Memórias do WTA 2017

XI 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
2017

Ficha catalográfica

Workshop de Tecnologia Adaptativa (11: 2017: São Paulo)
Memórias do WTA 2017. – São Paulo; EPUSP, 2017. 138p.

ISBN 978-85-86686-90-0

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 décima edição primeira do Workshop de Tecnologia Adaptativa realizou-se em São Paulo, Brasil, nos dias 26 e 27 de Janeiro de 2017, 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 ideias 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 ideias 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 (Ponta Porã, MS, Brasil)
- André Riyuiti Hirakawa (São Paulo, SP, Brasil)
- Angela Hum Tchemra (São Paulo, SP, Brasil)
- Anna Helena Reali Costa (São Paulo, SP, Brasil)
- Aparecido Valdemir de Freitas (São Paulo, SP, Brasil)
- Carlos Eduardo Cugnasca (São Paulo, SP, Brasil)
- Claudia Maria Del Pilar Zapata (Lima, Peru)
- Elisângela Silva da Cunha Rodrigues (Ponta Porã, MS, Brasil)
- Fabrício Augusto Rodrigues (Ponta Porã, MS, Brasil)
- Hemerson Pistori (Campo Grande, MS, Brasil)
- Ítalo Santiago Vega (São Paulo, SP, Brasil)
- João Eduardo Kögler Junior (São Paulo, SP, Brasil)
- Jorge Rady de Almeida Junior (São Paulo, SP, Brasil)
- Leoncio Claro de Barros Neto (São Paulo, SP, Brasil)
- Líria Matsumoto Sato (São Paulo, SP, Brasil)
- Marcos Pereira Barretto (São Paulo, SP, Brasil)
- Marcus Vinícius Midena Ramos (Petrolina, PE, Brasil)
- Ricardo Luís de Azevedo da Rocha (São Paulo, SP, Brasil)
- Ricardo Nakamura (São Paulo, SP, Brasil)
- Sérgio Donizetti Zorzo (São Carlos, SP, Brasil)
- Sidney Viana (São Paulo, SP, Brasil)
- Wagner José Dizeró (Lins, SP, Brasil)

Comissão Organizadora

- André Riyuiti Hirakawa (São Paulo, SP, Brasil)
- João José Neto, Chair (São Paulo, SP, Brasil)
- Newton Kiyotaka Miura (São Paulo, SP, Brasil)
- Paulo Roberto Massa Cereda (São Paulo, SP, Brasil)
- Ricardo Luís de Azevedo da Rocha (São Paulo, SP, Brasil)

Promoção do Evento

- LTA – Laboratório de Linguagens e Técnicas Adaptativas
- PCS – Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP

Apoio

- Escola Politécnica da Universidade de São Paulo (EPUSP)
- FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo, processo número 2016/21483-8
- IEEE – Institute of Electrical and Electronics Engineers
- Olos Tecnologia e Sistemas Ltda.
- Pagga Tecnologia de Pagamentos Ltda.
- São Paulo Convention & Visitors Bureau
- SBC – Sociedade Brasileira de Computação
- SPC – Sociedad Peruana de Computación
- Universidade de São Paulo

Memórias do WTA 2017

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 2017, o Décimo Primeiro Workshop de Tecnologia Adaptativa, realizado em São Paulo nos dias 26 e 27 de Janeiro de 2017. A exemplo da edição de 2016, 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 compilamos neste documento estas memórias com os artigos e textos técnicos apresentados no WTA 2017 – Décimo Primeiro Workshop de Tecnologia Adaptativa, realizado na Escola Politécnica da Universidade de São Paulo nos dias 26 e 27 de Janeiro de 2017.

Esta edição contou com 20 trabalhos, entre comunicações técnicas e artigos, relacionados à área de Tecnologia Adaptativa e aplicações nos mais diversos segmentos. Adicionalmente, foram ministrados dois tutoriais, a saber: *DInAton A^3: Adaptatividade em Ambientes de Aprendizagem*, por Ítalo Santiago Vega, e *Introspecção e compilação tardia em Python*, por Danilo de Jesus da Silva Bellini. O evento registrou uma participação efetiva de uma centena de pesquisadores durante os dois dias, constatando-se o sucesso do mesmo.

Esta publicação

Estas memórias espelham o conteúdo apresentado no WTA 2017. A exemplo do que foi feito no ano anterior, todo o material referente aos trabalhos apresentados no evento estará acessível no portal do WTA 2017, 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 2018.

Agradecimentos

Às instituições que apoiaram o WTA 2017, 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 às seguintes pessoas pelo valioso auxílio para a viabilização das necessidades locais do evento:

Apoio administrativo

- Ákio Nogueira Barbosa
- André Rodrigues Ribeiro
- Leia Sicília

Apoio operacional

- Daniel Costa Ferreira
- Edson de Souza
- Ian Silva Oliveira
- Haroldo Issao Guibu
- Newton Kiyotaka Miura
- Nilton Araújo do Carmo
- Renata Luiza Stange
- Rodrigo Kavakama
- Suellen Alves

Divulgação

- Amanda Rabelo
- Otávio Nadaleto

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 décima primeira edição do Workshop de Tecnologia Adaptativa. Também agradecemos ao Departamento de Engenharia de Computação e Sistemas Digitais pela disponibilização do auditório para a realização do evento e à Fundação de Amparo à Pesquisa do Estado de São Paulo pelo apoio financeiro. Por fim, agradecemos aos engenheiros Newton Kiyotaka Miura e Paulo Roberto Massa Cereda pelo valioso auxílio na organização.

São Paulo, 27 de Janeiro de 2017

João José Neto
Coordenador geral

Processo número 2016/21483-8, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)

As opiniões, hipóteses e conclusões ou recomendações expressas neste material são de responsabilidade do(s) autor(es) e não necessariamente refletem a visão da FAPESP.

Sumário

Lista de autores

Parte I

Comunicações técnicas

Botão *e-Call* adaptativo para segurança de cargas

B. S. Paterlini, L. B. Abrahão, J. J. Neto

***Resumo*— O tema em questão é a segurança pública e privada. Dentro deste contexto foi selecionado a sinalização instantânea em caso de situações de risco em geral, com foco principal no roubo de cargas. A partir do dispositivo de emergência denominado *e-call* (botão de pânico) é proposto um mecanismo adaptativo que auxilia motoristas e frotistas a selecionarem rotas mais seguras baseadas em informações atuais de acionamento do dispositivo por outros usuários, índice de roubo por tipo de carga, situação de tráfego da via e dados oficiais da Secretaria de Segurança Pública do Estado de São Paulo (SSP-SP).**

Keywords*— *e-call, botão de pânico, adaptativo, segurança de carga.

I. Introdução

A. Contexto Geral

Segundo dados da Associação Nacional do Transporte de Cargas e Logística (NTC) baseado em dados das Secretarias de Segurança Pública estaduais o estado de São Paulo concentrou 44,1% dos 19250 casos de roubo de carga que foram registrados no Brasil durante o ano de 2015. Prejuízo superior a R$1 bi [1]. Dados da Federação das Empresas de Transporte de Carga do Estado de São Paulo (FETCESP) apontam no 1º semestre de 2016 um todas de 4398 ocorrências no estado, sendo que 59,41% ocorreram na capital [2].

A secretaria de Segurança Pública do Estado de São Paulo (SSP-SP) divulga os dados oficiais de diversos tipos de crimes ocorridos divididos por regiões, permitindo mapear regiões com maior risco potencial de ocorrência de determinado crime, como ilustrado no gráfico da *Fig. 1*.

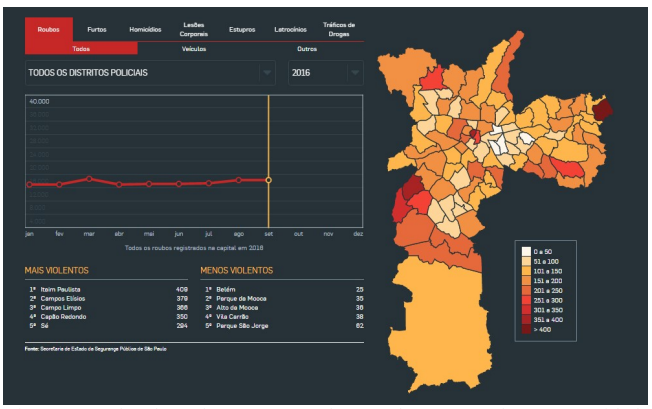


Figura 1. Dados de roubos até o mês de setembro no ano de 2016 na cidade de São Paulo. Fonte: Secretaria de Segurança Pública de São Paulo

Porém percebe-se que estes dados são pouco dinâmicos o que prejudica a qualidade desta informação, principalmente quando a velocidade com que essa informação é distribuída é fator determinante para que uma ação corretiva seja efetiva para aumentar a segurança de pessoas ou de uma carga em transporte. Para que a latência na comunicação desta informação não ocorra, aplicativos como "Onde fui roubado" auxiliam de grande forma as pessoas a evitar situação de alto risco, uma vez que é possível não somente identificar valores estatísticos de quantidade de roubos que aconteceram em determinado local por período, mas também visualizar denúncias realizadas a poucos instantes. Caso o aplicativo identifique uma denúncia em local próximo é emitida uma notificação ao usuário. Vide na *Fig. 2* que além de dados quantitativos o perfil do usuário e uma breve descrição na denúncia aumentam muito a qualidade da informação tornando-a muito mais completa.

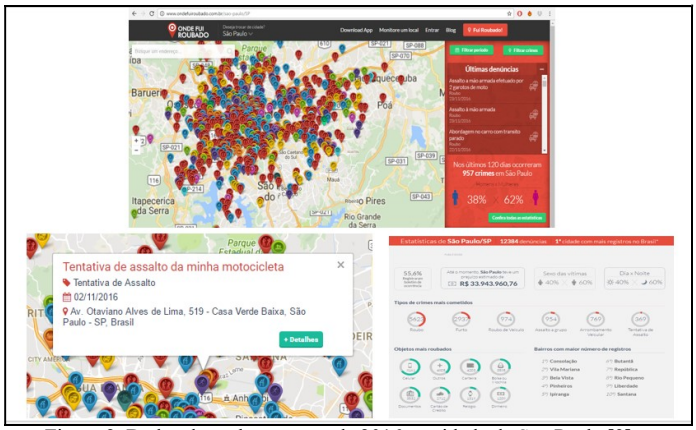


Figura 2. Dados de roubos no ano de 2016 na cidade de São Paulo [3].

B. Contexto das Transportadoras

As empresas de transporte apresentam grande interesse em sistemas de rastreamento e segurança de cargas. Grande partes dos caminhões de transporte de cargas possuem módulos de rastreamento com comunicação via GSM e GPS. Um dispositivo de uso frequente atrelado a estes módulos é o Botão de Pânico (*e-Call*) que apresenta pequena dimensão e é instalado no interior do veículo. Ao ser pressionado envia a uma Central de Controle e Operações (CCO) um alerta de uma situação de ameaça ou risco. Entre o condutor e a CCO é elaborado um roteiro de ações a serem executadas quando o alerta é recebido, que vão desde a tentativa de comunicação

com o motorista até a imobilização total do veículo. A *Fig.3* ilustra o conceito básico de rastreamento veicular.

Figura 3. Conceito básico de rastreamento veicular [1].

Estes módulos possuem dois modos de operação principais: normal e evento. O modo evento é ativado em caso de detecção de remoção da bateria interna ou quando o botão de emergência é pressionado como ilustra o diagrama da *Fig. 4*.

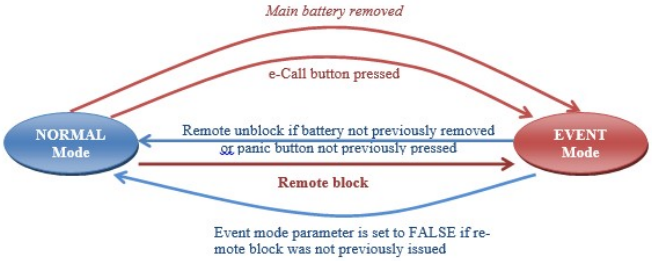


Figura 4. Modos de operação do módulo de rastreamento. Fonte: autores

Quando modo evento é ativado uma sequência de ações podem ser tomadas e é neste modo de operação que é proposto o mecanismo de adaptatividade para que além das ações corretivas executadas pela CCO sejam tomadas ações preventivas baseadas em diversos dados alimentados dinamicamente no sistema.

II. ADAPTATIVIDADE PARA SEGURANÇA DE CARGAS

Para determinar um modelo adaptativo que insira dinamismo e aumente a qualidade da informação para auxiliar a tomada de decisões em situações de potencial risco no transporte de cargas, devem ser levados em consideração alguns pontos:

- As cargas possuem entre si um grau de interesse diferente por parte dos assaltantes. Não necessariamente uma carga de maior valor comercial apresenta maior nível de interesse. A facilidade na venda, tamanho, peso entre outros fatores são mais determinantes. Este fator será chamado "Índice de Interesse de Roubo" (IIR).
- Informações históricas e atuais do índice roubo por região ajudam a determinar o risco de roubo na rota adotada. Quanto mais atual a informação maior a sua relevância.

  Para isso após definição da rota e início do percurso são determinados dois raios principais de interesse, a 1Km da carga e a 10Km da carga, como ilustrado na *Fig. 5*.

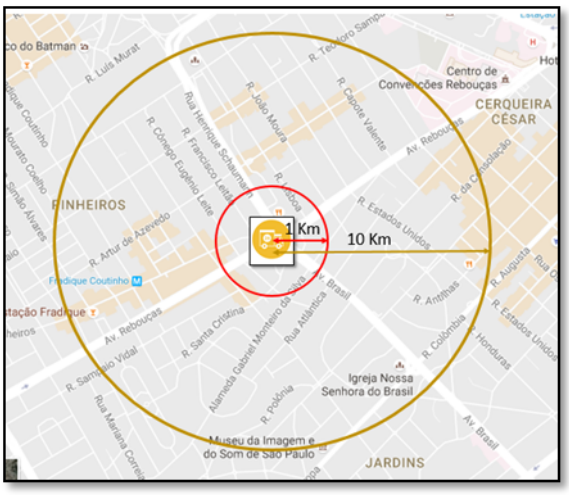


Figura 5. Raios de interesse. Fonte: Google Maps

- Quanto mais itens de segurança apresenta aquela carga, como trava eletrônica e bloqueio remoto menor chamado "Fator de Insegurança da Carga" (FIC).
- O acionamento do Botão de Pânico indica uma situação de risco atual na localização desta carga

Dados estes pontos é proposto um modelo com as seguintes características:

- Todos os veículos do modelo proposto possuem módulo de rastreamento com função de Botão de emergência e display com visualização do mapa (*Fig. 6*) com atualização a cada 1 minuto.

Figura 6. Painel de instrumentos com mapa [4].

- Todos os veículos do modelo possuem microfone e alto-falantes que permitem a troca de informações com a CCO.
- O botão de pânico pode ser acionado a qualquer momento pelo condutor e pode se dar de duas formas como descrito na *Tab.1*:

Tabela 1. Política de acionamento do Botão de Emergência.

Quantidade de vezes que foi pressionado	Significado para CCO	Política adotada pela CCO
1x	Potencial risco identificado	I. Aumenta amostragem do tempo de rastreamento de 5 minutos para 30 segundos. II. Tentativa de contato telefônico com o motorista, abrindo canal de áudio de voz.
2x	Risco atual	I. Aumenta amostragem do tempo de rastreamento de 5 minutos para 30 segundos. II. Busca por estratégias de bloqueio de carga e socorro. III. Abertura do canal somente de voz para "ouvir" ambiente do veículo.

A. Dados de Entrada do Sistema

i. Para início de coleta de dados do sistema é utilizado o mapa dado na figura 1, sendo as faixas de valores de 0 até o número máximo de roubos por região como valor de entrada sobre perigo "histórico" daquela região.

ii. Os dados de tráfego atual das vias é retirado das informações do site da CET (*Fig.7*). O cálculo da velocidade média nos próximos 2 Km da rota é atualizado a cada 1 minuto. Ressaltando que quanto menor a velocidade maior o risco.

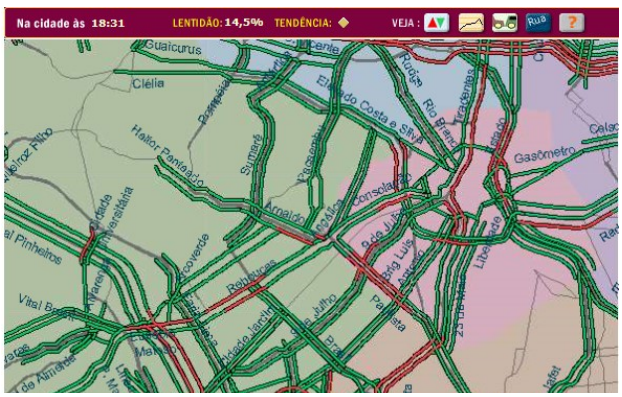


Figura 7. Mapa de trânsito atual CET-SP. Fonte: www.cetsp.com.br

iii. O FIC é um fator que varia entre 0,5 e 1, fornecido ela empresa de transporte em função da quantidade de itens de segurança que a carga apresenta. Quanto mais alto o valor menor a segurança da carga, como indica a *Tab. 2*.

Tabela 2. Fator de Insegurança de Carga.

Fator de Insegurança da Carga (FIC)	Segurança da carga
0,5-0,6	Baixa
0,7-0,8	Média
0,9-1	Alta

Os indicadores da tabela 2 são uma referência para dados de entrada, porém determinando algumas regras ele pode ser refinado de forma dinâmica:

- Regra 1: para cargas X e Y onde FIC(X) > FIC(Y). Se a quantidade de vezes que o botão de pânico foi pressionado para a carga Y for superior a X, os valore de X e Y se invertem.

A correção automática deste fator irá orientar o transportador a determinar dispositivos mais efetivos de segurança de carga.

iv. O IIR é um fator que varia entre 0,1 e 1, e a princípio é um dado de entrada da empresa de transportes baseado em dados históricos, onde quanto mais alto o valor mais alta a probabilidade de roubo, como indica a *Tab. 3*.

Tabela 3. Índice de interesse de Roubo.

Índice de Interesse de Roubo (IIR)	Probabilidade de roubo na rota
0,1-0,3	Baixa
0,4-0,7	Média
0,8-1	Alta

Os indicadores da tabela 3 são uma referência para dados de entrada, porém determinando algumas regras ele pode ser refinado de forma dinâmica:

- Regra 2: para cargas X e Y onde IIR(X) > IIR(Y). Se a quantidade de vezes que o botão de pânico foi pressionado para a carga Y for superior a X, os valore de X e Y se invertem.

Com esta regra além o IIR vai dinamicamente se refinando para definir qual a carga com maior índice de roubo, além disso permite visualizar ao longo do ano se os fatores se alteram e permite ao transportador prever em qual época determinada carga apresenta maior risco de roubo.

v. Para inserir um valor de determine a dinamicidade da informação quando algum usuário pressiona o Botão de Emergência será adotado o Fator de Risco Local (FRL):

Tabela 4. Fator de risco local em função do raio do mapa.

Quantidade de vezes que o Botão de Pânico foi pressionado	FRL em função do raio do mapa	
	1 Km	10 Km
1 x	1,5	1,2
2 x	2,0	1,5

Assim, partindo de um mapa estatístico dado na figura 1 será utilizado o algoritmo ilustrado no diagrama da *Fig. 8*, para a criação do mapa dinâmico.

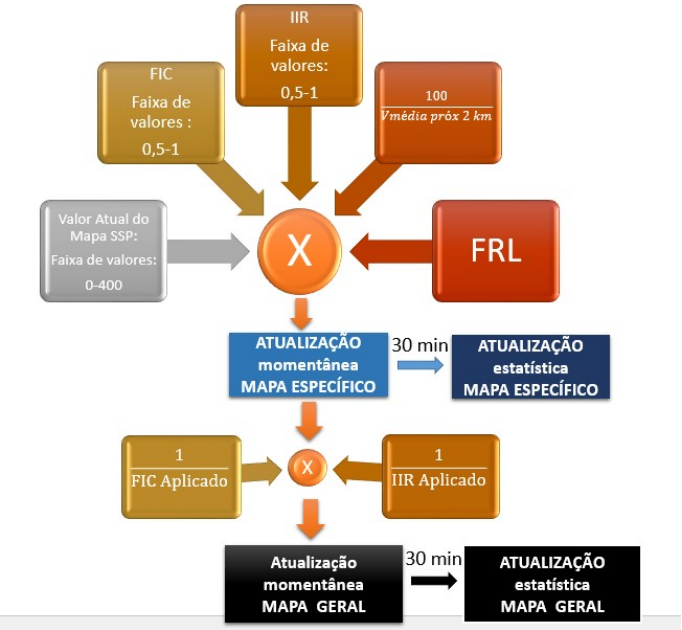


Figura 8. Fluxograma. Fonte: autores.

Algumas considerações sobre algoritmos:

1. O algoritmo faz sempre dois cálculos para os raios de 1 e 10 Km antes de atualizar os mapas.
2. Como os fatores IIR e FIC só tem interferência direta para a carga em que foram aplicados são utilizados dois mapas:
   i. Mapa Específico: considera os fatores IIR e FIC na atualização dinâmica das cores dos mapas, tornando o mapa característico da aplicação;
   ii. Mapa Geral: não considera dados específicos da carga (FIC e IIR), somente os dados relevantes gerados por outros usuários do sistema.
3. Para que o mapa tenha uma informação de maior relevância a o gráfico de cores é exibido em duas etapas:
   i. *Atualização Momentânea*: atualiza cores no mapa de acordo com o fluxograma expresso na figura 8 durante 30 minutos, mostrando assim que recentemente houveram alarmes nas redondezas. Além disso é exibido uma notificação no mapa na forma de alerta piscante pelos mesmo 30 minutos no raio de 10 Km;
   ii. *Atualização Estatística (AE):* somente considera a quantidade de vezes que o Botão de Pânico foi pressionado na região, sendo que qualquer condutor ao utilizar o fará aquele dado válido para o raio de 10Km ao redor do local.
   A soma da quantidade de vezes que o botão foi pressionado naquele raio, incluindo intersecções dará este valor. Ressaltando que cada condutor pode pressionar 1 ou 2 vezes e essa quantidade será utilizada como valor de AE.
4. Valores de atualização do mapa dados pelo cálculo do algoritmo ilustrado na figura 8 serão limitados até o máximo de vezes de o botão foi pressionado em um raio de 10Km, dado pela atualização estatística (AE). Para que possa ser aplicado ao mapa uma camada de cores correspondente ao risco de assalto da carga, como ilustrado na *Fig. 9*.

Figura 9. Escala de cores aplicadas ao mapa em função do risco calculado no algoritmo. Fonte: autores.

Caso no mapa de atualização momentânea quando o FRL for aplicado extrapolar o valor AE, a escala de cores permanece no valor máximo e a notificação na forma de alerta piscante sinaliza sobre o aumento de risco no local.

5. Intersecção de dois ou mais veículos: quando dois ou mais veículos apresentam intersecção em seus raios na atualização momentânea do mapa significa que o risco é ainda mais elevado. Neste caso é adotado na zona de intersecção o maior valor entre todos e é multiplicado pelo valor de acordo com a *Tab. 5* até o limite dado pela atualização estatística.

Tabela 5. Fator para intersecção entre.

Quantidade de veículos que acionaram o botão de emergência em uma zona de intersecção (1 ou 2x)	Fator de multiplicação
2	1,2
3	1,3
4	1,4
5	1,5
6	1,6
7	1,7
8	1,8
9	1,9
10 ou mais	2

B. *Resultados esperados do modelo*

Com o modelo proposto os seguintes resultados são esperados:

- ✓ Atualização Dinâmica dos Mapa Geral e Específico;
- ✓ Dados de entrada mais precisos;
- ✓ Clareza nas informações de risco;
- ✓ Informações de maior qualidade e relevância para tomada de decisão do operador;
- ✓ Previsões mais assertivas à respeito de rotas mais seguras de acordo com a carga, hora, dia ou até mesmo período do ano;
- ✓ Informações relevantes e personalizadas para cada carga transportada;
- ✓ Informações de maior qualidade e relevância ao motorista;
- ✓ Maior segurança para motorista e carga;
- ✓ Alteração automática de rota e pontos de parada;
- ✓ Substituição física do Botão de Pânico por uma função adaptativa

III. Conclusões

Buscou-se com o modelo apresentado utilizar conceitos de adaptatividade aplicados em um contexto do cotidiano que apresenta demanda imediata. A camada de adaptativa dentro do modelo foi gerado através da criação de fatores e índices que aumentam a qualidade da informação e fazem do modelo dinâmico. A *Fig. 10* ilustra esta camada.

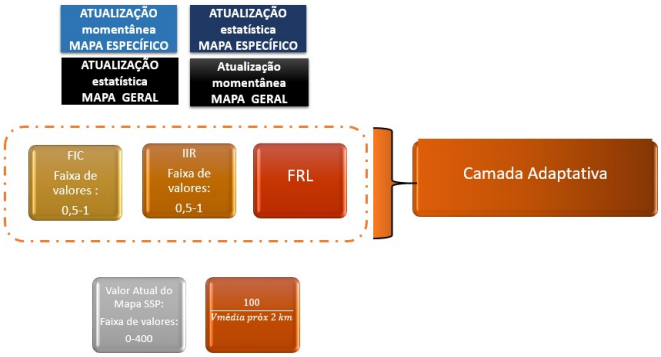


Figura 10. Camada adaptativa do modelo. Fonte: autores.

Este modelo se aplicado pode trazer um grande aumento para a segurança do condutor e redução dos prejuízos causados pelos roubos. Assim com o mapeamento realizado neste modelo e sua proposta foi criado um modelo adaptativo para um item de segurança já utilizado em larga escala dentro do contexto de transporte de cargas. Além disso, o principal ganho neste modelo está na qualidade da informação que pode tornar cada vez menos necessário o monitoramento online pelas CCOs, ou tornar a tomada de decisão da CCO ainda operante muito mais assertiva. Este modelo se aplicado pode trazer um grande aumento para a segurança do condutor e redução dos prejuízos causados pelos roubos de carga.

Referências

[1] Online: http://www.portalntc.org.br/. Acesso em 20/11/2016.

[2] Online: http://www.fetcesp.net/estatisticas-de-seguranca.php. Acesso em 21/11/2016.

[3] Online: http://www.ondefuiroubado.com.br/. Acesso em 18/11/2016.

[4] Online: http://blog.caranddriver.com/new-audi-tts-gauge-cluster-will-spread-to-three-other-models-by-2015/). Acesso em 19/11/2016.

[5] Online: http://cetsp1.cetsp.com.br/monitransmapa/painel/. Acesso em 21/11/2016.

Uso de Métricas de Roteamento através de Tabela de Decisão Adaptativa para Redes de Sensores Sem Fio

D. S. Miguel D. A. G. de Oliveira

***Resumo*—Redes de Sensores Sem Fio (RSSF) vem ganhando foco entre os pesquisadores, devido sua abrangência de aplicações como: detecção e acompanhamento, monitoramento de ambientes e indústrias, acompanhamento médico entre outros.**
O paradigma Internet das Coisas (Internet of Things (IoT)) torna possível a interconectividade entre dispositivos em qualquer momento e em qualquer lugar do planeta [9], porém, alguns dispositivos não possuem poder de processamento, memória ou, até protocolos para se comunicarem diretamente pela internet. As RSSF's desempenham um papel fundamental na IoT, tornando possível a conexão de pequenos dispositivos e objetos à internet. Adaptatividade é a capacidade que um sistema tem em modificar seu conjunto de regras.
Existem diversos trabalhos na literatura que empregam a adaptabilidade para flexionar as regras propostas para a solução de um problema em específico.
Observando o emprego de adaptatividade na literatura, observamos que seu uso se dá por modificações as regras definidas estaticamente e, geralmente empregadas junto à outras técnicas como de inteligência artificial, para flexionar as regras não só das soluçãoes propostas aos problemas como também às regras da própria adaptabilidade.

***Keywords*—Redes de Sensores Sem Fio, Internet das Coisas, Internet of Things, Adaptatividade, Adaptabilidade.**

I. Introdução

A. Redes de Sensores Sem Fio (RSSF's)

Redes de Sensores Sem Fio (RSSF) vem ganhando foco entre os pesquisadores, devido sua abrangência de aplicações como: detecção e acompanhamento, monitoramento de ambientes e indústrias, acompanhamento médico entre outros. O paradigma Internet das Coisas (Internet of Things (IoT)) torna possível a interconectividade entre dispositivos em qualquer momento e em qualquer lugar do planeta [9], porém, alguns dispositivos não possuem poder de processamento, memória ou, até protocolos para se comunicarem diretamente pela internet. As RSSF's desempenham um papel fundamental na IoT, tornando possível a conexão de pequenos dispositivos e objetos à internet.

Uma das maiores preocupações, quando programando um sensor (dispositivo sem fio), é o consumo de energia, uma vez que o sensor é energeticamente restrito. Muitos trabalhos de pesquisa focam em economizar por reduzir a carga de processamento ou desligando e ligando periféricos como sensores, rádio [10] e, até o processador. Uma alternativa seria saber a energia remanescente do nó para auxiliar nas decisões de economia e roteamento, uma vez que o rádio é o maior consumidor de energia no nó. Porém, adicionar hardware para medição encareceria o sensor além de consumir mais energia. Uma solução viável é estimar a energia remanescente do nó através de monitoria de uso do hardware [11]. É comum em RSSF's utilizar o ETX [12] como métrica de roteamento, porém, isso pode fazer com que um nó esgote sua energia muito antes de outros e, assim, acabar particionando a rede e presenciar alguns nós, ainda com energia, ficarem inacessíveis.

D. S. Miguel, Doctoral student at Universidade de São Paulo, email: dan.sm.mineiro@gmail.com.

D. A. G. de Oliveira, Master's student at Universidade de São Paulo, email: doriedson.ago@gmail.com.

B. Adaptatividade

A adaptatividade é a capacidade que um sistema ou dispositivo tem de se auto-modificar, alterando o comportamento de um processo em execução. A adaptatividade pode ser percebida quando um dispositivo definido com regras parametrizadas e comportamento invariante torna seu comportamento dinâmico, incorporando regras adaptativas. Diz-se que um dispositivo formal é adaptativo sempre que seu comportamento muda dinamicamente, em resposta direta a seus estímulos de entrada, sem interferência de agentes externos, mesmo seus usuários [1]. A adaptatividade pode ser aplicada aos formalismos tradicionais, tais como: Statechart, Autômatos Finitos, Redes de Markov e Tabela de Decisão [2].

Como descrito em [3] inúmeras aplicações potencias existem para a tecnologia derivada da adaptatividade, tais como inferência, processamento de textos em linguagem naturais, síntese de voz, reconhecimento de padrões, tomada de decisão, linguagens de programação adaptativas, otimização de código, meta-modelagem, computação evolutiva, engenharia de software e robótica. As aplicações potenciais enumeradas em [3] definem a tecnologia adaptativa como um campo de pesquisa emergente podendo ser inserida visando a resolução de problemas complexos apresentados em diversas áreas de interesse.

De maneira geral, as ações adaptativas permitem que regras sejam consultadas, eliminadas ou incluídas no sistema [4]. O comportamento de um sistema guiado por regras depende, exclusivamente, de um conjunto finito de regras, responsáveis por mapear cada possível configuração do sistema em uma configuração seguinte correspondente, em resposta a algum

	S	R	R	R	R
	1	2	3	4	5
State =		0	0	1	1
Carga <= 18		S	N	N	S
Carga > 18		N	S	S	N
State:=	0	0	1	1	0
ETX:=	100%	100%	0%	0%	100%
Energia:=	0%	0%	100%	100%	0%

Figura 1. Tabela de decisão para uso de ETX ou energia como métrica de roteamento.

estímulo de entrada recebido [5].

1) Tabelas de Decisão: A tabela de decisão representa ações a serem executadas, como resultado do comprimento de um conjunto de condições estabelecidas. As regras de decisão são pontos importantes para estrutura de uma tabela de decisão, a partir delas o conjunto de ações a serem executadas é definido, é necessário que um determinado conjunto de condições seja atendido para que tais ações sejam executadas. As tabelas de decisão podem auxiliar na formulação de questões de problemas, nem sempre claras em outras formas [6].

2) Tabelas de Decisão Adaptativas: Nas tabelas de decisão adaptativas o dispositivo subjacente é uma tabela de decisão convencional [7]. Como visto em [8] uma tabela de decisão convencional é composta por colunas que representam conjuntos de regras associadas a condições e ações. Novamente em [7] é descrito que na versão adaptativa de uma tabela de decisão convencional pode ser obtida adicionando-se a ela linhas, onde são incluídas as funções adaptativas. Além disso, a cada coluna que representa uma regra simples, deve ser adicionada uma chamada para uma função adaptativa associada à execução de uma regra em particular [7]. Com isso, sempre que uma regra adaptativa é aplicada, uma ação adaptativa é invocada, permitindo que sejam feitas mudanças no conjunto de regras [7].

II. Abordagem Proposta

Utilizando a energia remanescente do nó ao invés do ETX, conseguimos uma balanceamento de carga entre todos os nós de forma homogênea. Assim, os nós tendem a ficarem sem energia quase ao mesmo tempo, não havendo particionamento na rede devido a morte prematura de alguns nós.
Verficamos que ao usar somente energia como métrica de roteamento, os nós tendem a usar algumas rotas maiores se comparadas às rotas definidas por ETX, que tendem a usar o menor caminho (saltos) possível. Com isso, um pacote tende a gastar mais energia, na rede como um todo, em sua rota desde a origem até seu destino. Observando isso, concluímos ser interessante utilizar um método que equilibrasse o uso do roteamento priorizando o menor caminho (uso de menos energia no roteamento de um pacote) e o uso equilibrado de energia em toda rede para que um nó não morra prematuramente e assim particione a rede.
Assim, elaboramos uma tabela de decisão que define como

	H	–	–	–	+	+	+	H	–	–	–	+	+	+
State =		1	1	1	1	1	1		1	1	1	1	1	1
Carga <= 18		N	N	S	N	N	S		N	N	S	N	N	S
Carga > 18		S	S	N	S	S	N		S	S	N	S	S	N
Tempo > 300s		N	S	N	N	S	N		N	S	N	N	S	N
State:=		1	1	0	1	1	0		1	1	0	1	1	0
Tempo:=		0	0	0	0	0	0		0	0	0	0	0	0
ETX:=		80%	60%	100%	60%	40%	100%		60%	40%	100%	40%	20%	100%
Energia:=		20%	40%	0%	40%	60%	0%		40%	60%	0%	60%	80%	0%
K	A		V											
L						V		A		V				
M													V	
N														
O				V			V				V			V
p1				p1		60%	60%				p1		40%	40%
p2				p2		40%	40%				p2		60%	60%
p3				p3		40%	40%				p3		20%	20%
p4				p4		60%	60%				p4		80%	80%

Figura 2. Tabela de decisão adaptativa para métricas de roteamento.

estado inicial o uso somente de ETX como métrica de roteamento e, a partir do momento que houver desequilíbrio de energia entre os nós da rede, a regra é alterada para levar em conta energia como métrica de roteamento conforme figura 1.
Em uma abordagem inicial, trocaríamos o uso de ETX pelo uso de energia como métrica de roteamento para o reequilíbrio da energia nos nós da rede. Porém, isso poderia mudar drasticamente as rotas já definidas na rede implicando em sobrecarga no fluxo de pacotes de controle. Assim, definimos usar uma tabela de decisão adaptativa para balancear o uso de ETX e energia como métrica de roteamento a fim de suavizar o fluxo de pacotes de controle e restabelecer o balanceamento de energia da rede como mostrado na figura 2 3 e 4.
Para definirmos quando os nós estão desequilibrados energéticamente, definimos a seguinte fórmula:

Carga:= max(|M – minE|, |M – maxE|)

Onde,
M é a média da energia de todos os nós da rede.
minE é o valor da energia do nó com menos energia na rede.
MaxE é o valor da energia do nó com mais energia na rede.
Onde a carga da rede é definida pegando o maior valor dentre os módulos das diferenças da média energética da rede e o menor e maior valor de energia da rede.

III. Conclusão

Analisando a tabela adaptativa proposta, podemos notar que o equilíbrio energético da rede tende a se reestabelecer voltando a usar o ETX como métrica única de roteamento. Assim, priorizando o menor consumo de energia no roteamento de pacotes. Mesmo assim, se a rede demorar a reestabelecer o seu equilíbrio energético, a tabela adaptativa

	H	-	-	-	+	+	+	H	-	-	-	+	+
State =		1	1	1	1	1	1		1	1	1	1	1
Carga <= 18		N	N	S	N	N	S		N	N	S	N	S
Carga > 18		S	S	N	S	S	N		S	S	N	S	N
Tempo > 300s		N	S	N	N	S	N		N	S	N	nop	N
State:=		1	1	0	1	1	0		1	1	0	1	0
Tempo:=		0	0	0	0	0	0		0	0	0	0	0
ETX:=		40%	20%	100%	20%	0%	100%		20%	0%	100%	0%	100%
Energia:=		60%	80%	0%	80%	100%	0%		80%	100%	0%	100%	0%
K													
L													
M	A		V										
N						V		A		V			
O				V			V				V		V
p1				p1		20%	20%				p1		0%
p2				p2		80%	80%				p2		100%
p3				p3		0%	0%				p3		0%
p4				p4		100%	100%				p4		100%

Figura 3. Tabela de decisão adaptativa para métricas de roteamento.

	H	-	-	-	-	-	-	+	+	+	S 1	R 2	R 3	R 4	R 5	R 6
State =		1	1	1	1	1	1	1	1	1		0	0	1	1	1
Carga <= 18		N	N	N	N	N	S	N	N	S		S	N	N	N	S
Carga > 18		S	S	S	S	S	N	S	S	N		N	S	S	S	N
Tempo > 300s		N	nop	S	S	S	N	N	S	N		nop	nop	N	S	N
State:=		1	1	1	1	1	0	1	1	0	0	0	1	1	1	0
Tempo:=		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ETX:=		p1	p1	p3	p3	p3	100%	80%	60%	100%	100%	100%	80%	80%	60%	100%
Energia:=		p2	p2	p4	p4	p4	0%	20%	40%	0%	0%	0%	20%	20%	40%	0%
K									V						V	
L				V												
M					V											
N						V										
O	A						V			V						V
p1				p1	p1	p1	p1		80%	80%					80%	80%
p2				p2	p2	p2	p2		20%	20%					20%	20%
p3				p3	p3	p3	p3		60%	60%					60%	60%
p4				p4	p4	p4	p4		40%	40%					40%	40%

Figura 4. Tabela de decisão adaptativa para métricas de roteamento.

se encarregará de montar uma abordagem cada vez mais agressiva para o reestabelecimento do equilíbrio energético. Como trabalhos futuros, implementaremos a tabela descrita neste artigo em uma rede de sensores sem fio para análise de sua eficácia bem como suas nuances.

REFERÊNCIAS

[1] J. J. NETO, "Adaptative rule-driven devices – general formulation and a case study", In: CIAA'2001 Sixth International Conference on Implementation and Application of Automata, pages 234–250, Pretoria, South Africa, July 2001.

[2] SANTOS. Dispositivos adaptativos cooperantes. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85- 86686-86- 3, pp. 11-15. 28 e 29 de Janeiro, 2016.

[3] J. J. NETO, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa". Revista IEEE América Latina, 5.7, 1548-0992, 2007.

[4] TCHEMRA E CAMARGO, Aplicação da tecnologia adaptativa em sistemas de tomada de decisão, uma abordagem estratégica na seleção de fornecedores. In: Segundo Workshop de Tecnologia Adaptativa - WTA 2008. EPUSP, 2008.

[5] BALDI, CASTRO JUNIOR, RODRIGUES E RODRIGUES. Modelo adaptativo para um framework educacional. Em Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86- 3, pp. 70-79. 28 e 29 de Janeiro, 2016.

[6] TCHEMRA, A. H. Tabela de decisão adaptativa na tomada de decisão multicritério. Tese apresentada na Universidade de São Paulo, 2009.

[7] STANGE E JOSÉ NETO. Aprendizagem incremental usando tabelas de decisão adaptativas. In: Quinto Workshop de Tecnologia Adaptativa - WTA 2011. EPUSP, 2011.

[8] HUGHES, M. L., SHANK, R. M., STEIN, E. S.: Decision Tables. Midi Publications, Management Development Institute, Divisions of Information, Ind"ustries, Inc., Wayne, Pennsylvania, 1968.

[9] R. Want, B. N. Schilit, and S. Jenson, "Enabling the internet of things," Computer (Long. Beach. Calif)., vol. 48, no. 1, pp. 28–35, 2015.

[10] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," SICS Technical Report T2011:13 , ISSN 1100-3154, 2011. [Online]. Available: http://dunkels.com/adam/dunkels11contikimac.pdf. [Accessed: 23-Feb-2016].

[11] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," Proc. 4th Work. Embed. Networked sensors - EmNets '07, p. 28, 2007.

[12] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," ACM Trans. Sens. Networks, vol. 10, no. 1, pp. 1–49, 2013.

Uso da tecnologia adaptativa para reconhecimento do condutor de um veículo.

F. Leme; M. F. Martins; R. Vitor – Escola Politécnica da USP
Universidade de São Paulo
São Paulo – Brasil

***Resumo* — Os veículos atuais tornam-se a cada dia mais inteligentes, e adaptáveis as necessidades dos condutores, aplicando inúmeras tecnologias para se integrar com o ser humano. Indo do uso de reconhecedores de voz e gestos, ao limiar da direção autônoma e comunicação entre veículos. O presente trabalho pautou-se no objetivo de criar uma proposta de adaptabilidade do automóvel ao motorista. Usando da tecnologia adaptativa como forma de enquadrar o gerenciamento do motor ao perfil de cada usuário, sem a direta intervenção do mesmo.**

Palavras Chave* — *adaptatividade, reconhecimento do condutor, tomada de decisão.

I – INTRODUÇÃO

O uso de tecnologias adaptativas tem se tornado de fundamental apoio para novos desenvolvimentos tecnologicos nos dias de hoje, nos quais necessita-se que um maior número de decisões sejam tomadas pelos dispositivos, sem que haja a interferência externa. As tecnologias adaptativas buscam uma auto-modificação do comportamento de um processo em execução permitindo a solução de problemas em tempo real.

Utilizando dessa tecnologia, pretendemos tornar a condução de um veículo automotor uma experiência mais individualizada. Fazendo com que o veículo se adapte as características do condutor, sem a necessidade do mesmo dizer quem ele é, orientando-se apenas pela forma de dirigir de cada condutor. Assim podemos adaptar o gerenciamento do motor, às características do condutor, seja um condutor inexperiente, ou que goste de andar com uma velocidade mais elevada e retirar maior rendimento do propulsor. Porém sempre respeitando as regras de emissão e consumo.

