Integração da Plataforma SIGUS com a Ferramenta de Apoio a Aplicação de Tecnologia Adaptativa AdapTools

Denys G. Santos[1], **Hemerson Pistori**[1]
Amaury A. C. Junior[1], **João José Neto**[2]

[1]Departamento de Engenharia de Computação
Universidade Católica Dom Bosco
Av. Tamandaré, 6000, Jardim Seminário
79117-900 Campo Grande, MS

[2]Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158, tv. 3
05508-900 São Paulo, SP

gsdenys@gmail.com, pistori@ucdb.br, amaury@ec.ucdb.br, joao.jose@poli.usp.br

Abstract. *SIGUS is a platform of support to the development systems for digital inclusion of deficient people. Currently, this platform uses, to the machine learning, the WEKA application. Alternatively, this operation would be performed by adaptive automata in a tool named Adaptools. This paper describes the integration process among the SIGUS platform and the AdapTools.*

Resumo. *SIGUS é uma plataforma de apoio ao desenvolvimento de sistemas para a inclusão digital de pessoas portadoras de necessidades especiais. Atualmente, a plataforma SIGUS utiliza, para o aprendizado de máquinas, o aplicativo WEKA. Outra forma de realizar esse aprendizado é através de autômatos adaptativos com o auxílio da ferramenta AdapTools. Este artigo descreve o processo de integração entre a plataforma SIGUS e o AdapTools.*

1. Introdução

A plataforma de apoio ao desenvolvimento de sistemas para inclusão digital - SIGUS - conta com várias ferramentas que auxiliam na aprendizagem de máquina e no processamento de imagens. Atualmente o aprendizado de máquina é feito através do WEKA [6], que gera o conjunto de treinamento e faz a classificação, tendo como entrada parâmetros extraídos da imagem [5].

Essa classificação dos dados pode ser feita através de dispositivos adaptativos, tais como, os autômatos adaptativos, treinados para reconhecer cadeias de caracteres geradas pelos parâmetros discretos extraídos da imagem. A execução destes autômatos pode ser simulada pelo AdapTools, que é uma ferramenta de apoio ao uso da tecnologia adaptativa.

Um dispositivo adaptativo é constituído de um dispositivo subjacente de estrutura estática (autômatos, gramáticas [7, 8], árvores de decisão, entre outros) ao qual é acrescido um mecanismo adaptativo, que permite que a estrutura do dispositivo subjacente seja modificada dinamicamente. As primeiras aplicações da tecnologia adaptativa,

derivadas da teoria dos dispositivos adaptativos [9], concentram-se na área de construção de compiladores e, como quase toda a tecnologia envolvida nessa área, também já estão sendo utilizadas em diversas outras áreas da computação. Entre elas, podemos citar o processamento de linguagens naturais, robótica e visão computacional e problemas de reconhecimento de padrões [12].

Por sua natureza dinâmica, formalismos baseados em dispositivos adaptativos, como por exemplo, os autômatos adaptativos (Seção 2.), representam uma alternativa atraente na modelagem de software adaptativo. Neste artigo, descreve-se o AdapTools, os autômatos adaptativos e a integração entre o AdapTools[1] e a plataforma SIGUS, apresentando as modificações necessárias do AdapTools e da plataforma SIGUS para que a integração entre as duas se torne mais estáveis, rápida e transparente.

A Seção 2. define os autômatos adaptativos, que podem ser simulados no AdapTools. Para isso, inicia-se definindo, de forma mais geral, os dispositivos guiados por regras e de dispositivos guiados por regras adaptativos. Na seção 3. descreve-se a ferramenta AdapTools, através de suas características e aspectos de implementação. As Seções 4. e 5. descrevem, respectivamente, a ferramenta SIGUS e a sua integração com o AdapTools.

