária executada nas eleições brasileiras. No jogo de votação ponderada $v(S) = 1 \iff \sum w_i \geq q$, caso contrário $v(s) = 0$. Esse jogo é definido pelo par $[q, (w_1, w_2, \ldots, w_n)]$, onde q é denominada a quota do jogo e w_i o peso do agente i na votação. No caso da votação majoritária temos que $q = N/2 + 1$ e $w_i = 1 \forall i \in N$.

A. Formação de Coalizões

A formação da coalizão é o processo de constituição das coalizões em que os agentes avaliam iterativamente as condições do ambiente e as possibilidades de associação. Inclui 3 etapas [8]:

1) Geração da Estrutura Coalizional: formação de coalizões pelos agentes em que cada coalizão coordena suas atividades, mas sem coordenação entre coalizões. O conjunto de agentes é dividido em partições exaustivas e disjuntas chamadas de *estrutura coalizional*.
2) Resolver o problema de otimização de cada coalizão: Isso significa dividir as tarefas e recursos dos participantes da coalizão de forma a atingir o objetivo comum da coalizão.
3) Dividir o valor gerado entre os participantes da coalizão.

As 3 atividades interagem, e o processo de formação de coalizões entre agentes deve conter todas as etapas.

Os algoritmos de formação de coalizão usualmente adotam uma perspectiva centralizada. Por outro lado, agentes autônomos tomam decisões segundo sua própria agenda, agindo de forma descentralizada. Sendo assim, existe uma oposição entre as duas abordagens, com uma carência de algoritmos que sejam aplicáveis no contexto de sistemas multiagentes.

Dividimos a formação da coalizão em sistemas multiagentes nas seguintes etapas, que cobrem as etapas discutidas em [8] e apresentadas anteriormente.

- Identificação das necessidades: ao montar uma coligação, o agente participa dos custos e benefícios da mesma. Dessa forma, para poder avaliar a viabilidade de uma coligação, o agente precisa fazer o detalhamento dos seus objetivos. Esse detalhamento deve permitir que o agente classifique as coligações por ordem de preferência.
  Uma forma possível de detalhamento dos objetivos é através de planos de ações que, além de caracterizar as necessidades do agente, permitem a avaliação das associações com outros agentes sob a ótica do ganho que tais associações trariam para a realização dos planos de ações.
- Seleção de coalizões candidatas: estamos adotando o modelo de coalizão exclusiva, ou seja, o agente somente pode participar de uma coalizão por vez. Dessa forma, faz parte do processo de formação de coalizões a identificação de coligações candidatas e, dentre estar, as de maior ganho.
- Alteração da estrutura coalizional: uma vez selecionada a coalizão desejada, há a mudança da estrutura coalizional, com a formação de uma nova coalizão e a alteração ou exclusão da coalizão da qual o agente participava.

B. Estruturas Coalizionais

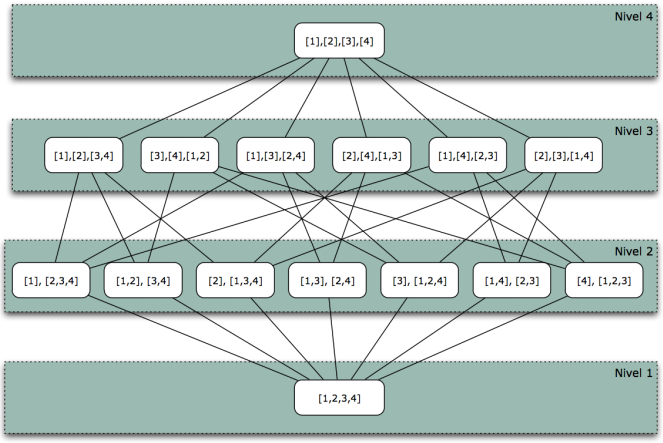


Figura 1. Estrutura Coalizional

Uma estrutura coalizional CS é uma partição de N em coalizões, ou seja, $CS = \{S_1, S_2, \ldots, S_k\}$, onde $N = \bigcup_i^{|S|} S_i$.

A estrutura coalizional é montada como resultado do processo de formação de coalizões. Na figura 1 temos o exemplo de todas as estruturas organizacionais para 4 agentes $N = \{1, 2, 3, 4\}$. Nessa figura cada nó do grafo representa uma estrutura coalizional. Os níveis separam as estruturas coalizionais por quantidade de coalizões. Assim o nível 2, por exemplo, possui 2 coalizões. As arestas representam modificações necessárias para levar de uma estrutura coalizional para outra. Dessa forma, a aresta entre as estruturas $\{[1], [2, 3, 4]\}$ e $\{[1], [2], [3, 4]\}$ representa a ação de abandono de coalizão do agente 2.

III. Estrutura Coalizional Adaptativa

Uma estrutura coalizional adaptativa é um dispositivo formado pela ênupla:
$CS_{adapt} = < CS, E, \Psi, \phi_1, \phi_2 >$, onde:

CS	Uma estrutura coalizional subjacente ao dispositivo.
E	O estado do ambiente.
Ψ	O conjunto de regras dos agente i, onde $\psi_i : CS \times E \rightarrow CS'$ é a regra de associação de cada agente, representando uma ação de abandono e opcionalmente junção a outra coalizão.
ϕ_1	A regra de ativação da função de associação, que implementa a decisão de que agente ativar para executar a transformação da estrutura coalizional subjacente.
ϕ_2	A função adaptativa que modifica o conjunto de regras, $\phi_2 : CS_{adapt} \times E \rightarrow CS'_{adapt}$.

A estrutura coalizional adaptativa tem a finalidade de modelar o processo iterativo de formação de coalizões através de um dispositivo adaptativo.

Agentes autônomos decidem individualmente, baseados nas suas regras e no estado do ambiente, qual é a melhor coligação possível de ser efetuada. Essas regras, representadas por ψ_i, podem ser homogêneas para todos os agentes, ou heterogêneas, dependendo do tipo de sistema multiagente (a

heterogeneidade usualmente está associada a sistemas abertos, como leilões digitais).

A reconfiguração das regras dos agentes é executada pela função adaptativa ϕ_2, e sua finalidade esta associada ao controle geral do sistema multiagente. A função adaptativa pode variar parâmetros de acordo com a etapa do ciclo de vida do sistema, ou mesmo alterar o comportamento do agente para a obtenção de objetivos comuns ao sistema.

Tome o caso de agentes que executam planejamento de ações baseado em aprendizado por reforço, ou mesmo um sistema multiagente com características evolutivas (seleção de indivíduos mais aptos e recombinação de características). Existem parâmetros que controlam o comportamento do algoritmo que podem ser reconfigurados pela função adaptativa.

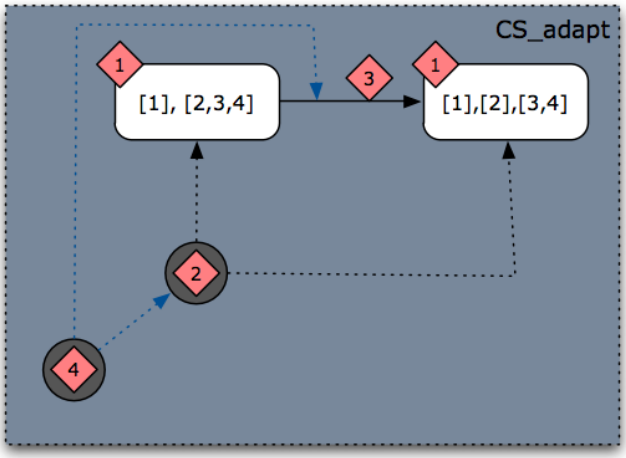


Figura 2. Estrutura Coalizional Adaptativa

Na figura 2 temos a representação da estrutura coalizional adaptativa (CS_{adapt}). Os losângulos numerados representam os elementos do CS_{adapt}. O primeiro representa a estrutura coalizional subjacente. A aresta de número 3 representa a regra de associação, nesse caso do agente 2, que abandona a coalizão modificando a estrutura coalizional subjacente. Para abandonar a coalizão o agente decidiu, através dessa regra, que seria vantajoso (obteria melhor utilidade, se esse for o critério) estar independente de qualquer coalizão. O nó de número 2 representa a regra ϕ_1 que decide qual agente ativar da estrutura coalizional subjacente. O nó 4, por sua vez, representa a regra ϕ_2 que é a função adaptativa, agindo sobre os elementos da estrutura coalizional adaptativa. Pelas arestas podemos observar que a função adaptativa modifica tanto a regra de ativação dos agentes, como as regras de associação dos agentes.