II – OBJETIVO

Como objetivo principal deste trabalho, tem-se a realização da identificação do motorista do veículo, tornando o mesmo adaptável a ele. Essa identificação será feita através de parâmetros que serão lidos em tempo real, assim o veículo se tornaria adaptável ao modo de condução e preferências individuais de diferentes condutores.

III – ADAPTATIVIDADE

Os dispositivos adaptativos são capazes de operar de forma dinâmica, adaptando-se à diferentes situações, fazendo-o a medida que diferentes estímulos são percebidos. De forma autônoma, estes dispositivos são capazes de modificar o conjunto de regras que definem seu comportamento. São capazes de alterar seu próprio código, em tempo real, sem a intervenção de um operador.

São formulados, para se moldarem às alterações dinâmicas do meio. De forma, que ao receber distintos tipos de dados de entrada possam decidir qual o melhor processamento a ser realizado. Podem ser formados a partir de dispositivos de natureza adaptativa ou de dispositivos não adaptativos, classificados como dispositivos ‘subjacentes’. Que ao receber uma camada adaptativa, ganham as características de adaptabilidade dos dispositivos adaptativos.

Pode-se então entender que estes dispositivos adaptativos, são dotados de uma certa adaptatividade que lhes garante uma considerável maleabilidade, e grande capacidade de processamento, equivalente a máquina de Turing. Permitindo a adaptação de sua operação às novas características do sistema.

Vale observar a definição de adaptatividade formulada por J.J. Neto:

> “ADAPTATIVIDADE, conforme a conotação aqui utilizada, é um termo que se refere a um conceito bastante simples: a capacidade que tem 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.” (NETO, 2007, p.1)

Desta forma, pode-se observar no conceito de adaptatividade, esta capacidade de modificação do dispositivo

por sua própria estrutura interna. Sem receber uma intervenção ou comando. A diferenciação de adaptabilidade e adaptatividade formulada por J.J. Neto e Rosalia E. C. Carhuanina torna-se aqui esclarecedora.

> "A adaptabilidade é caracterizada pela participação não ativa do sistema, e a escolha por parte de um agente externo de alguma funcionalidade pré-existente que implemente o comportamento correspondente com a alteração desejada. Adaptatividade, pelo contrário [...], é a propriedade que apresenta um sistema, processo ou dispositivo computacional, que lhe permite, sem a interferência de agentes externos, tomar a decisão de executar alguma mudança, de maneira autônoma e dinâmica, no seu próprio comportamento como consequência de algum estímulo, entendido como alguma variação nas condições no seu ambiente de execução, e a sua configuração corrente." (CARHUANINA & NETO, 2016, p.57)

Nesta condição, a de modificar-se de forma dinâmica, sem sofrer interferência de agentes externos, encontra-se a essência do conceito de adaptatividade. E partir disto, torna-se possível a confecção dos dispositivos adaptativos.

IV – METODOLOGIA

Para a execução do trabalho foram levantados dados de operação do veículo e por meio destes dados tornou-se viável a elaboração de uma tabela de tomada de decisões capaz de realizar o gerenciamento adaptativo da condução.

Alguns dos parâmetros a serem analisados são: velocidade angular do pedal do acelerador, velocidade/rotação durante a troca de marcha, velocidade média do veículo, interrupções abruptas do motor na saída, sequência na troca de marcha.

Após a aquisição destes dados, o sistema se torna adaptável, e o módulo do gerenciamento do motor altera o comportamento do mesmo, tornando-o mais agressivo, econômico, eficiente na saída e na troca de marchas (para motoristas com pouca prática) e também confortável.

Velocidade e rotação da troca de marcha: Com essa informação é possível criar um padrão para as trocas de marcha, devido ao fato de cada condutor ter comportamentos característicos ao realizar esta atividade. Exemplo, um motorista que tem como marca de sua condução realizar uma arrancada mais rápida, irá ter como padrão pessoal uma troca de marcha na qual a rotação será mais elevada e realizada numa velocidade maior. Diferente daquele condutor mais calmo, com troca de marchas menos constantes e direção em rotações mais baixas.

Velocidade média do veículo: Cada condutor tem uma preferência na velocidade de condução. A informação da velocidade média de condução associada à troca de marcha permite uma extrapolação para a compreensão do perfil do condutor.

E nas situações de cruzeiro, mesmo com as limitações impostas pelas leis de trânsito, que estipulam a velocidade máxima das vias, é possível adaptar este parâmetro ao motorista. Exemplo, um motorista dirige em uma autoestrada na qual o limite de velocidade é 110 km/h, porém esse condutor prefere ir a 100 km/h ou então a 120 km/h diminuindo apenas quando visualiza um radar. Seria feito a sincronia com o GPS para que se tenha a velocidade estipulada para cada via.

Interrupções abruptas do motor: Essa característica serve para a identificação de motoristas que ainda não possuem plena habilidade na condução de veículos.

Sequência na troca de marcha: Essa informação é utilizada pelo módulo adaptativo para melhorar o refinamento da busca pelo motorista. Muitos condutores após certo tempo de prática mudam as posições do cambio sem passar pela sequência habitual, de primeira a quinta marchas. Alguns acabam passando da primeira para a terceira ou da segunda para a quarta. Essa característica seria usada para melhorar os padrões de adaptação.

Sistema de conforto e conveniência: Cada indivíduo tem uma zona de conforto, e isso se reflete no uso do veículo. É possível analisar quais são as estações de rádio selecionadas no momento de direção, assim como os tipos de música.
Fazer a análise do sistema de climatização, no qual cada ocupante do veículo tem uma temperatura que se sente mais confortável. O sistema assim verificaria o uso do ar condicionado se está ligado ou não, e qual temperatura adequada para o usuário.

V – ANÁLISE DOS DADOS

Pela analise dos dados de condução coletados em relação a diferentes condutores é possível estabelecer as regras, ações e estados que formam a tabela de tomada de decisões. Nesta seção tem-se por objetivo fazer a analise destes dados e levantar os itens supramencionados.

Nos gráficos subseqüentes são demonstradas estas características de condução. Como já é possível observar no primeiro destes gráficos, o qual monitora as curvas de rotação do motor, velocidade do veículo e torque indicado pelo posicionador do corpo de borboleta do veículo. E por meio destas três variáveis podemos extrapolar um padrão de direção para um determinado individuo médio.

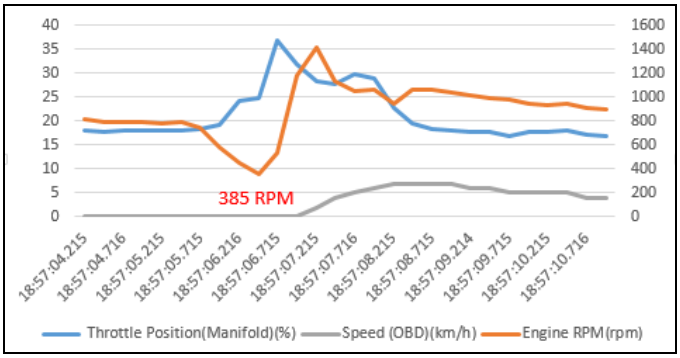


Gráfico 1 - Rotação atingindo pico mínimo de 385RPM com velocidade 0 km/h

Percebemos pelo gráfico que a rotação do motor cai a um valor muito baixo – 385RPM – pouco antes de o veículo arrancar. Esta informação indica a condução por um motorista com pouca experiência, que por não acelerar o veículo durante a arrancada acaba por deixar que o motor morra ou quase morra. Com isso formula-se a primeira das regras para a tabela de tomada de decisões. Esta regra avalia se, ao sair da inércia, a rotação do motor irá cair abaixo do valor da rotação de marcha lenta. Esta condição compõe a regra 1 da tabela.

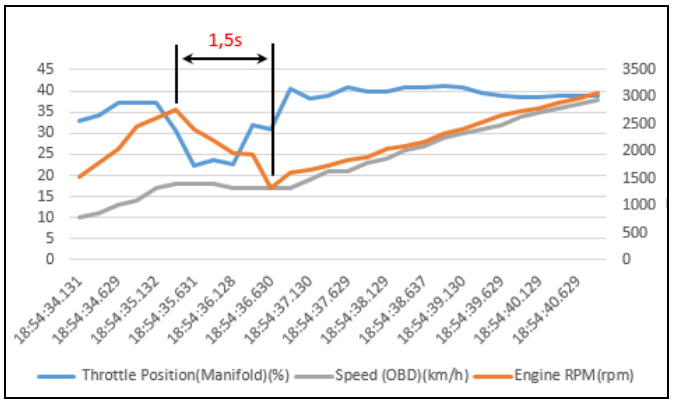


Grafico 2 - Tempo de troca de marca maior que 1,5s e rotação máxima para troca de marcha menor que 3000RPM

O gráfico acima demonstra uma situação em que o veículo já está em marcha. E verifica-se a que o tempo de troca de marcha é maior que 1,5 segundos e que a rotação máxima antes da troca da marcha é menor que 3000RPM. Estas condições indicam a existência de um condutor moderado. E por essas observação dos dados define-se duas outras regras. Uma regra avalia a velocidade para a troca de marchas, neste caso verificando se ela ocorre com um tempo superior a 1,5 segundos – regra definida como regra 7. Outra regra observada é a se a rotação máxima antes da troca de marchas está abaixo de 3000RPM. Esta última foi determinada como sendo a regra 5.

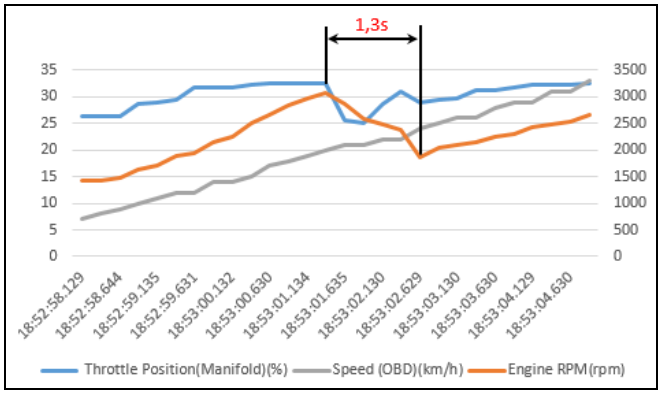


Gráfico 3 - Tempo de troca de marcha inferior a 1,5s e rotação máxima para troca de marcha maior que 3000RPM

Os dados do terceiro gráfico demonstram a condução realizada por um motorista mais agressivo no estilo de pilotar. Esta condição é revelada pela junção das características do tempo de troca de marcha menor do que 1,5 segundos e pela rotação máxima antes da troca ser maior do que 3000RPM.

Estão acima definidas mais duas regras a serem avaliadas. Em primeiro lugar se o tempo de troca de marcha é inferior a 1,5 segundos: regra 6. Em segundo se a rotação máxima antes da troca de marchas era superior a 3000RPM – regra 4.

Regra 2:

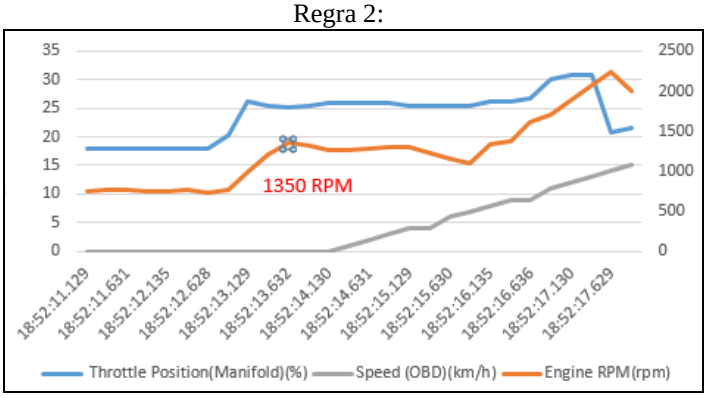


Gráfico 4 - Rotação máxima com o veículo ainda em 0 km/h entre 1000 e 1500RPM

No gráfico acima pode-ser observar que ao sair com o veículo do repouso a rotação do motor está numa faixa superior aos 1000RPM. Garantindo uma saída eficaz do estado de inércia. Pela analise observa-se que a maior rotação foi, neste momento, em torno de 1300RPM. Esta condição define outra regra, a de número 2. Que por sua vez avalia se ao arrancar com o veículo, a rotação imposta pelo motorista é maior que 1000RPM e menor do que 1500RPM.

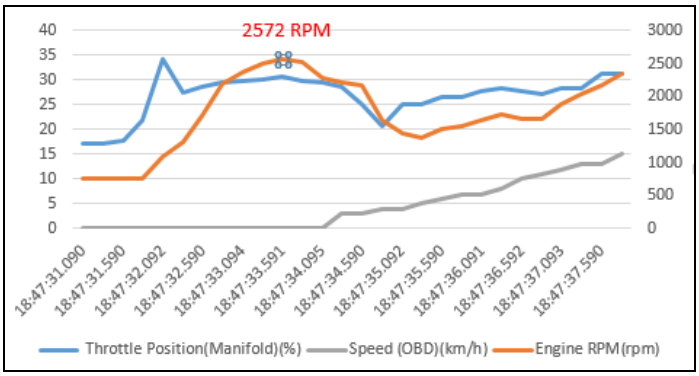


Gráfico 5 – Rotação máxima com o veículo ainda em 0 km/h acima de 1500RPM

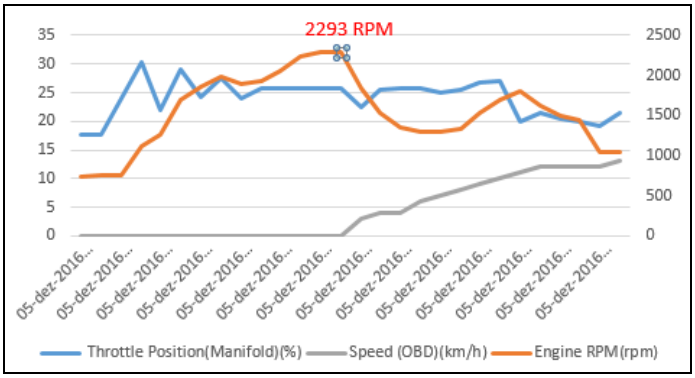


Grafico 6 - Rotação máxima com o veículo ainda em 0 km/h acima de 1500RPM

Os gráficos 5 e 6 demonstram duas situações nas quais para arrancar com o veículo o condutor eleva a rotação do motor acima de 1500RPM. Esta situação demonstra uma inexperiência do condutor que acelera o motor ainda sem carga e com a rotação já elevada vai liberando o curso da embreagem. Esta situação permite estipular a regra 3: rotação ao arrancar com o veículo superior a 1500RPM.

VI – TABELA DE DECISÕES

Tendo em vista as situações apresentadas pelos gráficos acima montou-se a tabela de tomada de decisões. Nesta tabela ocorre a validação das regras, a identificação do perfil do condutor e as ações que devem ser tomadas para cada condutor identificado.

A tabela de tomada de decisões é responsável pela inserção da adaptatividade ao gerenciamento do motor no modelo proposto por este trabalho. Sendo possível, por meio dela, inferir se as condições de operação, ou regras, que anteriormente foram estipuladas foram verificadas como ocorrências, ou não.

A combinação destas regras identifica o condutor que está no controle do veículo. E, além disto, aponta quais ações o sistema de gerenciamento deve tomar para tornar a condução mais agradável para o motorista.

Tabela de Decisões

		Motorista 1	Motorista 2	Motorista 3	Motorista 4	Motorista 5
r1	RPM max a 0km/h <700				x	
r2	RPM max a 0km/h 1000<RPM<1500			x		
r3	RPM max a 0km/h >1500	x	x			
r4	RPM max para troca de marcha >3000	x	x			
r5	RPM max para troca de marcha <3000			x		
r6	Tempo de troca de marcha <1,5s	x				
r7	Tempo de troca de marcha >1,5s		x	x	x	
		A1	A2	A3	A4	A5
r8	Velocidade máxima após 5s >15					
r9	Velocidade máxima após 5s <15					

São identificados pela tabela 5 tipos de condutor. Sendo que os quatro primeiros foram pré-determinados pela analise dos dados de condução, e pela ocorrência ou não das regras estipuladas durante sua pilotagem. O quinto condutor é determinado pela omissão das combinações de regras, ou seja, para os casos em que nenhuma das combinações de regras estipuladas sejam satisfeitas.

Pela combinação das regras 3,4 e 6, é identificado o motorista 1, que é um perfil de motorista mais agressivo. Que arranca com o veículo em velocidade maior, troca as marchas em tempo curto e estica as marchas até uma rotação maior.

A combinação das regras 3, 4 e 7 define o motorista 2, que é um motorista que usa a potência do motor de forma pouco habilidosa, e precisa de uma ajuda para dosar o uso do veículo.

Pela união da regras 2, 5 e 7, estipula-se o motorista 3, que configura-se como um condutor moderado, que não utiliza a totalidade de potencia do motor, que arranca com o carro em rotações acima de 1000RPM e abaixo de 1500RPM; que troca as marchas abaixo de 3000RPM e realiza as trocas num intervalo de tempo menos que 1.5 ms.

As regras 1 e 7, identificam o motorista 4. Este é um motorista inexperiente, que deixa o carro morrer ao arrancar com o mesmo, realiza as trocas de marcha com um intervalo de tempo alto.

Para cada um dos condutores identificados é realizada uma alteração no software de controle do gerenciamento do motor. Estas ações foram identificadas de A1 a A5 e são descritas na segunda parte da tabela de decisões exposta abaixo.

A1	Liberar borboleta 100% Aumentar ganho de resposta do pedal de acelerador
A2	Limitar posisão maxima da borboleta em 60% Diminuir ganho de resposta do pedal de acelerador
A3	Suavisar o ganho do pedal, para aceleraçãoes e desacelerações
A4	Acelerar marcha lenta para 1100 rpm Limitar posição da borboleta em 60%
A5	[+] r9, r8 [?] r8, A1 [?] r9, A3 Criar perfil do motorista 5 de acordo com as regras verdadeiras

Na construção do software de controle a tabela de decisões é aplicada utilizando uma série de "if"s". Inicialmente são declaradas as variáveis de entrada que serão lidas, como a rotação do motor, a velocidade do veiculo, etc. Depois de lidas elas são comparadas como condições – que são as regras estipuladas pela tabela de decisões. Estas regras são analisadas como variáveis de controle. E pela combinação das variáveis são definidos os conjuntos de ações a serem tomados pelo software.

VII – CONCLUSÃO

Após o sistema realizar a leitura dos parâmetros de entrada, o gerenciamento utiliza da tabela de decisões para tomar a ação adaptativa, identificando o motorista.

Essa tabela de decisões é vista pelo bloco de gerenciamento como uma sequência de "if"s", que quando um parâmetro é verdadeiro adiciona-se como estado verdadeiro, tornando a saída em nível lógico um, que após todas as sequencias feitas, irá identificar o motorista.

As regras contidas nessa tabela vão de r1 a r9, das quais tem como principal dado, a rotação do veículo e velocidade da troca de marcha.

Após a identificação do motorista as ações adaptativas são tomadas, representadas por A1 até A5. Ações essas que podem ser desde uma elevação da rotação no momento da saída do veículo, caso seja identificado um motorista inexperiente, ou então, aumentar o ganho no pedal do acelerador, assim um motorista que tem como característica acelerações mais fortes, ficando menos perceptível a ação de ter que "pisar fundo" no pedal do acelerador.

REFERÊNCIAS BBIBLIOGRÁFICAS

- PISTORI, Hemerson. Tecnologia adaptativa em engenharia de computação: estado da arte e aplicações. Edição Revisada. São Paulo: 2003.
- TCHEMRA, Ângela Hum. Tabela de decisões adaptativa na tomada de decisões multi-critérios. Edição Revisada. São Paulo: 2009.
- NETO, J.J. "Um Glossário sobre Adaptatividade", Memórias do WTA 2009 - Terceiro Workshop de Tecnologias Adaptativas, Laboratório de Linguagens e Técnicas Adaptativas, Departamento de Engenharia de Computação e Sistemas Digitais Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2009.
- NETO, J.J. Contribuições à Metodologia de Construção de Compiladores. Tese de Livre Docência, Escola Politécnica da USP, São Paulo, 1993.
- NETO, J.J. & CARHUANINA, Rosalia Caya. Personalização, "Customização", Adaptabilidade e Adaptatividade. Memórias do WTA 2016 – Décimo Workshop de Tecnologias Adaptativas, Laboratório de Linguagens e Técnicas Adaptativas, Departamento de Engenharia de Computação e Sistemas Digitais Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2016.

O Signo como Processo Adaptativo

OLIVEIRA, I. S.[*]; NETO, J. J.[*]; KOGLER Jr., J. E. [♣]; ROMANINI, A.V.[Ψ]

***Abstract*— Este trabalho investiga um modelo adaptativo para um processo de interpretação baseado na semiótica de C. S. Peirce. Aqui, a adaptatividade é empregada como um subsídio de apoio à formalização do comportamento de automodificação inerente à semiose, ou ação do signo genuinamente triádico peirceano.**

***Palavras-chave* — adaptatividade, computação semiótica, semiônica, semiótica, semiose, Peirce, modelagem e simulação.**

I. INTRODUÇÃO

O presente trabalho busca por um modelo adaptativo – apoiado pela teoria adaptativa originalmente proposta em [5] – para o processo de interpretação tal qual executado pela espécie humana, o que se encontra supostamente embasado pela teoria semiótica de Peirce, como instigam as suas definições mais gerais de signo. O signo, por sua vez, é um ser cuja evolução constitui o processo de semiose, ou ainda, o processo cuja representação é o alvo deste trabalho. Já foram realizadas especificações com base nessa mesma semiótica em [2] e [11], o que fez com que esses trabalhos fossem empregados como meio de confrontar as ideias originadas da interpretação dos presentes autores com relação à semiótica peirceana.

Um dispositivo adaptativo pode ser visto como "um sistema de malha fechada por um *loop* de realimentação (*feedback*) que possibilita ao sistema ajustar a si próprio em decorrência de eventos [que possam ocorrer] durante sua operação" [12, p. 1, adições entre colchetes]. Esses eventos podem ser vistos como os resultados dos disparos a partir de gatilhos pressionados por fontes internas ao próprio sistema (subsistemas) ou externas a ele – por exemplo, mensageria de outros sistemas, saídas de processos como detecção/identificação/classificação por sensores, atuação de efetores, entre outros. Em resumo, "a introdução da adaptatividade numa aplicação de software, normalmente implementada como um sistema de malha aberta, é convertível num sistema de malha fechada usando o laço de realimentação" [idem]. Esse *feedback* pode ter como origem o *self* e/ou o meio no qual se encontra imerso o mesmo *self*. Aqui entende-se por *self* o que for concebível (ou representável) como um sistema com autonomia para realizar o propósito ao qual foi destinado; já o contexto refere-se a tudo o que há no entorno do *self* (ambiente operacional), porém, disjunto dele e capaz de afetar as suas propriedades (de sistema) [1], como a sua lógica (caso modifique seu algoritmo). Outros sistemas à parte do *self*, por exemplo, um operador humano, constituem algo disjunto do *self*, mas que pode intervir indiretamente em seu funcionamento.

Tais dispositivos automodificáveis podem realizar atividades interessantes, como: "monitorar a si mesmos e ao ambiente [próximo] em que estão imersos, detectar mudanças significativas, decidir como reagir [às mesmas] e atuar para executar as decisões tomadas" [ibidem, adições entre colchetes]. Os elementos (atributos e procedimentos) que um dispositivo computacional exibe a fim de que ele seja qualificado como "adaptativo" são chamadas *propriedades self-** [12 *apud* [Kephart e Chess 2003; Babaoglu et al. 2005], p. 2]. Entretanto, o conjunto dessas propriedades muda de acordo com o entendimento dos pesquisadores em relação ao significado do termo "adaptativo" e, como efeito, há sistemas rotulados como adaptativos, em diferentes áreas do conhecimento, empregando o termo em sentidos não correlacionados – [1] e [12] apresentam taxonomias para "adaptativo" e outros conceitos relevantes bastante semelhantes, por exemplo, "customização" e "personalização".

Neste trabalho, o termo "adaptativo" refere-se a uma qualidade imputável apenas a sistemas capazes de incorporar, remover ou substituir [2] elementos do *self* computacional, autonomamente – isto é, sem requisitar uma intervenção humana "direta" [3] nos procedimentos necessários ao cumprimento de tais atos de automodificação. Essas rotinas adaptativas podem ser classificadas de acordo com o nível de abstração em que são aplicadas: 1º. Na arquitetura e na organização do sistema – alterando *design patterns* ou a topologia com que se dispõe os elementos constituintes do sistema – por exemplo, com vistas a aumentar a eficiência, diminuir o consumo de energia, reduzir o tempo de processamento e atender a outros requisitos não funcionais que são frequentes em problemas de performance envolvendo gargalos localizados. Para fins de ilustração, esse fato pode ocorrer numa camada específica de um sistema *multi-layer*, como uma camada específica interna a um protocolo de rede de computadores. Então, a camada problemática é tratável sem

[*] ian.oliveira@usp.br, jjneto@gmail.com — LTA – Dpto. de Engenharia de Computação — Esc. Politécnica
[♣] kogler@lsi.usp.br — LSI – Dpto. de Engenharia de Sistemas Eletrônicos — Esc. Politécnica
[Ψ] vinicius.romanini@usp.br — LTS – Dpto. de Comunicações e Artes — Esc. de Comunicações e Artes
Universidade de São Paulo

[1] Seja algo disjunto do *self*, tal que este seja incapaz de provocar modificações no (algoritmo do) *self*. Então, esta coisa não pertence ao contexto do *self*.
[2] Esta operação é opcional, uma vez que pode ser construída pela aplicação sucessiva de inclusões e de remoções em certa ordem.
[3] É possível conceber sistemas em que o ser humano atue no papel de supervisor. Nessa situação, por mais que este possa ser visto como um sistema externo fornecendo informações diretamente ao sistema em operação, ele *não* participa nas atividades internas ao sistema computacional, que podem ocorrer, sem novas intervenções, após o eventual recebimento da informação pelo supervisor. Assim sendo, a intervenção humana *direta* não ocorre nos processos-efeitos disparados a partir da informação fornecida. Com isso, faz parte do escopo deste trabalho essa possível comunicação entre sistemas, desde que preservem a autonomia sistêmica de cada um deles como um ser que tanto pode experienciar quanto é capaz de atuar no meio em que se insere.

afetar outras, acima ou abaixo dela[4]. 2º. Nas entidades (ou componentes) do sistema que possam ser vistos como subsistemas ou "caixas-pretas" substituíveis por outras peças sobressalentes, por exemplo: a) Interfaces (de entrada ou saída), como portas (USB, por exemplo) trocáveis por outros padrões ou versões atualizadas que conservam as interfaces com o meio externo, mesmo sendo internamente distintas com relação às suas antecessoras; ou ainda, b) Nós (lógicos e físicos) de uma rede, sendo possível incluir e remover sensores, roteadores e servidores inteiros, com todo o sistema em operação (*online*) ou desligado (*offline*); e 3º. Cada subsistema do *self* (módulo computacional) é alterável internamente, sem impactar outros – modificando-se ou a lógica computacional do módulo (algoritmos) e/ou a fisicalidade subjacente a ele (componentes físicos constituintes da parte em análise)[5].

Como definida e empregada aqui, a adaptatividade está presente em diversos processos, mas nem todos eles podem ser computáveis. "Processo" é algo que pode ser visto como uma abstração para um propósito, objetivo, meta. Ele pode ou não ser decomposto em subprocessos, mas, invariavelmente, é representável por uma sequência de atividades. Uma atividade pode ser entendida como algo semelhante a uma ação[6]. Sintetizando, *um processo adaptativo é algo que, por meio do emprego de automodificação, cumpre o seu propósito*.

[5] expõe uma formulação geral da adaptatividade, aplicável a um amplo espectro[7] de processos (ou dispositivos) adaptativos[8]. Essa formulação baseia-se em funções[9] / ações[10] adaptativas complementares a um dispositivo subjacente não-adaptativo, conferindo-lhe a habilidade de evoluir o seu próprio comportamento, em resposta a eventos (ou fatos), interpretáveis como gatilhos (*triggers*) de mudanças algorítmicas.

Além disso, o mesmo trabalho prevê uma série de aplicações dessa técnica de automodificação em várias áreas, como em: "comunicações digitais (protocolos, tratamento de erros), engenharia de software (especificações formais, interfaces homem-máquina, automação do *design* e implementação de atividades e ferramentas), ferramentas baseadas em sistemas de software (geradores de programas a partir de especificação formal), metaprogramação, inteligência artificial (mecanismos de reconhecimento de sensibilidade a contexto, processamento de linguagem natural, reconhecimento de padrões, dispositivos de aprendizagem, sistemas especialistas), tutoria auxiliada por computação, etc.". Uma dessas potenciais áreas de aplicação da adaptatividade é a semiótica; e a recíproca também parece ser verdadeira, até o presente momento.

Isso porque os interesses em comum estabelecem, entre outras, uma oportunidade de diálogo visando aprofundar o entendimento sobre como automodificações podem fazer um ser evoluir[11], durante seu tempo de existência, usando alguma linguagem como subsídio para atingir esse fim. Essas características inatas a processos que se automodificam denotam algo naturalmente aderente à alta expressividade computacional oferecida pela técnica adaptativa, capaz de atuar como subsídio de modelagem e implementação, em substrato computacional, dos processos internos a linguagens. E isso ocorre graças ao alto nível de abstração e à generalidade da referida formulação adaptativa, que pode ser vista como um modo relativamente simples para especificar e implementar automodificações apenas por meio da composição de rotinas computacionais elementares, necessárias à expressão da referida técnica de automodificação: incluir, remover e substituir elementos do *self*, com base em gatilhos internos e/ou externos a ele. A única ressalva para fácil utilização dessa sistemática é que o projetista consiga especificar um conjunto de regras[12] tal que ele possa ser considerado símbolo do processo adaptativo a automatizar; em outras palavras, a aplicação dessa formulação é realizável caso haja um conjunto de regras e este seja considerado uma boa[13] representação para o processo.

O atual diálogo com os estudos em semiótica também já revelou uma outra convergência essencial: que um dos objetos de estudo centrais dessa área são as linguagens naturais – incluindo sua interpretação – cada qual um sistema dinâmico e complexo. O processo no coração da semiótica é **a semiose pois ela constitui o processo descritor da evolução do signo** e a "semiótica é a ciência dos signos" [10, p. 1]. Já simplificando em muito essa teoria e demandando um pouco da sensibilidade do leitor para o emprego dos termos usados na definição a seguir, **o signo pode ser definido como *tudo o que for interpretável, ou insinuar-se como tal.*** [14] Por dedução, **a semiose também pode ser vista como *o processo descritor da evolução daquilo que for interpretável.***

A relevância pragmática da semiose à computação é destacada, por exemplo, em [2, p. 1]: "o modelo semiótico peirceano representa um elemento chave na construção da

[4] Supondo que as interfaces de comunicação com outras camadas sejam preservadas ou alteradas em todas as impactadas, isto é, considerando e tratando os efeitos colaterais indesejados resultantes dessas alterações.

[5] Note que é possível aplicar recursivamente essas visões, porque se tratam de meta-representações aplicáveis às representações do sistema e das suas subpartes. Assim, por exemplo, seria possível selecionar um dos subsistemas e, como no caso do sistema, analisar 1º. Sua arquitetura/organização; 2º. Suas partes; 3º. Os elementos (lógicos e físicos) internos a cada uma das partes.

[6] No entanto, considera-se que uma atividade possa ser interrompível – o tempo de execução de uma ação é desprezível porque é suficientemente pequeno. Pode ocorrer de uma atividade estar em progresso e um evento interrompê-la. Exemplo: alguém realizando sua corrida matinal é interrompido por alguém que deseja saber que horas são e, com isso, interrompe (mesmo que provisoriamente) a atividade que estava sendo feita. Uma outra diferença, mais técnica, é que "atividade" é associada ao estado da máquina, enquanto uma "ação" é atrelada a uma transição.

[7] É possível mostrar que essa formulação para dispositivos adaptativos pode representar qualquer mecanismo computacional Turing-computável.

[8] Todos eles obedecendo ao conceito de "adaptativo" já apresentado

[9] "Função" refere-se à especificação da entidade computacional, que é também descritível matematicamente.

[10] "Ação" refere-se à implementação de uma função adaptativa em alguma linguagem de programação.

[11] Aqui "evoluir" significa mudar na intenção de (tentar) cumprir um propósito.

[12] "Regra" pode ser entendido como um conjunto de declarações condicionais, em computação, descritíveis por conjuntos de sentenças *if-then-else*.

[13] Esta qualidade é uma avaliação que depende do julgamento dos projetistas e, com isso, pode ser considerada função da experiência de projeto e da familiaridade em relação ao assunto em análise que cada projetista apresenta.

[14] Ressalva: mesmo esta definição valendo para toda espécie de signo, este texto somente trata de signos genuínos, isto é, aquele tipo de signo que se arma, invariavelmente, em tríade ou a três relatos. Isso exclui do escopo deste trabalho os signos degenerados, que são casos especiais deste primeiro. Aos não iniciados neste assunto: podem ignorar esta nota pois não deverão sofrer com perdas de entendimento.

próxima geração de artefatos inteligentes capazes de transpor as limitações atuais da IA tradicional, como, por exemplo, o problema do *grounding* do símbolo", formulado originalmente em [3]. Resumidamente, esse problema é enunciável por duas questões complementares, diretamente correlacionadas com a investigação deste trabalho: 1ª. Como a semântica da interpretação de um sistema formal simbólico pode vir a ser algo próprio do sistema, ao invés de, exclusiva e simplesmente, parasitar os significados que habitam as mentes dos seus criadores?; e 2ª. Como *tokens* simbólicos – por exemplo, os símbolos das linguagens de computação – "desprovidos de significado próprio"[15] e manipuláveis só com base em suas formas (arbitrárias) podem ter fundamento (ou *grounding*) em coisas que não recaiam (incorram, ou ainda, que sejam equivalentes) a uma mera combinação de outros símbolos insignificantes, também "vazios de significado"? Até onde se sabe, este ainda é um problema em aberto. Aliás, uma instância dele é bem conhecida na área de inteligência artificial: o problema do "quadro" (*frame*), reincidentemente manifesto na abordagem puramente simbólica de modelagem do conhecimento. Neste, a dificuldade está em especificar formalmente (*framing*) o que varia (axiomas de efeito) e o que permanece constante (axiomas de quadro) num certo "domínio de conhecimento", eventualmente, combinando essas formulações lógicas (axiomas estado-sucessor). Na verdade, esse problema também pode ser enunciado como sendo a alta impossibilidade de elicitar todas as contingências[16] resultantes de todas as possíveis interações entre um ser artificial e um "mundo" em que está imerso. No entanto, é a essa atividade a que se dispõe um projetista ou programador cujo objetivo é o de simbolizar o conhecimento de uma cena (*frame*) – por exemplo, via representações munidas de símbolos desprovidos de *grounding*, como aqueles existentes em paradigmas computacionais puramente lógicos. Nesse sentido, o presente trabalho investiga a semiótica peirceana como subsídio estratégico para resolver esses problemas, uma vez que ela especifica um caminho a partir do qual se confere *grounding* (ou fundamento) aos símbolos – que constituem um caso particular de signo peirceano. Esse caminho é originado a partir da fusão entre dois paradigmas filosóficos: o idealismo (ou platonismo) e o naturalismo (ou realismo). Sem entrar em detalhes, uma contribuição relevante a partir da aplicação desse paradigma filosófico seria um caminho para a fusão do caráter arbitrário e convencional que é próprio dos símbolos[17] (em particular, os computacionais) com as qualidades subjacentes à experiência oriunda das interações com o mundo, da maneira como ele é capaz de se impor aos seres nele imersos, os quais, em correspondência (ou não) são capazes de o perceber (ou não) a partir dos seus atributos sensoriais inatos.

Posto isso, há outras questões gerais correlacionadas à essa investigação: 1. Até que ponto a computação estaria apta a simular processos naturais, como a interpretação executada pela espécie humana[18]? Qual a relação entre computação enquanto linguagem formal e outras linguagens não formais? Até que ponto é viável analisar uma linguagem formal sob a ótica do que é já sabido a respeito de linguagens naturais?

Caminhando em direção à solução desses problemas (e de outros correlacionados) este trabalho investiga o emprego da semiótica peirceana como base teórica para representação do processo de interpretação humana, vista como uma instância particular de semiose. Até onde é de conhecimento dos autores, ainda não é fato que a semiose é um processo computável; todavia, é verdade que: 1. Há modelos computacionais publicados – ver [2 e 11] – que se restringem a alguns aspectos desse processo, sem oferecer uma visão mais geral ou uma perspectiva do quão completo o modelo proposto seria em relação à teoria em que se apoia; e 2. O presente trabalho, inspirado nestes outros e na teoria semiótica subjacente, conseguiu, ao menos sob alguns aspectos, estabelecer uma descrição inicial adequada ao fenômeno em questão, considerando a base teórica estudada, principalmente no que tange às automodificações presentes como internas e naturais ao processo (de semiose). O fato deste processo ser representável sugere a hipótese de que este também seja algo automatizável, ao menos, até que isto seja refutado ou se prove um absurdo em pesquisas futuras.

Agora, no intuito de constatar se é fato que um dado processo é automatizável, primeiramente, é necessário encontrar uma modelagem (ou representação) adequada para ele, ao menos sob alguns aspectos específicos – isto é, uma representação que o simbolize com relação a essas qualidades. No caso da semiose vista como um processo de interpretação, é necessário entender três subprocessos que resultam em automodificações no objeto em que se aplicam,[19] que é o signo visto como uma cadeia semiótica, ou seja, como uma tríade que evolui em novas, a medida em que conhece mais (ou menos) sobre o seu objeto. Tais subprocessos constituem instâncias (ou formas) de *determinação* que é algo entendido, neste trabalho, como um efeito (lógico e/ou físico) de um primeiro resultando num segundo que não pode ser este primeiro que o gerou. Conhecidas essas características do processo, as técnicas adaptativas podem ser vistas como um subsídio formal bastante promissor para tentar compreendê-lo, especificá-lo e, quem sabe, realizá-lo computacionalmente.

Os assuntos abordados neste trabalho serão assim discutidos: na seção II, declara-se, objetiva e sinteticamente, o propósito do presente trabalho. Na seção III, apresenta-se o método de trabalho escolhido a fim de atingir os objetivos estabelecidos. Na seção IV, realiza-se a especificação da semiose vista como processo, seguindo a ordem declarada na seção III – Método, tal que: na seção IV.A, especifica-se, em alto nível de abstração, o processo de interpretação (ou semiose) como é definido pela teoria semiótica peirceana; na seção IV.A.2), apresenta-se um modelo não adaptativo para semiose; na seção IV.B, especificam-se os subprocessos adaptativos constituintes da

[15] Note que, na semiótica de Peirce, não há símbolo desprovido de significado. Essa é uma impressão que ocorre na computação, uma vez que é possível atribuir funcionalidades e significados específicos, selecionados pelo programador, aos símbolos computacionais também escolhidos por ele, o que revela não a falta de significado do símbolo, mas sim a sua natureza convencional e arbitrária, ao se colocar no lugar de seu objeto.

[16] De número que tende ao infinito, quanto maior a complexidade do ser e do contexto em que se encontra.

[17] E que lhes confere a tal impressão de serem "vazio em significado" até que alguma mente neles injete sentido.

[18] De tudo o que for (ou se insinuar como sendo) interpretável.

[19] Cada um desses subprocessos é trabalhado a partir da seção IV.A.2)b).

semiose; e, na seção IV.D, descreve-se como seria obtenível um modelo adaptativo, a partir dos resultados das seções anteriores. Na seção V, descrevem-se as contribuições deste trabalho. Na seção VI, explicitam-se os passos seguintes, a serem percorridos em trabalhos futuros. Por fim, na seção VII, dispõe-se tanto as referências, quanto informações adicionais sobre os autores.

II. Objetivo

Obter uma representação (mais) natural para a dinâmica da semiose, isto é, um modelo contendo os aspectos essenciais da dinâmica prevista para as automodificações inatas ao signo peirceano genuíno. Aqui, "natural" refere-se a elementos "inatos e observáveis num sentido puramente experiencial[20]" (CP, 1.204)[21] que possibilitam ao signo transformar-se, via semiose, incorporando informações sobre o objeto em que se põe no lugar, o que se dá em atos de interpretação. Empregando a teoria adaptativa proposta inicialmente em [5] como uma ferramenta de aproximação entre a representação (modelo) e o representado (o signo peirceano), deseja-se obter uma modelagem (mais) natural para o comportamento que o signo genuíno[14] apresenta.

III. Método

Com o intuito de satisfazer o propósito deste trabalho, foram pensadas as seguintes atividades, inseridas num ciclo de vida (de projeto) incremental e iterativo:

1. Especificar o processo de interpretação proposto, empregando a teoria semiótica destacada;
2. Apresentar um modelo não-adaptativo que satisfaça o processo estudado;
3. Considerar as partes adaptativas que integram o processo estudado; em outras palavras, detectar e explicar os subprocessos responsáveis por mudanças nos atributos de estado do dispositivo. Eles pressupõe a existência de automodificações que o dispositivo adaptativo é capaz de sofrer como efeitos, respostas ou reações a eventos que as disparem, conforme fundamenta a teoria adaptativa proposta em [5]. Para obter esses subprocessos que determinam (ou impõe) adaptações ao sistema, é necessário que eles sejam, nessa ordem:
    a. Internos (ou naturais) ao processo;
    b. Especificáveis (ou descritíveis) – em nível de projeto – como funções adaptativas ou usando uma forma de expressão alternativa ao conteúdo a ser exprimido, como imagens, diagramas e outros, desde que sejam potencialmente transdutíveis em algum tipo de especificação (ou modelo) formal;
    c. Realizáveis (ou implementáveis) computacionalmente, em algoritmos que constituam ações (ou rotinas) adaptativas, a partir das respectivas especificações obtidas no passo anterior. Se o resultado do passo b. for um modelo Turing-computável, então c. é um passo cujo resultado é um algoritmo.
4. Apresentar um modelo adaptativo como resultante do dispositivo não adaptativo original acoplado às funções adaptativas;
5. Refinar o resultado obtido, iterativamente, retornando do passo-em-análise a passos anteriores – se houver – até que a extensão e a profundidade, associadas ao passo-em-análise-atualizado, sejam tidas como razoáveis para avançar ao passo (ou atividade) seguinte.