2. Autômatos Adaptativos

O conceito de *dispositivo guiado por regras* generaliza a formalização de uma série de dispositivos, como por exemplo, autômatos finitos, autômatos de pilha e máquinas de Turing, que compartilham a característica fundamental de terem sua operação definida por uma conjunto fixo e finito de regras [7, 8]. Essas regras mapeiam cada possível configuração do dispositivo em uma nova configuração, levando em consideração um determinado *estímulo de entrada* e gerando algum *símbolo de saída*. Um dispositivo guiado por regras inicia seu funcionamento em uma determinada configuração, e segue aplicando sucessivamente uma regra do seu conjunto de regras, alternando entre as possíveis configurações, até que não existam mais estímulos de entrada ou até que se atinja uma configuração à qual nenhuma regra possa ser aplicada [9, 12].

Em um *dispositivo guiado por regras adaptativo*, ou simplesmente *dispositivo adaptativo*, o conjunto de regras passa a poder variar durante a leitura dos estímulos de entrada. Esta variação, no entanto, é completamente determinada por um outro nível de regras. Neste segundo nível, as *funções adaptativas* agem sobre o conjunto de regras original, modificando-o através da remoção e inserção de novas regras. Temos assim um dispositivo com duas camadas, a primeira, denominada *camada subjacente*, é representada por um dispositivo guiado por regras (não-adaptativo) e pelas *ações adaptativas*, que correspondem às chamadas de funções adaptativas do segundo nível. Ao segundo nível, dá-se o nome de *camada adaptativa*, constituída pelo conjunto de funções adaptativas. Dessa forma, define-se um mecanismo universal, capaz de transformar um dispositivo qualquer, não adaptativo, mas guiado por regras, em um dispositivo capaz de alterar sua estrutura interna (conjunto de regras) durante sua operação, a partir do acréscimo da camada adaptativa.

Um autômato adaptativo é um dispositivo guiado por regras adaptativo em que a camada subjacente consiste de um autômato de pilha estruturado e as ações adaptativas são implementadas através de *funções adaptativas*. As funções adaptativas determinam

[1] http://www.ec.ucdb.br/∼pistori/adaptools - seç ˜ao *Download AdapTools*

exatamente quais modificações devem ser realizadas na camada subjacente do dispositivo. No caso dos autômatos adaptativos, tais ações determinam quais as transições do autômato deverão ser removidas ou inseridas. O núcleo de uma função adaptativa consiste de uma lista de *ações adaptativas elementares*, que podem ser de três tipos: *ações elementares de consulta (?)*, que possibilitam a busca de padrões na estrutura definida pelas regras da camada subjacente, e as *ações elementares de inserção (+)* e de *remoção (-)*, que determinam, respectivamente, as regras que deverão ser inseridas ou removidas do conjunto de regras corrente da camada subjacente [12].

Uma função adaptativa pode ser formalmente descrita como uma 9-upla $FA = (F, P, V, G, C, R, I, A, B)$, onde:

- F é o nome da função adaptativa;
- P é uma lista de parâmetros formais;
- V é uma lista de identificadores de variáveis;
- G é uma lista de identificadores de geradores;
- C é uma lista de ações elementares de consulta;
- R é uma lista de ações elementares de remoção;
- I é uma lista de ações elementares de inserção;
- A é uma ação adaptativa inicial (opcional), e;
- B é uma ação adaptativa final (opcional).

As ações adaptativas são descritas na forma $AA = (F', P')$, onde F' é o nome da função adaptativa e P' é a lista de argumentos a serem passados para F'. A execução deste tipo de dispositivo se dá inicialmente com a execução de uma ação adaptativa inicial, caso exista, seguida da execução das ações elementares de *consulta, remoção e inserção*, nesta ordem. Por fim, se existir, é executada a ação adaptativa final [12].

As ações adaptativas inicial e final não são implementadas no AdapTools. No entanto, a ferramenta inclui a implementação das ações adaptativas anteriores e posteriores, que podem ser executadas, respectivamente, antes e depois da chamada de uma função adaptativa. A existência de tais ações também é opcional.