IV. Exemplo - Compartilhamento de Recursos

Para exemplificar a utilização da *estrutura coalizional adaptativa*, modelaremos um processo de compartilhamento de recursos entre agentes.

O conceito de recurso é utilizado em sistemas multiagentes para representar itens do ambiente que serão distribuídos representando um elemento do ambiente que pode ser transferido entre os agentes, e que de alguma forma influencia na realização dos objetivos do agente. Além disso, em muitos contextos [9] habilidades próprias dos agentes podem ser modeladas como recursos.

Alocação de recursos é uma distribuição particular dos recursos entre os agentes. No caso de recursos indivisíveis é uma partição do conjunto de recursos. O compartilhamento de recursos, por sua vez, é o processo de distribuição dos recursos comuns ao sistema no qual estes estão sob posse de agentes individuais, que podem negociá-los segundo critérios próprios.

Os procedimentos de compartilhamento de recursos são divididos em três categorias.

1) Leilão é o processo de alocar um recurso por um critério de preço. Esse procedimento é mais utilizado quando há escassez de recursos, estudado e aplicado em sistemas multiagentes, tanto no negócio de *leilões eletrônicos* como para distribuição de tarefas em sistemas distribuídos. [10]
2) Barganha, também conhecido como negociação é o procedimento utilizado quando há conflito de preferências no compartilhamento de recursos. Diversos procedimentos para negociação já foram desenvolvidos, com aplicações variadas [11], [12], [13] sendo uma forma de compartilhamento bastante utilizada, mas inadequada para compartilhamento de recursos entre uma quantidade grande de agentes.
3) Coalizões é um campo de estudo da teoria dos jogos [7], adequado para formação de grupos de agentes unidos com uma determinada finalidade. Para a aplicação da teoria dos jogos coalizionais em sistemas multiagentes devemos ter o contexto do SMA modelado como um jogo coalizional.

A. Jogo de Distribuição Ponderada de Recursos

Nessa seção definiremos o jogo utilizado na formação das coalizões. Primeiramente, os recursos tem uma representação cardinal na forma de um vetor de recursos. Um vetor de recursos de um agente é uma ênupla $(rc_i(k_1), rc_i(k_2), \ldots, rc_i(k_{|\mathcal{K}|})) \in \mathbb{N}^{|\mathcal{K}|}$ com uma posição $1 \leq j \leq |\mathcal{K}|$ para cada recurso do ambiente, sendo $rc_i(k_j) \in \mathbb{N}$ a quantidade do recurso referente à posição j.

Chamaremos de *vetor de recursos próprios* o vetor de inteiros que define a quantidade de recursos que um agente possui, denotado por rc_i. Definiremos o *vetor de recursos demandados pelo agente* por rc_i^{Π}, montado a partir dos recursos demandados para execução do conjunto de planos Π_i do agente i. A quantidade de recursos demandadas é definida na função de planejamento do agente. O agente monta um ou mais planos para serem executados e que dependem de recursos externos. A função de mapeamento do agente transforma o conjunto de planos em um vetor de recursos requeridos para a sua execução.

O *Jogo de Distribuição Ponderada de Recursos* é o jogo definido pela função característica que mede a contribuição

Algorithm 1 Procedimento de formação de coalizões

1: Cada agente constrói o seu plano
2: **repeat**
3: **for** cada agente i em ordem randômica **do**
4: Inicializa o vetor $u_i = (u_i^j)$.
5: **for** cada agente $j \notin S_i$ **do**
6: Testa coalizão S_i^j com j ou coalizão S^j:
7: $al_i^{S_i^j}$ é o valor alocado para i pela coalizão, seguindo a regra de proporcionalidade: $al_i(S) \approx w_i.rc_S$.
8: $u_i^j \leftarrow \frac{\sum al_i^{S_i^j}}{rc_i^\Pi}$
9: **end for**
10: **Dado que** $j \leftarrow \arg\max_j u_i^j$,
então se $u_i > \lambda.u_i^j$ monta a coalizão $S_j \cup \{i\}$
11: **end for**
12: **until** Não há melhora no bem-estar coletivo por MAX iterações

ponderada do agente com relação aos recursos da coalizão.

$$v(S) = \frac{\sum rc_i}{\sum rc_i^\Pi} \tag{1}$$

Ao participar de uma coalizão o agente troca recursos com os outros agentes, segundo um critério de alocação. O vetor de recursos resultante da alocação em uma coalizão é representado por $al_i(S)$. No nosso exemplo a alocação será proporcional ao peso do agente, dessa forma, temos:

$$w_i = \frac{rc_i - rc_i^\Pi}{rc_S - rc_S^\Pi} \tag{2}$$

$$al_i \approx w_i.rc_S \tag{3}$$

O valor da coalizão para um agente é medido pela utilidade da coalizão ($u_i(S)$).

$$u_i(S) = \frac{\sum al_i(S)}{\sum rc_i^\Pi} \tag{4}$$

O algorítmo 1 apresenta o procedimento para executar a formação de coalizão utilizando o *jogo de distribuição ponderada de recursos*.

B. Estrutura Coalizional Adaptativa para o Jogo de Distribuição Ponderada de Recursos

Considerando-se a *estrutura coalizional adaptativa* como dispositivo adaptativo descrito anteriormente, podemos representar o algorítimo 1 utilizando os elementos do dispositivo.

- ψ_i^{rc} Para qualquer agente j verifique a utilidade de participar da coalizão S_j (a coalizão da qual j participa). Se houver ganho, abandona a coalizão S e une-se a S_j. Ou seja, se $\lambda.u_i(S) < u_i(S_j)$, então abandona S e une-se a S_j, sendo u_i calculado segundo as equações (2), (3) e (4).
- Φ_1^{rc} Selecione ψ_i sendo i o agente com menor utilidade, e que não tenha sido ativado anteriormente.
- Φ_2^{rc} Modifique Ψ após k rodadas de ativações de tal forma que $\lambda = 1.5\lambda$. Com o incremento de λ estamos aumentando a estabilidade da estrutura coalizional de forma a termos um critério de interrupção da ação adaptativa.

V. Conclusão

Nesse artigo nos propusemos a apresentar o modelo de um dispositivo adaptativo que fosse capaz de conter os dados e processos associados a formação de coalizões. O modelo inicial atende as necessidades de representação do jogo coalizional utilizado como exemplo. Além disso, a especificação geral do dispositivo permite que o detalhamento das regras seja feito em nível de abstração mais alto do que no formato algorítimico, o que nos parece ser um ganho para a definição do processo iterativos utilizados em sistemas multiagentes.

Comparando-se a especificação do dispositivo adaptativo com o algorítimo, percebe-se que o primeiro é mais conciso nas definições, permitindo a abstração dos detalhes de implementação. Ainda, com a maturação do modelo a implementação do dispositivo adaptativo deverá permitir a implementação de um framework que poderá atender diversos jogos coalizionais.

Os próximos passos desse trabalho devem ser:

- Modelagem de situações coalizionais mais complexas enfocando problemas reais.
- Implementação do dispositivo adaptativo.

Referências

[1] P. Davidsson, "Categories of artificial societies," in *Proceedings of the Second International Workshop on Engineering Societies in the Agents World II*. London, UK: Springer-Verlag, 2001, pp. 1–9.

[2] V. Conitzer and T. Sandholm, "Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains," in *Proceedings of the 19th national conference on Artifical intelligence*. AAAI Press, 2004, pp. 219–225.

[3] A. C. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings, "Decentralised dynamic task allocation using overlapping potential games," *Comput. J.*, vol. 53, no. 9, pp. 1462–1477, 2010.