IV. Do Processo

A. *Especificação*

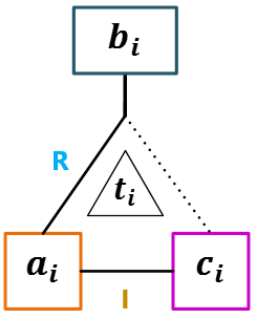
1) Geral

A seguir, o processo de semiose é apresentado, abstrata e genericamente – ver Fig. 1 – e especificado, em palavras – ver citação logo após a figura:

[20] Aqui, "experiência" é algo que se dá pelo emprego dos sentidos, tanto os de natureza física quanto os de natureza mental, como as ideias.

[21] NOTA DE CITAÇÃO: de acordo com a prática padrão, todas as referências a, ou citações extraídas de: 1. The Collected Papers of Charles Sanders Peirce, Vols. I-VI, ed. Charles Harshorne e Paul Weiss, Vols. VII-VIII, ed. Arthur Burks (Cambridge: Harvard University Press, 1931-35 e 1958); e 2. The Essential Peirce: selected philosophical writings, Vols. I-II, ed. Nathan Houser, são citadas assim: o número à esquerda do ponto decimal indica o número do volume; o número à direita do ponto decimal, designa o número do parágrafo.

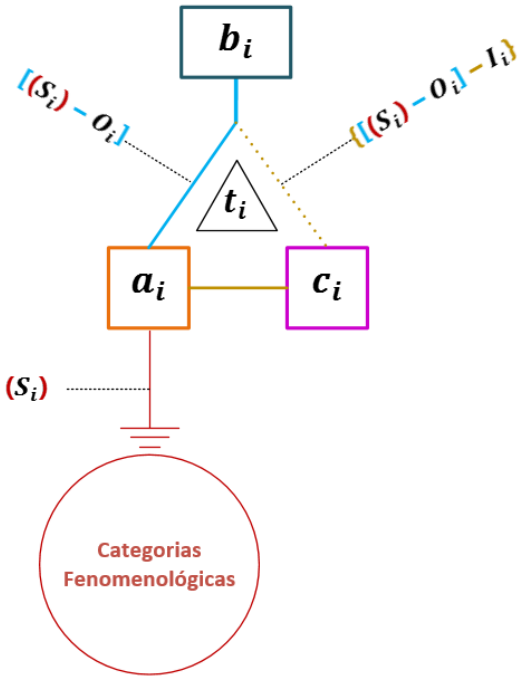
Fig. 1. Diagrama sintético referente à definição de signo de 1903, visto como uma relação triádica (t_i) ou armada a três relatos, sendo eles, nessa ordem: 1º. O signo em si mesmo, ou ainda, em função da sua própria natureza – a_i; 2º. O objeto do signo – b_i; e 3º. O interpretante do signo como terceiro relato sígnico – c_i. Além disso, a figura mostra "R" como sendo a relação entre o signo e o objeto; e "I" a relação entre o interpretante e o objeto, que é mediada pelo signo. Essas relações são desenvolvidas a seguir.



> *Um **Representamen**[22] é o Primeiro Correlato de uma relação triádica, o Segundo Correlato sendo nomeado como seu **Objeto**, e o **possível** Terceiro Correlato sendo denominado seu **Interpretante**, por cuja relação triádica o Interpretante é determinado como sendo o Primeiro Correlato da mesma relação triádica para o mesmo Objeto e para algum possível Interpretante. (CP, 2.242)*

Além disso, **a semiose**, ou o processo que descreve a evolução do signo, **é, no caso teórico geral, um processo potencialmente infinito, ordenado em tríades** (CP, 1.541; CP, 2.228; CP 2.303; CP, 8.332) **e que progride** (ou avança) – possibilitando ao signo incorporar mais informações sobre o seu objeto – **via atos de interpretação**. **Cada uma das tríades que integra esse processo estrutura-se como na Fig. 1 e constitui um signo**. Essa mesma figura também descreve que – invariavel e ordenadamente – o signo-como-tríade é descritível por três relações irredutíveis. "Irredutível" pode ser entendido com o mesmo significado da palavra "ortogonal" quando aplicada no domínio de vetores por exemplo: se cada uma das (três) relações que compõe o signo fosse (representada como) um vetor, ela seria ortogonal às duas outras relações, tal que elas poderiam ser vistas como um sistema de vetores linearmente independentes entre si. As relações que há na estrutura do signo-como-tríade são mostradas na Fig. 2 e descritas a seguir:

*Fig. 2. Diagrama referente à definição de signo de 1903, explicitando as relações internas à estrutura do signo-como-tríade t_i, sendo elas, nessa ordem: 1ª. A relação (S_i) do signo-como-primeiro-relato (a_i) em relação à sua própria natureza; 2ª. A relação entre o signo a_i e o seu objeto (b_i), simbolizada por "R" na Fig. 1 e, nesta imagem, escrita como **$[(S_i) - O_i]$**; e, 3ª. A relação entre o interpretante (c_i) e o objeto b_i, que é mediada pelo signo a_i, simbolizada por "I" na Fig. 1 e, aqui, registrada como **$\{[(S_i) - O_i] - I_i\}$***



Refinando a definição de signo-como-tríade, é importante estabelecer a visão da tríade semiótica (t_i) como sendo uma hierárquica composicional estabelecida pela conjunção e coexistência espaço-temporal de três relações, cada qual evidenciando um aspecto e papel intrínsecos ao conceito de signo-como-tríade. Em t_i, são elas:

1ª. A relação (S_i) do Signo (a_i) em referência à sua própria natureza, a qual expõe a categoria fundamental[23] a que o signo pertence, isto é, o fundamento (ou grounding) que constitui o signo;

2ª. A relação $[(S_i) - O_i]$ do signo com o seu objeto (b_i), em que o signo põe-se numa relação de representação, professando ser o objeto. Aqui a relação mostra como o signo tem a capacidade de se colocar no lugar do objeto, como se fosse um procurador no lugar de um cliente;

3ª. E última, a relação $\{[(S_i) - O_i] - I_i\}$ do signo com seu interpretante (c_i), este último sendo a ação ou o efeito (físico ou psíquico) potencial do signo ao ser

[22] Para os fins deste trabalho, é possível entender "representamen" como sinônimo de "signo-como-primeiro-relato", representado como "a_i" na Fig. 1. Aos curiosos: o conceito de representamen peirceano evoluiu a tal ponto que, se ele fosse visto como uma classe, então o signo poderia ser considerado uma subclasse desta primeira classe – ver (CP, 2.271; 1.540; 2.242) – quando entendemos por "classe" um identificador para um conjunto de qualidades.

[23] As categorias fundamentais peirceanas são construídas a partir da fenomenologia, que é a ciência da observação dos fenômenos, desde à chegada deles à mente *por meio e através* da percepção, até a possibilidade de resultarem em ações propositadas como interpretantes, isto é, efeitos da ação do signo ao ser interpretado. Como este trabalho trata apenas dos signos genuinamente triádicos, então, a categoria fundamental a que todos os fenômenos aqui analisados recaem – por exemplo, os símbolos e, mais especificamente ainda, os símbolos computacionais – é a Terceiridade. Essa categoria é identifica fenômenos que se armam a três elementos irredutíveis, sendo eles, nessa ordem: o signo, o objeto e o interpretante.

interpretado. No ato de interpretação, a tríade completa-se $\{[(S_i) - O_i] - I_i\}$ e, depois de interpretado, o signo (a_i) dá lugar ao seu interpretante-que-agora-é-o-signo-atualizado, ou ainda, o signo desenvolvido (a_{i+1} – ver Fig. 3). Este novo signo, por sua vez, também possuiria (eventualmente) um outro e novo interpretante (c_{i+1} – rever Fig. 3). Aqui se evidencia, com isso, o papel de mediador que o signo desempenha ao representar o objeto, pelo menos sob alguns aspectos específicos, para atualizar o interpretante, resultando no signo atualizado. Em suma, ***o ato de interpretação transforma um interpretante-como-signo-em-potencial (c_i), em t_i, num interpretante-como-signo-atualizado (a_{i+1}) que atua como signo (primeiro relato) na tríade seguinte, isto é, em t_{i+1}. Este novo signo nasce relacionando-se ao seu interpretante-como-signo-em-potencial (c_{i+1}), o que possibilita o signo corrente ser atualizado, num eventual ato futuro de interpretação.***

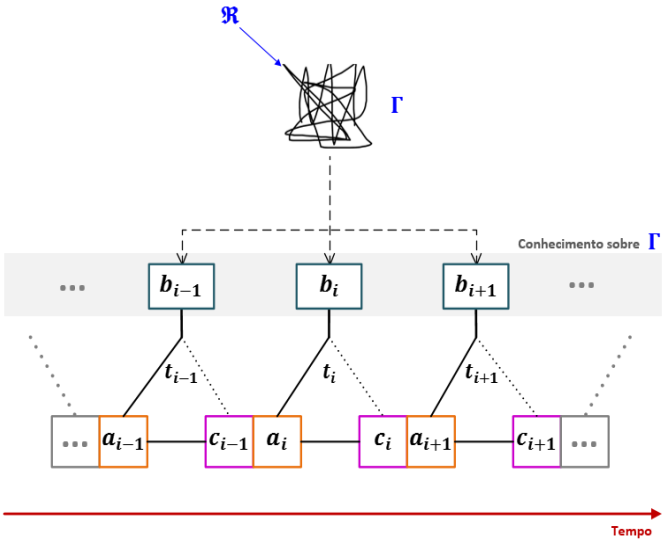
2) Das Restrições Teóricas Sobre o Processo Geral

As restrições impostas sobre o processo geral de semiose caracterizam-no como algo: 1. Ordenado em Tríade S-O-I e, no caso geral (ou teórico), potencialmente infinito; e 2. No qual os relatos sígnicos obedecem a três relações de determinação: 2.a. Do objeto (do signo) pelo "Real"[24]; 2.b. Do signo pelo seu objeto mirando a um interpretante; e 2.c. Do interpretante pelo objeto mediado pelo signo.

a) A Tríade Ordenada ((S – O) – I)

Seja a semiose vista como uma cadeia de tríades $T = \{\dots, t_{i-1}, t_i, t_{i+1}, \dots\}$ em que cada $t_i = (a_i, b_i, c_i)$ e $i \in \mathbb{Z}$ é um signo-visto-como-tríade, pertencente à T, como ilustra a Fig. 3. T é, com isso, nomeada uma cadeia semiótica de comprimento potencialmente infinito.

*Fig. 3. A semiose (ou o signo em evolução) como um processo ordenado em tríade e potencialmente infinito. É interessante considerar o modelo geral para $i \in \mathbb{Z}$ (e não $i \in \mathbb{N}$) porque assim, esse índice representa melhor as duas regressões ao infinito que ocorrem na semiose – uma em direção ao objeto inicial do signo ($b_{i \to -\infty}$) e o outra em direção ao interpretante final ($c_{i \to +\infty}$), no caso teórico geral. A imagem registra: 0. O objeto do "**real**", como sendo uma instância **Γ da "realidade" $\Re$**, incapturável na sua plenitude (flechas pontilhadas) através e por meio de signos; 1. Em laranja – ou $\forall a_i, i \in \mathbb{Z}$ – o Signo Ele Mesmo; 2. Em turquesa – ou $\forall b_i, i \in \mathbb{Z}$ – o Objeto **do signo**; e 3. Em roxo – ou $\forall c_i, i \in \mathbb{Z}$ – o Interpretante do signo. O pontilhado do objeto para o interpretante indica que a determinação do interpretante pelo objeto é, invariavelmente, mediada pelo signo.*



Inspirados em [3].

Assim a semiose corresponde à evolução dessa tríade irredutível (EP, 2.171) entre (Signo – Objeto) – Interpretante ([Burch 1991; Brunning 1997; Peirce SS 43; CP, 1.363; CP, 8.331; CP, 7.537] apud [2, p. 2]) que avança no decorrer do tempo, mas não necessariamente acoplada a ele, a não ser no sentido de incorporar informações sobre o objeto, por meio e através da determinação que sofre por ele, progressivamente; em outras palavras, quanto maior o intervalo transcorrido de tempo, maior é a possibilidade de o objeto ter determinado novos signos e, portanto, haver maior conhecimento sobre ele incorporado no signo-como-tríade que se adapta (ou se automodifica) pela semiose; no entanto, note que a semiose não depende do tempo enquanto atributo: haveria apenas uma correlação positiva entre o tempo e a "quantificação do conhecimento" incorporado no signo acerca do objeto.

Uma aplicação desse processo é a identificação da categoria (natural ou artificial) a que objetos (no sentido mais geral possível) pertencem, com base nos atributos invariantes **mínimos** (qualidades essenciais) para que sejam considerados membros da potencial classe (ou categoria). Observe que essas qualidades mínimas são obteníveis por operações realizadas "a partir dos objetos distais, por meio e através das projeções

[24] Entendendo-se por "Real" aquilo que está fora e seja independente do signo, embora seja também aquilo que seria representado caso a ação do signo, ou semiose, chegasse a um ponto ideal perfeito. Nessa última acepção, o "Real" constitui o que for representável num processo ideal (potencialmente infinito) de semiose e não qualquer representação (que é, invariavelmente, um signo).

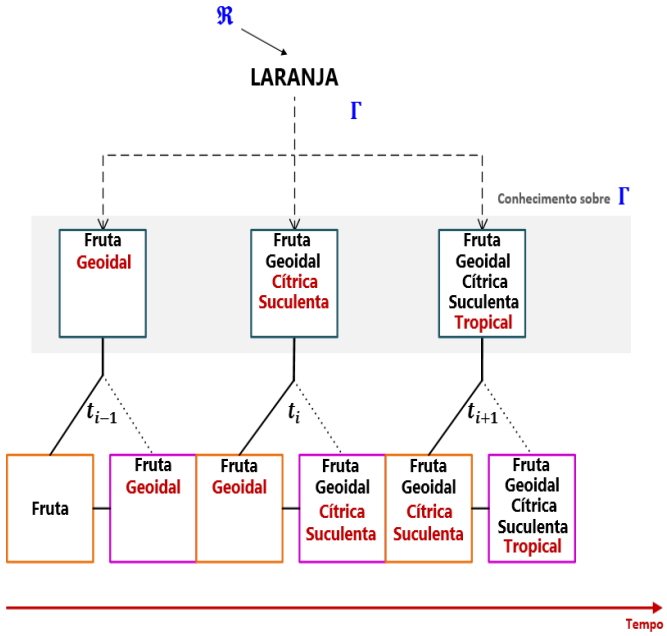
sensoriais proximais" [3, pp. 1, 6, 8-9] e constituiriam, em parte, os tais "atos de interpretação" para esta aplicação.

Um exemplo concebível referente à aplicação anterior pode ser construído a partir da discussão feita em [8, pp. 16-18] sobre a definição (ou o conceito) de "laranja". Imagine este objeto. Será que você imaginou uma fruta? Uma cor? Ou um intermediário humano atuando em transações fraudulentas? Eliminando o problema de ambiguidade do sujeito, considere a fruta. Agora, será que a laranja imaginada por você seria idêntica àquela resultante de um imaginário coletivo, isto é, da conjunção dos predicados de cada pessoa de um grupo, expondo seu respectivo conceito da fruta? É provável que não, porque características novas vão sendo listadas a medida em que as pessoas declaram suas experiências com laranjas[25], enunciando predicados[26] não declarados por outros do coletivo. Isso implica que o conceito (ou modelo/representação) de laranja pode evoluir em função (e a medida em que) as pessoas manifestam suas experiências individuais com as laranjas, resultando numa definição coletiva de "laranja". Esse processo de evolução do conceito-modelo-representação de "laranja" (fruta) é descrito na instanciação da Fig. 3 – caso geral – resultando na Fig. 4. Nela se retrata uma instância **Γ** do "Real" **ℜ** – isto é, uma laranja como ela é evocada por um de seus símbolos que, no caso da figura, possui suporte em linguagem textual, por ser uma palavra em língua portuguesa ("LARANJA"). **Γ, ou a laranja do "Real"** – que reside e resiste fora do signo – determina o objeto-como-segundo-relato do signo (ou objeto imediato) – quadrados rotulados com a letra "b" na figura.

Γ é incapturável, em sua plenitude, por qualquer uma de suas possíveis representações – isto é, pelos signos mostrados como quadrados rotulados com a letra "a" na figura – uma vez que elas se põem em seu lugar como se fossem a "verdadeira" laranja – mas nunca serão ela – a partir do conhecimento que detém sobre esta num certo tempo-espaço (objeto b_i), obtido a partir do conjunto de experiências com **Γ**. Tal conhecimento sobre **Γ** permite não só identificar, classificar e interpretar elementos da classe "LARANJA", como também possibilita agrupá-los ao estabelecer condições de identidade que permitem comparar objetos candidatos a ser "LARANJA", mas que, todavia, só o serão caso passem em testes mínimos (comparações) que supostamente "garantem" ou lastreiam a identidade entre os objetos avaliados e a classe "LARANJA". Esse lastro é a experiência com a classe em relação às experiências obtida ao experienciar os objetos a serem classificados. Uma identificação (ou categorização) como descrito acima pode ser realizada, por exemplo, detectando os predicados que forem compartilhados entre os objetos-alvos da comparação.

Outro ponto é, na Fig. 4, também se mostra que todo signo interpretado (ou atualizado) possui e exibe um conhecimento maior, na forma simples de um novo predicado, em relação aos seus antecessores, o que denota que ocorreu uma experiência particular, direta ou indiretamente, com "laranja". Reitera-se que informações sobre o objeto são incorporadas pelo signo ao ser atualizado, tornando-se conhecimento sobre ele, obtido *por meio e através da experiência* com relação a esse objeto.

Fig. 4. Uma aplicação da semiose, formulada para o caso teórico geral na Fig. 3, constituindo uma cadeia finita de signos que se referem a um objeto particular do "Real" **ℜ** *– aqui,* ***uma "laranja", que NÃO É a palavra (nem seria a imagem dela)****, é simbolizada por* ***Γ****. A representação de* ***Γ****, dada por meio da palavra "LARANJA", é um abuso intencional a fim de provocar o leitor quanto aos limites da capacidade humana de representar. Em outras palavras, o intuito é chamar a atenção para o fato de que a "verdadeira" laranja* ***Γ****, enquanto ser pertencente a* **ℜ***, nunca poderá será alguma de suas possíveis representações – como uma de suas imagens, por exemplo – o que implica que* ***esta palavra mostrada como sendo Γ não é senão uma sugestão (ou indicação) do legítimo objeto Γ****. Esta figura também exibe a semiose (ou evolução) do signo "laranja", partindo de uma representação mais simples, à esquerda para uma representação mais complexa, à direita, todas baseadas apenas em qualidades.*



Para completar o exemplo anterior, é razoável pensar que ninguém qualificaria uma laranja, por exemplo, como sendo de cor verde. Se alguém de boa saúde mental afirmar, contudo, que depois de ir à Marte encontrou algo verde, com as mesmas predicações da laranja terráquea, exceto pelo fato da cor ser verde e da procedência ser marciana, então o conceito de laranja potencialmente evoluirá, incorporando as laranjas de origem marciana, de cor verde, como sendo elementos da classe.

De volta à especificação, é importante lembrar que, como já foi apontado na base teórica vista até aqui – rever Fig. 3 – que "a relação triádica é o esquema analítico elementar de um processo de continuidade que tanto regride quanto se prolonga ao infinito" [9, p. 18]. No entanto e, ao mesmo tempo, como explica (Ransdell apud [9, p. 20]), uma semiose começa sempre *in media res* (ou no meio da cadeia), em qualquer ato de

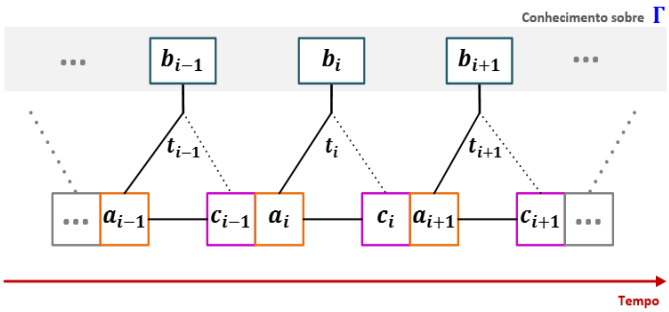
[25] Às vezes, estabelecendo paradoxos e contrastes umas com as outras, ao escancarar diferenças quanto a gostos, sensações e com isso, confrontar experiências individuais que diferem umas das outras opostas, a partir das comparações entre elas.

[26] E outros recursos de linguagem para fazer um objeto ser conhecido por meio da comunicação, por exemplo, a asserção de proposições.

interpretação. Com isso, apenas é possível interpretar, especificando ou abstraindo, tão extensiva quanto minuciosamente, quanto forem satisfeitos os propósitos que orientam uma interpretação. Assim que os satisfaz, a cadeia semiótica torna-se algo de comprimento finito, com início, meio e fim – rever Fig. 4.

Agora, a fim de especificar as relações lógicas existentes na dinâmica natural da semiose, é conveniente reexibir a Fig. 3, contendo apenas os relatos do signo-como-processo ordenado em tríades e infinito em potência.

Fig. 5. Fig. 3, sem o elemento do "Real": apenas os relatos do signo, o que o torna uma representação mais adequada da semiose.



A expressão lógica a seguir evidencia o signo como algo capaz de evoluir, incorporando informações sobre o objeto, ao ser interpretado, resultando no signo atualizado.

Expressão 1 – Relação lógica entre as "coisas" c_i e a_i – que são signos – enquanto relatos da tríade rotulada como t_i. Nessa tríade, c_i assume o papel de interpretante relativo ao signo a_i, que atua como primeiro relato da tríade t_i. Portanto, c_i é algo distinto de a_i, sendo concebido como o efeito (psíquico e/ou físico) que a_i pode engatilhar, ao ser interpretado.

$$\forall i \in \mathbb{Z}, (c_i \neq a_i)$$

Inspirados em (CP, 8.332).

É fundamental perceber que, nessa relação, c_i é interpretante de a_i (signo) e ambos coexistem na tríade t_i. c_i é um signo mais evoluído que a_i, mesmo que ambos se refiram ao mesmo objeto "real", porque ele é o resultado da interpretação de a_i, processo que incorpora informação sobre o objeto, resultando no signo atualizado a_{i+1}.

Na próxima relação lógica, é destacada a transição de certa tríade (t_i) para sua consecutiva (t_{i+1}), isto é, evidencia-se o passo evolutivo (ou progressivo) resultante de um ato de interpretação.

Expressão 2 – Relação lógica entre as "coisas" c_i e a_{i+1} – que são signos – enquanto relatos das tríades rotuladas como t_i e t_{i+1}, nessa ordem. Em t_i, c_i assume o papel de interpretante; e, em t_{i+1}, a_{i+1} atua como signo. O predicado deve ser lido como: " a_i determina [27] c_i , podendo gerar a_{i+1} ".

$$\forall i \in \mathbb{Z}, Determina(a_i, c_i) = \{\varepsilon \mid a_{i+1}\}$$

Inspirados em (CP, 8.332).

Explicando melhor a relação lógica anterior, na tríade t_i, o (papel de) interpretante c_i – que é um signo em potencial – ao ser interpretado com sucesso, faz c_i **tornar-se** a_{i+1}, o que significa fazê-lo deixar de atuar como (no papel de) interpretante, em t_i, e assumir o papel de signo na tríade sucessora: t_{i+1}. Essa expressão também sugere, por indução, a evolução do processo rumo a um interpretante final.

b) A Determinação e suas Espécies como Subprocessos

Como já foi mencionado na Introdução, mesmo que em altíssimo nível de abstração, a *determinação* também pode ser vista como um evento, fato ou ocorrência de natureza causal (física e/ou lógica, biológica ou artificial), que pode ser imaginada como um gatilho disparador de um ato interpretativo o qual, se levado às últimas consequências, pode ser representado pela incorporação de uma nova tríade à cadeia T.

A determinação também explica a ordem imposta aos relatos que formam a tríade sígnica, assim como dá indícios do ciclo de vida do signo, explicitando a dinâmica da semiose:

> *O signo é determinado pelo objeto relativamente ao interpretante, e determina o interpretante em referência ao objeto de um modo tal que o interpretante é determinado pelo objeto por meio e através da mediação do signo.*
> [MS 318:81, *apud* 4, p. 23]

Seja uma tríade $t_i = (S_i, O_i, I_i)$, $i \in \mathbb{Z}$, um elemento pertencente a uma cadeia semiótica T, composta de tríades, como especificada na seção anterior. Então, em t_i, há três espécies de determinação:

1. O_d determina O_i mirando a S_i
2. O_i determina S_i relativamente (ou mirando) a I_i; e
3. S_i determina I_i em referência a O_i.

O primeiro tipo de determinação explicita a imposição do objeto dinâmico (ou instância do "Real") sobre o objeto imediato (O_i), que é o segundo relato da tríade t_i. A geração de O_i mirar a S significa que o objeto já nasceria visando a potencial signo que o represente.

Já o segundo tipo de determinação significa que o signo (S_i) tem primazia lógica sobre o seu objeto (O_i), isto é, "embora o signo seja determinado pelo objeto, este, por sua vez, só é logicamente acessível *por meio e através* da mediação do signo"

[27] "Determinar" é um conceito aprofundado na seção IV.A.2)b)

[9, p. 25], ou ainda, o objeto só é acessível depois de se projetar num signo.

Pelo fato do objeto ser algo que transborda o signo e que resiste à sua evolução – do contrário, seriam ambos uma só coisa – **há a determinação do signo pelo objeto e não mera substituição**: ao mesmo tempo em que o signo é incapaz de substituir plenamente o objeto, colocando-se apenas em seu lugar, ele mira à ideia que o objeto é capaz de produzir ou modificar, isto é, seu interpretante.

É justamente essa aptidão gerativa – inata do signo (S_i) – que é expressa no terceiro tipo de determinação – item 3 da lista. Em resumo, o signo tem a capacidade (ou a potência) de gerar um efeito (ou ação), isto é, um interpretante (I_i), a partir da determinação que ele sofre do novo estado de conhecimento sobre o objeto (O_{i+1}), porém, note que o interpretante resulta sempre da mediação do objeto pelo signo – e não pela influência desintermediada (ou direta) do objeto.

B. Modelagem Formal Não Adaptativa Resultante

Como ilustra a Fig. 5, seja uma cadeia semiótica de tríades uma sequência reordenável $T = [\dots, t_{i-1}, t_i, t_{i+1}, \dots]$ em que $\forall i \in \mathbb{Z}, t_i = (a_i, b_i, c_i)$ é uma tríade-pertencente à cadeia – só definível no contexto de T – caso seu:

1. Primeiro termo seja (ou assuma o papel de) Signo (a_i);
2. Segundo seja (ou assuma o papel de) Objeto (b_i); e
3. Terceiro seja (ou assuma o papel de) Interpretante (c_i).

Já as restrições estudadas, impostas à cadeia T pela base teórica, significam que:

- Há um objeto dinâmico (**Γ**), incapturável pelas malhas dos signos porque ele resiste e existe no "Real" $\mathfrak{R}$, de tal modo que ele determina o Objeto (b_i) do Signo (a_i) como objeto imediato dele;

- É fundamental perceber que, em qualquer tríade $(\mathrm{t_i})$ da cadeia:
  - O objeto b_i determina o signo $\mathrm{a_i}$, que, ao nascer, já mira atualizar o interpretante-como-signo-potencial $\mathrm{c_i}$, no intuito de atualizar a_i em a_{i+1};
  - $\mathrm{c_i}$ é o interpretante de $\mathrm{a_i}$ (signo) e ambos coexistem na tríade $\mathrm{t_i}$. Ao ser atualizado, $\mathrm{c_i}$ incorpora mais informação sobre o objeto b_i e, por isso, ele constituirá um signo mais "evoluído" que o seu antecessor $(\mathrm{a_i})$, mesmo ambos se colocando no lugar do mesmo objeto.
    - ✓ Desse fato, infere-se que todo Interpretante (c_i) não é logicamente equivalente ao Signo (a_i) quando coexistem enquanto relatos numa mesma tríade (t_i) ; apenas ao ser atualizado, o interpretante-como-signo-potencial c_i torna-se uma versão atualizada de a_i – isto é, a_{i+1} – o que corresponde ao papel de signo na tríade seguinte à $\mathrm{t_i}$ – ou seja, $\mathrm{t_{i+1}}$ – Expressão 1.

- Na relação lógica restante – Expressão 2 – é destacada a transição de certa tríade $(\mathrm{t_i})$ para sua consecutiva $(\mathrm{t_{i+1}})$, ou seja, evidencia-se o passo evolutivo (ou progressivo) que resulta de um ato de interpretação, a partir de uma nova ocorrência de determinação do objeto, nesta situação, mediado pelo signo.

C. Elementos Dinâmicos

Abaixo, seguem os subprocessos adaptativos imperativos, intrínsecos e essenciais à semiose.

1) Previstos

Elicitam-se, aqui, três origens para as adaptações possíveis de ocorrer na cadeia semiótica, sem o intuito de ser exaustivo[28], mas sim, com o propósito de estabelecer um ponto de partida, em alto nível de abstração, para a dinâmica do funcionamento natural da tríade semiótica, vista aqui como um processo de semiose ordenado em tríades e potencialmente infinito.

Nesta seção, a ordem de apresentação das fontes de adaptação obedece uma sequência lógica, de tal modo que esta não requisita ou vincula-se a uma coordenada tempo-espacial. Com isso, é permitido (ou válido) que existam intervalos temporais em que há apenas adaptações sendo originadas de alguma dessas fontes, sem haver outras adaptações de outras fontes. Define-se essa propriedade como "princípio da autonomia das fontes"[29]. Por exemplo, é possível ocorrer adaptações originadas da primeira fonte, sem que haja outras nas duas fontes restantes, necessariamente.

Acoplada à ordem lógica acima, está uma dependência existencial hierárquica entre as fontes, menos no caso da primeira, que possui existência própria – o que implica o fato de essa fonte ser não só autônoma como também independente das outras. Então, excetuando-se o caso desta primeira fonte, as demais precisam que suas antecessoras tenham se manifestado para, só então, poderem manifestar-se, via produção e imposição (ou determinação) de adaptações à cadeia semiótica. Feitas essas considerações, vamos à descrição de cada uma:

A primeira fonte de adaptações é constituída a partir do choque entre um objeto do "Real" e uma cadeia semiótica associada a ele. Esse choque ocorre no sentido de o "Real" (ou objeto dinâmico) ser projetável. Ao projetar-se, a projeção resultante é o objeto (imediato), ou o segundo relato da tríade

[28] É possível conceber não só mecanismos gerativos, foco deste texto, como outros de eliminação, que atuariam como se fossem dispositivos de memória temporária ou limitada, removendo tríades da cadeia semiótica por uma questão de desuso do conhecimento armazenado sobre o objeto, por exemplo.

[29] Não confundir "autonomia" com "independência" lógica. Pode haver autonomia, com ou sem dependência de uma fonte em relação às outras, como o parágrafo seguinte desenvolve.

semiótica [30] . Em síntese, ao se impor (ou se manifestar forçosamente) o "Real" produz, ou determina, como efeito (físico e/ou psíquico), objetos (imediatos), cada qual podendo atuar (ou não) como (no papel de) segundo relato de alguma tríade que se coloca no lugar do objeto e, com isso, é signo dele.

Assim, o objeto (imediato) resulta da capacidade natural que o "Real" tem de se projetar ou impor-se. Ao ser percebido, nesse exato instante, o objeto torna-se sugestível e insinua-se à interpretação; em outras palavras, ele pode ser visto como uma entrada que adentra num ato de interpretação como (no papel de) signo, o que também pode ser escrito, muito economicamente, como: o signo é determinado pelo objeto. Essa determinação constitui a segunda fonte de adaptações, porque ela define que "coisas" são geradas a partir da existência do objeto imediato, mas nem todas elas constituem signos: apenas o fazem aquelas que são interpretáveis ou que se insinuam como sendo.

Uma vez obtida alguma interpretação para o objeto, isto é, algum signo deste – capaz de referenciar o objeto em que se coloca no lugar, ao menos sob alguns aspectos específicos – este ser recém-nascido já surge mirando a um interpretante, que é um outro signo no qual o primeiro é capaz de se desenvolver, uma vez mais, por meio de uma nova rodada de determinação do objeto do "Real". Contudo, esta segunda determinação não é igual à da primeira fonte já estudada: aqui ocorre mediação do que já é conhecido sobre o objeto – tanto por experiência colateral, como por conhecimento prévio guardado em outros signos anteriores – em síntese, ocorre uma intermediação pelo signo atual, que se encontra prestes a ser interpretado. Portanto, essa terceira espécie de determinação mostra que o signo só é capaz de representar o objeto do "Real" (ou dinâmico) sob certos aspectos específicos, todavia, jamais em sua plenitude. E isso é verdade, reiterando, porque, ao ser algo do "Real", o objeto dinâmico – que reside e resiste na sua própria existência, fora do signo – é algo inelutavelmente livre para se impor ao signo, que só o representa justamente com relação aos aspectos que ele manifestar/ou/ará (ou impor/s/rá). Assim sendo, por mais que o signo evolua (ou seja, incorpore informações sobre o objeto), ele ainda não deixará de ser (ou atuar no papel de signo) – pois jamais deixará de ser o que é para se constituir como aquilo que ele professa ser (seu objeto) – o que implica que o propósito do signo é lutar para ser o objeto do "Real", à medida em que esse objeto vai sendo descoberto, continuamente, por mais que isso seja, no absoluto, uma tarefa impossível. Para concluir, esse último fato implica que essa terceira espécie de determinação também constitui uma fonte de adaptação, a terceira, porque ao buscar ser o objeto, o signo só é capaz de o fazer pela atualização do interpretante-como-signo-em-potencial transformando-o em signo atual, o que resulta na sequência potencialmente infinita vista na seção IV.A.2)a). E esta atualização do interpretante resulta, invariavelmente, na geração de um novo interpretante ligado a esse antigo interpretante-que-se-tornou-signo, ou seja, este é, por fim, um subprocesso de automodificação e, sendo assim, é "adaptativo".

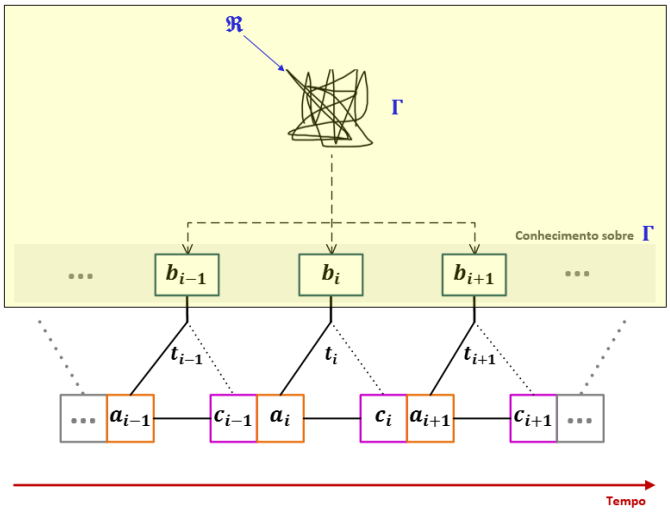
2) Especificáveis

Estudadas as três fontes de adaptatividade previstas, a especificação de cada uma será apresentada abaixo, em duas linguagens complementares: 1. Textual – ou via linguagem natural; e 2. Diagramaticamente – ou via imagens. Isso porque a notação formal para o processo estudado ainda está em fase de desenvolvimento, constituindo-se como uma atividade (de especificação) em progresso.

Sejam as fontes:

1. A imposição de uma instância do "Real" (objeto dinâmico) determinando um objeto imediato[31] que, por sua natureza, resulta da experiência colateral, ou ainda, do choque entre o "Real" e uma cadeia semiótica que o persegue com o propósito de o representar; esse objeto é inserido à cadeia como segundo relato de alguma nova potencial tríade, ainda a ser construída – ver Fig. 6.

Fig. 6. Semiose, com ênfase na representação da determinação do objeto imediato (b_i) como resultado da determinação do objeto dinâmico ***Γ*** *sobre a cadeia semiótica* ***T****.*



Como inferência formal possível a partir dessa determinação, poderia ser obtida, a título de ilustração, a expressão a seguir:

[30] A definição mais geral de signo não requisita uma percepção unida à mente. Aliás, tampouco demanda que haja uma mente ou uma percepção. Logo, ao definir o signo como algo de existência própria, é razoável tentar eliminar qualquer espécie de mentalismo.

[31] Para simplificar, o objeto imediato determinado será considerado, necessariamente, um objeto novo, o que não é verdade em todos os casos, porque ele (ou uma instância equivalente a ele) já pode existir como elemento da cadeia.

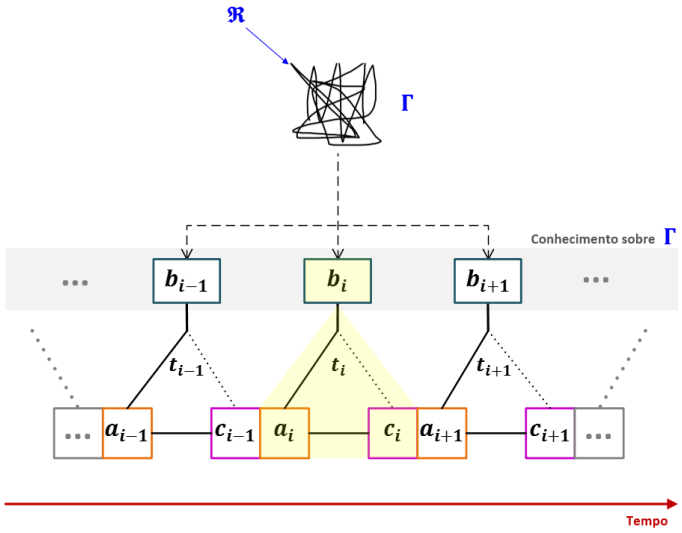
Expressão 3 – Relação de todo objeto imediato (b_i) com o objeto dinâmico Γ.

$$\forall i \in \mathbb{Z}, \exists \Gamma \in \mathfrak{R} \mid Determina(\mathfrak{R}, \Gamma) \wedge Determina(\Gamma, b_i) \wedge ObjetoImediato(b_i, \Gamma)$$

Fonte: Autor, baseado em (CP, 8.333, 4.536, 8.314, 8.183).

2. A determinação de um signo acoplado ao interpretante como resultado da determinação do objeto imediato – ver Fig. 7.

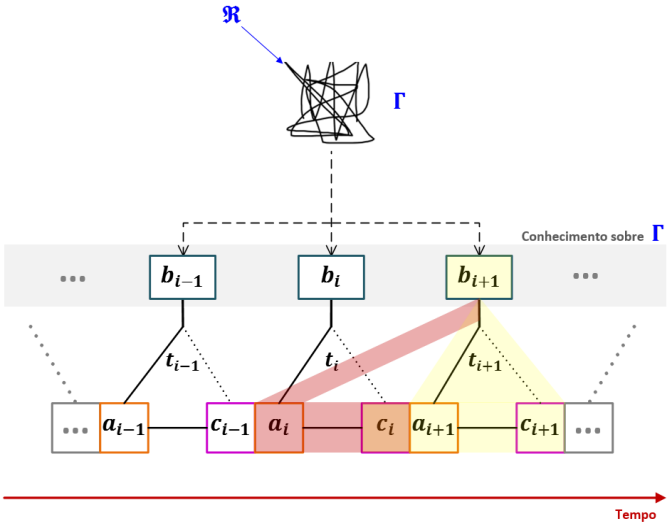
Fig. 7. Semiose, com ênfase na representação da determinação do signo (a_i) e o seu interpretante (c_i) a partir da determinação pelo objeto imediato (b_i).



E, finalmente,

3. A determinação do interpretante, via determinação do próprio objeto mediado pelo signo – ver Fig. 8.

Fig. 8. Semiose, com ênfase na representação da determinação do interpretante-como-signo-potencial (c_i) por meio e através do signo (a_i). Essa determinação ocorre não só com base no que já é conhecido sobre o objeto (b_i)[32] mas também no que está sendo conhecido sobre ele agora – o que poderia ser abstraído pela diferença: $b_{i+1} - b_i$. Em síntese, c_i torna-se a_{i+1} ao ser determinado por b_{i+1}, objeto que é mediado pelo signo a_i. Ao se constituir o signo a_{i+1}, seu interpretante-como-signo-potencial correspondente também é gerado – isto é, c_{i+1}. Com isso, houve a atualização da tríade t_i em $t_{i+1} = (a_{i+1}, b_{i+1}, c_{i+1})$, o que simboliza um passo evolutivo – também denominado "ato de interpretação" – do signo-como-tríade que é um ser cuja evolução impõe também a evolução da cadeia semiótica T, que pode ser vista como um histórico da trajetória de evolução do signo.



3) *Realizáveis (ou Implementáveis) Computacionalmente*

Essa análise pressupõe o término da atividade anterior, portanto, será reservada a trabalhos futuros. Contudo, já é possível adiantar algumas observações relevantes à fase de implementação dos subprocessos adaptativos elicitados:

- A primeira fonte de adaptações pressupõe a capacidade do "Real", instanciado em um objeto dinâmico, em se projetar, no intuito de gerar objetos imediatos. Com isso, essa capacidade implica em outra: a de que algo é capaz de sofrer a projeção imposta, mediante sensibilidade à alteridade do "Real";

- Ao existir, o objeto imediato é algo capaz de gerar signos-interpretantes, isto é, ele é um ser que pode resultar em outros, alguns dos quais sugerem ser ou são de fato passíveis de interpretação; portanto, além de poder sofrer a projeção do "Real" sobre si, o sistema tem que possuir a habilidade de "conectar" signos-interpretantes aos objetos recebidos como entradas iniciais; essa conexão é realizada no intuito de entender o "novelo sensorial" que constitui a projeção sensorial inicial, ao impor "Ordem" à ela, por exemplo:

[32] Isto se houver algum conhecimento sobre ele… senão, a determinação corrente é a primeira e uma nova cadeia T pode ser criada.

- A notando, isto é, percebendo sua existência (ou presença);
- A distinguindo (ou diferenciando) de outras coisas;
- A identificando (ou classificando);

❖ Um signo já disponível na cadeia, ao ser interpretado, deve ser capaz de produzir outro; e isso pressupõe a aptidão do signo em interpretar algo determinado pelo "Real" se projeta. Logo, o aprofundamento no subprocesso de interpretação – ou conversão do interpretante em signo atualizado a partir do signo desatualizado – é essencial.