A Figura 1 demonstra a estrutura de uma autômato adaptativo de estados finitos que faz o balanceamento de parênteses. Este, apresenta inicialmente, dois estados *q0* e *q1*, o estado *q0* (inicial) apresenta um *"loop"* para a entrada *"("* que chama a função adaptativa *.A*, e uma transição para a *q1* quando a entrada for cadeia vazia (ϵ). Durante a execução deste autômato, para toda entrada *"("* a estrutura inicial do autômato é replicada internamente na estrutura atual do mesmo, fazendo com que a cada parêntese aberto haja um novo estado, acessível através de uma transição *")"*, o que nos garante que se o autômato chegou ao estado final necessariamente este contém um fecha parênteses *")"* para cada abre parênteses *"("* lido [1].

3. Adaptools

O Adaptools é um ambiente computacional que permite a implementação, depuração e testes de autômatos adaptativos. O núcleo deste sistema é composto por uma máquina virtual que executa uma versão levemente modificada de um autômato adaptativo. Essas pequenas modificações no formalismo original facilitam a especificação de transições

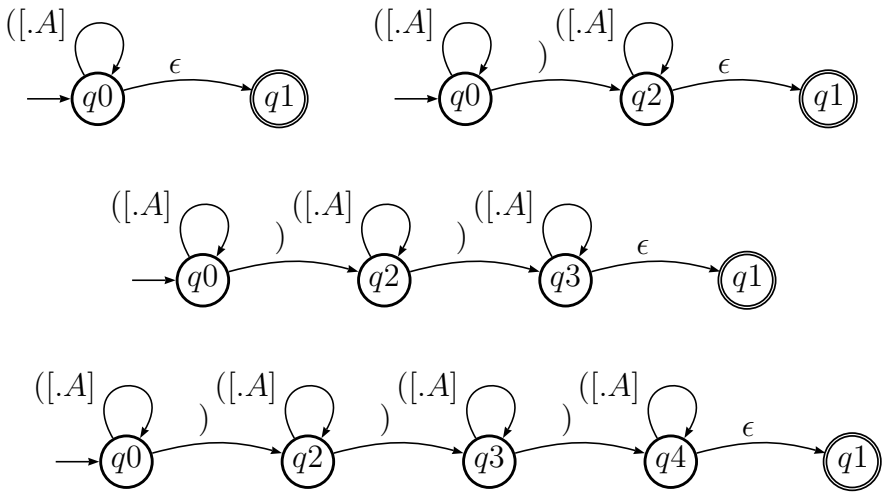

Figura 1. Execuç ˜ao de um autˆomato adaptativo de balanceamento de parˆenteses para a entrada: *((()))*.

internas, transições externas e das ações adaptativas elementares. Com isso, todos os elementos do dispositivo passam a ser apresentados através de uma única tabela. O ambiente computacional inclui ainda recursos de depuração, visualização de variáveis e pilhas e animação gráfica, bem como mecanismos para controle de projetos, execução simultânea de múltiplos dispositivos (com comunicação entre si) e uma série de exemplos de autômatos. Visto que os tradicionais autômatos de estados finitos, não-determinísticos com transições vazias (ϵ-automata) [7, 8] e os autômatos a pilha estruturados são especializações dos autômatos adaptativos, o Adaptools pode ser usado também como um laboratório virtual, através do qual pode-se praticar experimentalmente conceitos da teoria da computação. O autômato produzido pode, ainda, ser simulado na própria ferramenta através do ambiente gráfico mostrado na Figura 2 [3, 4, 12].