[4] T. Rahwan, S. D. Ramchurn, N. R. Jennings, and A. Giovannucci, "An anytime algorithm for optimal coalition structure generation," *J. Artif. Int. Res.*, vol. 34, no. 1, pp. 521–567, 2009.

[5] J. a. J. Neto, "Adaptive rule-driven devices - general formulation and case study," in *Revised Papers from the 6th International Conference on Implementation and Application of Automata*, ser. CIAA '01. Springer-Verlag, 2002, pp. 234–250.

[6] T. Pedrazzi, A. Tchemra, and R. Rocha, "Adaptive decision tables a case study of their application to decision-taking problems," in *Adaptive and Natural Computing Algorithms*, B. Ribeiro, R. F. Albrecht, A. Dobnikar, D. W. Pearson, and N. C. Steele, Eds. Springer Vienna, 2005, pp. 341–344.

[7] M. Osborne and A. Rubinstein, *A Course in Game Theory*. MIT Press, 1994.

[8] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, pp. 209 – 238, 1999.

[9] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J. A. Rodríguez-aguilar, and P. Sousa, "Issues in multiagent resource allocation," *Informatica*, vol. 30, 2006.

[10] D. Lehmann, R. Muller, and T. Sandholm, "The winner determination problem," in *Combinatorial Auctions*, P. Cramton, Y. Shoham, and R. Steinberg, Eds. MIT Press, 2006, pp. 297–317.

[11] J. Rosenschein and G. Zlotkin, *Rules of encounter: designing conventions for automated negotiation among computers*. the MIT Press, 1994.

[12] P. Dunne, "Extremal behaviour in multiagent contract negotiation," *J. Artif. Intell. Res. (JAIR)*, vol. 23, pp. 41–78, 2005.

[13] T. W. Sandholm, “Contract types for satisficing task allocation: I theoretical results,” in *IN PROC. AAAI SPRING SYMPOSIUM: SATISFICING MODELS*, 1998, pp. 68–75.

Frank Cara É bacharel em Ciëncias da Computação pela UNICAMP (2000). Atualmente é aluno do programa de mestrado em Engenharia Elétrica da Escola Politécnica / USP.

Márcio Lobbo Neto Graduado em Engenharia Elétrica - Eletrônica (1985), Mestre em Engenharia Eletrônica - Sistemas Eletrônicos (1990), ambos pela Escola Politécnica da USP, e Doutor em Informática pela Technische Universität Darmstadt (1996). Atualmente é professor livre docente da Escola Politécnica da Universidade de São Paulo.

Adaptatividade para pesquisas em Linguística de *Corpus*

J. L. Moreira Filho and Z. M. Zapparoli

***Abstract* — This paper introduces some of the main computational tools, methodology and problems for language teaching research in *Corpus* Linguists, attempting to argue for the need of investigation in the use of Adaptive Technology in the area as an improvement. In order to illustrate the viability and relevance of the proposal, examples are also presented and discussed. It may be considered as a starting point for future works on both areas.**

***Keywords* — *Corpus* Linguistics, Adaptative Technology, computational tools.**

I. INTRODUÇÃO

Este trabalho apresenta algumas das principais ferramentas computacionais, metodologia e problemas que pesquisadores na área de Linguística de *Corpus*, especificamente no campo de atuação do ensino-aprendizagem de língua estrangeira, enfrentam em suas pesquisas, na tentativa de argumentar sobre a necessidade de investigação do uso do conceito e técnicas adaptativas na área a fim de buscar soluções que possam avançar as práticas correntes.

Primeiro, destacamos a importância do uso de *corpus*, textos autênticos, no ensino de línguas, motivo pelo qual há grande interesse de utilização do material de *corpus* em pesquisas para elaboração de cursos, material pedagógico, avaliação de livros didáticos existentes, desenvolvimento de *software*, entre outras aplicações.

Em seguida, apresentamos brevemente as áreas da Linguística de *Corpus* e Tecnologia Adaptativa. Uma descrição um pouco mais detalhada sobre a Linguística de *Corpus* é fornecida, uma vez que será o foco de análise para as discussões que seguirão.

Logo após, para ilustrar as ferramentas e práticas correntes no contexto mencionado, são apresentadas e descritos os principais programas e a metodologia básica para pesquisas em Linguística de *Corpus*, principalmente no âmbito do ensino-aprendizagem de línguas,

Tendo fornecido uma visão geral das características do instrumental para as pesquisas, argumentamos sobre a necessidade de investigação da possibilidade de uso do conceito e técnicas adaptativas a partir de problemas e lacunas na prática de pesquisa em Linguística de *Corpus*.

Como fechamento, apresentamos as considerações finais e possíveis caminhos para futuros trabalhos em ambas as áreas.

II. A IMPORTÂNCIA DO USO DE TEXTOS AUTÊNTICOS

A utilização de textos autênticos, textos que não foram inventados para ensinar língua, é vital se o objetivo de ensino é a comunicação, a interação social e a execução de tarefas no mundo real [1]. A utilização de textos autênticos em sala de aula é benéfica para aprendizagem.

Entre os argumentos a favor dessa utilização está o de que ela promove um aumento da motivação, pois o aluno sente que está aprendendo a "língua de verdade".

Conforme [2], muitos livros didáticos, por uma série de questões, utilizam textos inventados para o ensino de línguas, o que pode trazer efeitos negativos para a aprendizagem: desmotivação, habilidades não plenamente desenvolvidas, e uso não natural da língua.

O trabalho com textos autênticos e o uso de *corpus* para fins pedagógicos na escola podem trazer alternativas significativas para a aprendizagem de língua. Com eles, é possível dar condições para que os alunos se conscientizem sobre as unidades pré-fabricadas da língua, que são os padrões léxico-gramaticais, e a característica probabilística da linguagem, isto é, como a língua realmente funciona.

Os exemplos extraídos de *corpus* são importantes para a aprendizagem de línguas porque expõem os alunos, desde os estágios iniciais do processo de aprendizagem aos tipos de frases e vocabulários que possivelmente serão encontrados em textos autênticos da língua ou no uso da língua em situações reais de comunicação.

Para isso, existe uma ampla instrumentação que pode ser aproveitada por professores e pesquisadores na área de ensino de línguas. Há uma série de *corpora* e ferramentas computacionais sendo desenvolvidas, ainda não totalmente conhecidas e utilizadas por esses profissionais, especificamente professores.

Contudo, é preciso verificar se as ferramentas disponíveis são acessíveis e adequadas para profissionais em diferentes contextos, refletindo sobre a viabilidade de sua introdução. A disponibilidade de toda a instrumentação é desejada, o que pode levar a questões de adaptação e criação de soluções de fácil uso.

III. TECNOLOGIA ADAPTATIVA

Conforme [3], a Tecnologia Adaptativa está relacionada a técnicas, métodos e disciplinas que estudam as aplicações da adaptatividade, que pode ser entendida como uma propriedade que um determinado modelo tem de modificar espontaneamente seu próprio comportamento em resposta direta a uma entrada, sem auxilio externo.

Um sistema adaptativo é aquele que possui a propriedade de se auto modificar a partir de determinada entrada, sem a necessidade de um agente externo.

Dentro da Tecnologia Adaptativa, há a noção de dispositivo, uma abstração formal. O dispositivo pode ser adaptativo ou não adaptativo. O dispositivo não adaptativo pode ser formado por um conjunto finito de regras estáticas

que, em linguagem de programação, pode ser representado na forma de cláusulas IF-THEN. A operação do dispositivo se dá pela aplicação das regras, tendo como retorno determinados estados. Quando o dispositivo não aplica nenhuma regra, a operação é terminada, gerando um erro. As ações de dispositivos adaptativos podem ser chamadas quando ocorre algum erro (quando nenhuma regra é aplicável), ou quando a operação do dispositivo não adaptativo está em um determinado estado.