D. Modelagem Formal Adaptativa Resultante

Seja o modelo formal não adaptativo para o processo estudado descrito na seção IV.B. Acoplando-se a ele os subprocessos adaptativos estudados na seção IV.C, obtém-se o modelo formal adaptativo resultante seguindo o método de especificação proposto em III.

Esta etapa é uma atividade de integração futura, somente realizável tendo sido obtidas as suas partes integrantes.

V. Resultados

Como resultante principal, está proposto um modelo inicial adaptativo consistente com o processo de evolução do signo peirceano genuíno, que se mostra naturalmente um processo de automodificação com adaptações originadas a partir das relações de determinação inatas a seu comportamento. O modelo descrito encontra-se lastreado na teoria semiótica peirceana e encontra-se em construção, apoiando-se sobre os conceitos já estabelecidos pela técnica adaptativa referenciada.

Incapaz de ser completo, por também ser (ou atuar como) signo, este modelo inicial oferece uma primeira visão – em altíssimo nível de abstração – relativa à dinâmica da semiose (ou evolução) do signo-como-tríade, sem a intenção de se aprofundar muito quanto às atividades que há em cada um dos três subprocessos estudados.

VI. Próximos Passos

Finda esta primeira iteração sobre o processo de semiose estudado, ainda é necessário:

1. Estabelecer uma notação formal para os subprocessos adaptativos;
2. Validar, iterativa e continuamente, o modelo resultante, de preferência, com o apoio de especialistas no processo semiótico estudado;
3. Investigar, em extensão e profundidade, cada subprocesso elicitado como parte da dinâmica relativa e inata ao signo. Caso necessário – por exemplo, por razões de complexidade – talvez seja interessante explodir cada subprocesso em outros de menor escopo, todavia, ainda sim, preservando a base teórica (ou não);
4. Inspecionar cada especificação, detalhando-a se necessário, aplicando alguma técnica de modelagem aderente: lógica formal, cálculo lambda e/ou outras;
5. Estudar a potencialidade de implementação (ou realização) dos subprocessos especificados e formatados como modelos;
6. Aplicar os modelos obtidos em casos de uso significativos, para fins de verificação e validação do propósito a que se destinam cumprir, isto é, atos de interpretação.

VII. Referências

1. CAYA, R. E. C.; NETO, J. J. **Personalização, "Customização", Adaptabilidade e Adaptatividade**Memórias do WTA 2016. **Anais**...São Paulo: WTA 2016, 2016Disponível em: <http://lta.poli.usp.br/lta/publicacoes/artigos/2016/memorias-do-wta-2016>
2. GOMES, A.; GUDWIN, R.; QUEIROZ, J. Towards meaning processes in computers from Peircean semiotics. **SEED Journal--Semiotics, Evolution, Energy, and Development**, v. 3, n. 2, p. 69–79, 2003.
3. HARNAD, S. The symbol grounding problem. **Physica D: Nonlinear Phenomena**, v. 42, n. 1–3, p. 335–346, 1990.
4. LISZKA, J. J. **A General Introduction to the Semeiotic of Charles Sanders Peirce**. Bloomington: Indiana University Press, 1996.
5. NETO, J. J. Adaptive automata for context-dependent languages. **ACM SIGPLAN Notices**, v. 29, n. 9, p. 115–124, 1 set. 1994.
6. PEIRCE, C. S. **Collected papers of charles sanders peirce**. [s.l.] Harvard University Press, 1974.
7. PEIRCE, C. S.; HOUSER, N. **The essential Peirce: selected philosophical writings**. [s.l.] Indiana University Press, 1998. v. 2
8. ROMANINI, V. **A contemporaneidade de Peirce no pensamento comunicacional**. São Paulo: [s.n.]. Disponível em: <http://www.academia.edu/30424930/A_contemporaneidade_de_Peirce_no_pensamento_comunicacional>. Acesso em: 29 dez. 2016.
9. SANTAELLA, L. **A teoria geral dos signos: como as linguagens significam as coisas**. São Paulo: Pioneira, 2000.
10. SANTAELLA, L.; NÖTH, W.; MENEZES, P. **O que é semiótica**. São Paulo: Editora Brasiliense, 1983.
11. SARBO, J. J.; FARKAS, J. I.; VAN BREEMEN, A. J. J. **Knowledge in Formation: A Computational Theory of Interpretation**. [s.l.] Springer Science & Business Media, 2011.
12. TAHVILDARI, M. S. AND L. "Self-adaptive software: Landscape and research challenges". **ACM Transactions on Autonomous and Adaptive Systems**, v. V, n. March, p. 1--41, 2009.

Ian Silva Oliveira é mestrando em Engenharia da Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP), atuando como membro do LTA – Laboratório de Linguagens e Tecnologia Adaptativa do PCS – Departamento de Engenharia de Computação e Sistemas Digitais. Graduado em Engenharia Elétrica – Ênfase em Computação, pela EPUSP (2015). Atua em pesquisa desde 2011, em Iniciação Científica, que durou até o fim da graduação. Seus interesses de pesquisa não se limitam a, mas incluem: modelagem de processos que se automodificam em ambientes dinâmicos; computabilidade teórica e aplicada à solução de problemas; lógicas formais e naturais; cognição e percepção; análise, processamento e síntese de linguagens, semiótica computacional.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo 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.

João Eduardo Kogler Jr. é correntemente membro do Departamento de Engenharia de Sistemas Eletrônicos da Escola Politécnica da Universidade de São Paulo (USP). Graduado em engenharia elétrica e em física, é doutor e mestre pela USP. Desenvolveu suas pesquisas de PhD e MSC, respectivamente, nas áreas de visão computacional e modelagem em neurociência. Desenvolveu seu trabalho de pesquisa de doutorado no Princeton SRC Research Lab, New Jersey, Estados Unidos, e de pós-doutorado como cientista visitante do INRIA Sophia Antipolis, França. Seus interesses atuais de pesquisa concentram-se em modelagem matemática e computacional da cognição e da percepção e em inteligência artificial, com aplicações à robótica cognitiva principalmente. Entre as abordagens que estuda encontram-se os modelos de sistemas dinâmicos baseados em mecânica estatística de redes, diagramas e teoria de categorias, computação adaptativa e aprendizado de máquina. Estuda modelos fundamentados na semiótica e filosofia da mente e cognição.

Anderson Vinícius Romanini é, atualmente, professor doutor pela Escola de Comunicações e Artes (ECA) da Universidade de São Paulo (USP), atuando como coordenador do LTS – Laboratório Transdisciplinar de Semiótica do CCA – Departamento de Comunicações e Artes. PhD em Semiótica pela Universidade de Indiana (EUA). Em Ciências da Comunicação, pela USP, recebeu: seu doutorado – modalidade sanduíche, com período na Universidade de Indiana, EUA; mestrado; e graduação. É o atual presidente da Sociedade Brasileira de Ciência Cognitiva (SBCC) na gestão de 2016 – 2018. É também pesquisador sênior no Centro de Lógica e Epistemologia (CLE) da Unicamp. Seus interesses correntes de pesquisa abrangem: a semiótica como lógica para uma teoria pragmática da comunicação; a semiótica como subsídio à análise dos fenômenos-objetos das ciências contemporâneas; e processos de auto-organização, estudados no domínio de sistemas complexos, com foco nos sistemas de naturezas informacional e/ou cognitiva.

Uma proposta de método adaptativo para seleção automática de preenchimento de texto

Francisco de Faria
Laboratório de Tomada de Decisão
Escola Politécnica da Universidade de São Paulo
São Paulo, Brasil

João Pereira
Escola Politécnica da Universidade de São Paulo
São Paulo, Brasil
raijoma@usp.br

***Resumo*—Esta pesquisa busca propor um método probabilístico para preenchimento automático de texto, com potencial aplicação em dispositivos computacionais como telefones celulares e computadores pessoais. O substrato formal utilizado como base na proposição do método é o autômato adaptativo, ao qual é associado o conceito frequentista de probabilidade, de forma a orientar o preenchimento de textos segundo uma dependência local de contexto. Esta dependência de contexto se traduz na probabilidade de reincidência de bigramas. Caso o contexto local não seja reconhecido no texto em curso de digitação, passa-se a uma análise livre de contexto baseada apenas na reincidência de palavras.**

***Palavras-chave*—Preenchimento de texto; autômato adaptativo; probabilidade; bigrama; dependência do contexto.**

I. Introdução

O conceito de adaptabilidade aplicado a autômatos finitos lhes confere a capacidade de se auto-modificar conforme as necessidades encontradas [1]. Essa metodologia adaptativa compreende-se de maneira suplementar à teoria dos autômatos [5].

O autômato adaptativo é uma máquina de estados e pode ser representado por um grafo direcionado, que inicialmente se apresenta como um autômato finito, porém, à medida que lhe são incutidas ações adaptativas durante o fluxo operacional, o grafo original pode ser modificado [3]. Estas ações adaptativas são geradas pela própria movimentação do autômato e resultam da aplicação de suas regras de transição. Se houver uma única regra aplicável à situação corrente, a próxima situação do dispositivo é determinada pela regra em questão, e caso haja mais de uma regra aderente à situação corrente do estado, as diversas situações seguintes são tratadas em paralelo. No caso de dispositivos voltados à sugestão de preenchimento de texto, baseados em análise léxico-morfológica, por exemplo, pode haver mais de uma regra a ser seguida, cada regra levando a uma cadeia diferente [2].

Os mecanismos adaptativos são formados por três tipos de ações adaptativas básicas: consulta (checa um conjunto de regras), exclusão de regras e inclusão de regras. As ações básicas de inclusão e exclusão são baseadas em padrões pré-definidos [4]. A execução de uma transição adaptativa acarreta mudanças na topologia do autômato. Isto faz com que uma nova máquina de estados substitua a anterior, caracterizando um novo passo do autômato no espaço das máquinas de estado [6].

O caminho em um espaço de máquinas de estado é gerado devido à máquina original não estar mais presente. Dessa forma, o autômato começa seu trabalho em uma máquina peculiar e vai continuá-lo em outra máquina, descrevendo uma trajetória em um espaço de trabalho que contempla todas as possibilidades de autômatos [4].

Devido a esta possibilidade de adaptação das cadeias às regras dentro do autômato, inúmeras pesquisas confirmam a viabilidade prática da utilização de autômatos adaptativos para preenchimento de texto não-estruturados [6].

A. Visão do problema

Os editores de texto atuais requerem o desenvolvimento de programas que sejam capazes de determinar uma estrutura de sequência lógica, a fim de completar palavras de maneira simples, sem qualquer análise léxico-morfológica ou pragmática. O histórico de ocorrência de uma palavra em uma frase ou texto pode ser um indicador da probabilidade de sua reutilização. Na linguagem coloquial é comum a reutilização de termos, palavras, bigramas e até trigramas.

Atualmente é muito comum que dispositivos computacionais, sejam celulares ou computadores, se utilizem de algum tipo de software para efetuar o trabalho de complemento de texto ou indicação de sugestões para complementar o texto. Há softwares que sugerem o complemento de frases ou palavras, porém, não seguem uma regra lógica ou estatística, de forma que as sugestões não são ótimas e muitas vezes não aderem ao texto.

A motivação deste trabalho é tornar precisas as recomendações de complemento de texto, não apenas na área tecnológica, mas também dentro de outras áreas que exijam este tipo de comportamento. Com essa finalidade é proposto um método adaptativo para reconhecimento e seleção automática de preenchimento de palavras à medida que os caracteres do texto são inseridos.

O arcabouço sugerido é baseado em um autômato adaptativo que pode ser visto como uma extensão do formalismo do autômato de pilha estruturado usado para reconhecimento de linguagens. O dinamismo ou adaptabilidade neste caso está na capacidade do autômato modificar sua própria topologia a fim de reconhecer palavras, bigramas ou frases reincidentes. Neste contexto, o autômato pode ser

utilizado para sugerir complementos para palavras em curso de digitação em um celular ou computador pessoal.

B. Objetivo

O objetivo deste trabalho é estudar a possibilidade de utilização de autômatos adaptativos para sugerir o preenchimento automático de palavras, segundo um método probabilístico e dependente de contexto baseado no histórico de ocorrências destas em textos anteriores.

Além do objetivo principal, pode-se ressaltar também a utilização da teoria de autômatos, retomando um caminho percorrido anteriormente pelos precursores na área de Inteligência Artificial, utilizando como base para o modelo de solução o autômato adaptativo no lugar do finito, já que propicia um tratamento mais adequado e uniforme às tratativas computacionais.

Um mecanismo baseado no arcabouço é desenvolvido a fim de efetuar testes de sugestão e as probabilidades de ocorrência de cada palavra em qualquer tipo de texto construído na etimologia latim/grega da língua portuguesa. Este mecanismo deve ser simulado em um computador, isto é, deve ser um mecanismo de software.

II. MÉTODO PROPOSTO

O método proposto baseia-se nas características do autômato adaptativo, sendo que a criação de novas máquinas de estado é regida pela ocorrência prévia de uma palavra ou bigrama. A cada estado do autômato corresponde um conjunto de sugestões de complemento. As sugestões de complemento são associadas à frequência estatística com a qual aquele estado culmina no complemento em questão. Esta abordagem estatística tem por objetivo maximizar as chances de aceitação dos complementos sugeridos.

A. Definição do método e evolução

O método proposto tem por finalidade:

- A sugestão de complementos para uma palavra em curso de digitação;
- O aprendizado de novas palavras e bigramas digitados.

O autômato inicial corresponde a um único estado de entrada, ao qual é associado um único estado de saída. À medida que textos são inseridos o autômato se adapta de forma a reconhecer palavras ou bigramas reincidentes. Quando um primeiro caractere é recebido, o autômato não o reconhece como entrada válida. Dessa maneira, uma vez que não há transições de estado associadas a este caractere, passa-se à fase de aprendizado, no qual novos estados são criados via ações adaptativas. Quando a primeira palavra é inserida por completo, o autômato conta com uma sequência de estados e transições capazes de reconhecê-la. O autômato já não se encontra no estado de entrada, mas sim em um novo estado que corresponde ao fim desta primeira palavra. A partir deste estado correspondente ao fim da primeira palavra registrada, o procedimento se repete até que a segunda palavra seja concluída e o autômato se torne capaz de reconhecer este primeiro bigrama inserido. Uma ação adaptativa desencadeada pelo fim desta segunda palavra liga este último estado ao estado inicial. Volta-se então a este estado inicial e repete-se o procedimento para esta segunda palavra também passe a ser reconhecida. Quando uma terceira palavra, diferente das duas primeiras, for inserida o autômato será capaz de reconhecer não apenas três palavras, mas também dois bigramas, o primeiro bigrama formado pela primeira e pela segunda palavra, e o segundo bigrama pela segunda e pela terceira palavra.

A fim de visualizar a dinâmica de funcionamento do método proposto, vejamos como seria processada a seguinte frase quando digitada: Enquanto a pata bota os ovos, os patos observam a pata e os ovos. Vamos supor que o autômato acaba de ser criado, ou seja, ainda não reconhece palavra alguma. Quando as primeiras palavras são inseridas, nenhuma sugestão é apresentada, pois o autômato ainda não as reconhece.

1) *Enquanto a pata bota os o_*
   - P(os | o_) = 1.0

Observa-se que quando se digita pela segunda vez a letra "*o*", esta já é reconhecida como parte de uma palavra conhecida, "os".

2) *Enquanto a pata bota os ovos, o_*
   - P(os | o_) = 0.5
   - P(ovos | o_) = 0.5

Quando se digita pela terceira vez a letra *o*, são sugeridas as palavras "os" e "ovos", ambas associadas a uma frequência de ocorrência igual a 50%, pois cada palavra apareceu apenas uma vez anteriormente.

3) *Enquanto a pata bota os ovos, os p_*
   - P(pata | p_) = 1.0

Já após digitar a letra "p" pela segunda vez, é sugerida apenas a palavra "pata", a única inserida anteriormente começando com "p".

4) *Enquanto a pata bota os ovos, os pa_*
   - P(pata | pa_) = 1.0
5) *Enquanto a pata bota os ovos, os pat_*
   - P(pata | pat_) = 1.0
6) *Enquanto a pata bota os ovos, os patos o_*
   - P(os | o_) = 0.66
   - P(ovos | o_) = 0.33
7) *Enquanto a pata bota os ovos, os patos observam a p_*
   - P(pata | a p_) = 1.0

Note que aqui, pela primeira vez, ocorre reincidência de um bigrama, "a p".

8) *Enquanto a pata bota os ovos, os patos observam a pa_*
   - P(pata | a pa_) = 1.0

*9) Enquanto a pata bota os ovos, os patos observam a pat_*

- P(pata | a pat_) = 1.0

*10) Enquanto a pata bota os ovos, os patos observam a pata e o_*

- P(os | o_) = 0.5
- P(ovos | o_) = 0. 25
- P(observam | o_) = 0. 25

*11) Enquanto a pata bota os ovos, os patos observam a pata e os o_*

- P(ovos | os o_) = 1.0

*12) Enquanto a pata bota os ovos, os patos observam a pata e os ov_*

- P(ovos | os ov_) = 1.0

*13) Enquanto a pata bota os ovos, os patos observam a pata e os ovo_*

- P(ovos | os ovo_) = 1.0

14) Enquanto a pata bota os ovos, os patos observam a pata e os ovos.

III. Conclusão

O método proposto baseia-se nas características do autômato adaptativo, sendo que a criação de novas máquinas de estado é regida pela ocorrência prévia de uma palavra ou bigrama. A cada estado do autômato corresponde um conjunto de sugestões de complemento. As sugestões de complemento são associadas à frequência estatística com a qual aquele estado culmina no complemento em questão. Esta abordagem estatística tem por objetivo maximizar as chances de aceitação dos complementos sugeridos.

Espera-se ter contribuído com esta pesquisa para a formulação de uma metodologia bem fundamentada do preenchimento automático de textos, que possa ser usada em simulações e também como modelo teórico para criação e execução de programas em diversos ambientes. Espera-se, ainda, ter contribuído com a utilização prática do autômato adaptativo, através da proposição de um dispositivo de software nele baseado.

Dentro dos objetivos traçados, a forma utilizada para alcançá-los foi interessante e profícua. Isto se deve ao fato de buscar uma visão integradora entre outras áreas como Probabilidade e, dentro dessa diversidade, tentar encontrar a resposta mais adequada às questões colocadas no início desta pesquisa.

O preenchimento automático de texto pode facilitar o trabalho de indivíduos que não dispõem de facilidade para digitação. O trabalho desenvolveu-se dentro das expectativas provando viabilidade do projeto e real vantagem na utilização de um método de sugestão baseado em probabilidade condicional, quando se utiliza linguagem coloquial.

Agradecimentos

Os autores gostariam de agradecer à Toshiba Corporation Japan pelo suporte ao primeiro autor. Agradecemos ainda a José Neto pela orientação deste projeto.

Referências Biográficas

[1] Barretto e José Neto. Uma arquitetura adaptativa. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, pp. 2-3. 28 e 29 de Janeiro, 2016.

[2] Canovas e Cugnasca. Um mapeamento de modelos adaptativos para dispositivos adaptativos guiados por regras. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, pp. 86-97. 28 e 29 de Janeiro, 2016.

[3] Carhuanina e José Neto. Personalização, customização, adaptabilidade e adaptatividade. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, pp. 52-59. 28 e 29 de Janeiro, 2016.

[4] Contier, Padovani e José Neto. O reconhecedor gramatical Linguístico: avanços em desambiguação sintática e semântica. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, pp. 27-36. 28 e 29 de Janeiro, 2016.

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

[6] Rocha, R. L. A. e Neto, J. J. Uma proposta de método adaptativo para a seleção automática de soluções. Proceedings of ICIE Y2K - International Congress on Informatics Engineering, Buenos Aires, 2000.

Adaptatividade na Estimativa da Capacidade de Baterias Estacionárias

Patrick Cadier D'Aquino e Baroni Santos
Escola Politécnica
Universidade de São Paulo
São Paulo, Brasil
patrickcdbs@gmail.com

***Resumo*— Neste artigo é proposto um método adaptativo para medir a capacidade e o envelhecimento de baterias estacionárias. Neste projeto usou-se redes neurais para estimar a capacidade de uma bateria com os valores da resistência e capacitância interna dessa bateria. Os dados recolhidos são apresentados e analisados. Com esses dados foi possível criar um sistema que estime a capacidade da bateria e que se adapte a novos dados apresentados.**

Palavras chaves—Baterias estacionárias, adaptatividade, SOH.

I. INTRODUÇÃO

Baterias de lítio-íon são essenciais para diversos projetos em eletrônica e o uso delas está aumentando devido aos avanços tecnológicos e ao barateamento dos circuitos eletrônicos. Uma aplicação para baterias de lítio são baterias estacionárias, elas fornecem energia quando a fonte principal não está presente, ou seja, elas funcionam como um buffer para a fonte principal. Esse tipo de aplicação é importante para sistemas de segurança ao se protegerem de quedas de energia. As baterias podem fornecer energia quando há um pico de demanda ou quando não há fornecimento de energia por outra fonte. Dois parâmetros essenciais para a análise do estado da bateria são o SOC (State Of Charge ou estado da carga) e o SOH (State Of Health ou estado de saúde).

A. *SOC: State Of Charge*

Suozzo [1] explica que o SOC descreve o quanto a bateria já foi descarregada em relação a sua capacidade nominal. Como esse parâmetro é escrito em porcentagem, 100% representa uma bateria com a carga igual a sua capacidade nominal e 0% representa uma bateria totalmente descarregada. Um dos métodos para prever o SOC é contagem de Coulomb como é descrito por Juang [2]. Medindo a corrente de saída é possível prever o quanto resta de carga na bateria. Os outros métodos descritos por Juang [2] são a estimativa pela tensão nos terminais ou pela impedância de saída. O problema de estimar o SOC pela tensão nos terminais é que a impedância da descarga ou da carga influencia na medição e muitas vezes não é possível medir a tensão em circuito aberto (Juang [2]).

B. *SOH: State Of Health*

O SOC é ruim para mostrar o estado de vida da bateria, pois uma bateria carregada terá um SOC de 100% independente se ela está envelhecida ou nova. Para isso temos o SOH. O SOH é definido por Chen et al [3], Barlak et al [4] e Le et al [5] como a relação entre a capacidade atual da bateria e a capacidade nominal da bateria. Como pode ser visto na equação, o SOH é um número expresso em porcentagem.

$$SOH = \frac{C_{bat}}{C_{nominal}} \times 100\% \quad (1)$$

Portanto o SOH quantifica o quanto a bateria envelheceu. Diferentes fontes (GPI [6], Groot [7], He et al [8] e Xing et al [9]) explicam que a bateria atinge o fim da vida (EOL – End Of Life) quando o SOH chega a 80%. He et al [8] afirma que uma bateria com SOH menor que 80% tem ciclos de carga e descarga muito instáveis, pois a sua perda de capacidade tem um caráter exponencial.

C. *Outros parâmetros*

Outro conceito usado no estudo de baterias é o SOF (State of Function ou estado de funcionalidade). Segundo Juang [2], o SOF é normalmente uma pergunta de Sim ou Não que indica se a bateria está apta a realizar determinada função de uma aplicação. O exemplo mais comum é se a bateria do carro tem carga suficiente para ligá-lo.

II. REDES NEURAIS

Mukherjee [10] compila as diferentes publicações que usam redes neurais para gerenciar baterias, medindo principalmente o SOC e o SOH de baterias. Mukherjee [10] e O'Gorman et al [11] justificam que o uso de redes neurais para esse propósito é vantajoso porque a análise deixa de ser estática e depende menos dos dados adquiridos em laboratórios. Como os parâmetros variam de modelo para modelo e de bateria para bateria, o uso de redes neurais na análise seria uma maneira de contornar isso. O'Gorman et al [11] explica outras vantagens das redes neurais como a baixa complexidade computacional desse método. Portanto usando redes neurais é possível economizar tempo na análise e na medição dos parâmetros (O'Gorman et al [11]).

Nesse projeto usou-se uma rede neural com duas camadas internas, a primeira com 9 neurônios e a segunda com 5 neurônios. Uma representação da rede pode ser vista a seguir.

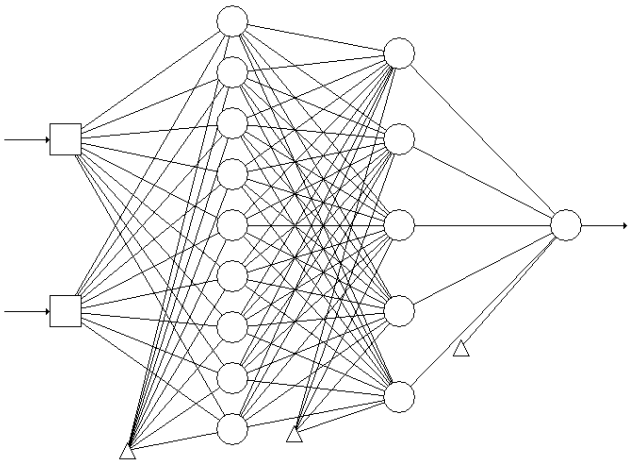


Fig. 1. Representação da rede neural.

III. ADAPTATIVIDADE

O conceito de adaptatividade é um conceito amplo, por isso temos na literatura várias formas de definir esse conceito. Santos [12] define um dispositivo adaptativo como um dispositivo que "pode ser representado por um conjunto de regras expressas em qualquer formato e um conjunto de operações associado a cada uma das regras". Com essa definição, a adaptatividade pode ser usada em diversas áreas basta que o algoritmo seja definido por um conjunto de regras. Assim as operações associadas a essas regras modificam as repostas do algoritmo. Santos [12] utiliza as formulações definidas por Neto [13]: o dispositivo adaptativo é composto por um dispositivo não adaptativo e um mecanismo adaptativo (Neto [13]). Além disso para definir o dispositivo é preciso definir um conjunto de regras não adaptativas, um conjunto de regras adaptativas, um conjunto de ações adaptativas e os mecanismos que relacionam uma ação adaptativa a uma regra adaptativa.

Whittle et al [14] definem sistemas adaptativos como sistemas com a "capacidade de maneira autônoma de modificar o seu comportamento" durante a execução. Segundo Whittle et al [14], um dos principais desafios de sistemas adaptativos é lidar com a incerteza, tanto a incerteza que deriva das entradas quanto a incerteza que deriva dos requisitos, que poderiam mudar de caso a caso.

Porém a formulação escolhida em redes neurais adaptativas é um pouco diferente. Cichocki et al [15] diminuem o tempo de convergência modificando a taxa de aprendizado de acordo com os dados de entradas. Diferente do que é feito em redes neurais não adaptativas onde é utilizada uma taxa de aprendizado fixa ou uma taxa de aprendizado que diminui com uma função pré-definida. Shun-ichi et al [16] fazem algo parecido ao usarem um algoritmo adaptativo para modificar o algoritmo de aprendizagem da rede neural em busca de maior eficiência estatística e menor tempo de convergência. Lin et al [17] apresentam uma solução parecida para um sistema hibrido entre rede neural e lógica fuzzy. O principal objetivo também é diminuir o tempo de convergência do algoritmo de aprendizado, o que é crítico principalmente para a inferência das regras do sistema de lógica fuzzy tradicional. No artigo, Lin et al [17] apresentam duas soluções hibridas e mostram que a adaptatividade do algoritmo de aprendizagem apresenta resultados melhores que o sistema tradicional.

Podemos concluir que a adaptatividade é principalmente usada em redes neurais para facilitar e simplificar o processo de aprendizado e diminuir o tempo de convergência, ou seja um pouco diferente do que é feito em outras áreas da adaptatividade.

IV. MÉTODO

Nesse estudo foi coletado 1050 dados de 21 baterias. Para cada dado foi coletado a resistência e a capacitância interna da bateria medida por um micro controlador. Além disso, mediu-se a capacidade da bateria naquele momento, o que permitiu calcular o SOH. Os gráficos da relação entre capacitância, resistência e SOH são os seguintes.

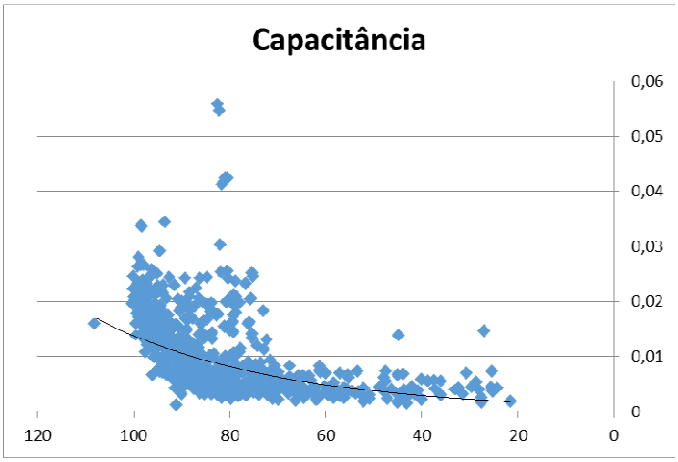


Fig. 2. Relação entre a capacitância e o SOH.

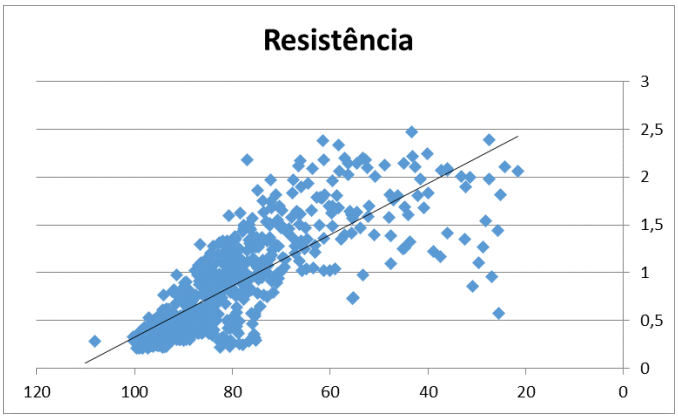


Fig. 3. Relação entre a resistência e o SOH.

Diferentemente do que é feito na literatura, inserimos adaptatividade na rede neural depois da rede neural já ter os seus pesos definidos. Primeiramente, foi usado os dados de 20 baterias (1000 conjuntos de dados) para treinar a rede neural, em seguida os 50 dados da última bateria restante foram usados para testar a rede neural e o algoritmo adaptativo, mudando os pesos da rede neural (ou as regras de execução da rede neural)

para que esta se adapte aos dados da última bateria, estimando com maior precisão a capacidade atual da bateria. Repetiu-se o processo 21 vezes, criando assim 21 redes neurais, uma para cada bateria, e o objetivo era atualizar esses dados de forma adaptativa e verificar que o erro médio fosse menor.

Nesse processo, simulamos a obtenção de dados que o dispositivo conectado à bateria conseguiria. Como a função da rede neural é prever a capacidade em um momento da bateria, e essa capacidade muda de tempo em tempo, o dispositivo terá que realizar diversas previsões. Assim, a vantagem de uma rede neural adaptativa é que as previsões ficam mais precisas para cada dado recebido. É importante notar que essa técnica só é possível em caso onde a capacidade da bateria pode ser medida de outra maneira. Por exemplo, se soubermos a corrente de descarga, é possível calcular a capacidade da bateria com o tempo de descarga. Portanto no estudo os dados possuem uma ordem definida e o objetivo será melhorar a previsão do próximo dado. Uma aplicação que essa técnica pode ser utilizada é na medição da qualidade de baterias de desfibriladores. Quando a bateria está nova, é conhecido o número de vezes que o desfibrilador pode ser usado com uma carga cheia. Assim, um modo de calcular o SOH dessa bateria é o número vezes que o aparelho foi usado. Obtém-se assim a evolução do SOH.

No final, o erro médio da rede neural adaptativa é comparado ao erro médio da rede neural inicial. O esperado era que o erro médio da rede adaptativa diminuísse para cada dado novo e que fosse menor que o erro médio da rede inicial. Como veremos nos resultados, a diminuição não é constante nem garantida. Por isso, criou-se um 2º algoritmo adaptativo que atualizava a rede somente quando a diferença entre o erro médio da rede inicial e o erro médio adaptativo aumentasse. A premissa é que os dados que obtiveram um erro maior na rede adaptativa são diferentes dos outros dados da bateria por razões aleatórias (temperatura ambiente, corrente de carga ou descarga, etc). Assim para obter um erro médio adaptativo menor, esse dado é ignorado na atualização da rede.

Além disso, o dispositivo pode escolher qual das duas redes (a adaptativa ou a inicial) ele utilizará para prever o próximo dado considerando qual tem um erro médio menor para os dados já adquiridos. Sendo assim, criou-se um 3º algoritmo que escolhe a previsão da rede com menor erro médio atual.

A premissa é que os dados mudam um pouco de bateria para bateria e que os dados de uma dada bateria são mais próximos entre si. Os algoritmos adaptativos buscam se aproveitar dessa particularidade para obter resultados mais precisos.

Esses algoritmos são simples de serem implementados em micro controladores, os mesmos necessários para fazer as medições da resistência e capacitância interna das baterias.

V. Resultados

A. *Primeiro algoritmo adaptativo*

Em um primeiro passo, criou-se um algoritmo adaptativo que simplesmente atualizava os seus pesos para cada dado obtido. Na seguinte tabela temos os resultados dos erros médios de cada rede para os 50 dados de cada bateria.

TABLE I. Erro médio no primeiro algoritmo

Bateria	**Erro médio no primeiro algoritmo**		
	Erro médio para os 50 dados	***Erro médio adaptativo para os 50 dados***	***Diminuição em %***
1	6,224	5,62	9,704
2	1,837	1,855	-1,005
3	6,392	4,359	31,806
4	6,535	6,026	7,795
5	10,603	9,886	6,768
6	8,497	5,655	33,45
7	8,708	9,055	-3,826
8	5,991	5,672	5,328
9	8,808	5,347	39,293
10	6,734	6,472	3,886
11	3,163	3,277	-3,454
12	5,495	4,634	15,665
13	5,726	6,517	-12,137
14	7,32	6,324	13,607
15	8,901	5,275	40,737
16	8,281	6,002	27,526
17	10,599	12,751	-16,876
18	15,268	9,769	36,015
19	10,209	10,13	0,78
20	8,059	7,808	3,106
21	8,906	10,122	-12,011
Média	7,727	6,788	12,143

Como podemos ver na tabela, algumas redes adaptativas possuem resultados piores que a rede inicial. A rede da bateria 17 por exemplo tem um aumento do erro médio da ordem de 16,9%. Entretanto a média de diminuição do erro é alta (12,14%) e mostra que com poucos dados já é possível especializar a rede neural para a determinada bateria.

Outros resultados importantes são as evoluções do erro médio da rede neural adaptativa em comparação com o erro médio da rede inicial. Essa análise permitiu modificar o algoritmo adaptativo para otimizá-lo.

Era esperado que a diminuição do erro médio seguisse uma curva parecida à curva da bateria 3, que pode ser vista na seguinte figura.

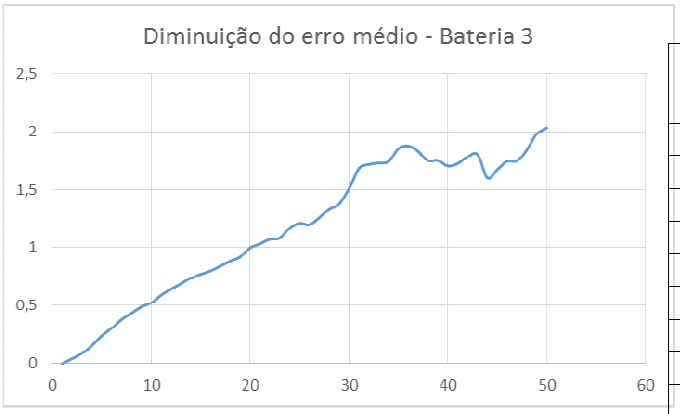


Fig. 4. Diminuição do erro médio da bateria 3 para cada dado novo no 1º algoritmo.

Como podemos ver na figura, a curva é em sua maior parte crescente e monótona, indicando que o erro médio diminuiu constantemente. Porém não é o que vimos na maioria dos casos. Um caso que pode ilustrar isso é o caso da bateria 5, como é visto no seguinte gráfico.

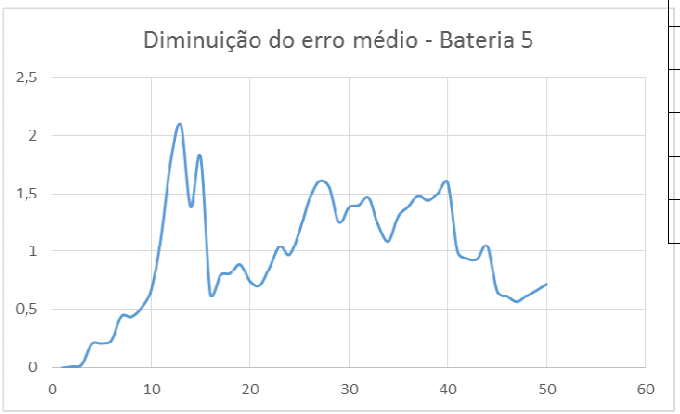


Fig. 5. Diminuição do erro médio da bateria 5 para cada dado novo no 1º algoritmo.

Apesar do erro médio adaptativo sempre ser menor que o erro médio inicial, já que a curva sempre é positiva, a curva tem diversos pontos de inflexão. Ou seja, alguns dados aumentam o erro médio adaptativo e seriam tratados melhores pela rede inicial. Assim, as últimas previsões não são as mais precisas do conjunto de dado. Claramente há espaço para otimizar esse algoritmo. Por isso criou-se um novo algoritmo descrito anteriormente para aproveitar somente os dados que possuem melhor relação entre si.

B. Segundo algoritmo adaptativo

Criou-se um novo algoritmo que não atualiza a rede neural adaptativa quando o novo dado diminuiu a diferença entre o erro médio da rede inicial com o erro médio da rede adaptativa, ou seja, toda vez que há uma queda na curva da diferença dos erros. O objetivo é que as curvas tenham uma forma mais parecida com a curva da bateria 3.

Na tabela a seguir podemos ver que houve uma melhora no algoritmo.

TABLE II. ERRO MÉDIO NO SEGUNDO ALGORITMO

Bateria	**Erro médio no primeiro algoritmo**		
	Erro médio para os 50 dados	***Erro médio adaptativo para os 50 dados***	***Diminuição em %***
1	6,786	6,383	5,951
2	2,061	2,518	-18,152
3	6,222	4,318	30,6
4	6,455	5,925	8,213
5	10,646	8,249	22,519
6	8,875	5,665	36,166
7	8,267	8,816	-6,227
8	7,067	6,104	13,62
9	9,047	5,749	36,447
10	7,556	8,861	-14,721
11	2,676	2,797	-4,339
12	5,614	4,351	22,505
13	6,705	7,077	-5,252
14	7,543	6,909	8,4
15	8,886	4,429	50,16
16	7,441	5,492	26,192
17	10,243	11,565	-11,424
18	15,627	10,573	32,344
19	10,269	10,551	-2,668
20	8,263	8,295	-0,377
21	9,882	10,652	-7,23
Média	7,911	6,918	12,553

Em algumas baterias o 2º algoritmo obteve um resultado pior, como é o caso da bateria 3, porém na média o resultado do 2º algoritmo é melhor.

Uma das baterias que obtiveram uma melhora considerável foi a bateria 5, como podemos ver na figura.

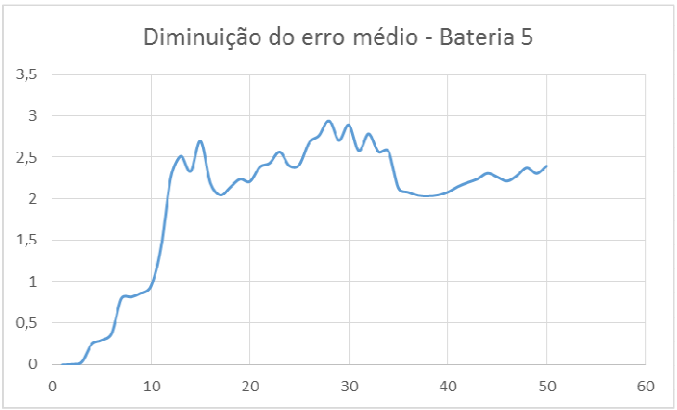


Fig. 6. Diminuição do erro médio da bateria 5 para cada dado novo no 2º algoritmo.

Podemos ver que nesse exemplo o objetivo da premissa foi atingido já que as quedas que ocorrem na curva do 1º algoritmo ocorrem com menor intensidade no 2º algoritmo, diminuindo os efeitos negativos de um dado que diverge do restante do conjunto de dados.

C. *Terceiro algoritmo adaptativo*

Nesse 2º algoritmo, é necessário calcular os resultados das duas redes neurais e algumas vezes a rede neural inicial obtém resultados melhores. Por isso, criou-se um 3º algoritmo que escolhe o resultado da rede neural (rede neural inicial ou rede neural adaptativa) que tem o menor erro médio no conjunto de dados já adquiridos daquela bateria. Essa diferença é simples de projetar e evita que a previsão seja piorada.

Os resultados desse 3º algoritmo estão na seguinte tabela.

TABLE III. ERRO MÉDIO NO TERCEIRO ALGORITMO

Bateria	**Erro médio no primeiro algoritmo**		
	Erro médio para os 50 dados	***Erro médio adaptativo para os 50 dados***	***Diminuição em %***
1	6,786	6,383	5,951
2	2,061	2,119	-2,725
3	6,222	4,318	30,6
4	6,455	5,998	7,088
5	10,646	8,249	22,519
6	8,875	5,665	36,166
7	8,267	8,269	-0,022
8	7,067	6,104	13,62
9	9,047	5,749	36,447
10	7,556	7,761	-2,635
11	2,676	2,808	-4,708
12	5,614	4,39	21,807
13	6,705	6,845	-2,04
14	7,543	6,909	8,4
15	8,886	4,429	50,16
16	7,441	5,492	26,192
17	10,243	10,795	-5,107
18	15,627	10,934	30,035
19	10,269	10,643	-3,514
20	8,263	8,336	-0,868
21	9,882	10,013	-1,313
Média	7,911	6,772	14,401

Como vemos, a vantagem do 3º algoritmo em relação ao 2º se dá nos conjuntos de dados onde o 2º algoritmo é pior que a rede inicial. Para esses casos o erro diminuiu consideravelmente. Porém para os casos onde o 2º algoritmo é melhor que o 3º (bateria 4 por exemplo), a diferença entre os dois não é considerável.

Podemos ver a vantagem do 3º algoritmo no gráfico da bateria 10. Enquanto o 2º algoritmo é melhor que a rede inicial, o 3º algoritmo não oferece vantagem, porém a partir da 34ª medida, o 3º algoritmo evita que erro aumente.