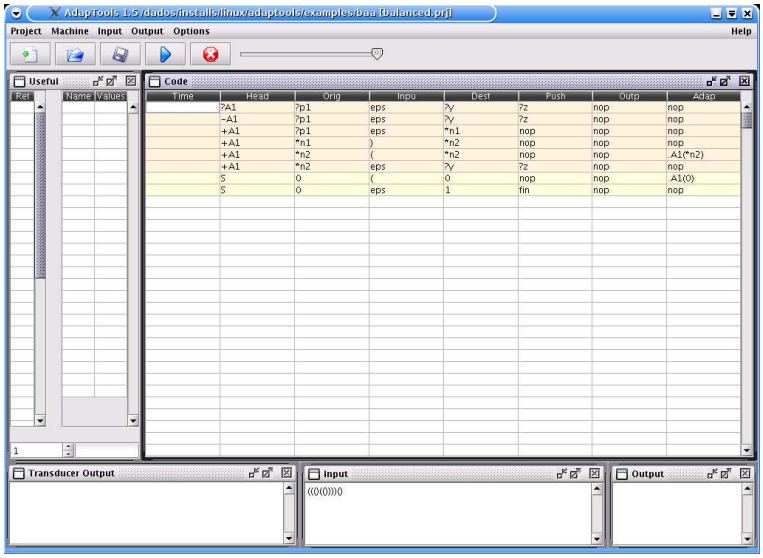

Figura 2. Janela principal do Adaptools

3.1. Aspectos de Implementação

No AdapTools os autômatos são representados através da tabela de transição. Esta tabela é composta por sete colunas: a coluna (1) (*Head*) dá um nome à sub máquina ou da função

adaptativa contida na linha. Quando o conteúdo desta coluna for o nome de uma função adaptativa, estas devem ser identificadas pelas ações adaptativas elementares consulta ($?$), inserção ($+$) e remoção ($-$). A coluna (2) (*Orig*), bem como a coluna (3) (*Dest*), mantém identificadores que correspondem, respectivamente, aos estados de origem e destino da transição. A coluna (4) (*Inpu*) mantém o símbolo que pode ser lido a partir de um estado incluindo transições vazias (ϵ). A coluna (5) (*Push*) contém um endereço de chamada de sub-máquina. A coluna (6) (*outp*) é utilizada para enviar uma saida para um transdutor. A coluna (7) (*Adap*) pode ser utilizada para a chamada de funções adaptativas [4, 3].

Visando a não utilização de estruturas de dados extras, o AdapTools foi codificado de forma que possa ser utilizado uma série de palavras reservadas que estão descritas abaixo.

eps: Representa cadeia vazia (ϵ).
nop: Nas colunas *Push*, *Outp* e *Adap*, indica função está inativa.
pop: Na coluna *Dest* indica uma transição de retorno de sub-máquinas.
fin: Na coluna *Push* indica que o estado de destino é um estado final.
spc: Na coluna *Inpu*, representam os caracteres ASCII.
digit: Representa o intervalo [0..9].
letter: Representa os intervalos [a..z] e [A..Z]
special: símbolos diferentes de letras e números.
other: Símbolo que não possa ser consumido estando no estado de origem da transição.
a..z: Faixa de valores ASCII de uma letra até uma outra.

3.1.1. Características do sistema

A codificação do AdapTools separa com muita clareza a máquina virtual que implementa o mecanismo subjacente, a parte adaptativa e os tipos de dados de apoio a execução [3].

Os tipos de dados de apoio a implementação são constituídos pela classe abstrata *Vmds*[2], que é implementado por outras duas classes: a *Viewer* que implementa a parte gráfica do AdapTools e a *VmdlImpl* que da suporte ao ambiente textual (*Runner*).

A implementação da classe *Runner*, além de possibilitar a execução textual, permite também que outros softwares desenvolvidos em Java utilizem o AdapTools em sua estrutura, através da chamada:

String args[ ]={"-m", Machine_File, "-I", Input_String};
String Return_Data = new Runner().run(args);

onde é criada uma cadeia que informa o arquivo da máquina de entrada e a seqüência a ser processada. A cadeia é, então, processada através da chamada do método run da classe $Runner$.