Basicamente, os dispositivos adaptativos são formados por três ações adaptativas elementares [4]: i. Consulta de regras/estados; ii. Exclusão de regras; iii. Inclusão de regras. Seu uso está ligado a situações complexas em que há a necessidade de tomadas de decisões não triviais, por exemplo, na área de estudos da linguagem, resolução de ambiguidades em programas de anotação (morfológica, sintática, etc.).

IV. Linguística de *Corpus*

A Linguística de *Corpus* é uma área que estuda a língua por meio da observação de grandes quantidades de dados linguísticos reais, isto é, textos falados ou escritos provenientes da comunicação no mundo real língua em uso), com o auxílio de ferramentas computacionais.

De forma geral, o conjunto de dados linguísticos reais criteriosamente coletados e utilizados em estudos de Linguística de *Corpus* é chamado de *corpus* (plural: *corpora*). O *corpus* deve ser constituído de dados autênticos (não inventados), legíveis por computador e representativos de uma língua ou variedade da língua que se deseja estudar.

A Linguística de *Corpus* faz uso de uma abordagem empirista e tem como central a noção de linguagem enquanto sistema probabilístico. De acordo com essa noção, os traços linguísticos não ocorrem de forma aleatória, sendo possível evidenciar e quantificar regularidades (padrões). É comum na área afirmar que a linguagem é padronizada (*patterned*), isto é, existe uma correlação entre os traços linguísticos e os contextos situacionais de uso da linguagem.

Na Linguística de *Corpus*, a padronização evidencia-se por colocações, coligações ou estruturas que se repetem significativamente

Algumas das áreas de interesse da Linguística de *Corpus* são: compilação de *corpora* e desenvolvimento de ferramentas para sua análise, descrição de linguagem, exploração do uso de descrições baseadas em *corpora* para várias aplicações, tal como o ensino de línguas, processamento de linguagem natural, reconhecimento de voz e tradução.

O computador desempenha um papel importante para os estudos na área, que foi impulsionada por seus avanços. As ferramentas computacionais são geralmente utilizadas para reorganização e extração de informações no *corpus* para observação e interpretação de dados, fornecendo novas perspectivas para a análise linguística.

Por meio das ferramentas computacionais, as pesquisas linguísticas podem ganhar velocidade, precisão e confiabilidade. Há uma comparação muito comum na literatura segundo a qual o computador e seus recursos seriam para o desenvolvimento da Linguística equivalentes ao do microscópio para a biologia.

Assim, surge uma grande variedade de *software* disponível para os estudos de Linguística de *Corpus*. Muitos *softwares* exibem e permitem a manipulação de extensas listas de frequência de palavras, fraseologias e colocações.

[5] apresenta uma série de ferramentas computacionais que podem ser utilizadas em pesquisas de Linguística de *Corpus*, as quais serão citadas, brevemente, em três categorias: i. programas para coleta de *corpus*; ii. Etiquetadores; iii. programas para análise da padronização.

V. Principais ferramentas utilizadas em pesquisas de Linguística de *Corpus*

Programas para coleta de *corpus*, como o *WinHTTrack*, um *off-line browser*, disponível em http://www.httrack.com/, permite o download de um site inteiro da Internet para o diretório de um computador local, incluindo HTML, imagens e outros tipos de arquivos. Em suas opções avançadas, é possível especificar tipos de arquivos e condições de busca para a coleta. Embora o programa não seja destinado especificamente para a pesquisa linguística, suas funções vão ao encontro da necessidade da coleta de grandes quantidades de textos na área, a qual também pode ser conseguida por meio de outros programas e *scripts* de linguagem de programação.

Outro tipo de programa, geralmente necessário após uma coleta automática de textos na Internet, é um conversor de HTML para textos sem formatação e códigos de script característicos. Tais programas podem ser encontrados facilmente na Web.

Também é possível escrever *scripts* em linguagem de programação Shell ou Python, por exemplo, que utilizem expressões regulares para a remoção de determinados códigos e textos indesejados, fazendo uma 'limpeza' nos textos do *corpus* para o prosseguimento das análises. Para os usuários de Windows, uma opção é a instalação do Cygwin, um emulador do sistema operacional Unix para Windows. Os scripts podem ser escritos e executados em um terminal de comandos.

Etiquetadores online e desktop, tais como o etiquetador do site VISL, Brill e TreeTagger, são utilizados para inserir automaticamente etiquetas que podem indicar, dependendo do tipo de etiquetagem, marcações morfossintáticas, sintáticas, semânticas ou discursivas. Um exemplo de tais marcações é ilustrado a seguir:

```
( [(] PU @PU #1->0
Reuters [Reuter] N P NOM @APP #2->0
) [)] PU @PU #3->0
- [-] PU @PU #4->0
The [the] ART S/P @>N #5->7
Arctic [Arctic] ADJ POS @>N #6->7
zone [zone] N S NOM @SUBJ> #7->8
has [have] <aux> V PR 3S @FS-ACC> #8->38
moved [move] <mv> V PCP2 AKT @ICL-AUX< #9->8
into [into] PRP @<SA #10->9
a [a] ART S @>N #11->12
warmer [warm] ADJ COM @P< #12->10
, [,] PU @PU #13->0
greener [green] ADJ COM @>N #14->19
« [«] PU @PU #15->0
new [new] ADJ POS @>N #16->19
normal [normal] ADJ POS @>N #17->19
» [»] PU @PU #18->17
phase [phase] N S NOM @SUBJ> #19->0
, [,] PU @PU #20->0
which [which] <rel> INDP S @SUBJ> #21->22
means [mean] <mv> V PR 3S @FS-N< #22->19
less [little] DET COM S @>N #23->24
```

Fig. 1. Exemplo de texto etiquetado

A maioria das ferramentas disponíveis permite uma utilização avançada, em que o conjunto de etiquetas pode ser aumentado e/ou o usuário pode fazer um treinamento dos dados para melhor satisfazer as necessidades específicas de sua pesquisa.

Programas para análise da padronização, como listadores de palavras e concordanciadores, que são programas que permitem a busca por palavras específicas em um *corpus*, fornecendo exaustivas listas para as ocorrências da palavra em contexto.

Geralmente esses programas são encontrados em uma única solução/*software*. É o caso de *software* como o famoso Wordsmith tools, com versões disponíveis para download em http://www.lexically.net/wordsmith/, um dos mais utilizados em pesquisas de Linguística de *Corpus*, cuja versão mais estável possui três ferramentas principais: Wordlist (lista palavras por frequência no *corpus*), KeyWords (extrai palavras-chave do *corpus*) e Concord (gera linhas de concordância).

Outros *software* que executam funções similares são: MicroConcord (http://www.lexically.net/software/index.htm), AntConc (http://www.antlab.sci.waseda.ac.jp/software.html), Unitex (http://www-igm.univ-mlv.fr/~unitex/) e Kitconc (http://www.fflch.usp.br/dl/li/x/?p=435), uma versão desktop simples em português.

Vejamos a interface de um programa que executa a função de contagem da frequência de palavras em corpora:

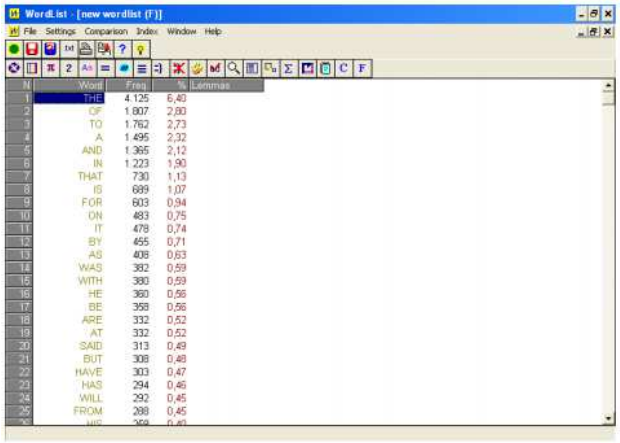


Fig. 2. Lista de frequência de palavras no Wordsmith tools 3.0

A partir de listadores de palavras, é possível descobrir quais são as palavras mais utilizadas em um texto ou conjunto de textos. Uma das opções permite listar também as expressões mais ocorrentes formadas por duas ou mais palavras, chamadas de n-gramas de um texto ou conjunto de textos.