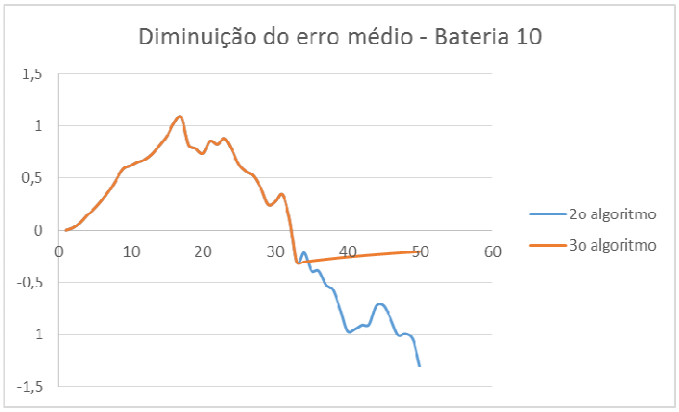


Fig. 7. Comparação da diminuição do erro médio para o 2º e 3º algoritmo com a bateria 10.

Em suma, três algoritmos adaptativos foram criados para reduzir o erro médio da previsão do próximo dado a ser coletado. A diminuição do erro médio obtida pelo 3º algoritmo (o mais completo) para os 21 conjuntos de dados foi de 14,4%. Esses algoritmos podem ser facilmente inseridos no código do micro controlador que controla a bateria.

VI. CONCLUSÃO

É possível concluir que o objetivo foi atingido já que um algoritmo adaptativo foi projetado e apresentou resultados satisfatórios para o conjunto de testes. Além disso, o algoritmo não oferece grande esforço computacional, garantindo que possa ser usado em diversas aplicações que usem baterias estacionárias.

O algoritmo no final permite minimizar erros decorrentes de peculiaridades das baterias usadas e melhorar a previsão da capacidade da bateria.

REFERÊNCIAS

[1] C. Suozzo, Lead-acid battery aging and state of health diagnosis, 2008

[2] L.W. JUANG, Online battery monitoring for state-of-charge and power capability prediction, 2010.

[3] Z. Chen, C.C. mi, Y. Fu, J. Xu, X. Gong, “Online battery state of health estimation based on Genetic Algorithm for electric and hybrid vehicle applications.” Journal of Power Sources, 240: 184-192, 2013.

[4] C. Barlak, Y. Özkazanç, “Determination of battery state-of-health via statistical classification.” Commun. Fac. Sci. Univ. Ank. Series A2-A3 vol. 52(2) pp 1-9 2010.

[5] D. Le, X. Tang, “Lithium-ion Battery State of Health Estimation Using Ah-V Characterization.” Annual Conference of the Prognostics and Health Management Society, 2011

[6] "Lithium Ion technical handbook" Gold Peak Industries Ltd. November 2003.

[7] J. Groot, State-of-Health Estimation of Li-ion Batteries: Cycle Life Test Methods

[8] W. He, N. Williard, M. Osterman, M. Pecht, “Prognostics of lithium-ion batteries based on Dempster–Shafer theory and the Bayesian Monte Carlo method.” Journal of Power Sources 196, 10314– 10321, 2011

[9] Y. Xing, E.W.M Ma, K.L. Tsui, M. Pecht, “Battery management systems in electric and hybrid vehicles.” Energies, 4, 1840-1857 ,2011

[10] A. Mukherjee, Advances in battery management using neural networks and fuzzy logic. 2003.

[11] C.C. O'gorman, D. Ingersoll, R.G Jungst, T.L. paez, "Artificial Neural Network Simulation of Battery Performance." IEEE 98, 2008.

[12] J.M.N. Santos, "Dispositivos Adaptativos Cooperantes", X Workshop de Tecnologia Adaptativa", 2016.

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

[14] J. Whittle; P. Sawyer; N. Bencomo; B.H.C. Chengy; J.M. Bruel, "RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems", Requirements Engineering Conference, 2009.

[15] A. Cichocki, S. Amari, M. Adachi, W. Kasprazak, "Self-Adaptive Neural Networks For Blind Separation Of Sources",IEEE International Symsposium On Circuits And Systems, 1996.

[16] S. Amari, A. Cichocki, "Adaptive Blind Signal Processing – Neural Network Approaches", Proceedings Of The IEEE, vol. 86, no. 10, 1998

[17] C. Lin, C.S. Lee, "Neural Network Based Fuzzy Logic Control And Decision System", IEEE Transactions On Computers, vol 40, no 12, 1991

Emprego de adaptatividade em mineração de dados jurídicos - Uma primeira análise

Leme, Pedro C.
Departamento de Engenharia de Sistemas Eletrônicos
Escola Politécnica da USP
São Paulo, Brasil
pedro.leme@yahoo.com

***Resumo*—A extração de informações de dados judiciais públicos no Brasil, surgida principalmente a partir de 2006, tem se tornado uma área de crescente visibilidade. O uso de robôs de coleta desses dados a partir de diferentes sistemas e sites requer codificação específica e orquestração de funcionamento baseado em diversos fatores, travas e características de cada site. Devido a essas características o controle de muitos robôs e máquinas de maneira quase autônoma se faz necessário, não sendo mais possível um ajuste manual para cada tipo de sistema-alvo de maneira satisfatória. Pretende-se analisar o cenário e futuramente aplicar os conceitos de Adaptatividade no desenvolvimento de um mecanismo adaptativo que irá atuar sobre o subsistema de controle de instâncias de modo que o funcionamento deste passe a requerer menor envolvimento humano e apresente melhora em qualidade.**

Palavras-chave—adaptatividade; mineração de dados; aplicações

I. Introdução

Com o alto nível de judicialidade[1] e a determinação do Conselho Nacional de Justiça (CNJ)[2] de digitalizar os processos judiciais no país todo a partir de 2006, surgiu uma área de atuação, já explorada em países como os EUA, especializada na extração, compilação e geração de valor de dados legais e judiciais públicos no Brasil.

A empresa Digesto[3] surgiu em 2012 a partir dessa ideia, e hoje extrai dados diariamente de 500 fontes de diários oficiais e mais de 50 tribunais em todo o território brasileiro de maneira digital e automática, tendo 100 milhões de processos e mais de dois bilhões de andamentos processuais, dentre outras bases. A extração dos dados se dá a partir de sites e portais de tribunais e órgãos públicos brasileiros, cujos dados são abertos por lei.

Robôs e sistemas desenvolvidos especialmente para cada site ou sistema realizam as requisições via método HTTP/S ou *webservices*, depois fazem o *parsing* do retorno e análise dos dados retornados, quer seja HTML, JSON, XML ou texto puro, para a extração de informações que são então tratadas e enviadas para um banco de dados relacional. Essa informação depois é usada para geração de relatórios e inferência de classificações diversas via aprendizado por máquina a fim de obter e inserir inteligência sobre esses dados.

II. Objetivo

Cada sistema usado pelos diferentes órgãos públicos têm características diferentes, seja em termos de caminho de acesso, estrutura interna, apresentação dos dados, horários de funcionamento, travas e controles de número de acesso, necessidade ou não de *login* além de instabilidade do serviço, e isso se reflete nas características dos robôs (*workers*), sendo necessárias diretivas específicas para cada um, e tais parâmetros são atualmente codificados num sistema de execução e ajustados manualmente a partir de observações e controles manuais, diários ou semanais.

Esse sistema, responsável por orquestrar a extração dos dados, tem a capacidade de iniciar e paralisar as máquinas para cada robô, decidir o horário em que devem trabalhar, além de configurar parâmetros como número de processos e *threads* simultâneas e estratégia de coleta de dados e controle de filas de pedidos de informação com base nas variáveis inseridas nele por um operador.

Com o crescimento dos serviços, robôs e volume de dados obtidos diariamente, o controle manual desses parâmetros tem se mostrado cada vez mais complexo e demorado, além de apresentar resultados muitas vezes aquém do esperado, resultando em falta de dados por tempo inadequado de execução de determinado robô, gasto excessivo com máquinas ligadas esperando por tarefas ou ainda problemas mais urgentes, como bloqueio de IP's e *logins* em sites devido a grandes volumes de requisições em um curto espaço de tempo.

O objetivo do estudo é fazer um levantamento do cenário atual e analisar as possibilidades de uso da Adaptatividade no sistema de gerenciamento dos robôs de extração, de modo que as características dos sistemas e análise do funcionamento destes em tempo real possibilitem a troca automática das estratégias usadas no acesso a cada um dos sites e portais, melhorando a qualidade dos dados extraídos ao mesmo tempo em que aumenta sua quantidade e reduz custos com infraestrutura.

III. Infraestrutura e fluxo atual de dados

O fluxo do sistema inicia-se com robôs que diariamente coletam e extraem informações de diários oficiais do país todo. Esses robôs são ligados automaticamente por agendamento e através de requisições HTTP e *parsing* de HTML baixam arquivos, em sua maioria pdf, dos diários oficiais de tribunais

de todo o Brasil. Nos diários constam pequenos trechos de andamentos e solicitações de processos, assim como uma relação de todos os processos novos criados no dia anterior.

Com tais dados em mãos um sistema extrai dos textos números num formato específico, compatíveis com o padrão de numeração única instituído pelo CNJ, e os envia como mensagens para um sistema de filas específicas para cada tribunal, o mesmo de onde cada número foi extraído do respectivo diário.

A partir do enfileiramento de mensagens com os números num sistema de mensageria, outros robôs, especializados em extração de dados a partir de páginas web, começam a consumir esses números e a fazer as requisições nos tribunais de justiça, a fim de obter informações mais detalhadas acerca dos processos e seus andamentos. Tal extração faz uso de requisições HTTP/S e de *webservices*, além de *parsing* de HTML e geração e navegação em árvores estruturadas a partir deste. Encontrando e absorvendo essas informações dos dados devolvidos pelo tribunal, os robôs realizam uma limpeza e normalização para depois inseri-los em um banco de dados relacional, capaz de armazená-los e garantir suas propriedades e conexões.

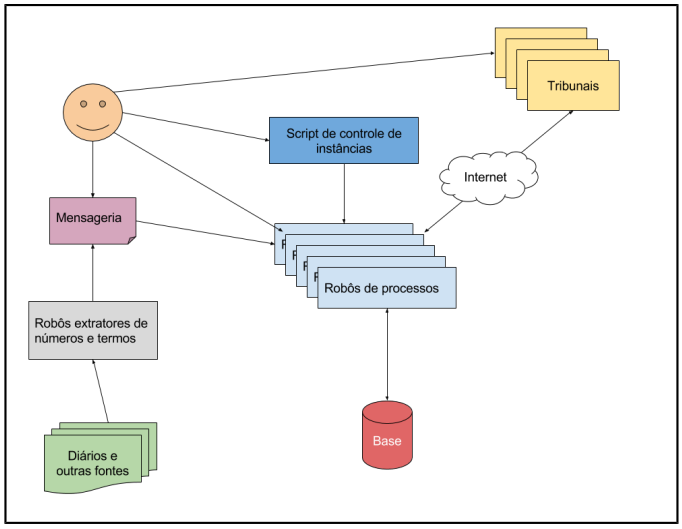


Fig. 1. Representação simplificada do sistema atual

Da representação simplificada do sistema atual, na Figura 1, nota-se que os robôs têm seu funcionamento orquestrado por um *script* de controle de instâncias que armazena parâmetros de configuração, tempo de execução e quantidade de instâncias de cada *worker*, para então ligá-las e desligá-las automaticamente seguindo essas variáveis. Atualmente os parâmetros são configurados manualmente por um operador, que monitora o funcionamento dos robôs, a quantidade de mensagens nas filas e o andamento das requisições nos tribunais a fim de decidir a quantidade de instâncias de cada robô e seus horários de execução para ajuste individual no curto prazo.

IV. Dificuldades do cenário

Com um número pequeno de sites de tribunais e robôs o controle manual das variáveis é relativamente rápido e efetivo, pois é possível acompanhar em tempo real o funcionamento e ajustar os parâmetros praticamente *on-the-fly* no *script* de controle. No entanto, com o crescimento da quantidade e cobertura dos *workers*, tal procedimento tem se tornado custoso e sujeito a falhas, quer seja pela alta frequência de modificações nos robôs ou pela grande variação no funcionamento de cada um deles.

Um ponto a se considerar no desenvolvimento de robôs específicos para uma grande gama de sites e serviços diferentes é a quantidade de características existentes, assim como o número expressivo de possibilidades de coexistência de uma ou mais destas características num mesmo robô. Tem-se como exemplos a existência de *captcha*, necessidade de *login*, horários específicos de acesso, uso de diferentes rotas dependendo do dado que se busca, controle de número de requisições concorrentes ou diárias, bloqueio de acesso a determinadas informações e principalmente instabilidade e indisponibilidade de diversos sites e serviços públicos.

Nos robôs em funcionamento atualmente boa parte dessas características já é levada em conta, quer seja pelos mecanismos de acesso com *login* e detecção de falhas e indisponibilidade, mas pelos sites-alvo serem sistemas sobre os quais não há qualquer conhecimento interno do funcionamento ou decisão dos mantenedores, os robôs e o sistema de controle não é preparado para tomar qualquer medida minimamente inteligente de maneira automática caso alguma dessas características mude.

Há de se juntar a esse problema de modificação sem aviso prévio o fato de haver flutuação nas quantidades de mensagens de pedidos nas filas, quer seja por algum fator externo, como um dia sem publicação de diário oficial ou publicação dupla ou um novo pedido com números já conhecidos que precisem de atualização, ou mesmo interno, como um ajuste de horário e quantidade de máquinas errado que aos poucos deixe acumular uma grande quantidade de mensagens a serem consumidas até que fuja do controle.

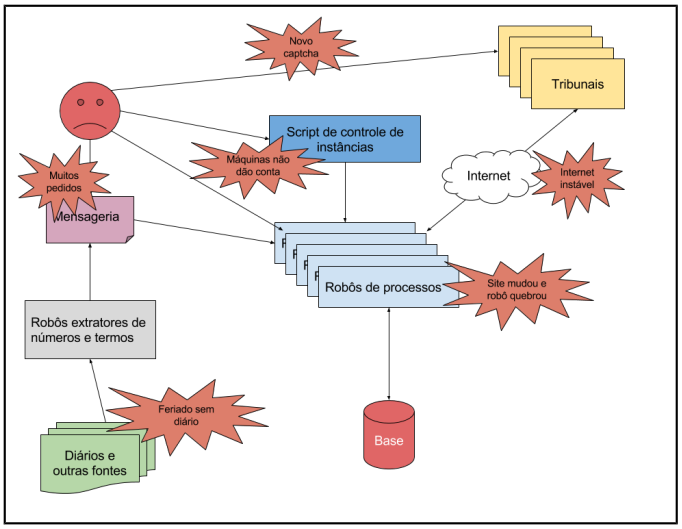


Fig. 2. Ilustração de problemas frequentes no funcionamento dos sistemas

Com a adição de todos os problemas possíveis e conhecidos que ocorrem numa escala semanal, senão diária, multiplicados

pela quantidade de sistemas-alvo na cobertura de extração, o trabalho manual de ajuste do *script* de controle de instâncias já não apresenta eficiência aceitável, e um novo mecanismo para esse subsistema na infraestrutura é necessário.

V. Adaptatividade

O conceito de Adaptatividade, conforme especificado por Neto, J. J.[4], descreve um sistema no qual um dispositivo adjacente é capaz de ter seu conjunto de regras modificado por um mecanismo adaptativo. Tal mecanismo é capaz de observar o dispositivo adjacente e seus estados, e realizar modificações na estrutura de regras deste conforme estados ou entradas condizentes com as ações adaptativas existentes se manifestem.

São três as ações adaptativas possíveis, de consulta, inclusão e exclusão, e são capazes de observar e realizar mudanças na estrutura de regras do dispositivo adjacente, de modo que este passe a trabalhar de outra maneira depois da execução de uma ação adaptativa. Vale notar que as regras das ações adaptativas devem estar codificadas no mecanismo adaptativo, para que este saiba quando usá-las e para que efeito.

Há aplicação e formalismo de Adaptatividade para diversos dispositivos, tais como o Autômato Adaptativo, tabelas de estado e de decisão, gramáticas, cadeia de Markov e árvore de decisão, o que abre possibilidades acerca da melhor estratégia a ser adotada.

VI. Possibilidades de aplicação da adaptatividade

As aplicações da adaptatividade em data mining são muitas e a área é uma das que mais se beneficia da possibilidade de modificação adaptativa *on-the-fly* durante seu funcionamento. Passando pela obtenção de dados a partir de requisições web, *parsing* das árvores de dados, busca e extração da informação desejada, até o controle de acesso às fontes, ser capaz de modificar as regras pelas quais um dispositivo funciona conforme seu estado e entradas é muito valioso para quem trabalha com extração de informação, quer seja estruturada, semi ou não-estruturada.

Devido à aplicação e natureza dos dados judiciais extraídos pelos robôs atualmente, todas as informações coletadas precisam ser o mais específicas e corretas possível, o que inviabiliza num primeiro momento robôs adaptativos capazes de modificar-se para continuar extraindo dados mesmo com mudanças no sistema-alvo, já que assim passa-se a validação dos dados de dentro dos robôs para dentro da base de dados onde os processos são armazenados, com maior risco de envenenamento e empobrecimento da qualidade dos dados já obtidos. Ainda assim, espera-se que no futuro e com maior tempo para pesquisa na área possa-se aplicar algum desenvolvimento nesse sentido.

Já outra área tanto ou mais crítica no funcionamento da infraestrutura, e com menor risco em relação à diminuição da qualidade dos dados extraídos, é o controle de funcionamento dos diversos robôs que rodam durante 24 horas todo dia. As dificuldades no controle manual do funcionamento dos robôs, conforme descrito anteriormente, tem tomado uma proporção muito grande, e um sistema adaptativo, capaz de ter suas regras de funcionamento modificadas a partir das características de cada robô e sistema-alvo, podem ter um grande impacto positivo em termos de cobertura de extração, diminuição de custos e de tempo empregado na manutenção do funcionamento deste.

Atualmente cada robô tem codificada uma gama de informações a respeito de seu sistema-alvo, como tratamento de indisponibilidade, uso de *login*, bloqueio por *captcha* e mudanças de estratégia para acesso à informação, enquanto o *script* de controle dos robôs tem outros dados complementares já obtidos e tratados ao longo do tempo com respeito principalmente a horários de acesso e volume de acesso paralelo permitido. No entanto, não há um sistema que tome conhecimento dessas informações todas e as transforme em tomada de decisão automática. A geração de uma árvore ou tabela de decisão universal que leve em conta todos esses dados de maneira única é complexa e foge ao controle depois de algumas iterações, devido principalmente às enormes possibilidades de interação entre as características presentes em cada sistema-alvo.

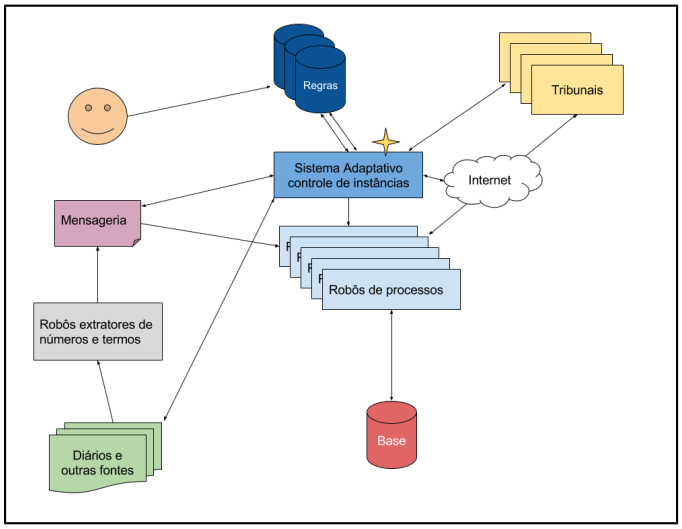


Fig. 3. Proposta de Sistema Adaptativo

A ideia de emprego da adaptatividade nesse ponto é a criação de um mecanismo adaptativo que atue sobre o subsistema de controle de instâncias e robôs e consiga analisar todos os dados sobre o alvo, presentes tanto nos robôs quanto no *script*, podendo então realizar ações adaptativas para modificar uma tabela de decisão padrão gerando um controle específico para cada robô. Além disso, com base no andamento das filas de processos, consiga ajustar de maneira iterativa essa tabela para um ajuste fino entre o solicitado e o atendido pelos robôs diariamente.

Outro ponto que se espera alcançar no desenvolvimento do mecanismo adaptativo é a capacidade deste de emitir alertas e avisos quando as regras e ações adaptativas existentes não forem capazes de garantir o funcionamento dos robôs e o consumo das filas em níveis aceitáveis, de modo que assim um operador possa realizar, ainda que manualmente, um segundo nível de adaptatividade, ajustando as ações adaptativas e a tabela de decisão para uma melhor conformidade com uma situação não antecipada.

VII. Futuros desenvolvimentos

A partir da decisão do caminho a ser seguido, os próximos passos agora serão a geração da tabela de decisão não-adaptativa do dispositivo adjacente, seguida da compilação em tabela das características de cada sistema-alvo para então codificação das ações adaptativas possíveis e os efeitos pretendidos na tabela de decisão.

Para o desenvolvimento do mecanismo adaptativo usar-se-á das linguagens de programação Python ou Clojure. A escolha tem dois pontos: enquanto Python é uma linguagem fácil de se trabalhar, flexível e atualmente usada em todos os nossos sistemas em produção, Clojure é um dialeto LISP com vários novos conceitos, capaz de rodar em cima da JVM do Java (abrindo portas para sistemas enterprise existentes), além de toda a possibilidade que ser um LISP traz, como macros, sintaxe simplificada e poderosa (código é dado) e alto nível.

Como base de regras para as ações adaptativas espera-se fazer uso de JSON ou uma base SQLite, facilitando a edição das ações mesmo por um desenvolvedor que não tenha tido contato com os conceitos mais avançados, como a teoria formal de adaptatividade, autômatos adaptativos, dentre outros.

Com o sistema pronto, a etapa seguinte é o funcionamento em malha aberta (sem ação real) paralelamente ao sistema não-adaptativo existente, como forma de validação e benchmark, para depois entrar em produção.

VIII. Resultados esperados

Busca-se finalizar o desenvolvimento inicial do mecanismo adaptativo em fevereiro de 2017 para início dos testes, com entrada em produção em março do mesmo ano garantido o funcionamento satisfatório.

Como funcionamento satisfatório entende-se a redução, de forma automática, de pelo menos 60% no tamanho das filas acumuladas no sistema de mensageria semanalmente, com atendimento correto de pelo menos 95% das mensagens delas.

Espera-se também, ainda que não seja possível uma quantificação adiantada, a diminuição de alertas e avisos falso-positivos, uma melhora na qualidade desses avisos e uma menor necessidade de intervenção humana para o consumo completo das filas de tarefas sem aumento expressivo no custo mensal com infraestrutura.

Referências

[1] Site O Globo, “Conflagrado nos tribunais, Brasil tem um processo em andamento para cada dois habitantes”, http://oglobo.globo.com/brasil/conflagrado-nos-tribunais-brasil-tem-um-processo-em-andamento-para-cada-dois-habitantes-16822691, visitado em 15/12/2016.

[2] LEI Nº 11.419, DE 19 DE DEZEMBRO DE 2006 - Lei de Informatização do Processo Judicial. Cópia obtida em http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2006/lei/l11419.htm, visitado em 15/12/2016.

[3] Site da empresa Digesto, http://digesto.com.br/servicos, visitado em 15/12/2016.

[4] Neto, João José. Autômatos em Engenharia de Computação - uma Visão Unificada. Primera Semana de Ciencia y Tecnología de la Sociedad Chotana de Ciencias y la Red Mundial de Científicos Peruanos Ciudad de Chota, Perú, Junio 22-27, 2003.

Adaptatividade em Process Mining

S. G. Dada e J. José Neto

***Resumo*— O mapeamento de processos de negócio é fundamental para a eficiência de um serviço prestado. No mapeamento dos processos é possível descobrir quais tarefas são dependentes umas das outras e quais são gargalo da operação. Atualmente, o projeto do processo não é o que ocorre na realidade. Assim, criou-se um conceito chamado de Process Mining. Com o Process Mining, a partir do processo já funcionando, é mapeado o fluxo que realmente está acontecendo no serviço. Essa técnica utiliza algoritmos que transformam logs em redes de Petri. Um dos algoritmos mais utilizados é o algoritmo Alpha. A proposta deste estudo é sanar, com a implementação da adaptatividade, duas desvantagens apresentada pelo algoritmo alpha: (1) tratamento de ruídos, (2) criação da rede de Petri em tempo real.**

***Keywords*— process mining, adaptatividade, redes de petri, algoritmo alpha, log de eventos.**

INTRODUÇÃO

Todos querem produzir mais em menos tempo. Uma maneira de realizar isso é através da boa definição de um processo de negócio que explicite as tarefas dependentes entre si e as tarefas que poderiam ser executadas em paralelo. Em uma situação ideal, o fluxo de tarefas é desenhado antes de se iniciar as atividades e é seguido. Na prática, gasta-se muito tempo desenhando o processo e muitas vezes ele não é seguido [1].

Uma maneira de simplificar o mapeamento de processos é utilizando uma técnica chamada Process Mining [1]. Process Mining são técnicas que permitem extrair informações através dos logs de eventos. Com essa metodologia, é possível checar, reparar, construir e fazer predições de processos [2].

Um dos algoritmos básicos utilizados para transformar logs em processo é o algoritmo alpha. A saída deste algoritmo é uma rede de Petri. Uma das desvantagens do algoritmo alpha é que ele não trata ruído [3]. Outra desvantagem é que o algoritmo alpha deve ler toda a base de logs para depois gerar a rede de Petri, isso faz perder o dinamismo da descoberta de algumas tarefas que são gargalos do processo.

Com o objetivo de sanar essas duas desvantagens do algoritmo alpha, este artigo propõe a implementação de adaptatividade no algoritmo.

A área de estudo de adaptatividade permite que a rede de Petri se modifiquem dinamicamente, sem influências externas [4] [5]. Com essa característica, a adaptatividade irá permitir que o algoritmo alpha seja construído dinamicamente, conforme logs vão sendo criados, e irá contar a frequência que cada fluxo ocorre para retirar ruídos.

PROCESS MINING

Process Mining cria fluxos, redes de Petri, através da leitura de log de eventos. A figura 1 exemplifica essa área de estudo:

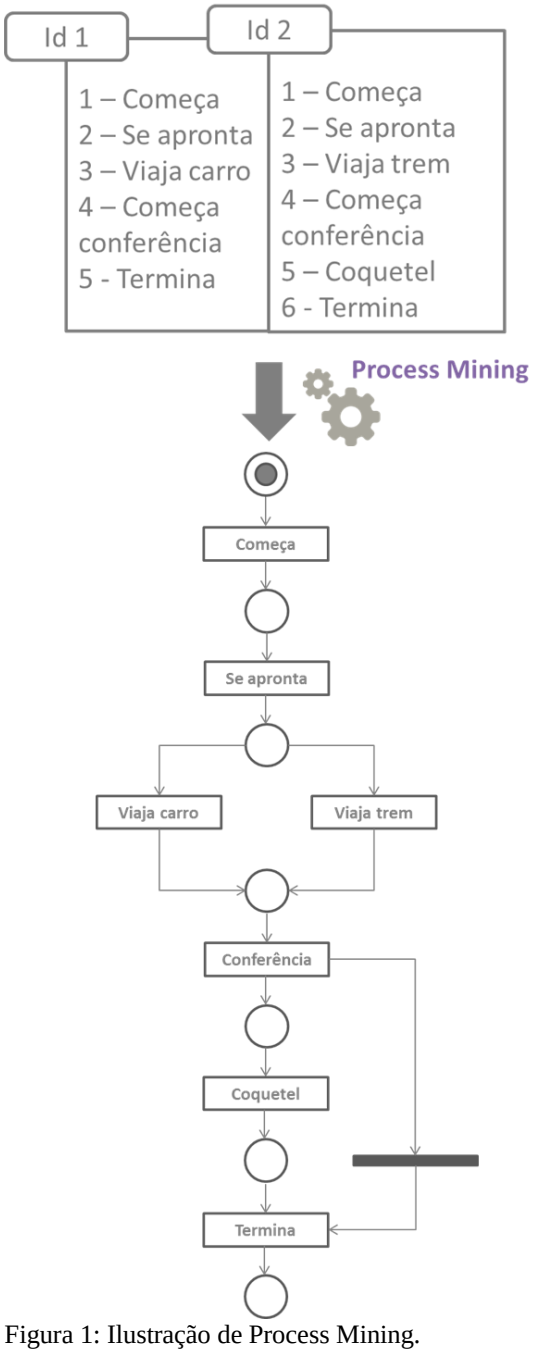


Figura 1: Ilustração de Process Mining.

ALGORITMO ALPHA

O algoritmo alpha é um dos algoritmos mais usados no Process Mining para criar a rede de Petri [3].

O algoritmo alpha é descrito a seguir. Passo 1: todo o log do processo é lido. Passo 2: todas as possíveis sequências de eventos são mapeadas. Passo 3: analisa-se a relação entre os eventos. São elas: (a) sucessão direta x>y – se após o acontecimento do evento x, acontece o y; (b) casualidade x→y – se sempre o evento x acontece antes de o y e x>y; (c) paralelo x || y – se o evento x acontece antes do y e o y acontece antes do x (x>y e y>x); (d) escolha x#y – se o após o evento x não ocorre o y e após o y não ocorre o x (não x>y e não y>x). Passo 4: cria-se a rede de Petri através de essas relações, para cada relação há uma configuração diferente da rede de Petri. A figura 2 [6] apresenta qual a configuração da rede de Petri a partir da relação existente entre os eventos.

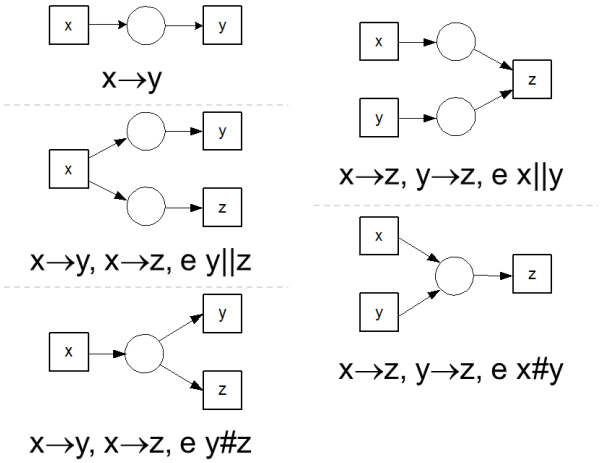


Figura 2: relações e suas respectivas redes de Petri. Apresentada por AALST e HEE [6].

A figura 3 apresenta um exemplo do algoritmo Alpha. A figura 4 mostra a codificação do algoritmo. Elas foram baseadas na explicação de AALST e HEE [6].

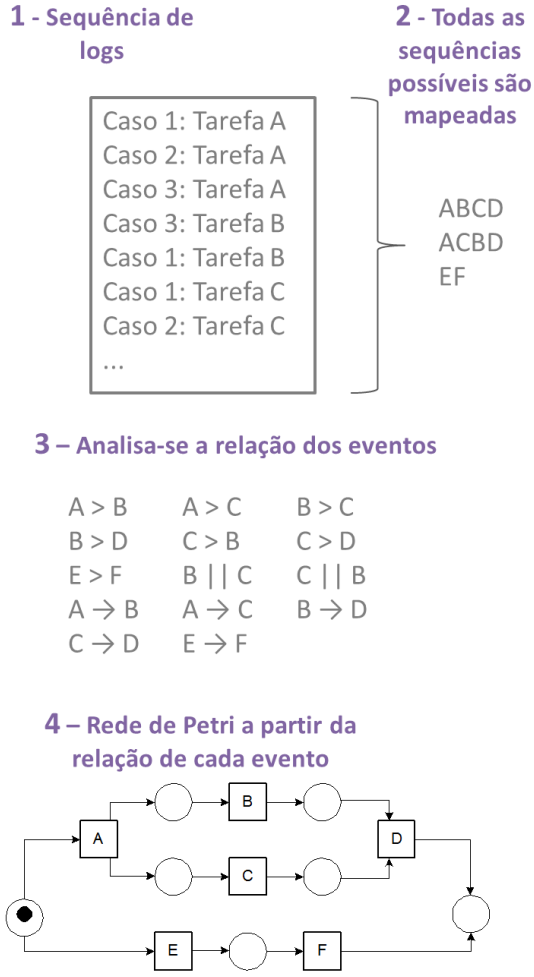


Figura 3: Exemplo do algoritmo alpha apresentado por AALST e HEE [6].

Let W be a workflow log over T. $\alpha(W)$ is defined as follows.

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = first(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = last(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
7. $F_W = \{ (a, p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p_{(A,B)}, b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W, t) \mid t \in T_I \} \cup \{ (t, o_W) \mid t \in T_O \}$, and
8. $\alpha(W) = (P_W, T_W, F_W)$.

Figura 4: codificação do algoritmo Alpha apresentado por AALST e HEE [6].

ADAPTATIVIDADE

Um dispositivo adaptativo se modifica dinamicamente, sem influências externas. Ele é influenciado somente através das entradas do sistema. Ele é formado por uma camada não adaptativa e uma adaptativa, composta de conjuntos de ações de Consulta, Adição e Eliminação [4].

Existe um modelo de Rede de Petri Adaptativa desenvolvido por GOMES e CAMOLESI [5]. Este modelo visa

à consulta das regras de entrada em uma transição e a adição ou eliminação de transições e posições da Rede de Petri. Quais regras serão definidas é um dos pontos a ser desenvolvido.

PROPOSTA

Este estudo tem o objetivo de (1) retirar os ruídos da análise de logs e (2) criar a rede de Petri através dos logs dinamicamente, sem necessitar ler toda a base existente antes de iniciar o algoritmo. A adoção de adaptatividade para cumprir o objetivo possibilita um maior dinamismo para o serviço.

Para se retirar o ruído utiliza-se um autômato adaptativo. O autômato adaptativo atua como um coletor de palavras na rede de Petri e faz a contagem de quantas vezes aquele caminho do fluxo é percorrido. Para se identificar um caminho fora do padrão utiliza-se o conceito apresentado por HAWKINS [7]. Neste conceito, caminhos que aparecem na frequência do interquartil×1,5 estão dentro da normalidade.

Para se fazer as podas da rede de Petri, após analisar que aquele caminho apresenta uma frequência baixa, então possivelmente é um fora do padrão, utiliza-se do conceito de tabela de decisão adaptativa. Se o caminho possui uma frequência alta, nenhum estado é alterado, se o caminho possui uma frequência baixa (operação de consulta), os estados envolvidos são retirados (operação de subtração) e nenhum estado é adicionado (operação de soma).

Para se fazer a rede de Petri dinamicamente, conforme logs vão sendo criados a rede de Petri vai se modificando, utiliza-se o conceito apresentado por PISTORI [8]. PISTORI [8] apresentou uma técnica de árvore de decisão adaptativa. Com a adaptatividade, foi possível elaborar um algoritmo incremental para gerar a árvore de decisão. Assim, durante o processo de classificação, foi permitida a incorporação de novos exemplos. Além disso, foi possível reordenar a árvore periodicamente.

RESULTADOS

O estudo proposto será válido se, após a aplicação de adaptatividade, o algoritmo alpha deixar de considerar ruídos em seus dados e se ele construir a rede de Petri conforme logs vão sendo criados, em tempo real. O algoritmo adaptativo não deve alterar significativamente o tempo de processamento do algoritmo alpha original. Essa análise é uma proposta para um próximo estudo.

REFERÊNCIAS

[1] MEDEIROS, A.K.A; SCHOENSLEBEN, Paul. **Workflow Mining: Current Status and Future Directions**. CoopIS/DOA/ODBASE 2003, LNCS 2888, pp. 389–406, 2003.

[2] AALST, Wil van Der. **Process Mining: Discovery, Conformance and Enhancement of Business Processes**. Berlin: Springer-verlag Berlin Heidelberg, 2011.

[3] BANERJEE, Anoopam; GRUPTA, Preeti. **Extension to Alpha Algorithm for Process Mining**. International Journal of Engineering & Computer Science, 2015.

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

[5] GOMES, Wilson C. M.; CAMOLESI, Almir R.. **Visualizador gráfico para Redes de Petri Adaptativa.** WTA - Workshop de Tecnologia Adaptativa. São Paulo, out. 2008.

[6] AALST, Wil van Der.; HEE, Van K.M.. **Workflow Management: Models, Methods, and Systems.** MIT press, Cambridge, MA, 2002.

[7] HAWKINS, D. **Identification of Outliers.** Chapman and Hall, London, 1980.

[8] PISTORI, Hemerson. **Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.** Tese de Doutorado, EPUSP, São Paulo, 2003.

Sheila Genesine Dada, graduada em engenharia elétrica com ênfase em computação pela Escola Politécnica da Universidade de São Paulo (2014). Atualmente é mestranda do Programa de Pós-Graduação em engenharia de computação da Escola Politécnica da Universidade de São Paulo, atuando como aluna pesquisadora no LTA – Laboratório de Linguagens e Técnicas Adaptativas – do PCS. Curriculum Lattes: http://lattes.cnpq.br/1162729917105344

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Técnicas Adaptativas 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. Curriculum Lattes: http://lattes.cnpq.br/4091709928353457.

Estudo comparativo de algoritmos de compressão de textos baseada em gramática

Newton K. Miura e J. José Neto

***Resumo*—A compressão baseada em gramática busca inferir a menor gramática possível para descrever sem perdas uma cadeia de símbolos. Este documento apresenta uma breve descrição de algoritmos de compressão gramatical encontrados na literatura nos quais o uso da tecnologia adaptativa será investigado para verificar potenciais vantagens trazidas por esta abordagem na simplificação da expressão de soluções, no aprendizado incremental da sintaxe de textos.**

***Palavras-chave:*—compressão baseada em gramática, inferência de gramáticas, gramáticas livres de contexto, autômatos adaptativos.**

I. Introdução

A compressão de texto baseada em gramática é um método para representar uma cadeia de caracteres por meio de uma gramática livre de contexto (GLC) definida conforme hierarquia de Chomsky [1], que gera uma única cadeia idêntica à original. Pesquisada desde a década de setenta [2], esta abordagem de compactação de dados baseia-se na ideia de que uma GLC pode representar de forma compacta as estruturas repetitivas existentes dentro de um texto. Intuitivamente, uma maior compressão seria obtida para cadeias de entrada que contenham uma maior quantidade de subcadeias repetidas e que consequentemente seriam representadas por uma mesma regra de produção gramatical. Exemplos de dados com estas características são as sequências de genes de uma mesma espécie, textos com controle de versão.

A definição naturalmente hierárquica de uma GLC permite que algoritmos de manipulação de cadeias possam realizar operações diretamente nas suas representações compactadas, sem a necessidade de uma descompressão prévia [3], [4], [5], [6]. Isso traz a vantagem de diminuir a o espaço de armazenamento temporário requerido para a manipulação de dados, além de abrir possibilidade dos algoritmos apresentarem tempos menores de execução ao se diminuir o dado a ser processado [3]. Essas características são atraentes por melhorar a eficiência no processamento de grandes quantidades de dados cuja demanda tem crescido principalmente em aplicações de *Big Data* e manipulação de genomas.

Operações em textos compactados gramaticalmente [7] incluem busca de cadeias, cálculos de distância de edição, frequência de repetição de cadeias ou caractere, cálculo do acesso a uma posição específica da cadeia original, obtenção da primeira ocorrência de um caractere e indexação para acesso randômico. Exemplos de aplicações de compressão baseada em gramática como mineração de estruturas repetitivas, reconhecimento de padrões, mineração de dados utilizando são citados em [8], [4] e [3].

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: nkmiura@usp.br e jjneto@usp.br.

O processo de obtenção deste tipo de gramática com características específicas, conforme descritas no item II-A, é um processo de inferência de gramática, que consiste num campo de estudos dentro das ciências da computação [9] desde a década de sessenta [10].

Este trabalho foca na compressão de textos baseada em gramática adotando um dispositivo adaptativo guiado por regras [11] para a identificação de dados repetitivos.

Um dispositivo adaptativo [12] tem a capacidade auto modificação, ou seja, ele altera as regras de seu funcionamento em tempo de execução sem a necessidade de intervenção externa. Um exemplo é o autômato adaptativo, que é uma máquina de estado com poder computacional equivalente à máquina de Turing [13], capaz de alterar a sua configuração de acordo com a cadeia de entrada.

Na seção II-A é apresentado o conceito básico desse tipo de compressão e alguns algoritmos encontrados na literatura. Na seção III é apresentada uma implementação baseada em autômato adaptativo.

II. Revisão da literatura

A. Compressão baseada em gramática

A compressão de texto é realizada por meio de um tipo de GLC $G = (\Sigma, V, D, X_s)$ [14] em que: Σ é o conjunto finito de símbolos terminais de tamanho $m = |\Sigma|$; V é o conjunto de símbolos não-terminais (ou variáveis) com $\Sigma \cap V = \emptyset$; $D \subset V \times (V \cup \Sigma)^*$ é o conjunto finito de regras de produção com tamanho $n = |V|$; e $X_s \in V$ é o não-terminal que representa o símbolo inicial da gramática. A compressão gramatical de uma cadeia S é uma GLC que produz somente S deterministicamente, ou seja, para qualquer $X \in V$ existe somente uma regra de produção em D e não há ciclos. A árvore sintática de G é uma árvore binária ordenada em que os nós internos são etiquetados com os símbolos não-terminais de V e as folhas com os terminais de Σ, ou seja, a sequência de etiquetas nas folhas corresponde à cadeia de entrada S. Cada nó interno Z corresponde a uma regra de produção $Z \rightarrow XY$ com os nós filhos X à direita e Y à esquerda. Como G pode ser expressa na forma normal de Chomsky [1] qualquer gramática de compressão é um *Straight Line Program* (SLP) [15], [14], [3] que é definida como uma compressão gramatical sobre $\Sigma \cup V$ e regras de produção na forma $X_k \rightarrow X_i X_j$ onde X_k, X_i, $X_j \in \Sigma \cup V$, e $1 \leq i, j < k \leq n + m$.

A figura 1 apresenta um exemplo de compactação da cadeia $S = (a, b, a, a, a, b, c, a, b, a, a)$, que resulta na sequência compactada $S^4 = \{X_4, c, X_3\}$ e o dicionário de derivações

$D = \{X_1 \rightarrow ab, X_2 \rightarrow aa, X_3 \rightarrow X_1X_2, X_4 \rightarrow X_3X_1\}$, que corresponde à floresta de árvores sintáticas. As linhas tracejadas de uma mesma cor identificam trechos da árvore que possuem nós pais com uma mesma etiqueta.