4. Plataforma SIGUS

SIGUS é uma plataforma de apoio ao desenvolvimento de softwares para a inclusão digital de pessoas portadoras de necessidades especiais. Esta plataforma tem como principal objetivo incentivar e facilitar o aumento da produção de softwares que dêem suporte para o uso por parte de portadores de necessidades especiais [5, 11].

[2]Virtual Machine Data Structures

A plataforma SIGUS proporciona ao programador um conjunto de classes e métodos para facilitar o uso de dispositivos de captura de vídeo, sobre os quais pode-se aplicar as técnicas de processamento de imagens e aprendizagem de máquinas. Portanto, a plataforma SIGUS integra técnicas e tecnologias para facilitar o desenvolvimento de aplicações que dêem suporte para uso de pessoas portadores de necessidades especiais.

Com o auxilio desta plataforma o programador pode desenvolver softwares (na linguagem java) para a inclusão digital de pessoas com necessidades especiais, como por exemplo, um software para controle de uma cadeira de rodas através de movimentos da cabeça ou dos olhos, tradutor de uma linguagem de sinais para um linguagem escrita ou oral, jogos interativos e outros.

Atualmente, a plataforma SIGUS é composta por um conjunto de classes para fazer o processamento de imagens e a aprendizagem de máquina. Estas classes interagem com aplicativos como o IMAGEJ, que faz a validação dos métodos de processamento de imagens, o JMF, responsável pela captura de vídeo e WEKA para aprendizado de máquina.

Pistori [12] descreve dois protótipos que utilizam da plataforma SIGUS, o TTT (*Tic Tac Toe*), que pode ser jogado utilizando apenas movimento com a cabeça, e o LIBRAS, que quando treinado pode reconhecer as letras, números e símbolos da linguagem brasileira de sinais a partir de imagens capturadas por uma *webcam*.

5. Integração SIGUS-Adaptools

Como mostrado em [10, 13] os autômatos adaptativos têm um enorme potencial para o reconhecimento de linguagens, incluindo-se as linguagens naturais. Esse potencial foi empregado para o reconhecimento de sinais visuais na plataforma SIGUS, através de uma integração com o software AdapTools.

As técnicas de aprendizagem de máquina na plataforma SIGUS, antes da integração, eram desenvolvidas somente através da ferramenta WEKA, através da qual gerava-se um conjunto de treinamento e, a partir desse conjunto, a ferramenta WEKA classificava a imagem capturada. O conjunto de treinamentos consiste de uma cadeia de caracteres, extraída da imagem, seguida de seu significado. Dessa forma, pode-se modelá-lo através de um autômato adaptativo e a ferramenta AdapTools pode substituir a ferramenta WEKA, na etapa de aprendizagem, fazendo assim a classificação da imagem.

A integração tornou-se possível através da criação das classes *SigusEffectAdaptools*, obtida a partir da classe *SigusEffect* e *Larner_Adaptools*, obtida a partir da classe *Larner_*. A primeira é responsável pela integração entre os módulos da plataforma SIGUS e a segunda pela classificação da imagem. Essas Alterações permitiram que a plataforma SIGUS, que antes utilizava somente o WEKA, também pudesse utilizar o AdapTools para processar os resultados extraídos do processamento da imagem capturada pela *webcam*.

A classe *SigusEffectAdapTools* faz a chamada da máquina (autômato) que será utilizada para processar a cadeia de caracteres que representa o conjunto de características extraídas da imagem. A classe *Larner_Adaptools* executa a ferramenta AdapTools sobre a máquina especificada na classe anterior, processando a cadeia gerada através dos parâmetros extraídos da imagem. Este era o processo que era realizado, anteriomente, somente pelo WEKA.

Além das duas alterações descritas, foi criada a classe *Discretize* para fazer a discretização da cadeia constituída pelos parâmetros extraídos da imagem, o que torna mais fácil a tarefa de criação do autômato que irá processar essa cadeia.