A lista de palavras é uma listagem ordenada por frequência de todas as formas que ocorrem em um *corpus*. A partir da lista de frequência podemos definir quais são as palavras mais interessantes para análise do *corpus*.

A ideia é a de que palavras que possuem uma ocorrência maior são mais relevantes, visto que há uma probabilidade maior de serem encontradas em outro *corpus* ou textos. Por exemplo, para um aprendiz inicial de língua estrangeira, aprender palavras mais frequentes é extremamente essencial, visto que podem aparecer em diversos contextos.

Ao analisar a ocorrência das palavras gramaticais, podemos tentar identificar quais palavras se destacam em relação ao tipo de *corpus*, texto, gênero ou registro as quais pertencem. Geralmente, a palavra mais frequente (número um da lista) em textos em língua portuguesa é a preposição 'de'. Se alguma outra palavra gramatical ocupar esta posição, será uma ocorrência marcada e merecedora de verificação.

A manipulação das listas por meio de funções de classificação permite, quando o pesquisador julgar necessário, juntar frequências de palavras, como é o caso da lematização. No exemplo abaixo, o pesquisar poderia ter a necessidade de juntar todas as frequências das diferentes formas do verbo 'ABRIR' sob uma única forma (infinitivo):

3	ABERTO	2	00,02		
4	ABRA	4	00,05		
5	ABRE	3	00,04		
6	ABRIR	4	00,05		

Fig. 3. Exemplo de palavras marcadas para lematização

Podemos também levantar informações estatísticas gerais do *corpus*, como número total de palavras/itens (*tokens*), número de vocábulos/formas (*types*), razão forma/item (*TypeToken Ratio*), que pode ser calculada da seguinte maneira: ((número total de formas / número de itens)*100). A porcentagem resultante pode indicar a proporção de repetição das palavras no *corpus*. Um valor baixo da razão forma/item indica que há muita repetição no *corpus*, ou seja, um número menor de formas. Um valor alto da razão forma/item indica pouca repetição das palavras no *corpus*, um número mais de formas.

Outra maneira de fazer um recorte em relação às palavras que devem ser analisadas é a extração de palavras-chave. Muitas vezes, a lista de palavras-chave fornece uma filtragem mais apurada das palavras que se destacam em *corpus* ou texto.

Em um *corpus* formado por textos de anúncios de emprego em inglês, palavras como '*EXPERIENCE*', '*SKILLS*', '*ABILITY*', '*REQUIREMENTS*' e '*SALARY*' seriam palavras-chave, uma vez que podem ser consideradas especificas do gênero, revelando parte de sua estrutura padronizada:

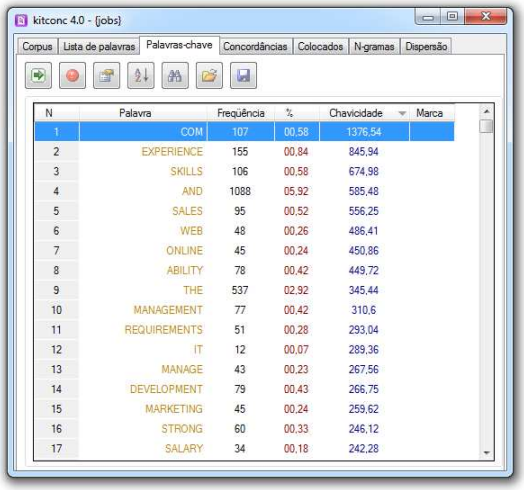


Fig. 4. Palavras-chave de no Kitconc 4.0

Em programas que executam extração de palavras-chave, é comum haver um recurso que possibilita a comparação automática de uma lista de frequência de palavras do *corpus* de estudo com uma lista de frequência de palavras de um *corpus* de referência, geralmente muito maior, a partir de um *corpus* de língua geral.

Em programação, tal comparação pode ser feita utilizando uma fórmula como mostra o *script* abaixo:

```
def loglikelihood(self,ntokensinsc,ifreqiinsc, ntokensincr,ifreqincr):
    #convert to float
    nt1 = float(ntokensinsc)#number of tokens in study corpus
    nt2 = float(ntokensincr)#number of tokens in ref corpus
    f1 = float(ifreqiinsc)#freq of item in study corpus
    f2 = float(ifreqincr) #freq of item in ref corpus
    #calculate
    e1 = nt1 * ((f1+f2) / (nt1+nt2))
    e2 = nt2 * ((f1+f2) / (nt1+nt2))
    ll = 2 *  ( (f1 * math.log(f1/e1) ) + (f2 * math.log(f2/e2) ) )
    #return value
    return ll
```

Fig. 5. Função *loglikelihood* para extração de palavras-chave

A função é aplicada para cada palavra na lista de frequência. Os argumentos da função, em ordem, são: frequência da palavra no *corpus* de estudo, número total de *tokens* no *corpus* de estudo, número total de *tokens* no *corpus* de referência, frequência da palavra no *corpus* de referência.

O resultado da comparação retorna uma lista classificada em que as principais palavras do *corpus* de estudos estarão no topo, geralmente palavras de conteúdo, em contraponto a uma lista de frequência, em que as primeiras palavras são gramaticais.

Uma concordância é a listagem das ocorrências de uma palavra de busca de um *corpus*, a qual fica centralizada, com uma quantidade definida de contextos em ambos os lados (esquerda e direita).

Em um primeiro momento, é comum associar/confundir o termo concordância apresentado aqui com o termo utilizado em gramática.

Quando utilizamos o termo concordância, queremos nos referir a um tipo de visualização privilegiada do uso de palavras, conforme já descrita. No exemplo abaixo, exibimos linhas de concordância da palavra 'tendência' em seu formato mais comum, KWIC (*key word in context*).

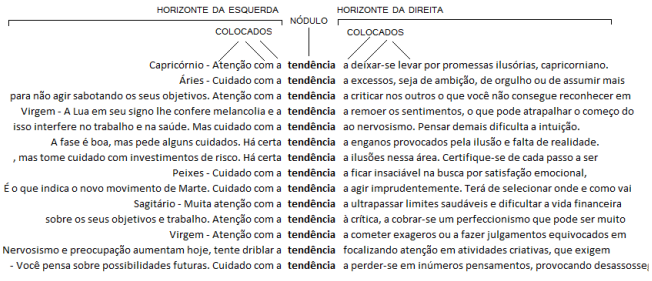


Fig. 6. Exemplo de linhas de concordância

Diferentemente de um texto normal, as linhas de concordâncias são geralmente lidas a partir de seu nódulo, palavra ou expressão de busca, a qual fica centralizada para análise. Na leitura, buscamos a existência de padrões de uso por meio da análise de colocados, palavras à esquerda e à direita que ocorrem significativamente com o nódulo. Por isso, em primeiro, não aconselhamos ficar buscando a totalidade dos fragmentos, a fim de completar a frases/sentenças.

Como exemplo, podemos primeiramente buscar quais palavras da esquerda e da direita geralmente se colocam com a palavra de busca a fim de determinar padrões. Em seguida, podemos analisar quais tipos de palavras formam tais padrões, classe gramatical, campo semântico, etc. Por fim, são identificados os sentidos de cada padrão. Vejamos a padronização da palavra '*resume*' nas linhas de concordância a seguir:

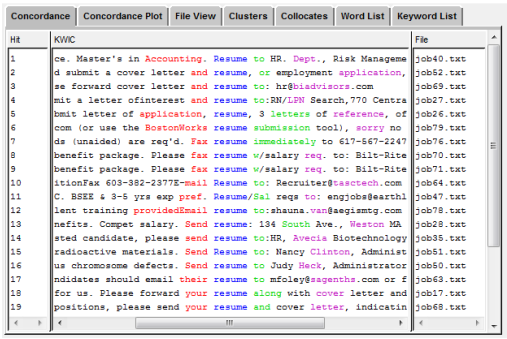


Fig. 7. Concordâncias da palavra '*resume*' no AntConc

Podemos analisar linhas de concordância e identificar três tipos de padrões: colocação, coligação e prosódia semântica. Colocação refere-se à junção significativa, co-ocorrência, entre palavras (*send*/*fax* + *resume* + *to*). Coligação é a relação gramatical mantida pelas palavras (verbo + resume + preposição). Prosódia semântica negativa, positiva ou neutra (o padrão *send/fax resume to* ocorre tipicamente com palavras de semântica e contextos neutros).