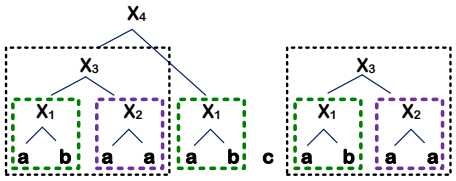


Figura 1. Exemplo de compressão baseada em gramática.

A taxa de compressão será maior quanto menor for a GLC inferida, o que constitui no principal desafio da compressão baseada em gramática, pois este problema foi demonstrado ser intratável. Storer e Szymanski [16] demonstraram que dada uma cadeia S e uma constante k a obtenção de uma GLC de tamanho k que gere S é um problema NP-completo, e Charikar et al. [17] demonstrou que a GLC mínima é aproximável a uma taxa logarítmica calculando que $8569/8568$ é o limite desta taxa de aproximação dos algoritmos para a obtenção do menor valor de k, assumindo que $P \neq NP$. Além dos algoritmos de inferência das gramáticas, outro objeto de pesquisa é a obtenção de uma estrutura de dados para armazenar a GLC inferida que tenham características adequadas para suportar operações no texto compactado [5] e sejam também compactas. Conforme analisado por [4] vários algoritmos iniciais adotaram a codificação de Huffman para compactar o dicionário embora ele não permita acesso aleatório das cadeias, e mais recentemente tem-se usado o método de representação *succint data structure*.

São apresentadas a seguir breves descrições de algumas pesquisas desenvolvidas.

B. Algoritmos de compressão

Nas descrições abaixo N é o tamanho da cadeia de entrada, g é o tamanho da gramática GLC mínima e $\log$ refere-se a $\log_2$.

O algoritmo Sequitur proposto por Nevill-Manning [18] opera de forma incremental em relação à cadeia de entrada com a restrição de que cada bigrama esteja presente somente uma única vez numa regra de derivação da gramática inferida e que cada regra seja usada mais de uma vez. Sequitur opera num espaço e tempo de execução linear em relação ao tamanho da cadeia de entrada.

O algoritmo RePair desenvolvido por Larsson e Moffat [19] constrói uma gramática substituindo iterativamente pares de símbolos, terminal ou não, por um símbolo não-terminal fazendo um processamento *off-line* do texto completo, ou de trechos longos, e adotando uma representação compacta do dicionário. Embora exija um espaço de memória acima de 10 vezes o tamanho da cadeia de entrada, o algoritmo apresenta um tempo de execução linear, sendo eficiente principalmente na descompressão, favorecendo operações de busca no dado compactado.

Rytter [20] e Charikar et al. [17] desenvolveram algoritmos que aproximam o tamanho da GLC obtida em $O(\log N/g)$ a partir da transformação para GLC da representação dos dados campactados com o método LZ77 de Lempel e Ziv [21]. Rytter propõe o uso de uma gramática cuja árvore de derivação é uma árvore binária AVL balanceada [22] em que a altura de dois sub-árvores filhas diferem somente de uma unidade, o que favorece operações de casamento de padrões. Charikar et al. impôs a condição de que a árvore de derivação binária seja balanceada em largura, favorecendo operações de união e divisão.

Sakamoto [23] adotou estratégia similar ao RePair [19] e obteve um algoritmo com exigência de um espaço em memória menor, realizando a substituição iterativa de pares de símbolos distintos e repetições de um mesmo símbolo, usando listas duplamente ligadas para armazenar a cadeia de entrada e uma fila de prioridade para a frequência dos pares.

O algoritmo de [23] foi modificado por Jez [24] que obteve uma gramática de tamanho $O(g \log(N/g))$ para cadeias de entrada com alfabeto Σ que possam ser identificados por números de $\{1, ..., N^c\}$ para uma constante c. Diferentemente das pesquisas anteriores a cadeia de entrada nas demonstrações e nas análises não foi baseada na representação Lempel-Ziv.

Maruyama et al. [6] desenvolveu um algoritmo baseado numa subclasse de gramática dependente de contexto Σ-$sensitive$ por ele proposto com o objetivo de otimizar a operação de casamento de padrão no dado compactado.

Tabei et al. [5] elaborou um algoritmo escalável para regressão parcial de mínimos quadrados baseado numa representação compactada por gramática de matrizes com alta dimensionalidade e que permite acesso rápido a linhas e colunas sem a necessidade de descompactação. Quando comparada a técnicas probabilísticas, esta abordagem mostrou superioridade em termos de precisão, eficiência computacional e facilitou a interpretação da relação entre os dados e as variáveis de etiquetas e respostas.

A compressão *online* de texto é o foco no algoritmo *Fully-online compression algorithm* (FOLCA) proposto por Maruyama et al. [25] que infere árvores de *parsing* parciais [20] cujos nós internos são percorridos *post-order* e armazenados numa representação sucinta. Para a classe $C = \{x_1, x_2, ..., x_n\}$ de n objetos, $\log n$ é o mínimo de bits para representar qualquer $x_i \in C$. Se o método de representação requer $n + O(n)$ bits para qualquer $x_i \in C$, a representação é chamada de sucinta [4]. São apresentados resultados experimentais comprovando a escalabilidade do algoritmo em espaço de memória e tempo de execução no processamento de genomas humanos com alta quantidade de textos repetitivos com presença de ruídos.

Fukunaga et al. [15] propôs um algoritmo *online* para identificação aproximada de padrões frequentes em dados compactados gramaticalmente com menor consumo de memória comparado com métodos *offline*. O *edit-sensitive parsing* (ESP) [26], que mede a similaridade de duas cadeias de símbolos pela distância de edição, é usado para a comparação de sub-árvores de gramáticas.

C. Adaptatividade em inferência gramatical

A compressão baseada em gramática é um caso específico de inferência gramatical cuja finalidade é a identificação de padrões sintáticos que permitem estabelecer as regras de formação de uma linguagem a partir de amostras [9]. Neste caso a amostra corresponde ao texto a ser compactado que é a única sentença que a gramática permite obter.

As pesquisas brevemente descritas abaixo empregaram técnicas baseadas em dispositivos adaptativos para a inferência gramatical.

Neto e Iwai [27] propuseram o seu uso para inferir um reconhecedor com capacidade de aprendizado de uma linguagem regular a partir do processamento de amostras positivas e negativas de cadeias pertencentes a essa linguagem. O reconhecedor é obtido a partir da aglutinação de dois autômatos adaptativos. Um deles constrói a árvore de prefixos a partir das cadeias de entrada e o outro produz uma árvore de sufixos a partir da sequência inversa da mesma cadeia. Matsuno [28] implementou este algoritmo baseado em autômatos adaptativos, e apresentou uma aplicação do algoritmo de Charikar [17] para obter uma GLC a partir de amostras da linguagem definida por essa gramática.

Tabela I
LISTA DE ELEMENTOS.

Elemento	Significado
Q	Conjunto dos estados do autômato finito
$F \subset Q$	Subconjunto dos estados de aceitação
$q_0 \in Q$	Estado inicial
Σ	Alfabeto da cadeia de entrada
D	Conjunto das funções adaptativas
P	$P : D \cup \{\epsilon\} \times Q \times \Sigma \mapsto Q \times \Sigma \cup \{\epsilon\} \times D \cup \{\epsilon\}$, Relação de mapeamento
$\sigma \in \Sigma$	Qualquer símbolo do alfabeto
$\mathcal{A}(q, x)$	Função adaptativa $\mathcal{A} \in D$ com argumentos q, x executado antes do consumo de um símbolo
$\mathcal{B}(y, z)$	Função adaptativa $\mathcal{B} \in D$ com argumentos y, z executado após o consumo de um símbolo
g_i^*	Gerador usado em funções adaptativas que associa nomes com novos estados criados
$-(q_i, \sigma) \rightarrow (q_j)$	Ação adaptativa elementar que remove a transição de q_i para q_j e que consome σ
$+(q_i, \sigma) \rightarrow (g_i^*, \epsilon), \mathcal{A}$	Ação adaptativa elementar que adiciona uma transição a partir do estado q_i, consumindo o símbolo σ, levando a um novo estado criado g_1^* pela função adaptativa $\mathcal{A}$ a ser executado antes do consumo do síbolo σ
Out_i	Ação semântica a ser executada em alguns estados, como na máquina de Mealy [1].

III. COMPRESSÃO BASEADA EM GRAMÁTICA COM IMPLEMENTAÇÃO ADAPTATIVA

A abordagem adaptativa adotada neste trabalho é inspirada no algoritmo RePair [19] e utiliza um autômato adaptativo no processo de identificação de padrões de símbolos repetidos na cadeia de entrada a serem substituídos por uma regra de produção de gramática.

O autômato adaptativo modifica a configuração de um autômato finito tradicional, que pode ser representado por uma tupla (Q, Σ, P, q_0, F). O significado de cada elemento desta tupla e os outros elementos utilizados nesta seção estão descritos na tabela I para referência. Foi adotada uma notação similar a apresentada por Cereda et al em [29].

A modificação no autômato subjacente ocorre por meio de funções adaptativas a serem executadas antes ou depois das transições, ou seja, de acordo com os símbolos consumidos da cadeia de entrada. A função adaptativa executada antes da transição é representada pelo caractere '$\cdot$' grafado após o nome da função (ex. $\mathcal{A}\cdot$) e função posterior pelo mesmo caractere grafado antes do nome (ex. $\cdot\mathcal{B}$). Elas podem modificar a configuração do autômato realizando ações adaptativas elementares de consulta, deleção ou inserção de regras. As funções adaptativas podem usar variáveis e geradores para as ações de alteração do autômato. As variáveis são preenchidas somente uma única vez durante a execução. Os geradores são um tipo especial de variáveis usadas para associar um nome único, sem ambiguidade, para cada estado novo criado no autômato e são identificados pelo símbolo '$*$' (ex. g_1^*, g_2^*).

O autômato adaptativo apresentado neste trabalho analisa trigramas presentes na cadeia de entrada para definir o par de símbolos mais adequado a ser substituído por uma regra de produção de gramática numa iteração do algoritmo RePair. Desta forma, a cada trigrama o autômato reinicia a sua operação a partir do estado inicial q_0. Isso requer a obtenção do conjunto de todos os trigramas presentes na cadeia de entrada. Por exemplo, considerando a sequência $abcab$, o conjuto das entradas para o autômato será $\{abc, bca, cab\}$.

O autômato totaliza a ocorrência de cada trigrama e de seus bigramas constituintes, prefixo e sufixo, da cadeia de entrada. Isso é realizado por meio de contadores que são incrementados por funções de ação semântica Out_i executados nos estados em que um bigrama ou trigrama é reconhecido. Por exemplo, para um trigrama abc, são contabilizados a quantidade total de repetição do trigrama, do prefixo ab e do sufixo bc.

Cada bigrama terá 2 contadores, um para prefixo e outro para sufixo, que podem ser incrementados em qualquer estado do autômato que tenha a função de ação semântica associada. Com base nestes contadores, após o processamento de todos os trigramas será possível escolher o par a ser substituído. Um exemplos de critério de escolha é o par com a maior quantidade de repetições dentro do trigrama mais frequente. Outros critérios pode ser a escolha do prefixo ou sufixo deste trigrama.

Partindo do estado terminal no qual o trigrama mais frequente é reconhecido, ao percorrer o autômato no sentido inverso das transações, ou seja, em direção ao estado inicial, é possível identificar os bigramas constituintes e obter os valores de seus contadores. No contador de bigrama associado ao estado terminal do trigrama é obtido o valor da contagem para o bigrama sufixo, e no estado anterior o contador do bigrama prefixo.

O uso da adaptatividade orienta o projeto deste autômato de forma a considerar como ele deve ser construído de forma incremental para que à medida que se consome os símbolos da cadeia de entrada o objetivo acima seja atingido. Do estado inicial até qualquer estado final ele terá no máximo 3 transições, pois ele analisa apenas 3 símbolos.

A seguir é apresentada a evolução das configurações deste

autômato adaptativo para o primeiro trigrama $\sigma_0\sigma_1\sigma_2$ de uma cadeia de entrada.

A figura 2 apresenta a configuração inicial simplificada do autômato composto por um único estado q_0 e transições para cada símbolo $\sigma \in \Sigma$ executando a função adaptativa $\mathcal{A}_0(\sigma, q_0)$ antes do consumo. Para facilitar a visualização, a representação do conjunto destas transições foi substituída por um único arco em que o símbolo a ser consumido está indicado como $\forall\sigma$. Esta mesma simplificação de representação foi adotada na descrição do algoritmo 1 da função adaptativa $\mathcal{A}_0$. Ela remove a transição que consome o primeiro símbolo σ_0 do trigrama, cria um novo estado q_1 e a transição para ele consumindo σ_0, e também transições de q_1 para ele mesmo, associados ao consumo de $\forall\sigma \in \Sigma$, e outra função adaptativa $\mathcal{A}_1$.

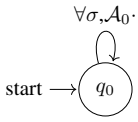


Figura 2. Configuração inicial do autômato adaptativo.

Algoritmo 1: Função adaptativa $\mathcal{A}_0$

função adaptativa $\mathcal{A}_0(s,\ q_x)$
Geradores: g_1^*
$-(q_x, s) \rightarrow q_x$
$+(q_x, s) \rightarrow (g_1^*, \epsilon)$
$+(g_1^*, \forall\sigma) \rightarrow (g_1^*, \epsilon), \mathcal{A}_1$
fim

A figura 3 apresenta a nova configuração do autômato adaptativo após o consumo do primeiro símbolo σ_0. O algoritmo 2 apresenta a função adaptativa $\mathcal{A}_1$. A sua operação é similar à de $\mathcal{A}_0$, criando um novo estado q_2, porém ela prepara o consumo de um terceiro símbolo inserindo uma transação com a função adaptativa $\mathcal{A}_2$, que está descrita no algoritmo 3.

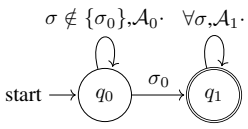


Figura 3. Configuração do autômato após o consumo do primeiro símbolo σ_0.

Algoritmo 2: Função adaptativa $\mathcal{A}_1$

função adaptativa $\mathcal{A}_1(s,\ q_x)$
Geradores: g_1^*
$-(q_x, s) \rightarrow q_x$
$+(q_x, s) \rightarrow (g_1^*, \epsilon),$
$+(g_1^*, \sigma) \rightarrow (g_1^*, \epsilon), \mathcal{A}_2$
fim

O algoritmo 3 modifica a configuração do autômato removendo a trasição de q_2 para ele mesmo consumindo o símbolo σ_3, cria um novo estado q_3 e a transição para ele consumindo σ_3, o terceiro símbolo do trigrama. O novo estado criado será associado à função semântica Out_1.

Algoritmo 3: Função adaptativa $\mathcal{A}_2$

função adaptativa $\mathcal{A}_2(s,\ q_x)$
Geradores: g_1^*
$-(q_x, s) \rightarrow q_x$
$+(q_x, s) \rightarrow (g_1^*, \epsilon)$
fim

Após o consumo do segundo e do terceiro símbolo da cadeia de entrada é obtido o autômato da figura 4.

Os estados q_2 e q_3 são associados a funções de saída Out_0 and Out_1 respectivamente correspondentes a ações semânticas nestes estados.

Out_0 é a função responsável por incrementar o contador de ocorrências do bigrama prefixo $\sigma_0\sigma_1$. Out_1 é responsável por incrementar o contador de ocorrências do bigrama sufixo $\sigma_1\sigma_2$, e o contador do trigrama $\sigma_0\sigma_1\sigma_2$.

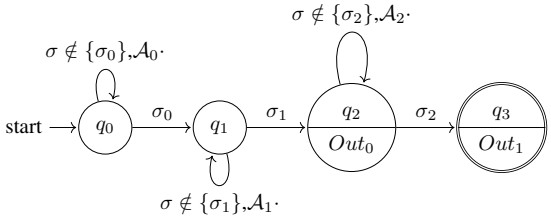


Figura 4. Configuração do autômato após o consumo do trigrama inicial $\sigma_0\sigma_1\sigma_2$.

Para melhor ilustrar o funcionamento do autômato adaptativo, a figura 5 apresenta a sua configuração após o processamento da cadeia exemplo $abcaba$. O índice i dos estados q_i corresponde à sequência em que foram inseridos pelo algoritmo.

IV. Experimentos

Está em elaboração um sistema de testes que implementa o autômato adaptativo utilizando a biblioteca AA4J [1] [30], as funções de saída correspondentes às ações semânticas e os processos iterativos de substituição de pares por regras de produção gramatical, de forma similar ao algoritmo RePair.

Para os ensaios serão utilizados corpora publicamente disponíveis [2].

Nos ensaios serão mensurados e comparados as taxas de compressão, tempo de execução e requisitos de armazenamento temporário para os diferentes critérios de escolha de padrões repetitivos que o algoritmo permite.

V. Conclusão

Para obter uma regra de substituição num algoritmo de compressão baseada em gramática, neste trabalho apresentamos o autômato adaptativo como dispositivo para identificar

[1] https://github.com/cereda/aa
[2] http://pizzachili.dcc.uchile.cl/repcorpus.html

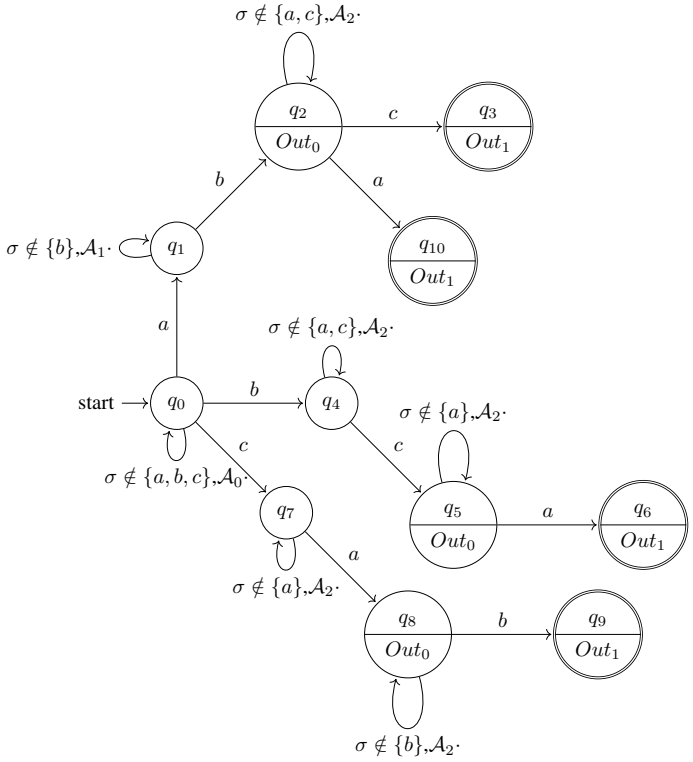


Figura 5. Configuração do autômato após o processamento da cadeia $abcaba$.

bigramas, prefixo ou sufixo, que mais se repetem numa cadeia de símbolos e que também ocorrem dentro de um trigrama mais repetido.

Como trabalho futuro, o autômato adaptativo poderá ser expandido para analisar n-gramas maiores que trigramas. Além disso, um estudo comparativo de desempenho poderá ser feito com outras técnicas. Outro ponto a ser analisado é a adoção de um formalismo gramatical adaptativo [31] na descrição da gramática inferida com o objetivo de viabilizar alguma operação diretamente no dado compactado.

A tecnologia adaptativa possibilita a construção de autômato de grandes dimensões e complexos por meio da simplificação da representação do problema. Este tipo de autômato é projetado por meio da especificação da forma como o dispositivo deve ser incrementalmente modificado de acordo com os dados de entrada, a partir de uma configuração inicial simples, visando a suas configurações intermediárias desejadas, e também a saída a ser obtida pelas ações semânticas associadas.

REFERÊNCIAS

[1] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1997.

[2] F. Rubin, "Experiments in text file compression," *Commun. ACM*, vol. 19, no. 11, pp. 617–623, Nov. 1976. [Online]. Available: http://doi.acm.org/10.1145/360363.360368

[3] M. Lohrey, "Algorithmics on slp-compressed strings: A survey." *Groups Complexity Cryptology*, vol. 4, no. 2, pp. 241–299, 2012. [Online]. Available: http://dx.doi.org/10.1515/gcc-2012-0016

[4] H. Sakamoto, "Grammar compression: Grammatical inference by compression and its application to real data," in *Proceedings of the 12th International Conference on Grammatical Inference, ICGI 2014, Kyoto, Japan, September 17-19, 2014.*, 2014, pp. 3–20. [Online]. Available: http://jmlr.org/proceedings/papers/v34/sakamoto14a.html

[5] Y. Tabei, H. Saigo, Y. Yamanishi, and S. J. Puglisi, "Scalable partial least squares regression on grammar-compressed data matrices," in *22nd KDD*, 2016, pp. 1875—-1884.

[6] S. Maruyama, Y. Tanaka, H. Sakamoto, and M. Takeda, "Context-sensitive grammar transform: Compression and pattern matching," *IEICE Transactions on Information and Systems*, vol. E93.D, no. 2, pp. 219–226, 2010.

[7] Y. Tabei, "Recent development of grammar compression," *Information Processing Society of Japan Magazine*, vol. 57, no. 2, pp. 172–178, jan 2016.

[8] A. Jeż, *A really Simple Approximation of Smallest Grammar*. Springer International Publishing, 2014, pp. 182–191.

[9] C. De la Higuera, *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

[10] M. E. Gold, "Language identification in the limit," *Information and Control*, vol. 10, no. 5, pp. 447–474, 1967.

[11] J. José Neto, "Adaptive automata for context-sensitive languages," *SIGPLAN Notices*, vol. 29, no. 9, pp. 115–124, set 1994.

[12] ——, "Adaptive rule-driven devices - general formulation and case study," in *CIAA 2001 6th International Conference on Implementation and Application of Automata*, ser. Lecture Notes in Computer Science, B. W. Watson and D. Wood, Eds., vol. 2494. Pretoria, África do Sul: Springer-Verlag, Julho 2001, pp. 234–250.

[13] R. L. A. Rocha and J. José Neto, "Adaptive automaton, limits and complexity compared to the Turing machine," in *Proceedings of the I LAPTEC*, 2000, pp. 33–48.

[14] Y. Takabatake, Y. Tabei, and H. Sakamoto, *Online Self-Indexed Grammar Compression*. Springer International Publishing, 2015, pp. 258–269.

[15] S. Fukunaga, Y. Takabatake, T. I, and H. Sakamoto, "Online grammar compression for frequent pattern discovery," *CoRR*, vol. abs/1607.04446, 2016.

[16] J. A. Storer and T. G. Szymanski, "The macro model for data compression," in *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, ser. STOC '78. New York, NY, USA: ACM, 1978, pp. 30–39. [Online]. Available: http://doi.acm.org/10.1145/800133.804329

[17] M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat, "The smallest grammar problem." *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2554–2576, 2005.

[18] C. G. Nevill-Manning and I. H. Witten, "Identifying hierarchical structure in sequences: A linear-time algorithm," *J. Artif. Intell. Res.(JAIR)*, vol. 7, pp. 67–82, 1997.

[19] N. J. Larsson and A. Moffat, "Offline dictionary-based compression," in *Data Compression Conference, 1999. Proceedings. DCC '99*, Mar 1999, pp. 296–305.

[20] W. Rytter, *Application of Factorization to the Approximation of Grammar-Based Compression*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 20–31.

[21] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theor.*, vol. 23, no. 3, pp. 337–343, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1977.1055714

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[23] H. Sakamoto, "A fully linear-time approximation algorithm for grammar-based compression," *Journal of Discrete Algorithms*, vol. 3, no. 2–4, pp. 416–430, 2005, combinatorial Pattern Matching (CPM) Special IssueThe 14th annual Symposium on combinatorial Pattern Matching. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570866704000632

[24] A. Jez, "Approximation of grammar-based compression via recompression," *Theoretical Computer Science*, vol. 592, pp. 115–134, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397515004685

[25] S. Maruyama and Y. Tabei, "Fully online grammar compression in constant space." in *DCC*, A. Bilgin, M. W. Marcellin, J. Serra-Sagristà, and J. A. Storer, Eds. IEEE, 2014, pp. 173–182.

[26] G. Cormode and S. Muthukrishnan, "The string edit distance matching problem with moves," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 1, pp. 2:1–2:19, 2007.

[27] J. José Neto and M. K. Iwai, "Adaptive automata for syntax learning," in *Anais da XXIV Conferência Latinoamericana de Informática - CLEI 98*, 1998, pp. 135–149.

[28] I. P. Matsuno, "Um estudo dos processos de inferência de gramáticas regulares e livres de contexto baseados em modelos adaptativos," Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2006.

[29] P. R. M. Cereda and J. José Neto, "A recommendation engine based on adaptive automata," in *Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 2: ICEIS*, 2015, pp. 594–601.

[30] ——, "AA4J: uma biblioteca para implementação de autômatos adaptativos," in *Memórias do X Workshop de Tecnologia Adaptativa – WTA 2016*, 2016, pp. 16–26.

[31] M. K. Iwai, "Um formalismo gramatical adaptativo para linguagens dependentes de contexto," Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2000.

Newton Kiyotaka Miura é graduado em Engenharia de Eletricidade pela Escola Politécnica da Universidade de São Paulo (1989) e mestre em Engenharia de Sistemas pela Universidade de Kobe, Japão (1993). Atualmente é pesquisador da Olos Tecnologia e Sistemas e doutorando do Programa de Pós-Graduação em Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo, na área de concentração Engenharia de Computação, atuando como aluno pesquisador no Laboratório de Linguagens e Técnicas Adaptativas do Departamento de Eng. de Computação e Sistemas Digitais. Tem experiência na área de ciência da computação, com ênfase em sistemas de computação, atuando principalmente nos seguintes temas: tecnologia adaptativa, autômatos adaptativos, dispositivos adaptativos, análise e processamento de linguagem natural.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980)e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Técnicas Adaptativas 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.

Parte II

Submissões

O Reconhecedor Gramatical Adaptativo Linguístico: Experimentos e Resultados Comparativos

A. T. Contier, D. Padovani e J. 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, um reconhecedor gramatical que utiliza autômatos e gramáticas adaptativas como tecnologia subjacente. O presente artigo é uma continuação de trabalhos anteriores, com uma nova seção para apresentar os experimentos realizados e os resultados obtidos comparados aos do estado da arte.**

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

I. Autômatos Adaptativos

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

O dispositivo convencional pode ser uma gramática, um autômato, ou qualquer outro dispositivo que respeite um conjunto finito de regras estáticas. Este dispositivo possui uma coleção de regras, usualmente na forma de cláusulas if-then, que testam a situação corrente em relação a uma configuração específica e levam o dispositivo à sua próxima situação. Se nenhuma regra é aplicável, uma condição de erro é reportada e a operação do dispositivo, descontinuada. Se houver uma única regra aplicável à situação corrente, a próxima situação do dispositivo é determinada pela regra em questão. Se houver mais de uma regra aderente à situação corrente do dispositivo, as diversas possíveis situações seguintes são tratadas em paralelo e o dispositivo exibirá uma operação não determinística. Os mecanismos adaptativos são formados por três tipos de ações adaptativas elementares: consulta (inspeção do conjunto de regras que define o dispositivo), exclusão (remoção de alguma regra) e inclusão (adição de uma nova regra).

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

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

II. Processamento da Linguagem Natural: Revisão da Literatura

O processamento da linguagem natural requer o desenvolvimento de programas que sejam capazes de determinar e interpretar a estrutura das sentenças em muitos níveis de detalhe. As linguagens naturais exibem um intricado comportamento estrutural visto que são profusos os casos particulares a serem considerados. Uma vez que as linguagens naturais nunca são formalmente projetadas, suas regras sintáticas não são simples nem tampouco óbvias e tornam, portanto, complexo o seu processamento computacional. Muitos métodos são empregados em sistemas de processamento de linguagem natural, adotando diferentes paradigmas, tais como métodos exatos, aproximados, pré-definidos ou interativos, inteligentes ou algorítmicos [5]. Independentemente do método utilizado, o processamento da linguagem natural envolve as operações de análise léxico-morfológica, análise sintática, análise semântica e análise pragmática [6]. A análise léxico-morfológica procura atribuir uma classificação morfológica a cada palavra da sentença, a partir das informações armazenadas no léxico [7]. O léxico ou dicionário é a estrutura de dados contendo os itens lexicais e as informações correspondentes a estes itens. Entre as informações associadas aos itens lexicais, encontram-se a categoria gramatical do item, tais como substantivo, verbo e adjetivo, e os valores morfossintático-semânticos, tais como gênero, número, grau, pessoa, tempo, modo, regência verbal ou nominal. Na etapa de análise sintática, 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. 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. O mapeamento é feito identificando as propriedades semânticas do léxico e o relacionamento semântico entre os itens que o compõe [8]. Já a análise pragmática procura reinterpretar a estrutura que representa o que foi dito para determinar o que realmente se quis dizer. Inserem-se nessa categoria as relações anafóricas, correferências, determinações, focos ou temas, dêiticos e elipses [9].

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.

III. RECONHECEDOR ADAPTATIVO: SUPORTE TEÓRICO LINGUÍSTICO

A Moderna Gramática Brasileira de Celso Luft [12] foi escolhida como suporte teórico linguístico do reconhecedor aqui proposto. A escolha foi feita em função da forma clara e precisa com que Luft categoriza os diversos tipos de sentenças de língua portuguesa, diferenciando-se 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 frasais ou oracionais, compostos por elementos denominados sintagmas (Tabela 1).

TABELA 1.
ELEMENTOS FORMADORES DE SINTAGMAS

Sintagmas	
Substantivo	Quantitativos+Pronomes Adjetivos+ Sintagma Adjetivo1+Substantivo+ Sintagma Adjetivo2+ Sintagma Preposicional+ Oração Adjetiva
Verbal	Pré-verbais+ Verbo Auxiliar+ Verbo Principal

TABELA 1.
CONTINUAÇÃO

Sintagmas	
Adjetivo	Advérbio de Intensidade+ Adjetivo+ Sintagma Preposicional
Adverbial	Advérbio de Intensidade+ Adverbio+ Sintagma Preposicional
Preposicional	Preposição+ Sintagma Substantivo

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

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. Os padrões são classificados em cinco tipos (Tabela 2):

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).
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.
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.

TABELA 2
PADRÕES ORACIONAIS DE LUFT

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			
Padrões Pessoais Verbo-Nominais				
SS	Vtpred	SS	SS	
SS	Vtpred	SS	Sadj	
SS	Vtpred	SS	SP	
SS	Vtpred	SS	Sadv	
SS	Vtpred	SS		
SS	Vtpred	Sadj		
SS	Vtpred	SP		
Padrões Impessoais Nominais				
	Vlig	SS		
	Vlig	Sadj		
	Vlig	Sadv		
	Vlig	SP		
Padrões Impessoais Verbais				
	Vtd	SS		
	Vti	SP		
	Vi			

Os sintagmas e os padrões oracionais de Luft foram mapeados em uma gramática para que pudessem ser processados computacionalmente, ficando da seguinte forma:

S → [Conec] [SS] SV [Conec]
Conec → S
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:
S – Sentença
SS – Sintagma substantivo
SV – Sintagma verbal
SP – Sintagma preposicional
SN – Sintagma nominal
Sadv – Sintagma adverbial
Sadj – Sintagma adjetivo
Adv – adverbio
Adj – adjetivo
ArtDef – artigo definido
ArtInd – artigo indefinido
Aux – Partícula auxiliar (apassivadora ou pré-verbal)
Conec – Conector (conjunção ou pronome relativo)
Dem – pronome demonstrativo indefinido
Ind – pronome indefinido
Neg – partícula (negação)
PrA – pronome adjetivo
PrPes – pronome pessoal
Prep – preposição
Quant – numeral
Sc – substantivo comum
Sp – substantivo próprio
V– verbo
Vlig – verbo de ligação
Vi – verbo intransitivo
Vtd – verbo transitivo direto
Vti – verbo transitivo indireto
Vtdi – verbo transitivo direto e indireto

IV. PROPOSTA DE UM RECONHECEDOR GRAMATICAL

O Linguístico é uma proposta de reconhecedor gramatical composto de cinco 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 (Fig.1).

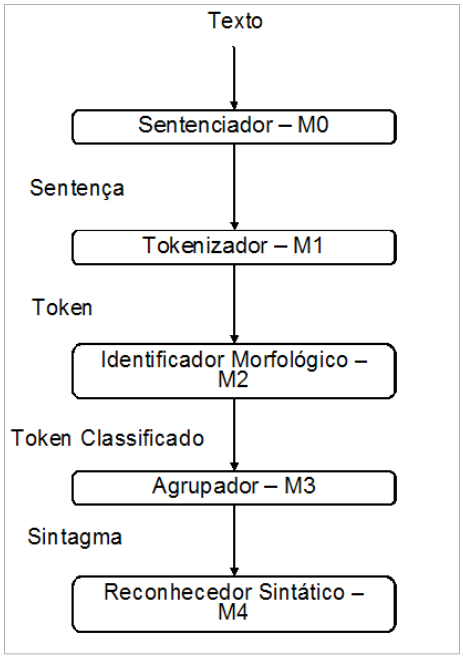


Figura 1. Estrutura do Linguístico.

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 *Token*izador, 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, chamado Identificador Morfológico, foi concebido para utilizar tecnologias adaptativas (Fig.2.1). O Identificador Morfológico é composto por um Autômato Mestre e um conjunto de máquinas especialistas que acessam bases de dados de regras de acesso às classificações morfológicas. A prioridade é obter as classificações do corpus Bosque [13]; caso o termo procurado não seja encontrado, o Identificador Morfológico procura por substantivos, adjetivos e verbos, no formato finito e infinito (flexionados e não flexionados), através de uma máquina de formação e identificação de palavras, que usa, como base, o vocabulário do TeP2.0 [14]; por fim, o Identificador Morfológico 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 [15].

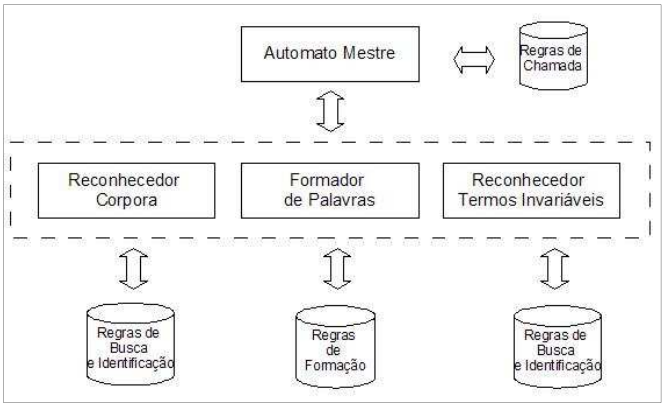


Figura 2.1. Arquitetura do Identificador Morfológico.

O Autômato Mestre (Fig.2.2) é responsável pelo sequenciamento das chamadas às máquinas, de acordo com um conjunto de regras cadastradas em base de dados. Ele inicia o 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 máquina de processamento (M1), uma transição entre o estado 1 e M1, e uma transição que aguarda o estado final da máquina 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 máquina 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.

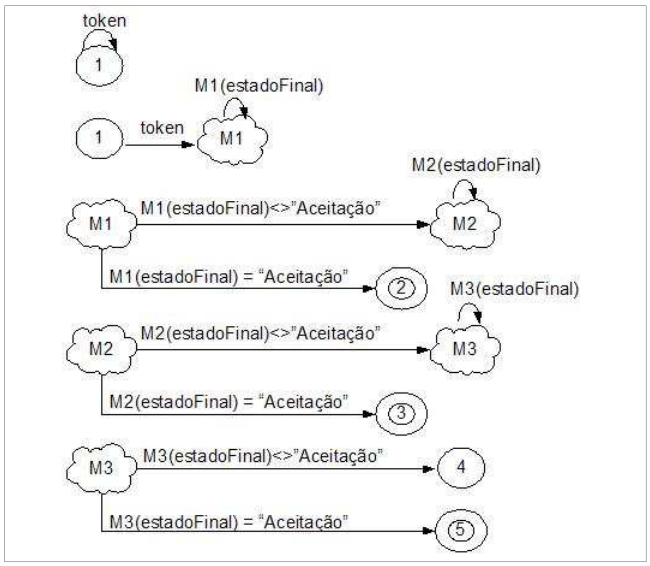


Figura 2.2. Estrutura Adaptativa do Autômato Mestre.

O processo se repete quando M2 chega ao final do processamento, com a criação da máquina 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 máquinas do Reconhecedor de Corpus, 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. Embora o Autômato Mestre processe um número pré-determinado de transições de estado e inicialmente não apresente variação do fluxo, optou-se pelo uso da tecnologia adaptativa devido à flexibilidade com que regras podem ser incluídas ou alteradas, o que é particularmente útil, caso seja necessário usar novas técnicas de reconhecimento.

As máquinas M1, M2 e M3 também foram projetadas de acordo com a tecnologia adaptativa. A Máquina M1 usa um autômato adaptativo que se auto modifica de acordo com o tipo de *token* que está sendo analisado: palavras simples, palavras compostas, números, valores e símbolos. A Fig. 2.3 apresenta a estrutura adaptativa de M1.

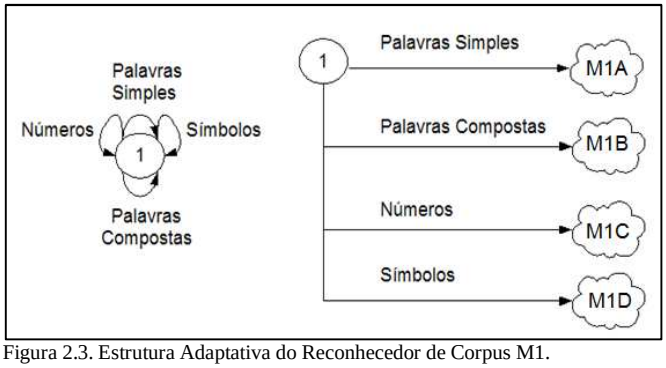


Figura 2.3. Estrutura Adaptativa do Reconhecedor de Corpus M1.

Inicialmente o autômato é composto por um único estado e por transições para ele mesmo (Fig.2.3, à esquerda). Ao identificar o tipo de *token* que será analisado (obtido no processo de *token*ização), o autômato cria máquinas e as transições correspondentes. As alternativas de configuração são apresentadas na Fig.2.3, à direita. As submáquinas M1A, M1B, M1C e M1D reconhecem os *tokens* através de outro tipo de autômato que processa as unidades conforme apresentado na Fig. 2.4.

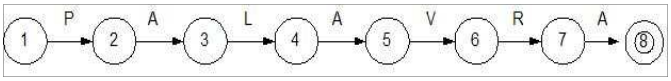


Figura 2.4. Reconhecedor de Palavras Simples.

No exemplo apresentado na Fig.2.4, o *token* "Palavra" é processado pela submáquina M1A; se o processamento terminar em um estado de aceitação, o *token* é reconhecido. As submáquinas M1A, M1B, M1C e M1D são criadas previamente por um programa que lê o Corpus 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 Corpus Bosque.

O Formador de Palavras, máquina M2, também é montado previamente por um programa construtor, usando o vocabulário do TeP2.0 como léxico de formas previamente construídas e o conjunto de regras de prefixação, sufixação e regressão verbal descrito por Margarida Basílio [16] para construir novas formas. As regras de formação são mapeadas previamente ao conjunto de palavras ao qual podem ser aplicadas, evitando repetições e permitindo a seleção das formas pertinentes no momento em que a palavra em questão é analisada. No entanto, são necessários alguns cuidados para evitar a criação de estruturas que aceitem palavras inexistentes. A Fig. 2.5 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.

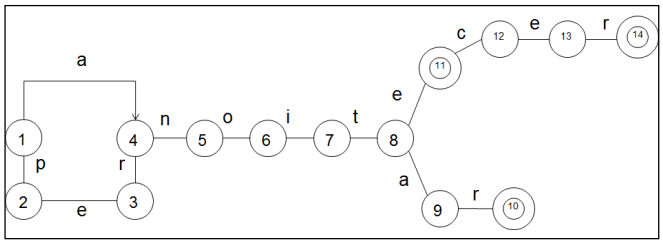


Figura 2.5. Autômato Formador de Palavras.

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 [17]. 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 [16], 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.

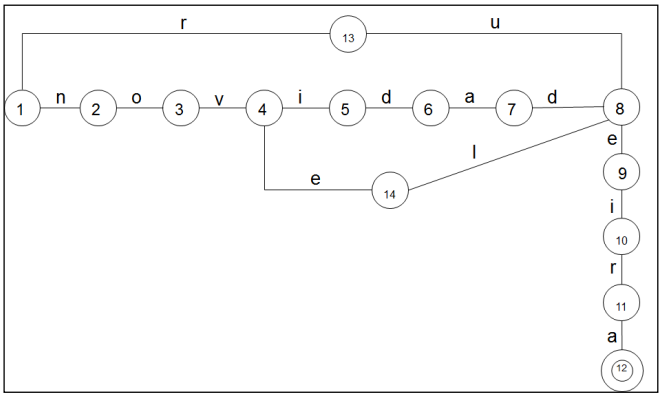


Figura 2.6. Autômato Formador de Palavras.

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. 2.6 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.

A máquina 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. 2.7 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)".

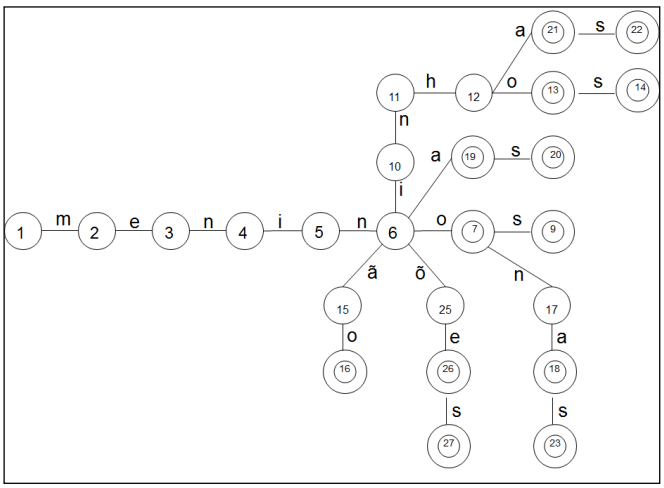


Figura 2.7. Autômato Formador de Flexões Nominais.

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. 2.8 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”.