Para validar a integração foi desenvolvido mais dois protótipos baseados nos já existentes, esses novos protótipos são o TTT_Adaptools e o LibrasAdaptools.

6. Conclusão

Atualmente, embora a teoria adaptativa esteja bem desenvolvida, existe ainda a necessidade do desenvolvimento de ferramentas de software que façam uso mais geral desses conceitos. O desenvolvimento e a implementação de dispositivos adaptativos concentram-se fortemente em projetos de aplicação ou mecanismos subjacentes específicos e que, muitas vezes, não podem ser facilmente utilizados em outras situações. Nesse contexto, a integração com o AdapTools mostra-se como uma alternativa interessante, uma vez que trata-se de uma máquina virtual, cuja "linguagem de máquina"descreve um autômato adaptativo e é capaz de simular seu modo de operação, permitindo o projeto e a produção de código que possa ser simulado para uso em diferentes domínios de aplicação.

A integração do AdapTools com a plataforma SIGUS contribuiu com o aumento da variedade de ferramentas que podem ser utilizadas. A integração não foi total, pois, neste momento, não é possível o acesso ao ambiente gráfico do AdapTools através da plataforma SIGUS. No entanto, esta melhoria pode ser implementada em futuras versões da plataforma.

Entre as possíveis propostas de melhoria, também podemos destacar o tratamento de não-determinismos [7], a criação de um compilador que converte uma linguagem de mais alto nível de abstração para a notação do AdapTools, o desenvolvimento de uma versão distribuída do AdapTools [2] e a criação de uma interface gráfica para a plataforma SIGUS.

Referências

[1] Zuffo F. A. Utilização de autômatos adaptativos para tradução texto-voz. In *Departamento de Engenharia de Computação*, Campo Grande, MS, Brasil, Dezembro 2004.

[2] M. V. Steen A. S. Tunenbaum. *Distributed Systems*. Pearson Education, New Jersey, USA, 2002.

[3] J.J. Neto H. Pistori. Adaptools: Aspectos de implementação e utilização. *Boletim técnico PCS - USP*, 2003.

[4] J.J. Neto e E.R. Costa H. Pistori. A free software for the development of adaptive automata. *IV International Forum on Free Software*, 2, 2003.

[5] J.J. Neto e E.R. Costa H. Pistori. Utilização de tecnologia adaptativa na detecção da direção do olhar. *SPC Magazine*, 2, 2003.

[6] E. Frank I. H. Writen. *Data Mining: Pratical Machine Learner Tools end Techiniques with Java Implementations*. Academic Press, USA, 2002.

[7] R. Motwani J. E. Hopcroft, J. D. Ullman. *Introdução de Autômatos, Linguagens e Computação*. Campus, São Paulo SP, Brasil, 2002.

[8] P. B. Menezes. *Linguagens Formais e Automatos*. Sagra Luzzatto, Porto Alegre RS, Brasil, 2002.

[9] J. J. Neto. Adaptative rule-driven devices - general formulation anda case study. In *CIAA'2001 Sixth International Conference on Implementation and Application of Automata*, pages 234–250, Pretoria, South Africa, July 2001.

[10] J. J. Neto and Miryam de Moraes. Formalismo adaptativo aplicado ao reconhecimento de linguagem natural. *Anais da Conferencia Iberoamericana en Sistemas, Cibernética e Informática*, July 2002.

[11] H. Pistori. Plataforma de apoio ao desenvolvimento de sistemas guiados por sinais visuais. http://www.gpec.ucdb.br/sigus, 2004.

[12] Hemerson Pistori. *Tecnologia Adaptativa em Engenhara de Computação: Estado da Arte e Aplicações*. PhD thesis, Universidade de São Paulo, São Paulo, Brasil, 2003.

[13] C. Y. O. Taniwaki and J. J. Neto. Autômatos adaptativos no tratamento sintático de linguagem natural. Technical report, PCS-POLI-USP, São Paulo, Brasil, 2001.