A identificação de padrões pode ser auxiliada também por outros tipos de visualização, como lista de colocados:

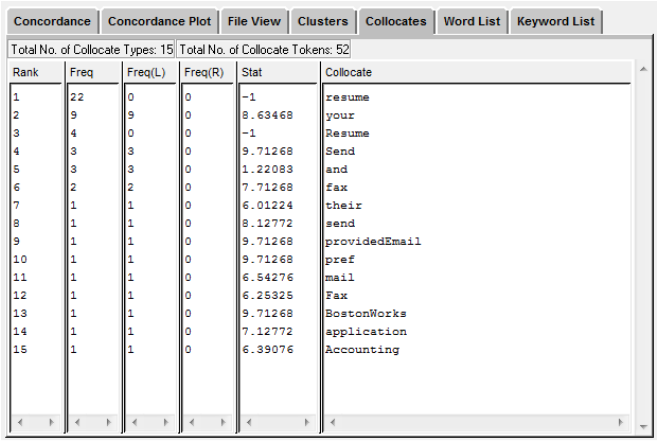
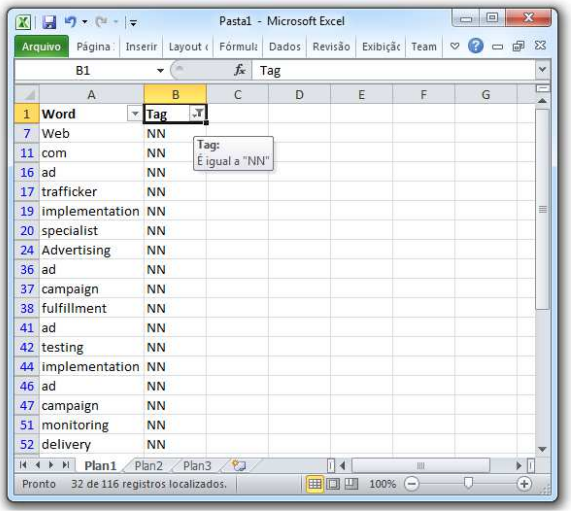


Fig. 8. Lista de colocados da palavra *resume* no AntConc

Na lista de colocados, podemos identificar a ocorrência das palavras '*send*' e '*fax*', colocados que formam o padrão identificado.

Quando há a necessidade de uso de algum recurso não encontrado nessas ferramentas para manipulação de seus dados, é comum a utilização de programas complementares.

Um desses programas pode ser a planilha eletrônica do Excel, que disponibiliza opções de filtro, classificação, cálculos de fórmulas, entre outras. Vejamos a filtragem de substantivos, identificados pela etiqueta 'NN', a partir de um texto etiquetado:



	Word	Tag
7	Web	NN
11	com	NN
16	ad	NN
17	trafficker	NN
19	implementation	NN
20	specialist	NN
24	Advertising	NN
36	ad	NN
37	campaign	NN
38	fulfillment	NN
41	ad	NN
42	testing	NN
44	implementation	NN
46	ad	NN
47	campaign	NN
51	monitoring	NN
52	delivery	NN

Fig. 9. Utilização do recurso de filtro em dados anotados

Embora o exemplo de utilização pareça simples, em conjunto com outras funções, o aplicativo pode suprir necessidades de determinadas pesquisas.

Um exemplo é a pesquisa de [6], que comparou os resultados de diferentes ferramentas de análise de *corpus* na tarefa de identificação de palavras candidatas a termo em textos sobre câncer de mama e desenvolveu uma metodologia com o auxílio do Excel para a filtragem dos candidatos mais comuns entre todas as ferramentas e candidatos exclusivos de cada uma.

VI. Uso das ferramentas computacionais em metodologia básica para pesquisas

As ferramentas descritas anteriormente podem ser aplicadas nos seguintes passos em pesquisas, compondo uma metodologia básica, em nível menos profundo, para a elaboração de cursos e extração de material para confecção de atividades pedagógicas. É possível que os passos descritos também sejam utilizados em pesquisas com diferentes objetivos além dos especificados, tal como lexicografia análise de *corpora* de aprendizes.

Vejamos os seguintes passos:

- Coleta do *corpus*;
  - Manual ou automática a partir de programas específicos.
- Preparação do *corpus* para as análises;
  - Limpeza e organização dos textos.
- Análise das frequências das palavras do *corpus*;
  - Busca de indícios de palavras que se destacam no *corpus*.
- Análise de palavras-chave do *corpus*;
  - Busca de palavras que se destacam no *corpus*.
- Seleção de palavras para análise;
  - Delimitação de um número de palavras para possível análise.
- Análise da padronização das palavras selecionadas por meio de concordâncias, listas de colocados e n-gramas;
- Descrição da padronização das palavras selecionadas como resultado.
- Utilização dos padrões em objetivos seguintes (confecção de atividades, por exemplo).

A pequena metodologia descrita é ponto de partida inicial para uma gama de pesquisas em Linguística de *Corpus*. Os passos podem ser cíclicos e envolver algum tipo de recurso adicional durante o processo.

Ainda que seja uma metodologia básica, exige grande esforço e trabalho do pesquisador, tanto na preparação do material, *corpus*, como na análise, identificação de padrões, por exemplo.

Dado o conhecimento das principais ferramentas e noções metodológicas passaremos a seguir para uma reflexão sobre lacunas e pontos a serem otimizados no processo de pesquisa e possíveis oportunidades para pesquisa e desenvolvimento de recursos da adaptatividade em Linguística de *Corpus*.

VII. O microscópio da Linguística de *Corpus* pode ter aspectos adaptativos

Sem sobra de dúvida, o computador com suas ferramentas é imprescindível para pesquisas em Linguística de *Corpus*, e em muitas outras áreas, o que justifica a citação da metáfora do microscópio para a área. Muitas descobertas sobre a língua têm sido reveladas por meio de sua instrumentação, a ponto de se considerar uma nova maneira de estudar a língua.

Contudo, é preciso sempre fazer uma avaliação dos métodos e recursos utilizados a fim de obter avanços e facilitar o trabalho de pesquisa. É natural que se deseje um microscópio cada vez mais preciso, abrangente e dinâmico, que possa ser utilizado facilmente em diferentes tarefas.

O argumento é o de que muitas das tarefas executadas em pesquisas de Linguística de *Corpus* podem ser beneficiadas com a introdução do conceito de Adaptatividade, principalmente aplicações desenvolvidas para determinada atividade.

Novamente citando a pesquisa de [6], embora houvesse uma série de ferramentas disponíveis, as quais algumas delas já foram descritas, uma nova ferramenta foi criada especificamente para o trabalho de identificação de possíveis candidatos a termo, ZExtractor, disponível em http://www.fflch.usp.br/dl/li/x/?p=559, a qual utilizou funções similares às de concordanciadores e listadores de palavras. O fato de que nem sempre as ferramentas disponíveis serão totalmente adequadas às necessidades da pesquisa é natural e, por isso, há a necessidade de adaptá-las.

No caso, o que se tem feito é tornar as ferramentas adaptáveis, quando possível. Como é a ferramenta criada ZExtractor.