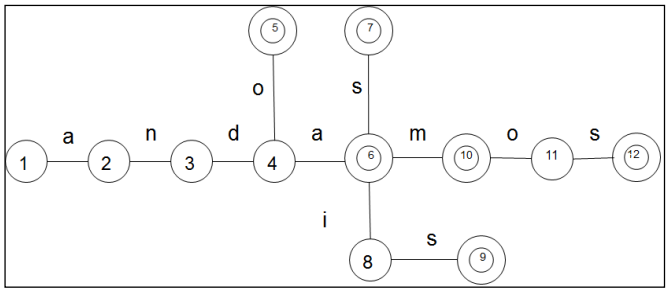


Figura 2.8. Autômato Formador de Flexões Verbais.

A estrutura de armazenamento da máquina 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 máquina M3 é um autômato que varia em função do tipo de termo (conjunções, preposições e pronomes) e utiliza uma estrutura arbórea similar a submáquina M1A. A Fig. 2.9 apresenta um exemplo de autômato usado no reconhecimento de termos deste domínio.

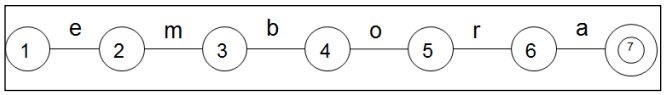


Figura 2.9. Autômato Reconhecedor de Conjunções, Preposições e Pronomes.

A estrutura de armazenamento da máquina 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 Francisco. Por exemplo, o termo “embora” seria classificado como cj+Ba –conjunção da base de dados de apoio.

O Autômato Mestre também é responsável por desambiguar as classificações morfológicas e informar ao Agrupador apenas as que forem mais adequadas. Como as palavras podem ter mais do que uma classificação morfológica, é necessário utilizar uma técnica para selecionar aquela que é mais apropriada para o contexto analisado. Por exemplo, a palavra “casa” pode ser classificada como substantivo comum, feminino, singular ou como verbo flexionado na 3ª pessoa do singular no tempo presente, modo indicativo. No entanto, tendo em vista o contexto em que palavra se encontra, é possível escolher a classificação mais provável. Por exemplo, se a palavra “casa” vier precedida de um artigo definido, é mais provável que ela seja um substantivo; já se a palavra antecessora for um substantivo próprio, é mais provável que “casa” seja um verbo.

Collins [18] apresenta um modelo estatístico que leva em consideração 2 classificações anteriores à palavra analisada para fazer a desambiguação, criando distribuições nas quais a etiqueta de máxima probabilidade é o resultado da função:

P = max p(E,S), sendo:
E = etiquetas
S = palavras da sentença
p = probabilidade de E, dado S
max = máxima probabilidade

Por exemplo, para etiquetar a frase “O advogado entrevista a testemunha”, a função receberia as palavras da sentença e as sequências de etiquetas possíveis para elas (obtidas a partir de um Corpus de testes), calculando, então, a sequência de maior probabilidade. No entanto, Collins alerta que a complexidade para executar tal função inviabiliza sua utilização, pois ela cresce exponencialmente em função do número de palavras da sentença (Em uma sentença de “n” palavras e “K” possíveis classificações, existiriam $|K|^n$ possíveis etiquetas de sequencia).

Para evitar o problema da complexidade da execução, Collins propõe o uso do algoritmo Viterbi [19]. O algoritmo Viterbi é usado para encontrar a sequência mais provável de estados ocultos que resultam da observação de uma sequência de eventos. No caso da identificação das etiquetas morfológicas, os eventos são as palavras do texto analisado e os estados são as etiquetas de identificação morfológica. Para uma dada sentença ou sequencia de palavras, o algoritmo Viterbi escolhe a etiqueta que maximiza a seguinte fórmula:

P(token|etiqueta) * P(etiqueta| n etiquetas anteriores) (1)

O algoritmo Viterbi pode ser implementado por meio de um autômato finito. No entanto, um autômato com estas características poderia ficar muito grande e, consequentemente, consumir elevados recursos computacionais, devido ao tamanho do léxico usado para montá-lo. Uma alternativa para evitar este tipo de problema, é usar a tecnologia adaptativa. A Fig. 2.10 apresenta um exemplo da estrutura do autômato que implementa o algoritmo Viterbi usado pelo desambiguador morfológico do Linguístico. A sequência de *tokens* analisados é “A casa”. O passo apresentado mostra como o autômato seleciona a etiqueta do token “casa”. A primeira etiqueta foi identificada como “Início” e a segunda como “Artigo”. Supõe-se, no exemplo, que além de “casa”, “bicicleta” e “cidade” sejam *tokens* possíveis, dadas as sequências de etiquetas analisadas (Início-> Artigo->Substantivo e Início -> Artigo -> Verbo).

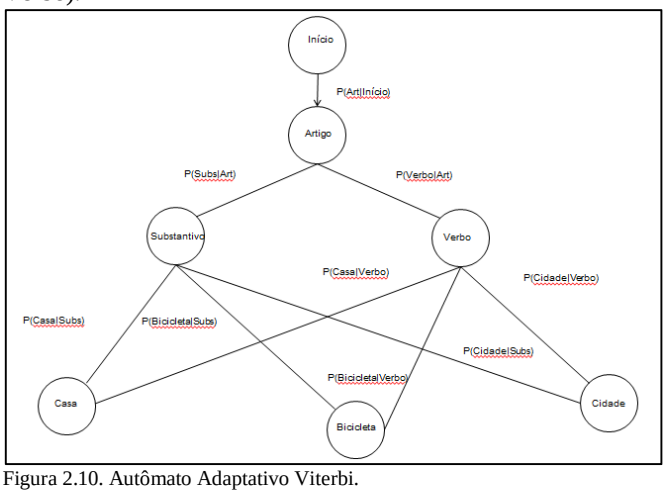


Figura 2.10. Autômato Adaptativo Viterbi.

O autômato monta dinamicamente os estados e transições de acordo com as etiquetas e probabilidades, identificando, ao final do processamento, a etiqueta mais provável. No exemplo apresentado, a seleção é feita através da comparação das seguintes sequências:

Início -> Artigo -> Substantivo -> Casa
Início -> Artigo -> Substantivo -> Bicicleta
Início -> Artigo -> Substantivo -> Cidade
Início -> Artigo -> Verbo -> Casa
Início -> Artigo -> Verbo -> Bicicleta
Início -> Artigo -> Verbo -> Cidade

Ao término das comparações, o autômato seleciona a etiqueta que maximiza a probabilidade de ocorrência de acordo com a fórmula geral de Viterbi (1), no caso, "Substantivo".

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.3). Inicialmente, o Agrupador recebe do Identificador as classificações morfológicas dos *tokens* e as agrupa em sintagmas de acordo com as regras de 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}

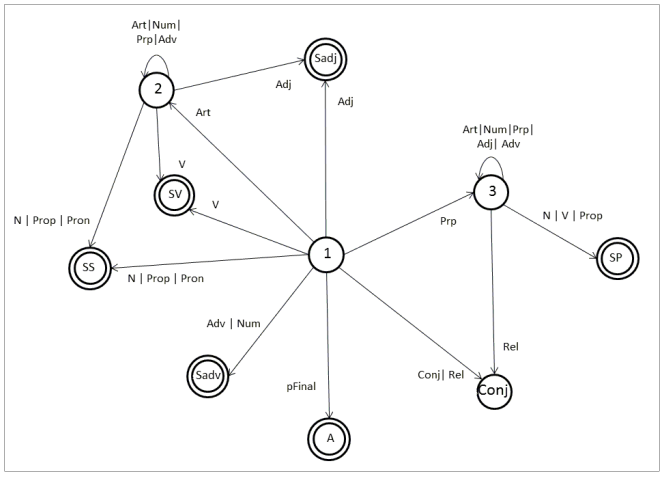


Figura 3. Configuração Completa do Autômato Construtor de Sintagmas.

Por exemplo, no caso de sintagmas substantivos teríamos:
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 transições:
PrA Sc → SS; A Sc → SS; A casa → SS
Já o Agrupador recebe o *token* "A", identificado pelo *Token*izador 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. Para isso ele usa a matriz apresentada na Tabela 3. 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.

TABELA 3.
MATRIZ DE AGRUPAMENTO DE SINTAGMAS

SA / US	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 oracionais 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]}

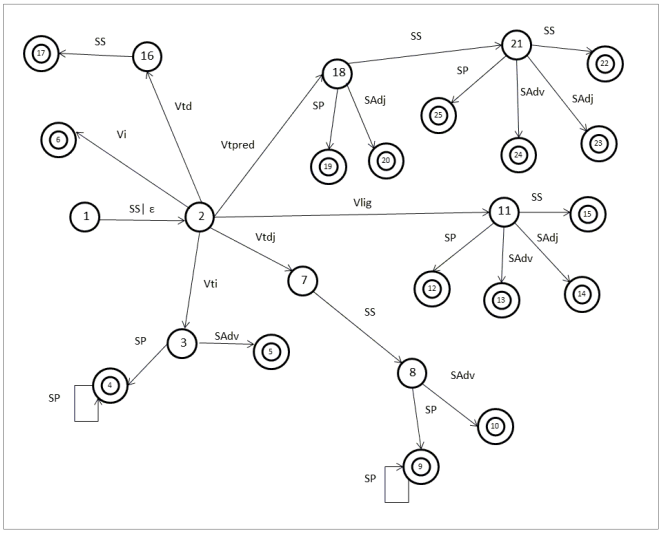


Figura 4.1. Configuração Completa do Reconhecedor Sintático.

No entanto, para que a análise sintática seja feita, não são necessárias todas as ramificações da configuração completa do autômato (Fig. 4.1). 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. 4.2 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.

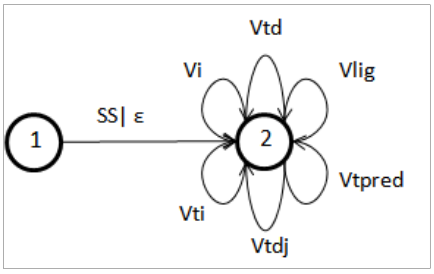


Figura 4.2. Configuração Inicial do Reconhecedor Gramatical.

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. 4.3 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. 4.1.

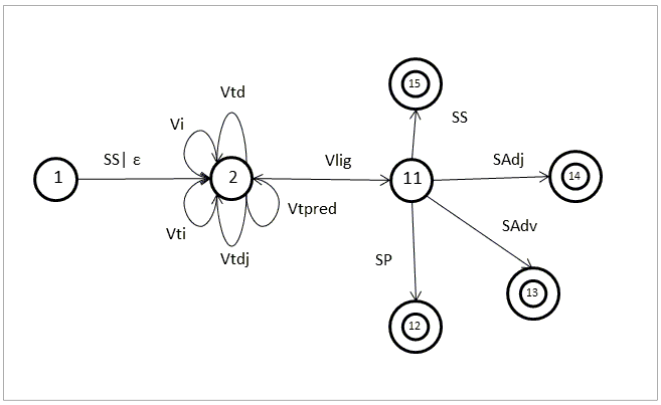


Figura 4.3. Configuração do autômato após o processamento do verbo de ligação.

V. DESAMBIGUAÇÃO SINTÁTICA E SEMÂNTICA

Iwai [20] apresenta um formalismo, denominado Gramáticas Adaptativas, no qual as regras gramaticais são criadas dinamicamente a partir do processamento da cadeia de entrada e de informações relacionadas ao contexto em que se apresentam, permitindo a identificação e resolução de ambiguidades semânticas. A autora demonstra, ainda, a equivalência computacional entre as Gramáticas Adaptativas e os autômatos adaptativos, o que permite que frases da linguagem gerada por uma gramática adaptativa sejam reconhecidas por autômatos adaptativos.

O Linguístico incorpora o formalismo proposto pela autora, através do mapeamento dos sintagmas e padrões oracionais de Luft para uma gramática adaptativa. As ações adaptativas são usadas para modificar as regras de produção em função do contexto definido pelo tipo do verbo encontrado no texto e pela presença de conjunções ou pronomes relativos. O resultado final é o seguinte:

$G = (G^0, T, R^0)$, onde:

$G^0 = (VN^0, VT, VC, PL^0, PD^0, S)$

R^0 = relação que associa regras de produção às ações adaptativas.

VN^0 = {S, con, SS, SV, SS, Sadj, SP,num, PrA, Sc, Sp, PrPes, Neg, Aux, PreV,V, Vlig ,Vtda,Vtdna , Vtdi, Vtpred ,Vti ,Vi, Sadv, Prep, Ind, ArtDef, ArtInd, Dem, Pos, PrRel, Adv, Adj, pont }

VT = {{léxico},{,}, ; , . ,:,!,?,1, 2, 3, 4, 5, 6, 7,8, 9, 0}

$VC = \varnothing$

$PD^0 = \varnothing$

PL^0 = { S → [con] [PrRel] [SS] SV pont [S]

SS → [Sadj] SS [Sadj | SP] [S] | S

SS → (([num | PrA] Sc) | Sp | PrPes)

SV→ [Neg] [Aux | PreV] V [SS | Sadj | Sadv | SP] | ((SS SP) | (SS Sadj) | (SP SP) | (SS SS) | (SS Sadv) | (SS SP SP))

V → (Vlig | Vtda | Vtdna | Vtdi | Vtpred |Vti |Vi)

SP → Prep (SS | Sadj)

Sadj → Sadj [SP] [S] | S

Sadj → [Adv] Adj [S]

Sadv → Sadv [SP] [S] | S

Sadv → [Adv] Adv [S]

PrA → [PrA] (Ind | ArtDef | ArtInd| Dem| Pos)

con → (aditiva | adversativa | alternativa | conclusiva| explicativa | integrante | causal | comparativa | concessiva | condicional | conformativa | consecutiva | final | proporcional | temporal)

num→ (0 |1 |2 | 3 |4 |5 |6 |7 |8 |9)| (cardinal| ordinal| multiplicativo| fracionario)

pont → (,| ; | . |: |! |?)}

Sendo:

S – Sentença
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
Pos – pronome possessivo
Ind – pronome indefinido
Neg – partícula (negação)
PrA – pronome adjetivo
PrPes – pronome pessoal
Prep – preposição
PreV – pré-verbais
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

A análise semântica das sentenças é feita de acordo com os termos processados, que podem gerar novas regras de produção e exclusão de regras existentes. Por exemplo, quando o Linguístico encontra um verbo transitivo indireto, ele dispara uma ação adaptativa A (Vti) que elimina regras iniciais definidas na gramática G^0 e cria outras, relacionadas aos complementos oracionais exigidos por este tipo de verbo. A representação destas alterações é feita da seguinte forma:

A (t= Vti) = L (t) = { + [R: $_G^0$ P26-66, 69,77-78, 87, 93, 101-102, 113-114:
$_G^1$V ← χ\V]
+ [$_{G1}$χV → Vti]
- [R: $_G^0$ P67-68,70-76,79-86, 88-100,103-112,115-120: $_G^1$∅]

A sintaxe da instrução R é a seguinte:
R: <$_G^n$> <P_{I-F}> : < $_G^{n+1}$ > P'
onde:
R: replace
$_G^n$: gramática anterior à atualização
P_{I-F}: regras de produção para atualização, sendo I = Inicio de intervalo e F= fim de intervalo. Acrescenta-se mais de um grupo de regras usando a vírgula para separação.
$_G^{n+1}$: gramática posterior à atualização
P': nova regra de produção
∅ : símbolo que indica que as regras assinaladas em P_{I-F} serão removidas

V ← χ\V: indica a aplicação de contexto ao não-terminal V

χV→Vti: indica a derivação de um não terminal V, dependente de contexto, em um não terminal Vti

O formalismo apresentado por Iwai é bastante prático e fornece suporte para incluir características adicionais que o *parser* pode utilizar para resolver problemas em que o contexto seja importante para definição das regras de produção. Por exemplo, o *parser* pode verificar se a frase analisada está correta ou resolver possíveis ambiguidades caso tenha à disposição informações sobre o tipo do verbo, tempo, modo e pessoa do verbo analisado. No exemplo abaixo, a ação adaptativa A utiliza como parâmetros de entrada a identificação do tipo do verbo, modo e pessoa para definição das funções adaptativas.

T = { A (verbo=Vlig, tempo = Presente, modo = Singular, pessoa = Primeira) = F (verbo, tempo, modo, pessoa) =
{ + [R: $_G^n$ P_{I-F}: $_G^{n+1}$V ← χV]
+ [$_G^{n+1}$χV → Vlig, Presente, Singular, Primeira]
+ [R: $_G^n$ P_{I-F}: $_G^{n+1}$∅] }

Analogamente, é possível incluir regras para análise da concordância nominal ou qualquer outro tipo de conteúdo semântico, sem necessidade de usar qualquer outro elemento fora do formalismo. As gramáticas adaptativas também podem ser usadas para representar o uso de informações probabilísticas na seleção das regras de produção. Tal técnica é usada quando existe mais de uma regra de produção aplicável e não existem informações sintáticas e semânticas suficientes para desambiguá-las. Neste caso, é possível usar a probabilidade de ocorrência das regras como fator de escolha e o formalismo de Iwai pode ser usado representar este tipo de informação contextual. No exemplo abaixo, a ação adaptativa A utiliza como parâmetro de entrada a probabilidade de ocorrência das regras de produção avaliadas e uma indicação para usar a regra de máxima probabilidade.

SVprob → V SP 10%
SVprob → V (SS SP) 90%

T = { A (t=SV, u = prob, v=max) = F (SV, prob, max) =
{ + [R: $_G^n$ P_{I-F}: $_G^{n+1}$SV ← χSVprob]
+ [$_G^{n+1}$ $χSV_{prob}$ → $χSV_{max}$]
? [$_G^{n+1}$ $χSV_{max}$]
+ [$_G^{n+1}$ $χSV_{max}$ → V (SS SP)]
+ [R: $_G^n$ P_{I-F}: $_G^{n+1}$∅] }

VI. EXPERIMENTOS E RESULTADOS COMPARATIVOS

Os módulos apresentados na seção IV foram desenvolvidos em linguagem Python, com o apoio do NLTK [21], uma biblioteca de processamento de linguagem natural. O Sentenciador e o *Token*izador foram implementados exatamente como haviam sido descritos previamente. Já o Identificador Morfológico e o Analisador Sintático foram implementados a partir do corpus CINTIL-Treebank [22] e das regras de produção da Teoria X Barra [23] na qual ele se apoia. O CINTIL-Treebank, corpus da Língua Portuguesa desenvolvido pela Universidade de Lisboa, foi escolhido nesta etapa da pesquisa por usar o mesmo sistema de notação da Universidade da Pennsylvania – Treebank, que é considerado padrão no processamento de Linguagem Natural, e por se apoiar em uma gramática estrutural e não descritiva, assim como a de Luft, fornecendo padrões oracionais equivalentes, sem perda efetiva do poder de análise. A mudança permitiu a realização de testes que não seriam possíveis com a gramática de Luft, pois ela não está disponível em corpus pré-anotados e, ainda, a comparação dos resultados de trabalhos publicados em diferentes línguas.

O algoritmo Viterbi descrito na seção IV, Fig. 2.10, foi implementado através de um mecanismo de *fall-back*, analisando trigramas, bigramas e unigramas, e assumindo a classificação mais comum do corpus, que é substantivo, caso não tenha encontrado a classificação do *token* analisado. O treino utilizou 90% do corpus CINTIL-Treebank e os testes foram feitos em 10%, escolhidos aleatoriamente. A Tabela 4 apresenta uma comparação dos resultados do Linguístico e dos analisadores morfológicos do estado da arte [24,25].

TABELA 4.
COMPARAÇÃO DE RESULTADOS

Sistema	Linguístico	TBL	TnT	MXPost	QTag
Precisão	91,41%	97,09%	96,87%	97,08%	89,97%

Observa-se que os resultados obtidos estão abaixo daqueles apresentados por analisadores morfológicos do estado da arte, que chegam a 97,09% de precisão. No entanto, é possível obter melhorias, visto que foi usado um corpus de treinamento de tamanho reduzido e também por não terem sido utilizados recursos semânticos e contextuais de desambiguação, que serão incorporados nos próximos experimentos.

O Reconhecedor Sintático utilizou as regras de produção capturadas do CINTIL-Treebank, enriquecidas com as probabilidades de ocorrências obtidas a partir de um subconjunto do mesmo corpus. As regras de produção foram extraídas diretamente do CINTIL-Treebank, usando bibliotecas do NLTK e realizando algumas adaptações, visando evitar a repetição de regras e incorporar o *encoding* usado na Língua Portuguesa (ISO-Latin-1). A Fig.5 apresenta um fragmento das regras obtidas.

AP -> A CONJP [0.00766]	VP -> V ADV [0.01065]
ADV_ -> PNT ADV_ PNT [0.05036]	A_ -> ADV A_ [0.01639]
S -> V_ AP [0.00019]	AP -> AP CP [0.00128]
AP -> ADV_ A [0.00766]	NP -> NP QNT [0.00003]

Figura 5 – Fragmento das Regras de Produção Extraídas do CINTIL – Treebank.

As regras de produção foram estruturadas de forma a apresentar as derivações dos não terminais em outros não terminais, seguidos da probabilidade calculada para a regra, em parênteses. Usando, como exemplo, o fragmento da Fig.5, o não terminal AP deriva para um não terminal A e outro não terminal CONJP com probabilidade de 0.00766 ou 0.766 %. As derivações de não terminais para terminais foram excluídas da gramática, sendo substituídas pelas classificações geradas pelo Identificador Morfológico.

Os resultados foram medidos utilizando o método Parseval [26], considerado padrão para apurar o grau de correção das árvores de derivação geradas por *parsers*. Os testes foram feitos com 1070 frases do corpus CINTIL-Treebank, escolhidas aleatoriamente, representando aproximadamente 10% do total. A Tabela 5 apresenta os resultados obtidos pelo Reconhecedor Sintático, comparando-o a outros *parsers*. O índice FParseval (média harmônica de precisão e cobertura) foi de 91,49%, melhor, portanto, que os resultados apresentados pelos *parsers* Bikel, Stanford e Berkeley aplicados ao mesmo corpus - 84,97%, 88,07% e 89,33%, respectivamente, chegando a níveis semelhantes aos obtidos com a língua inglesa [22]. Os resultados, no entanto, precisam ser relativizados pelo fato de tanto o treino quanto o teste terem sido realizados em um mesmo corpus, o que pode ter gerado algum tipo de viés, mesmo dividindo o corpus em subconjuntos diferentes para cada atividade.

TABELA 5.
COMPARAÇÃO DE RESULTADOS

Sistema	Linguístico	Bikel	Stanford	Berkeley
FParseval	91,49 %	84,97%	88,07%	89,33%,

VII. 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. Procurou-se também apresentar cenários em que a gramática utilizada pelo Linguístico é estendida, incorporando critérios semânticos para auxiliar na resolução de ambiguidades. Ao final, foram apresentados os experimentos realizados com o Linguístico, comparando os resultados com os do estado da arte.

O Identificador Morfológico apresentou resultados um pouco abaixo dos analisadores morfológicos de referência. Dado que foram os primeiros experimentos e que não houve uso de informações semânticas e contextuais para desambiguação, espera-se melhora com o desenvolvimento da pesquisa. Já o Reconhecedor Sintático apresentou resultados melhores do que os *parsers* do estado da arte aplicados a textos em português e chegou a níveis semelhantes aos apresentados em análises de textos em inglês.

Os resultados observados nos encorajam a avançar na investigação, testando o Linguístico com outros corpus e implementando melhorias já identificadas nesta etapa do trabalho. Espera-se também ampliar o escopo da pesquisa, com a inclusão de aspectos de dependência do contexto, equivalências entre gramáticas, dialetos e regionalismos, estendendo o uso das técnicas e métodos adaptativos no processamento de linguagem natural.

REFERÊNCIAS

[1] http://www.pcs.usp.br/~lta/

[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. 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] M. G. V.Nunes 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] http://www.linguateca.pt/

[14] http://www.nilc.icmc.usp.br/tep2/

[15] http://www.portalsaofrancisco.com.br/alfa/materias/index-lingua-portuguesa.php/

[16] Basilio, M. Formação e Classes de Palavras no Português do Brasil. Ed.Contexto, 2004.

[17] Basilio, M. Teoria Lexical. Ed.Atica, 1987.

[18] http://www.cs.columbia.edu/~mcollins/hmms-spring2013.pdf/

[19] Forney, G.D. J. IEEE. Proceedings of the IEEE, Volume 61, Issue 3, 1973.

[20] Iwai, M.: Um Formalismo Gramatical para Linguagens Dependentes de Contexto. São Paulo, 2000. Escola Politécnica da Universidade de São Paulo. Tese de Doutorado (2000).

[21] Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

[22] Branco, António, Francisco Costa, João Silva, Sara Silveira, Sérgio Castro, Mariana Avelãs, Clara Pinto and João Graça, 2010, "Developing a Deep Linguistic Databank Supporting a Collection of Treebanks: the CINTIL DeepGramBank ", In Proceedings, LREC2010 - The 7th international conference on Language Resources and Evaluation, La Valleta, Malta, May 19-21, 2010.

[23] Mioto C., Silva M. C. F., Lopes, R. Novo Manual de Sintaxe. Ed. Contexto. 2013.

[24] Branco, A., Silva, J.: Evaluating solutions for the rapid development of state-of-the-art POS taggers for Portuguese. In: Proceedings of the 4th Language Resources and Evaluation Conference (LREC). (2004) 507–510.

[25] Silva, J.: Shallow processing of Portuguese: From sentence chunking to nominal lemmatization. Master's thesis, University of Lisbon (2007) Published as Technical Report DI-FCUL-TR-07-16.

[26] Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Marcus, M., Santorini, B.: A procedure for quantitatively comparing the syntactic coverage of English grammars. In: Proceedings of the Workshop on the Evaluation of Parsing Systems. (1991) 306–311.

Ana Teresa Contier formou-se 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 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 atua no Laboratório de Dados da Serasa 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.

Agentes adaptativos reativos: formalização e estudo de caso

R. L. Stange, P. R. M. Cereda e J. José Neto

***Resumo*—Um dos objetivos da Inteligência Artificial é construir agentes. No contexto deste trabalho, um agente caracteriza-se como qualquer dispositivo que recebe estímulos de um ambiente e age sobre este ambiente para realização de uma tarefa. Existem diferentes teorias, arquiteturas e linguagens de agentes, com o objetivo de fazê-los desempenhar melhor suas tarefas. Este artigo apresenta uma formulação para um agente reativo com capacidade de automodificação, chamado de *agente reativo adaptativo*. Para isso, foi aplicado o conceito de adaptatividade ao agente, permitindo que este mude o seu comportamento baseado nas suas percepções. Para avaliar o desempenho do agente adaptativo, foi utilizado um estudo de caso para o problema do mundo de wumpus. Os resultados foram comparados a outros tipos de agentes, sendo o agente adaptativo mais vantajoso em alguns aspectos, tais como simplicidade e eficiência.**

***Palavras-chave:*—dispositivos adaptativos, agentes inteligentes, inteligência artificial**

I. Introdução

Agentes e sistemas multiagentes representam uma forma de analisar, projetar e implementar sistemas complexos de software. Pesquisadores e desenvolvedores de diferentes disciplinas têm abordado questões relacionadas a agentes, tais como na Inteligência Artificial [1], Programação Orientada a Objetos [2] e Interface Humano-Computador [3].

Assim como outros conceitos de computação, a definição de agentes sofre por não possuir um consenso universalmente aceito [4], [5]. Este trabalho tem o enfoque na Inteligência Artificial e adota, portanto, a definição e classificação de agentes baseadas em [1]. Russell e Norvig [1] definem um *agente* simplesmente como "algo que age". De acordo com esta definição, é possível vislumbrar um programa de computador como um agente. Entretanto, existem cenários nos quais é esperado que um agente computacional seja racional e opere de forma autônoma, percebendo seu ambiente e eventualmente sendo suscetível a mudanças. Considerando aspectos de racionalidade, um *agente racional* é aquele que realiza uma tarefa que considera correta de acordo com sua lógica, isto é, este age para alcançar os melhores resultados (ou o melhor resultado esperado) [1]. O estudo de *agentes inteligentes* é um dos temas centrais da Inteligência Artificial e seus conceitos podem ser aplicados desde sistemas relativamente pequenos (por exemplo, filtros de e-mail) até sistemas complexos (por exemplo, controle de tráfego aéreo) [4].

A *adaptatividade* é a característica atribuída ao comportamento automodificável de sistemas computacionais que ocorre em decorrência de estímulos de entrada e ao histórico de operação desses sistemas [6]. As pesquisas em adaptatividade investigam soluções para diversos problemas da computação, incluindo tomadas de decisão e aprendizado de máquina [7], [8] e processamento de linguagem natural [9]. O objetivo deste trabalho é aplicar adaptatividade em agentes, de forma que estes possam obter melhores resultados na execução de suas tarefas, sem alterar suas propriedades e características originais.

Este artigo está organizado da seguinte forma: a Seção II introduz o conceito de agentes, enfatizando a forma reativa. Posteriormente, aborda a estrutura de agentes, apresentando a organização elementar de projeto de agentes reativos utilizada neste trabalho. A proposta de um agente reativo com a incorporação da adaptatividade é formulada na Seção III. A Seção IV apresenta experimentos para a avaliação de um agente adaptativo reativo utilizando problema do mundo de wumpus como estudo de caso. As considerações finais sobre os resultados obtidos com a incorporação da adaptatividade em uma agente reativo estão na Seção V.

II. Estrutura de agentes

Um *agente* é definido como um dispositivo que percebe seu ambiente através de *sensores* e age sobre esse ambiente através de *atuadores* [1], conforme ilustra a Figura 1. A percepção refere-se às entradas perceptivas do agente e uma *sequência de percepções* consiste no histórico de todas as percepções até então. A escolha de uma ação pode depender tão somente da percepção atual ou da sequência de percepções obtida até um dado instante no tempo.

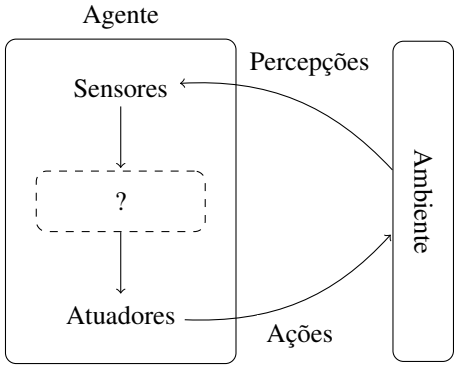


Figura 1. Agentes interagem com ambientes [Adaptada de [1]]

Em termos matemáticos, o comportamento do agente é dado

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: rlstange@usp.br, paulo.cereda@usp.br e jjneto@usp.br.

pela *função do agente*, que realiza o mapeamento de qualquer sequência de percepções em uma ação correspondente. O *programa do agente* é a implementação concreta desta função. O conjunto de sensores e atuadores necessários para que este programa seja executado em um dispositivo computacional é chamado de *arquitetura*. Assim, pode-se afirmar que um agente é composto por um programa e uma arquitetura.

Russell e Norvig [1] apresentam quatro tipo básicos de programas de agentes que possuem princípios subjacentes a quase todos os sistemas inteligentes:

1) *Agente reativo simples:* age em função da percepção atual, ignorando o histórico de percepções.
2) *Agente reativo baseados em modelo:* conserva um estado interno que depende do histórico de percepções e utiliza este modelo para tomar decisões.
3) *Agente baseado em objetivos:* utiliza o conceito da definição de objetivos a serem alcançados, de tal forma que o agente combina o modelo interno, idêntico ao agente baseado em modelos, com informações acerca dos objetivos para escolher as ações que levem a atingi-los.
4) *Agente baseado em utilidade:* utiliza um termo chamado *utilidade* para mensurar o quão satisfatória foi a solução encontrada para atingir um objetivo. A função de utilidade é essencialmente a internalização da medida de desempenho do agente.

É importante destacar que os quatro tipos básicos de programas de agentes apresentados compartilham elementos comuns.

A. Agentes reativos

A função do *agente reativo* é simplesmente realizar o mapeamento de uma percepção de entrada em uma ação de saída. Em diversas aplicações na área de robótica, a ausência de um estado interno que representa o histórico das percepções pode ser limitante; em outras áreas, como jogos, tal ausência não é impactante e pode tornar o agente mais eficiente [10].

A vantagem dos agentes reativos concentra-se no fato de que as regras do tipo $\langle$*condição* $\mapsto$ *ação*$\rangle$ fornecem uma representação inteligível, modular e eficiente. Porém, não há uma representação explícita do conhecimento, além da ausência de representação interna simbólica do ambiente e memória das ações.

Máquinas de estados finitos constituem um método simplificado e prático para a construção de agentes reativos. Tais máquinas são inerentemente modulares, permitindo a construção de uma biblioteca de comportamentos primários juntamente com transições, baseadas em combinações condicionais. O comportamento pode então ser ligado pelas transições usando uma representação gráfica, como um autômato [10]. Porém, apesar da simplicidade conceitual dos métodos baseados em máquinas de estados finitos, a representação interna pode tornar-se inviável com o tempo e de acordo com a quantidade das informações processadas. Em média, a manutenção atinge $\mathcal{O}(n^2)$ transições entre estados.

Para a implementação de um programa de agente reativo, uma abordagem geral e flexível é, em um primeiro momento, construir um interpretador de regras $\langle$*condição* $\mapsto$ *ação*$\rangle$. Em seguida, cria-se um conjunto de regras para ambientes de tarefas específicas.

III. Agentes reativos adaptativos

Esta seção apresenta a proposta de agentes adaptativos, em particular, na forma reativa. São introduzidas as definições de dispositivos adaptativos dirigidos por regras, bem como a organização de um agente adaptativo.

A. Dispositivos adaptativos

Um dispositivo é dito adaptativo quando este possui a capacidade de modificação de seu próprio comportamento, sem a interferência de agentes externos [11], [12]. Tal modificação espontânea ocorre em resposta ao histórico de operação e aos dados de entrada, resultando em comportamentos distintos ao longo do tempo [13]. A adaptatividade é adicionada como uma extensão aos formalismos já consolidados, aumentando seu poder de expressão [14], [15]. Os dispositivos adaptativos são definidos formalmente a seguir.

Definição 1 (dispositivo guiado por regras)**.** Um dispositivo guiado por regras é definido como $ND = (C, NR, S, c_0, A, NA)$, tal que ND é um dispositivo guiado por regras, C é o conjunto de todas as possíveis configurações, $c_0 \in C$ é a configuração inicial, S é o conjunto de todos os possíveis eventos que são estímulos de entrada do dispositivo, $\epsilon \in S$, $A \subseteq C$ é o subconjunto de todas as configurações de aceitação (da mesma forma, $F = C - A$ é o subconjunto das configurações de rejeição), NA é o conjunto de todos os possíveis símbolos de saída de ND como efeito da aplicação das regras do dispositivo, $\epsilon \in NA$, e NR é o conjunto de regras que definem ND através de uma relação $NR \subseteq C \times S \times C \times NA$. □

Definição 2 (regra do dispositivo)**.** Uma regra $r \in NR$ tem a forma $r = (c_i, s, c_j, z)$, $c_i, c_j \in C$, $s \in S$ e $z \in NA$, indicando que, em resposta a um estímulo s, r muda a configuração corrente c_i para c_j, consome s e gera z como saída [6]. Uma regra $r = (c_i, s, c_j, z)$ é dita compatível com a configuração corrente c se, e somente se, $c_i = c$ e s é vazio ou igual ao estímulo de entrada corrente; neste caso, a aplicação da regra r move o dispositivo para a configuração c_j, denotada por $c_i \Rightarrow^s c_j$, e adiciona z ao fluxo de saída. □

Definição 3 (aceitação de um fluxo de estímulos de entrada por um dispositivo)**.** Um fluxo de estímulos de entrada $w = w_1 w_2 \dots w_n$, $w_k \in S - \{\epsilon\}$, $k = 1, \dots, n$, $n \geq 0$, é aceito por um dispositivo ND quando $c_0 \Rightarrow^{w_1} c_1 \Rightarrow^{w_2} \dots \Rightarrow^{w_n} c$ (de modo resumido, $c_0 \Rightarrow^w c$), e $c \in A$. Da mesma forma, w é rejeitado por ND quando $c \in F$. A linguagem descrita pelo dispositivo guiado por regras ND é representada pelo conjunto $L(ND) = \{w \in S^* \mid c_0 \Rightarrow^w c, c \in A\}$. □

Definição 4 (dispositivo adaptativo guiado por regras)**.** Um dispositivo guiado por regras $AD = (ND_0, AM)$, tal que ND_0 é um dispositivo e AM é um mecanismo adaptativo, é considerado adaptativo sempre que, para todos os passos de operação $k \geq 0$ (k é o valor de um contador embutido T iniciado em zero e que é incrementado de uma unidade toda

vez que uma ação adaptativa não nula é executada), AD segue o comportamento do dispositivo subjacente ND_k até que a execução de uma ação adaptativa não nula inicie o passo de operação $k+1$ através de mudanças no conjunto de regras; em outras palavras, a execução uma ação adaptativa não nula em um passo de operação $k \geq 0$ faz o dispositivo adaptativo AD evoluir do dispositivo subjacente ND_k para ND_{k+1}. □

Definição 5 (operação do dispositivo adaptativo)**.** O dispositivo adaptativo AD inicia sua operação na configuração c_0, com o formato inicial definido por $AD_0 = (C_0, AR_0, S, c_0, A, NA, BA, AA)$. No passo k, um estímulo de entrada move AD para uma configuração seguinte e inicia seu passo de operação $k+1$ se, e somente se, uma ação não-adaptativa for executada; dessa forma, estando o dispositivo AD no passo k, com o formato $AD_k = (C_k, AR_k, S, c_k, A, NA, BA, AA)$, a execução de uma ação adaptativa não nula leva a $AD_{k+1} = (C_{k+1}, AR_{k+1}, S, c_{k+1}, A, NA, BA, AA)$, onde $AD = (ND_0, AM)$ é um dispositivo adaptativo com um dispositivo subjacente inicial ND_0 e um mecanismo adaptativo AM, ND_k é o dispositivo subjacente de AD no passo de operação k, NR_k é o conjunto de regras não adaptativas de ND_k, C_k é o conjunto de todas as configurações possíveis para ND no passo de operação k, $c_k \in C_k$ é a configuração inicial no passo k, S é o conjunto de todos os eventos possíveis que são estímulos de entrada para AD, $A \subseteq C$ é o subconjunto as configurações de aceitação (da mesma forma, $F = C - A$ é o subconjunto de configurações de rejeição), BA e AA são conjuntos de ações adaptativas (ambos contendo a ação nula, $\epsilon \in BA \cap AA$), NA, com $\epsilon \in NA$, é o conjunto de todos os possíveis símbolos de saída de AD como efeito da aplicação de regras do dispositivo, AR_k é o conjunto de regras adaptativas definido pela relação $AR_k \subseteq BA \times C \times S \times C \times NA \times AA$, com AR_0 definindo o comportamento inicial de AD, AR é o conjunto de todas as possíveis regras adaptativas para AD, NR é o conjunto de todas as possíveis regras não-adaptativas subjacentes de AD, e AM é o mecanismo adaptativo, $AM \subseteq BA \times NR \times AA$, a ser aplicado em um passo de operação k para cada regra em $NR_k \subseteq NR$. □

Definição 6 (regras adaptativas do dispositivo)**.** Regras adaptativas $ar \in AR_k$ são da forma $ar = (ba, c_i, s, c_j, z, aa)$ indicando que, em resposta a um estímulo de entrada $s \in S$, ar inicialmente executa a ação adaptativa anterior $ba \in BA$; a execução de ba é cancelada se esta elimina ar do conjunto AR_k; caso contrário, a regra não-adaptativa subjacente $nr = (c_i, s, c_j, z)$, $nr \in NR_k$ é aplicada e, finalmente, a ação adaptativa posterior $aa \in AA$ é executada [6]. □

Definição 7 (função adaptativa)**.** Ações adaptativas podem ser definidas em termos de abstrações chamadas funções adaptativas, de modo similar às chamadas de funções em linguagens de programação usuais [6]. A especificação de uma função adaptativa deve incluir os seguintes elementos: *(a)* um nome simbólico, *(b)* parâmetros formais que referenciarão valores passados como argumentos, *(c)* variáveis que conterão valores de uma aplicação de uma ação elementar de inspeção, *(d)* geradores que referenciam valores novos a cada utilização, e *(e)* o corpo da função propriamente dita. □

Definição 8 (ações adaptativas elementares)**.** São definidos três tipos de ações adaptativas elementares que realizam testes nas regras ou modificam regras existentes, a saber:

1) *ação adaptativa elementar de inspeção:* a ação não modifica o conjunto de regras, mas permite a inspeção deste para a verificação de regras que obedeçam um determinado padrão.
2) *ação adaptativa elementar de remoção:* a ação remove regras que correspondem a um determinado padrão do conjunto corrente de regras.
3) *ação adaptativa elementar de inclusão:* a ação insere uma regra que corresponde a um determinado padrão no conjunto corrente de regras.

Tais ações adaptativas elementares podem ser utilizadas no corpo de uma função adaptativa, incluindo padrões de regras que utilizem parâmetros formais, variáveis e geradores disponíveis. □

Observe que o mecanismo adaptativo pode ser visto como uma simples extensão do dispositivo não-adaptativo subjacente, o que preserva a integridade e as propriedades deste [6].

B. Agentes adaptativos

Um agente adaptativo é um dispositivo que observa seu ambiente através de sensores e age no próprio ambiente através de atuadores, de acordo com uma função f que determina seu comportamento, podendo o conjunto de regras de f modificar-se espontaneamente ao longo do tempo, em resposta ao histórico de operação e às percepções. A função f do agente realiza o mapeamento de um conjunto de percepções em uma ação [1], tal que $f\colon 2^P \mapsto AC$, no qual P é o conjunto de percepções e AC é o conjunto de ações. A modificação do conjunto de regras da função do agente ao longo do tempo ocorre por meio do mecanismo adaptativo, disparada pela execução de regras adaptativas.

Definição 9 (agente adaptativo)**.** Um agente adaptativo é definido como $AA = (A_0, AM)$, tal que A_0 é um agente e AM é um mecanismo adaptativo. Para todos os passos de operação $k \geq 0$ (k é o valor de um contador embutido T iniciado em zero e que é incrementado de uma unidade toda vez que uma ação adaptativa não nula é executada), AA segue o comportamento do agente subjacente A_k até que a execução de uma ação adaptativa não nula inicie o passo de operação $k+1$ através de alterações no conjunto de regras da função f do agente. Se o dispositivo subjacente A_0 possuir características adaptativas, o agente AA é classificado como adaptativo multinível. □

A Figura 2 apresenta uma possível organização de um agente adaptativo, estendida a partir do modelo proposto por Russell e Norvig [1]. Observe que o agente adaptativo é formado por um agente convencional acrescido de um mecanismo adaptativo.