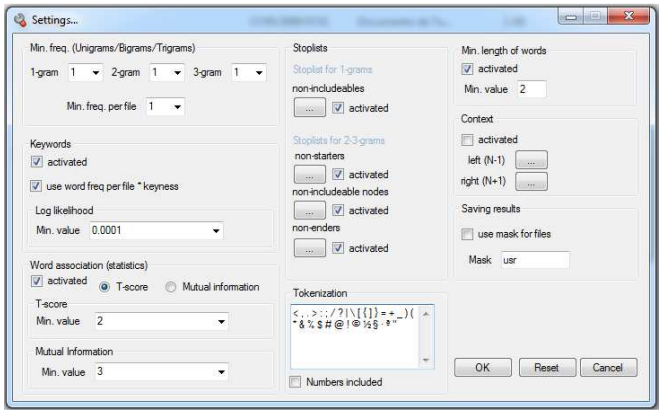


Fig.10. Configurações do Programa ZExtractor

O programa permite que o usuário faça uma série de ajustes para que a execução das funções possa retornar os resultados desejados ou próximos. As opções envolvem a definição de: frequência mínima de ocorrência do candidato no *corpus*, ocorrência mínima do candidato nos arquivos de texto do *corpus*, definição de *stoplists*, valor de corte de chavicidade, frequência do candidato por arquivo e estatísticas de associação de palavras.

Porém, as funções são estáticas e puramente quantitativas, dependendo da calibragem do pesquisador que analisará, a cada saída, os resultados obtidos. Não que seja algo errado, mas se há um modo que possivelmente melhore a execução das tarefas, é preciso investigá-lo, como a própria pesquisa citada o faz, ao testar as várias ferramentas e a partir delas desenvolver uma metodologia para extração de candidatos a termo.

Deste modo, fazemos uma reflexão sobre as ferramentas computacionais e metodologia básica descrita, com o objetivo de apontar alguns problemas enfrentados e possíveis pontos para o desenvolvimento de novas soluções para pesquisa. Ao fazê-lo, tentamos relacionar a pesquisas e trabalhos que utilizam o conceito da Adaptatividade.

VIII. Adaptatividade para pesquisas em Linguística de *Corpus*

Nesta seção, fazemos a descrição de alguns problemas e limitações em pesquisas de Linguística de *Corpus*, entendendo que o conceito de adaptatividade e técnicas adaptativas poderiam ser empregados para criação de soluções dinâmicas. É importante destacar que as descrições fornecidas estão sob a ótica de pesquisas relacionadas ao uso da Linguística de *Corpus* para o desenvolvimento de aplicações, cursos e materiais pedagógicos para o ensino de línguas.

Em relação à coleta e compilação de *corpus*, às vezes não é possível contar com uma grande quantidade de textos para um *corpus*, por várias restrições (direitos autorais, por exemplo), o que pode prejudicar sua utilização em determinada aplicação em que necessite de grandes quantidades de dados para obter um resultado confiável. É o caso de aplicações baseadas em dados estatísticos. O uso de dispositivos baseados em regras poderia auxiliar em uma solução.

Em muitas pesquisas, os objetivos, a metodologia de análise e aplicação podem ser os mesmos, tendo apenas a distinção do *corpus* e/ou gênero, como é o caso de algumas pesquisas que procuram extrair a padronização de palavras e fraseologias para uso pedagógico. Seria desejável o desenvolvimento de uma solução comum que pudesse lidar com as diferenças, sem que o pesquisador tenha que passar por todos os passos novamente. Em termos metodológicos, pode parecer que pesquisadores estejam fazendo sempre o mesmo esforço. O caminho poderia ser encurtado.

Algumas ferramentas computacionais para análise de *corpus* possuem limites de processamento de textos. O Wordsmith tools 3.0, por exemplo, no Concord, limita o número de linhas de concordância retornadas. Se a palavra ocorre mais do que o limite, não é possível analisá-la em sua totalidade, sendo necessário fazer um recorte e estimativas.

O recorte é muito comum nas pesquisas. Pode estar relacionado à quantidade de itens a ser analisada, tempo e esforço para a pesquisa. Contudo, todos esses fatores poderiam ser minimizados, sem que o pesquisador tivesse que executar a mesma pesquisa em dois momentos, como condição o aumento quantitativo dos itens analisados. O desenvolvimento de soluções de treinamento e aprendizado de máquina poderia auxiliar em tarefas que filtrassem itens ou executassem análises com dados previamente armazenados.

O recurso de extração de palavras-chave para filtragem de palavras a serem analisadas, que nas ferramentas aqui descritas são identificadas a partir de fórmulas estatísticas, poderia ser incrementado com regras dinâmicas para a separação, por exemplo, de palavras relacionadas especificamente ao assunto/conteúdo do texto, em relação às palavras comuns ao gênero dos textos do *corpus*, uma necessidade típica da pesquisa em Linguística de *Corpus*.

Retornando ao exemplo do programa ZExtractor, ao invés de especificar regras/configurações comuns para todas as palavras do corpus, o conjunto de regras poderia ser aplicado dinamicamente para cada possível candidato, podendo em alguns casos até solicitar a intervenção do usuário e memorizar sua ação para aplicações futuras.

O reconhecimento de padrões, uma das tarefas mais importantes de pesquisas em Linguística de *Corpus* poderia ser beneficiado por soluções adaptativas, uma vez que na área da Tecnologia Adaptativa há vários trabalhos nesse sentido.

Como já é de conhecimento, o computador lida melhor que o ser humano em tarefas repetitivas, em que o nível de atenção deve ser alto e contínuo. Descobertas interessantes em análises poderiam surgir.

A partir dessas poucas considerações, é possível afirmar que estudos que busquem adição do conceito e técnicas adaptativas em pesquisas da Linguística de *Corpus* poderiam transformar a maneira em que seu microscópio é utilizado.

Na seção a seguir, citamos aspectos de uma pesquisa em andamento que apresenta passos iniciais para o uso da Adaptatividade no desenvolvimento de um sistema de montagem automática de atividades de leitura em língua inglesa com *corpora*. O objetivo é salientar as possíveis contribuições da Tecnologia Adaptativa para a área de Linguística de *Corpus*, especificamente em relação à linha de pesquisa que se preocupa com o desenvolvimento de aplicativos e ferramentas para análise de corpora e ensino de línguas.

IX. Gerador de Atividades de Leitura Adaptativo

O desenvolvimento de um sistema de geração e montagem de atividades de leitura tem como objetivo prático de suprir a necessidade de professores que desejam utilizar materiais baseados em *corpora* em suas aulas, mas que não estão familiarizados com o uso de ferramentas de processamento e exploração de *corpora* e/ou que não possuem muito tempo para preparar atividades.

O projeto está baseado em um estudo realizado em uma pesquisa de mestrado [7], que teve como produto final um *software desktop* para preparação semiautomática de atividades de leitura em inglês, tomando como entrada um texto selecionado pelo usuário com fins pedagógicos e conduzindo-o, através de etapas, como um assistente eletrônico, até a publicação de uma unidade didática com exercícios baseados em concordâncias (*data-driven learning*), predição, léxico-gramática e questões para leitura crítica, utilizando análises automáticas do texto selecionado por meio de fórmulas estatísticas: lista de frequência, palavras-chave, possíveis palavras cognatas, etiquetagem morfológica, possíveis padrões (*clusters*) e densidade lexical do texto.

Nesse primeiro protótipo, com base no conceito de '*standard exercise*' (atividade padrão) introduzido em [8] para cursos de ESP (*English For Specific Purposes*), embora os resultados obtidos tenham demonstrado a viabilidade e o potencial da solução, há ainda a necessidade de muita pesquisa e desenvolvimento para melhorias: diminuição de erros de análise, aumento da variedade de exercícios disponíveis e adequação do conjunto de exercícios ao texto de entrada.

Tendo em vista tais necessidades, a atual pesquisa estuda a possibilidade do uso da Tecnologia Adaptativa para o desenvolvimento de uma aplicação dinâmica, em que o conjunto de exercícios seja variado, flexível e adequado à entrada (texto/gênero). Em contraponto a um modelo padrão de atividade, como no primeiro protótipo, propõe-se o conceito de um gerador adaptativo de exercícios.

O esquema representado a seguir ilustra as etapas básicas para o funcionamento do sistema para que seja possível a inclusão de técnicas adaptativas na geração automática de exercícios de leitura, tendo como partida um texto e opções pré-definidas de um usuário.

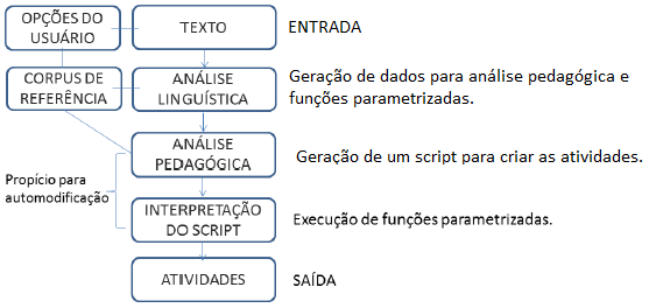


Fig. 11. Esboço para uma aplicação dinâmica

A implementação do sistema será feita utilizando a linguagem Python. Inicialmente, criamos um módulo para análise linguística de texto e corpora. Os dados da análise servirão posteriormente para tomadas de decisão em um módulo de análise pedagógica do texto, que auxiliará na composição de um script em xml, conforme a seguir:

```
<?xml version="1.0" ?>
<activity>
 <label content='Qual conjunto de palavras possui substantivos?\n' />
 <label content= '\na) ' /> <pos tag='NN' limit='5' random='true' order='none' delimiter=' - '
 case='upper' length='5'  />
 <label content= '\nb) ' /> <pos tag='JJ' limit='5' random='true' order='none' delimiter='; '
 case='upper' />
 <label content= '\nc) ' /> <pos tag='VB|JJ' limit='5' random='true' order='none' delimiter=' '
 case='upper' />
 <label content= '\nd) ' /> <pos tag='VB' limit='5'  />
 <label content='\n\n' />

 <label content='Qual das alternativas possui o maior numero de palavras cognatas?\n' />
 <label content= '\na) ' /><cognates limit='8' random='false' order='none' case='lower' length=
 '0' delimiter='; ' />
 <label content= '\nb) ' /><keywords limit='8' random='false' order='desc' case='lower' length=
 '0' stoplist='true'  delimiter='; ' />
 <label content= '\nc) ' /><keywords limit='8' random='false' order='asc' case='lower' length='0'
  stoplist='true'  delimiter='; ' />
 <label content= '\nd) ' /><keywords limit='8' random='true' order='asc' case='lower' length='4'
 stoplist='true'  delimiter='; ' />
</activity>
```

Fig. 12. Exemplo de script para criação de exercícios

Após a montagem do script, sua interpretação é realizada por meio de funções parametrizadas. O resultado final é a impressão de exercícios para as atividades:

```
Problems  Console
<terminated> C:\Users\home\Documents\workspace\TEXT\test_publish.py
Qual conjunto de palavras possui substantivos?

a) FLOES - COMPANIES - IMPLICATIONS - SCIENTISTS - DEVELOPMENT -
b) NEW; ACIDIC; TALLER; LAND-BASED; DARK;
c) FOUND LIKELY REPORTED WARMER OPEN
d) sent warmed affect said bears

Qual das alternativas possui o maior numero de palavras cognatas?

a) global; including; previous; percent; annual; absorbs; international; report;
b) arctic; ice; sea; report; normal; scientists; polar; u.s.;
c) summer; taller; clog; mingle; help; corps; increasingly; when;
d) setting; national; carbon; research; never; 2007,; until; at;
```

Fig. 13. Interpretação de script para criação de exercícios

As descrições do sistema em questão caminham para o entendimento e aplicação de técnicas adaptativas. A montagem dinâmica do script para a criação dos exercícios, com base nas análises linguística e pedagógica, considerando as opções definidas pelo usuário, pode conferir aspectos adaptativos ao sistema, o que elevariam o conceito de 'atividade padrão' a um conceito de 'atividade adaptativa' para elaboração de materiais pedagógicos para o ensino de leitura em língua estrangeira.

Embora haja muito trabalho a ser realizado, a investigação é importante para o estabelecimento de um diálogo concreto entre Linguística de *Corpus* e Tecnologia Adaptativa, especificamente no desenvolvimento de aplicações dinâmicas para análise de corpora e ensino de línguas.

X. Considerações Finais

O trabalho buscou mostrar as principais práticas e necessidades em pesquisas de Linguística de *Corpus*, em uma linha de desenvolvimento de ferramentas para análise de *corpus* e ensino de línguas, abordando questões de uso de ferramentas computacionais e metodológicas, as quais poderiam ser foco de investigação em trabalhos futuros com o diálogo entre áreas da Linguística de Tecnologia Adaptativa.

Podemos considerar as informações apresentadas como um primeiro esboço para tal diálogo. Esperamos que seja útil para fomentar uma série de discussões sobre o assunto.

REFERÊNCIAS

[1] Guariento W. & Morley J. (2001). Text and task authenticity in the EFL classroom. ELT Journal. 55/4 October, 1-7.

[2] GILMORE, A. (2004). A comparison of textbooks and authentic interactions. ELT Journal,58/4 October, 1-12.

[3] DIZERÓ, W. Formalismos Adaptativos Aplicados na Modelagem de *Software*s Educacionais. Tese de Doutorado, EPUSP, São Paulo, 2010.

[4] PADOVANI, D.; CONTIER, A.; JOSÉ NETO, J. J.; Tecnologia Adaptativa Aplicada ao Processamento da Linguagem Natural. In: WTA 2010; Quarto Workshop de Tecnologia Adaptativa, 2010, São Paulo. Memórias do WTA 2010: Quarto Workshop de Tecnologia Adaptativa. São Paulo: Laboratório de Linguagens e Técnicas Adaptativas, 2010. p. 35-42.

[5] BERBER SARDINHA, T. (2004) *Linguística de Corpus*. São Paulo: Manole.

[6] SILVA E TEIXEIRA, R.B. Termos de (Onco)mastologia: uma abordagem mediada por corpus. 2010. Dissertação de mestrado. Pontifícia Universidade Católica de São Paulo.

[7] MOREIRA FILHO, P. Desenvolvimento de um software para preparação semiautomática de atividades de leitura em inglês. Dissertação de Mestrado Inédita, LAEL, PUC-SP, 2007.

[8] Disponível em: < http://wordsmithtools.com/downloads/corpus_linguistics/Standard%20Exercise.pdf>. Acesso em 01/02/2013.

José Lopes Moreira Filho é Doutorando em Semiótica e Linguística Geral (USP). Possui Mestrado em Linguística Aplicada e Estudos da Linguagem pela Pontifícia Universidade Católica de São Paulo (PUCSP). Possui graduação em Letras – Português e Inglês (Bacharelado Tradução) pela Universidade de Mogi das Cruzes (UMC). Atualmente, é Professor Coordenador de Centro de Línguas da Diretoria Regional de Ensino da SEE-SP, mantendo interesses na área de Linguística, Linguística Aplicada, Linguística Informática, Linguística de *Corpus*, Processamento de Linguagem Natural, atuando principalmente no desenvolvimento de ferramentas computacionais para exploração de *corpora*, ensino de línguas, entre outras aplicações que envolvem linguagem e tecnologia.

Zilda Maria Zapparoli é professora associada aposentada junto ao Departamento de Linguística da Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo, instituição em que obteve os títulos de Mestre, Doutor e Livre-Docente, e onde continua desenvolvendo atividades de ensino, pesquisa e orientação no Curso de Pós-Graduação em Linguística, área de Semiótica e Linguística Geral, linha de pesquisa Informática no Tratamento de *Corpora* e na Prática da Tradução. Desde 1972, atua em Linguística Informática, com tese de doutorado, tese de livre-docência, pós-doutorado na Université de Toulouse II e trabalhos publicados na área. É líder do Grupo Interdisciplinar de Pesquisas em Linguística Informática, certificado pela USP e cadastrado no Diretório de Grupos de Pesquisa no Brasil do CNPq, em 2002. Integrou comissões e colegiados na USP, destacando-se os trabalhos relativos ao processo de informatização da FFLCH-USP, enquanto membro da Comissão Central de Informática da USP e presidente da Comissão de Informática da FFLCH-USP por cerca de treze anos.

Parte II

Transcrição da mesa redonda

A transcrição da mesa redonda está em fase de revisão. Em breve, esta seção será substituída pelo texto adequado.