De acordo com a Figura 2, o mecanismo adaptativo é representado na forma de um tratador de ações adaptativas.

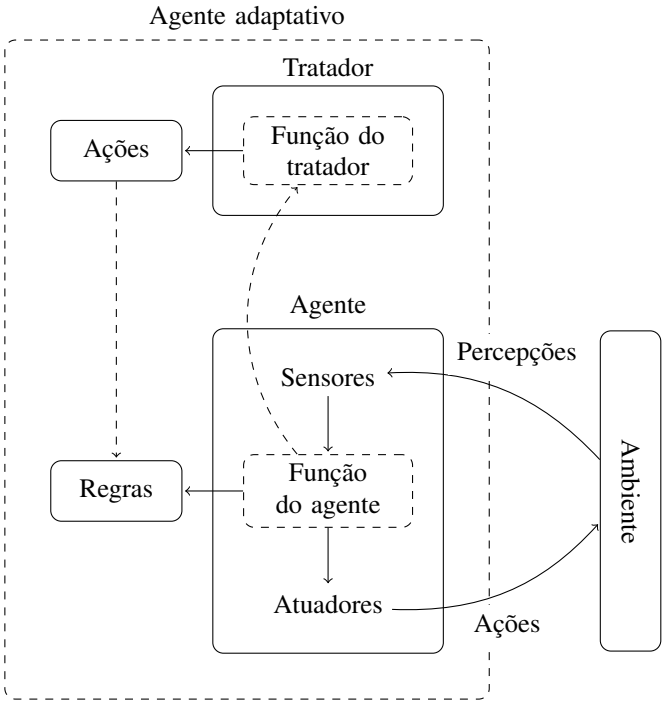


Figura 2. Organização de um agente adaptativo, estendida a partir do modelo proposto por Russell e Norvig [1].

Este intercepta pedidos de tratamento a partir da função do agente subjacente e realiza as modificações de acordo com padrões estabelecidos previamente em seu conjunto de ações. A modificação do conjunto de regras da função f do agente, através da execução de ações adaptativas, proporciona a evolução do agente subjacente no tempo, conforme ilustra a Figura 3.

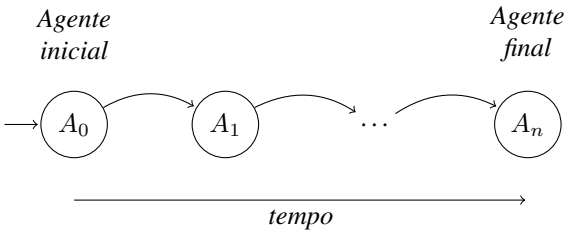


Figura 3. Evoluções sucessivas do agente subjacente no tempo, através da efetivação de ações adaptativas.

O conjunto de regras da função f do agente é acrescido de ações adaptativas anteriores e posteriores, conforme ilustra a Figura 4. Tais ações disparam pedidos de tratamento ao componente tratador do agente, que efetivamente realiza as modificações no conjunto de regras.

A Figura 4 apresenta partes hachuradas que indicam a existência de ações adaptativas associadas às regras correspondentes. No exemplo, a regra 1 possui apenas uma ação adaptativa anterior, a regra 2 não possui ações adaptativas associadas, e a regra n possui duas ações adaptativas, uma anterior e uma posterior.

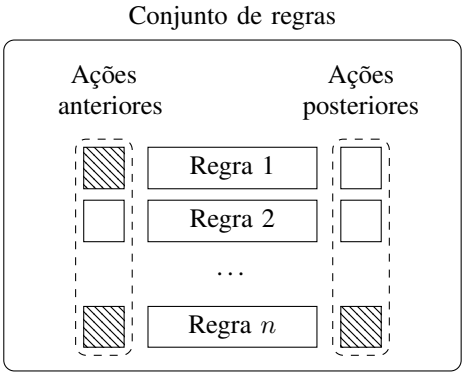


Figura 4. Formato de um conjunto de regras acrescido de ações adaptativas. As partes hachuradas indicam a existência de ações adaptativas associadas às regras correspondentes.

É possível utilizar adaptatividade multinível [16], [17], [18] na especificação de um agente, conforme ilustra a organização apresentada na Figura 5. Observe que um agente adaptativo de nível n possui n componentes tratadores, tal que um tratador $k+1$ intercepta pedidos de tratamento do nível k, hierarquicamente inferior, e realiza alterações no conjunto de regras ou ações disponível em tal nível. É importante notar que o agente encontra-se no nível 0 da organização do modelo.

A utilização de um agente adaptativo na resolução de um problema pode conferir expressividade ao modelo e simplificação da representação da lógica interna em relação ao seu correspondente não-adaptativo.

C. Agentes reativos como dispositivos subjacentes

A função do agente pode ser implementada utilizando-se estratégias básicas de projeto que refletem quais informações serão utilizadas no processo de tomada de decisão. O tipo mais simples é o agente reativo, que atua somente sobre a percepção corrente, ignorando o histórico de percepções recebidas; a tomada de decisão baseia-se estritamente em um conjunto de regras condicionais [1].

Definição 10 (agente reativo)**.** Um agente reativo é definido como $RA = (Q, P, AC, \Gamma, \delta)$, tal que Q é o conjunto de estados internos do agente, P é o conjunto de percepções, AC é o conjunto de ações, Γ é o mapeamento, $\Gamma\colon Q \times 2^P \mapsto Q$, e δ é uma função que mapeia regras em ações, $\delta\colon P \mapsto AC$. □

Agentes reativos possuem uma característica limitante: a tomada de decisão atua apenas sobre a percepção corrente, implicando na necessidade da observação total do ambiente; caso contrário, o agente potencialmente falhará [1]. A inclusão de uma camada adaptativa em um agente reativo permite tratar situações previstas, mas não esperadas no contexto corrente; através da execução de ações adaptativas, o agente tem seu conjunto de regras modificado, acomodando, portanto, um novo contexto a partir da observação do ambiente. A Figura 6 ilustra as modificações do conjunto de regras da função do agente reativo adaptativo ao longo do tempo para acomodar novos contextos.

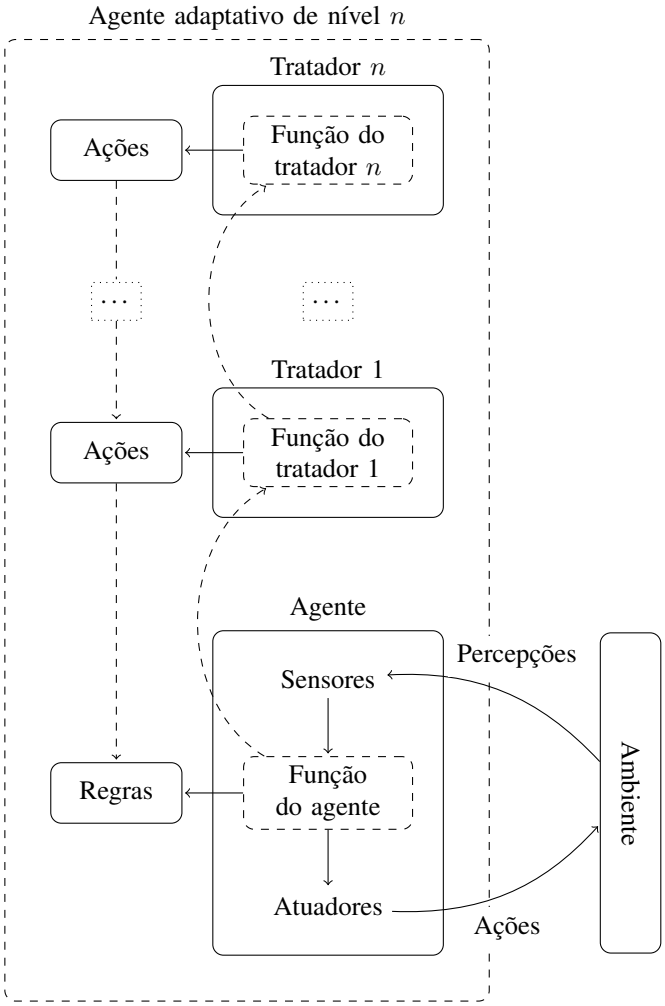


Figura 5. Organização de um agente adaptativo multinível, estendida a partir do modelo apresentado na Figura 2.

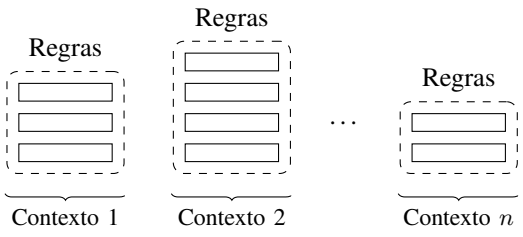


Figura 6. Modificações do conjunto de regras da função do agente reativo adaptativo ao longo do tempo para acomodar novos contextos.

Novos contextos proporcionam resolução incremental, através da divisão de um problema em subproblemas menores [19], [20]. A resolução de um problema em um espaço de abstração mais simples pode ser utilizada para o tratamento de outros problemas em níveis de abstrações superiores [21]. O processo é então repetido até que o problema original seja resolvido em seu espaço original de abstração [22]. Tal estratégia de resolução pode produzir reduções significativas no espaço de busca [23], [24], [25].

Observe que, dado um agente reativo simples, é possível adicionar uma camada adaptativa de tal forma q este continue reativo em sua estrutura, mas agindo por contexto na execução.

IV. Experimentos

Esta seção apresenta um estudo de caso para avaliação de um agente adaptativo reativo utilizando o tradicional problema do *mundo de wumpus*, uma variante de um jogo de computador [26] apresentada por Genesereth [1] como ambiente de testes para técnicas em inteligência artificial. O mundo de wumpus consiste em uma grade bidimensional contendo um número determinado de buracos, um monstro e um agente.

No mundo de wumpus, o agente inicia sua iteração na posição $(1, 1)$ da grade e sua tarefa consiste em evitar o monstro e os buracos, encontrar ouro e sair do ambiente pela mesma posição a qual iniciou. O agente pode perceber uma brisa em posições adjacentes aos buracos, um fedor em posições adjacentes ao monstro, e um brilho em posições adjacentes ao ouro. Existem variações do mundo de wumpus que permitem ao agente armar-se com uma ou mais flechas, podendo atirar no monstro quando encontrá-lo [27].

Em geral, os ambientes do mundo de wumpus permitem que o agente recupere o ouro e saia em segurança. Entretanto, existem situações em que o ouro estará inalcançável (por exemplo, dentro de um buraco) ou que o agente deverá enfrentar o monstro, podendo morrer no confronto [27], [1].

A. *Especificação do agente adaptativo reativo*

O agente adaptativo reativo proposto para o mundo de wumpus foi especificado de acordo com as seguintes regras iniciais:

- O agente pode locomover-se pelo ambiente, evitando buracos e o monstro, de acordo com a observação das percepções correntes.
- Ao perceber a existência do monstro em uma posição adjacente, o agente poderá optar por disparar uma flecha, na tentativa de matá-lo.
- Na percepção de um brilho em uma posição adjacente, o agente tentará recuperar o ouro.

O contexto inicial do agente prevê que este locomova-se e sobreviva no ambiente, e procure pelo ouro. Adicionalmente, as seguintes regras disparam ações adaptativas:

- Caso o agente opte por disparar uma flecha, na tentativa de matar o monstro, e este disparo for bem-sucedido, haverá uma percepção de um grito no ambiente, indicando que o monstro está morto. Neste caso, na percepção de um grito, uma ação adaptativa removerá a regra que trata da percepção de fedor.

- Caso o agente recupere o ouro, através da percepção de brilho, uma ação adaptativa removerá a própria regra que determina a recuperação do ouro e adicionará uma regra que determina a percepção da entrada do ambiente, permitindo que o agente saia pela mesma posição que entrou.

Observe que, ao matar o monstro, não existe mais a necessidade do agente em possuir a regra referente ao confronto (a menos, é claro, que o agente esteja atuando em uma variação do mundo de wumpus que admite a existência de mais monstros). Da mesma forma, não existe a necessidade de possuir a regra referente à percepção do ouro, dado que este já esteja recuperado pelo agente. A Figura 7 ilustra os possíveis contextos tratados pelo agente adaptativo reativo, de acordo com a execução das ações adaptativas.

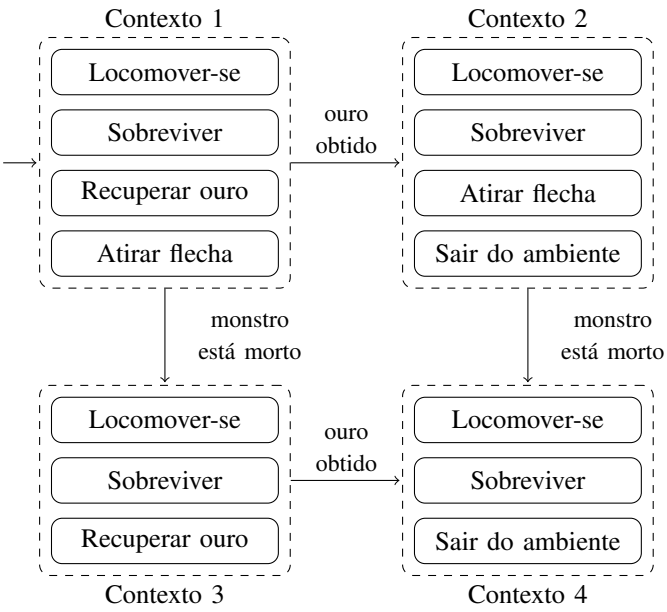


Figura 7. Possíveis contextos tratados pelo agente adaptativo reativo para o problema do mundo de wumpus.

É importante destacar que o conjunto de regras do agente adaptativo reativo representa o conhecimento necessário e suficiente para tratamento do contexto corrente. As ações adaptativas permitem que o conjunto de regras modifique-se ao longo do tempo para acomodar situações previstas, mas não esperadas.

B. Simulação do ambiente

A interação do agente adaptativo reativo no mundo de wumpus foi simulada utilizando-se um software de especificação e simulação de agentes desenvolvido por Jill Zimmerman[1] (Goucher College, Estados Unidos). Foram gerados 14 ambientes de dimensão 4×4, com diferentes quantidades e localizações de obstáculos, conforme ilustra a Figura 8.

O software de simulação utilizado para este experimento apresenta as seguintes medidas de desempenho da interação do agente com o ambiente [1], permitindo o cálculo da pontuação total ao final da simulação:

1) Bônus de 1000 pontos ao recuperar o ouro (tal ação é uma parte importante da tarefa do agente no ambiente).
2) Penalização de -1000 pontos quando o agente morre no ambiente. Esta situação ocorre quando o agente entra em uma posição contendo um buraco ou um monstro (ainda vivo).
3) Penalização de -1 ponto para cada ação tomada pelo agente no ambiente (mesmo que o resultado da ação seja um bônus).
4) Penalização de -10 pontos por utilizar uma flecha, na tentativa de matar o monstro.

O experimento consistiu na simulação da interação de quatro tipos de agentes – reativo simples, baseado em modelos, baseado em objetivos e adaptativo reativo – nos ambientes ilustrados na Figura 8, avaliando seus desempenhos globais. A simulação foi repetida 10 vezes para cada par ordenado $\langle \textit{agente}, \textit{ambiente} \rangle$, totalizando 560 execuções. Os resultados obtidos são exibidos na Figura 9.

De acordo com a Figura 9, é possível observar semelhanças na execução dos agentes em um mesmo ambiente. De modo particular, os agentes baseados em modelos e objetivos apresentaram execuções significativamente próximas. O agente adaptativo reativo apresentou uma execução relativamente melhor em relação ao agente reativo simples e, em alguns ambientes, teve pontuação próxima a dos agentes baseados em modelos e objetivos. Como melhoria, um refinamento das regras que determinam o trajeto a ser percorrido pode contribuir com a redução do número de passos do agente no ambiente e aumentar suas chances de sobrevivência.

V. Considerações finais

Este artigo apresentou uma proposta de organização de agentes adaptativos como uma alternativa viável às iniciativas existentes. A utilização de novos contextos através do fenômeno da adaptatividade apresenta benefícios importantes, tais como:

1) divisão de trabalho, permitindo que cada contexto tenha um comportamento específico dado um subproblema, reduzindo o espaço de regras para um subconjunto necessário e suficiente.
2) abstração, proporcionando representações distintas de acordo com o nível do subproblema tratado, sem entrar no mérito da representação do problema original.
3) reuso, de tal forma que contextos possam ser reaproveitados parcial ou totalmente, conforme a similaridade entre subproblemas.
4) facilidade de manutenção, na qual um contexto pode ter sua lógica interna atualizada sem comprometer os demais (desde que a integridade da resolução do subproblema seja preservada).

Agentes adaptativos apresentam conveniências na especificação de um conjunto de regras, organizando-as por contexto. Tal característica permite a simplificação do problema original, dividindo-o em subproblemas menores. Adicionalmente, a execução torna-se mais eficiente, dado que um contexto só

[1]Disponível em `http://phoenix.goucher.edu/~jillz/cs340/`.

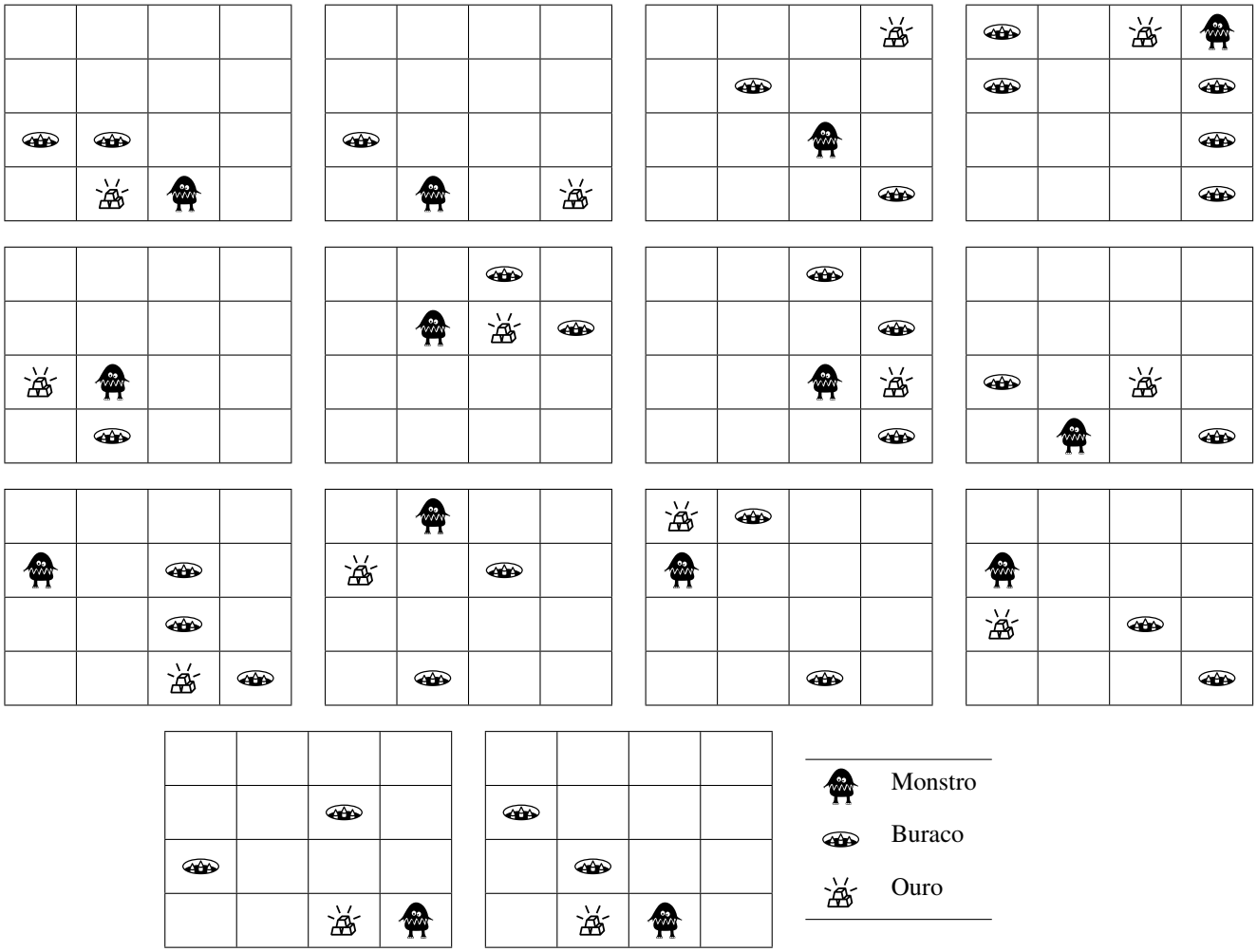


Figura 8. Ambientes gerados com diferentes quantidades e localizações de obstáculos para simulação da interação do agente adaptativo reativo no mundo de wumpus.

será ativado se efetivamente for necessário. Outras técnicas existentes podem ser combinadas [28], de forma a tratar especificamente cada subproblema.

REFERÊNCIAS

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Pearson Education, 2003.

[2] G. Booch, *Object-Oriented Analysis and Design with Applications*, 2nd ed., D. J. et al., Ed. Addison Wesley Longman, Inc, 1994.

[3] P. Maes, "Agents that reduce work and information overload," *Communications of the ACM*, vol. 37, no. 7, p. 31–40, 1994.

[4] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Autonomous Agents and Multi-Agent Systems*, vol. 1, pp. 7–38, 1998.

[5] A. Ahmad, M. S. Ahmad, and M. Z. Yusof, "An exploratory review of software agents," *International Symposium on Information Technology*, 2008.

[6] J. José Neto, "Adaptive rule-driven devices: general formulation and case study," in *International Conference on Implementation and Application of Automata*, 2001.

[7] T. Pedrazzi, A. H. Tchemra, and R. L. A. Rocha, "Adaptive decision tables – a case study of their application to decision-taking problems," in *Proceedings of International Conference on Adaptive and Natural Computer Algorithms – ICANNGA 2005*, 2005.

[8] H. Pistori and J. J. Neto, "Adaptree – proposta de um algoritmo para indução de Árvores de decisão baseado em técnicas adaptativas," in *Anais da Conferência Latinoamericana de Informática – CLEI 2002*, Montevideo, Uruguai, Novembro 2002.

[9] M. C. and J. José Neto, "Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos," in *V PROPOR – Encontro para o processamento computacional de Português falado e escrito*, 2000.

[10] G. M. Heckel, F. W. P.; Youngblood and N. S. Ketkar, "Representational complexity of reactive agents," in *IEEE Symposium on Computational Intelligence and Games (CIG)*, 2010, p. 257–264.

[11] J. José Neto, "Contribuições à metodologia de construção de compiladores," Thesis (Livre-docência), Escola Politécnica, Universidade de São Paulo, 1993.

[12] ——, "Adaptive automata for context-dependent languages," *SIGPLAN Notices*, vol. 29, no. 9, pp. 115–124, 1994.

[13] ——, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa," *IEEE Latin America Transactions*, vol. 5, pp. 496–505, 2007.

[14] H. Pistori, "Tecnologia em engenharia de computação: estado da arte e aplicações," PhD thesis, Escola Politécnica, Universidade de São Paulo, 2003.

[15] P. R. M. Cereda and J. José Neto, "Utilizando linguagens de programação orientadas a objetos para codificar programas adaptativos," in *Memórias do IX Workshop de Tecnologia Adaptativa – WTA 2015*, 2015, pp. 2–9.

[16] R. I. Silva Filho, R. L. A. Rocha, and R. H. G. Guiraldelli, *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence: Papers from the Ray Solomonoff 85th Memorial Conference, Melbourne, VIC, Australia, November 30 – December 2, 2011*. Springer Berlin Heidelberg, 2013, ch. Learning in the Limit: A Mutational and Adaptive Approach, pp. 106–118.

[17] R. I. Silva Filho and R. L. A. Rocha, *Adaptive and Natural Computing Algorithms: 10th International Conference, ICANNGA 2011, Ljubljana,*

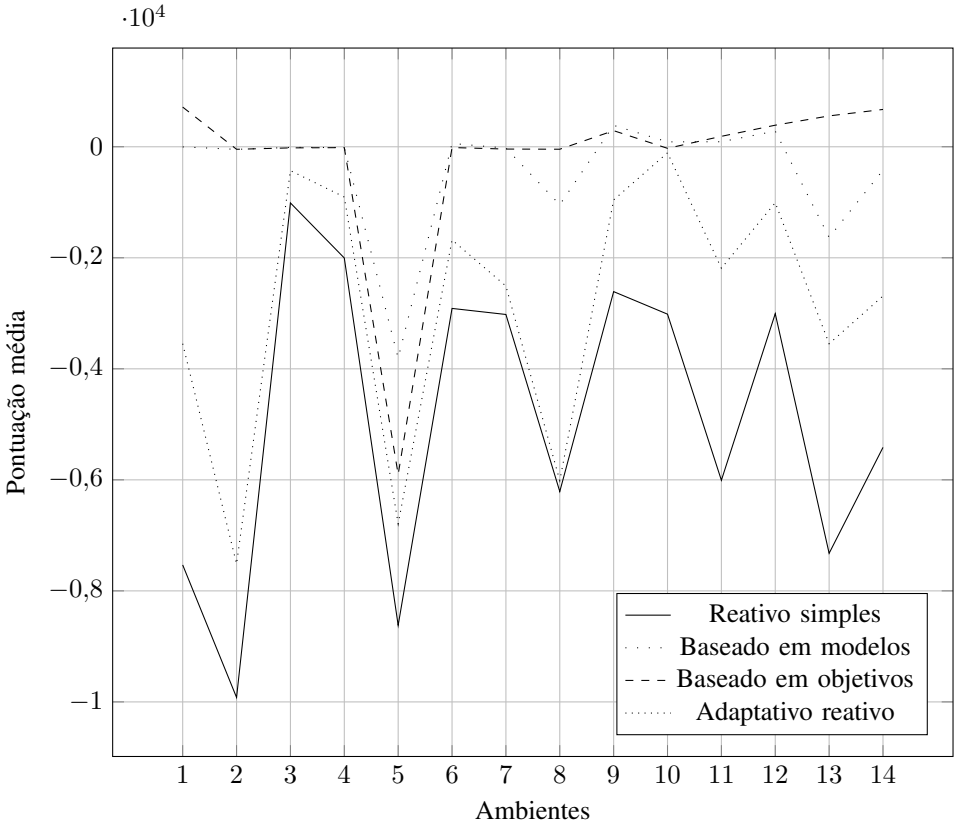


Figura 9. Pontuação média dos quatro agentes (reativo simples, baseado em modelos, baseado em objetivos e adaptativo reativo) em 14 ambientes do mundo de wumpus.

Slovenia, April 14-16, 2011, Proceedings, Part II. Springer Berlin Heidelberg, 2011, ch. Adaptive Finite Automaton: A New Algebraic Approach, pp. 275–284.

[18] R. I. Silva Filho, "Uma nova formulação algébrica para o autômato finito adaptativo de segunda ordem aplicada a um modelo de inferência indutiva," Tese de doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 2011.

[19] C. Knoblock, *Generating Abstraction Hierarchies: An Automated Approach to Reducing Search in Planning*. Springer US, 1993.

[20] Y. Anzai and H. A. Simon, "The theory of learning by doing," *Psychological Review*, no. 86, pp. 124–140, 1979.

[21] C. Knoblock, "Learning abstraction hierarchies for problem solving," in *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990, pp. 923–928.

[22] G. Pólya, *How to Solve It*. Princeton University Press, 1945.

[23] M. Minsky, *Computers and Thought*. New York, NY: McGraw-Hill, 1963, ch. Steps toward artificial intelligence, pp. 406–450.

[24] A. Newell and H. A. Simon, *Contemporary Approach to Creative Thinking*. Atherton Press, 1962, ch. The processes of creative thinking, pp. 63–119.

[25] C. Knoblock, "Search reduction in hierarchical problem solving," in *AAAI'91 Proceedings*, 1991, pp. 686–691.

[26] G. Yob, "Hunt the wumpus," *People's Computer Company*, vol. 2, no. 1, 1973.

[27] G. M. Khan, J. F. Miller, and D. M. Halliday, *Advances in modeling adaptive and cognitive systems*. UEFS, 2010, ch. Intelligent agents capable of developing memory of their environment, pp. 77–114.

[28] P. R. M. Cereda and J. José Neto, "Adaptive data mining: Preliminary studies," *IEEE Latin America Transactions*, vol. 12, no. 7, pp. 1258–1270, October 2014.

Renata Luiza Stange atualmente é doutoranda do Programa de Pós-Graduação em Engenharia de Computação do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, atuando como pesquisadora no Laboratório de Linguagens e Técnicas Adaptativas do PCS. Mestre em Ciências, também pelo Programa de Pós-Graduação em Engenharia de Computação do PCS/EPUSP (2011). Bacharel em Análise de Sistemas pela Universidade Estadual do Centro-Oeste (2002). Atua como docente na Universidade Tecnológica Federal do Paraná (UTFPR). Tem experiência na área de Ciência da Computação, particularmente em dispositivos adaptativos, tecnologia adaptativa e suas aplicações à inteligência artificial, aprendizagem de máquina e reconhecimento de padrões.

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 doutorando do Programa de Pós-Graduação em Engenharia de Computação do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, atuando como aluno pesquisador no Laboratório de Linguagens e Técnicas Adaptativas do PCS. Tem experiência na área de Ciência da Computação, com ênfase em Teoria da Computação, atuando principalmente nos seguintes temas: tecnologia adaptativa, autômatos adaptativos, dispositivos adaptativos, linguagens de programação e construção de compiladores.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de S ão Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Técnicas Adaptativas 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.

XML2AA: geração automática de autômatos adaptativos a partir de especificações XML

P. R. M. Cereda e J. José Neto

***Abstract*—Este artigo apresenta uma ferramenta para geração automática de autômatos adaptativos, utilizando a linguagem Java, a partir de especificações XML, de forma consistente e aderente à teoria. Aspectos técnicos do desenvolvimento da ferramenta são apresentados e discutidos, incluindo modos de execução e exemplos de especificações de autômatos, em formato XML, para reconhecimento de linguagens regulares, livres de contexto e dependentes de contexto.**

***Palavras-chave:*—Autômato adaptativo, linguagens de programação, bibliotecas, adaptatividade.**

I. Introdução

Este artigo apresenta uma ferramenta chamada XML2AA para geração automática de autômatos adaptativos, utilizando a linguagem Java, a partir de especificações XML, de forma consistente e aderente à teoria original proposta por José Neto [1]. Espera-se que esta ferramenta ofereça subsídios para a incorporação de elementos adaptativos em programas através de uma interface simplificada, além de atuar como material didático no ensino de autômatos adaptativos.

A organização deste artigo é a seguinte: a Seção II apresenta uma breve revisão bibliográfica da teoria. A Seção III apresenta aspectos técnicos da ferramenta, incluindo seus modos de operação, e discussões sobre implementação. Exemplos de especificações de autômatos, em formato XML, e execução no terminal de consulta integrado da ferramenta são contemplados na Seção IV. As considerações finais são apresentadas na Seção V.

II. Conceitos iniciais

O autômato adaptativo [2], [1] é um extensão do formalismo do autômato de pilha estruturado [1] que permite o reconhecimento de linguagens recursivamente enumeráveis. O termo *adaptativo*, neste contexto, pode ser definido como a capacidade de um dispositivo em alterar seu comportamento de forma espontânea. Um autômato adaptativo, portanto, tem como característica a possibilidade de alterar sua própria topologia durante o processo de reconhecimento de uma dada cadeia, em resposta a algum estímulo [3]. A possibilidade de alteração do autômato ocorre através da utilização de ações adaptativas, que lidam com situações esperadas, mas ainda não consideradas, detectadas na cadeia submetida para reconhecimento pelo autômato [4].

Definição 1 (espaço de autômatos)**.** Ao executar uma transição que contém uma ação adaptativa associada, o autômato sofre mudanças, obtendo-se uma nova configuração. Para a aceitação de uma determinada cadeia, o autômato percorrerá um caminho em um espaço de autômatos; haverá, portanto, um autômato E_0 que iniciará o reconhecimento de uma cadeia w, autômatos intermediários E_i que serão criados ao longo do reconhecimento, e um autômato final E_n que corresponde ao final do reconhecimento de w. Em outras palavras, seja a cadeia $w = \alpha_0\alpha_1 \ldots \alpha_n$; o autômato M descreverá um caminho de autômatos $\langle E_0, \alpha_0\rangle \rightarrow \langle E_1, \alpha_1\rangle \rightarrow \ldots \rightarrow \langle E_n, \alpha_n\rangle$, onde E_i representa um autômato correspondente à aceitação da subcadeia a_i. □

Definição 2 (autômato adaptativo)**.** Um autômato adaptativo M é definido como $M = (Q, A, \Sigma, \Gamma, P, Z_0, q_0, F, E, \Phi)$, onde Q é o conjunto finito não-vazio de estados, A é um conjunto de submáquinas, definidas a seguir, Σ é o alfabeto do autômato, correspondendo ao conjunto finito não vazio dos símbolos de entrada, Γ é o conjunto finito não-vazio de símbolos de pilha, armazenados na memória auxiliar do autômato, P é a relação de transição de estados, $q_0 \in Q$ é o estado inicial (da primeira submáquina), Z_0 é o símbolo marcador de pilha vazia, $F \subseteq Q$ é o conjunto dos estados de aceitação do autômato (da primeira submáquina), E representa o modelo de autômato utilizado (inicialmente, o reconhecimento inicia-se no autômato E_0 do caminho de autômatos), e Φ é o conjunto de funções adaptativas, aplicáveis às transições [1], [3]. □

Definição 3 (submáquina do autômato adaptativo)**.** Uma submáquina $a_i \in A$ do autômato adaptativo M é definida como $a_i = (Q_i, \Sigma_i, P_i, q_{i,0}, F_i, \Phi_i)$, onde $Q_i \subseteq Q$ é o conjunto de estados de a_i, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada de a_i, $q_{i,0}$ é o estado de entrada da sub-máquina a_i, $P_i \subseteq P$ é a relação de transição de estados de a_i, $F_i \subseteq F$ é o conjunto de estados de retorno da sub-máquina a_i, e $\Phi_i \subseteq \Phi$ é o conjunto de funções adaptativas aplicáveis. □

Definição 4 (relação de transição do autômato adaptativo)**.** A relação de transição P é definida como $P \subseteq (\Gamma \times Q \times \Sigma \times (\Phi \cup \{\epsilon\})) \times (\Gamma \times Q \times \Sigma \times (\Phi \times \{\epsilon\}))$, na forma $(\gamma g, e, s\alpha), \mathcal{B} \rightarrow (\gamma g', e', s'\alpha), \mathcal{A}$, onde e, e' são os estados corrente e de destino, repectivamente, s é o símbolo consumido, α é o restante da cadeia de entrada, g é o topo da pilha, g' é o novo topo da pilha, γ é o restante da pilha, $\mathcal{B}$ é uma função adaptativa anterior, e $\mathcal{A}$ é uma função adaptativa posterior. Uma configuração para um autômato adaptativo é um elemento de $E \times Q \times \Sigma^* \times \Gamma^*$, e uma relação entre configurações sucessivas $\vdash$ é definida como:

– *Consumo de símbolo:* $(E_i, q, \sigma w, uv) \vdash (E_i, p, \sigma' w, xv)$, com $p, q \in Q$, $u, x \in \Gamma$, $v \in \Gamma^*$, $\sigma, \sigma' \in \Sigma \cup \{\epsilon\}$, $w \in \Sigma^*$, se σ foi consumido pelo autômato, $x = u$, e

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: paulo.cereda@usp.br e jjneto@usp.br.

$(\gamma, q, \sigma\alpha) \rightarrow (\gamma, p, \sigma'\alpha) \in P$.

- *Chamada de sub-máquina:* $(E_i, q, w, uv) \vdash (E_i, r, w, xv)$, com $q, r \in Q$, $u \in \Gamma$, $v, x \in \Gamma^*$, $w \in \Sigma^*$, $x = pu$, com chamada da sub-máquina R, estado inicial r, retorno em p, e $(\gamma, q, \alpha) \rightarrow (\gamma p, r, \alpha) \in P$.
- *Retorno de sub-máquina:* $(E_i, q, w, uv) \vdash (E_i, p, w, v)$, com $p, q \in Q$, $u, x \in \Gamma$, $v \in \Gamma^*$, $w \in \Sigma^*$, $u = p$, com retorno de sub-máquina para p, e $(\gamma g, q, \alpha) \rightarrow (\gamma, g, \alpha) \in P$.
- *Funções adaptativas:* $(E_i, q, \sigma w, uv) \vdash^{\mathcal{B},\mathcal{A}} (E_{i+2}, p, \sigma' w, u'v)$, indicando $(E_i, q, \sigma w, uv) \rightarrow^{\mathcal{B}} (E_{i+1}, q, \sigma w, uv) \vdash (E_{i+1}, p, \sigma' w, u'v) \rightarrow^{\mathcal{A}} (E_{i+2}, p, \sigma' w, u'v)$, com $p, q \in Q$, $u, u' \in \Gamma$, $v \in \Gamma^*$, $w \in \Sigma^*$, se $(\gamma u, q, \sigma\alpha), \mathcal{B} \rightarrow (\gamma u, p, \sigma'\alpha), \mathcal{A} \in P$. □

Definição 5 (chamada de função adaptativa)**.** Uma chamada de função adaptativa $\mathcal{F}_i$ assume a forma $\mathcal{F}_i(\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,m})$, onde cada $\tau_{i,j}$ representa um argumento passado à função. A declaração de uma função adaptativa $\mathcal{F}_i$ com m parâmetros consiste de um cabeçalho da forma $\mathcal{F}_i(\Theta_{i,1}, \Theta_{i,2}, \ldots, \Theta_{i,m})$ e um corpo contendo nomes (uma lista de identificadores que representam objetos no escopo da função, incluindo variáveis e geradores) e ações (lista de ações adaptativas elementares precedida por uma ação adaptativa inicial e seguida por outra final). A inicialização de uma função adaptativa preenche os geradores e os parâmetros passados e, a seguir, as ações são efetivamente aplicadas. □

Definição 6 (ações adaptativas elementares)**.** São definidos três tipos de ações adaptativas elementares que realizam testes no conjunto de regras ou modificam regras existentes (relação de transição de estados P), a saber:

1) *ação adaptativa elementar de inspeção:* a ação não modifica o conjunto de regras, mas permite a inspeção deste para a verificação de regras que obedeçam um certo padrão.
2) *ação adaptativa elementar de remoção:* a ação remove regras que correspondem a um determinado padrão do conjunto corrente de regras.
3) *ação adaptativa elementar de inclusão:* a ação insere uma regra que corresponde a um determinado padrão no conjunto corrente de regras. □

Definição 7 (linguagem reconhecida por um autômato adaptativo)**.** A linguagem reconhecida por um autômato adaptativo M é dada por $L(M) = \{w \in \Sigma^* \mid (E_0, q_0, w, Z_0) \vdash^* (E_n, f, \epsilon, Z_0), f \in F\}$. □

III. Aspectos técnicos e implementação

A ferramenta `XML2AA` permite a transformação automática de uma especificação XML de um autômato adaptativo em uma instância da classe `AdaptiveAutomaton` da biblioteca `AA4J` [5], pronta para uso direto em aplicações ou para consulta de cadeias através de um terminal integrado. Esta seção apresenta os aspectos técnicos e discussões sobre implementação.

A. Descrição formal da especificação XML

A ferramenta `XML2AA` requer uma descrição formal acerca do vocabulário e estrutura da especificação XML de um autômato adaptativo. Tal conjunto de regras é definido na Figura 1, em formato DTD (acrônimo de *Document Type Declaration*).

```
<!DOCTYPE adaptiveAutomaton [
   <!ELEMENT adaptiveAutomaton
        (transitions,submachines,actions?)>
   <!ELEMENT transitions (transition+)>
   <!ELEMENT transition
        (preAdaptiveFunction?,postAdaptiveFunction?)>
   <!ATTLIST transition to CDATA #REQUIRED>
   <!ATTLIST transition from CDATA #REQUIRED>
   <!ATTLIST transition symbol CDATA #IMPLIED>
   <!ATTLIST transition call CDATA #IMPLIED>
   <!ELEMENT parameter (#PCDATA)>
   <!ELEMENT submachines (submachine+)>
   <!ELEMENT submachine (state+)>
   <!ATTLIST submachine name CDATA #REQUIRED>
   <!ATTLIST submachine main CDATA #FIXED "true">
   <!ELEMENT state EMPTY>
   <!ATTLIST state name CDATA #REQUIRED>
   <!ATTLIST state start CDATA #FIXED "true">
   <!ATTLIST state accepting CDATA #FIXED "true">
   <!ELEMENT actions (adaptiveAction+)>
   <!ELEMENT adaptiveAction
        (parameter*,variable*,generator*,action+)>
   <!ATTLIST adaptiveAction name CDATA #REQUIRED>
   <!ELEMENT variable (#PCDATA)>
   <!ELEMENT generator (#PCDATA)>
   <!ELEMENT action
        (preAdaptiveFunction?,postAdaptiveFunction?)>
   <!ATTLIST action type (add|remove|query) #REQUIRED>
   <!ATTLIST action to CDATA #REQUIRED>
   <!ATTLIST action from CDATA #REQUIRED>
   <!ATTLIST action symbol CDATA #IMPLIED>
   <!ATTLIST action call CDATA #IMPLIED>
   <!ELEMENT preAdaptiveFunction (parameter*)>
   <!ATTLIST preAdaptiveFunction name CDATA #REQUIRED>
   <!ELEMENT postAdaptiveFunction (parameter*)>
   <!ATTLIST postAdaptiveFunction name CDATA #REQUIRED>
]>
```

Figura 1. Vocabulário e estrutura da especificação XML de um autômato adaptativo, em formato DTD.

De acordo com a Figura 1, a especificação XML de um autômato adaptativo é considerada bem formada se esta é aderente às regras gerais XML e válida se esta obedece às regras DTD que definem as etiquetas e atributos correspondentes, bem como a estrutura sintática.

B. Etiqueta principal

A especificação XML de um autômato adaptativo deve iniciar-se com a etiqueta principal `<adaptiveAutomaton>`, que denota a descrição dos componentes do dispositivo propriamente dito. Respectivamente, a etiqueta `</adaptiveAutomaton>` conclui a especificação.

De acordo com a Figura 1, a especificação do autômato deve conter, ao menos, um conjunto de transições e um conjunto de submáquinas. O terceiro componente é opcional, descrevendo as ações adaptativas presentes no modelo. A Figura 2 apresenta uma estrutura geral para a especificação XML de um autômato adaptativo.

```
<adaptiveAutomaton>
  <transitions> ... </transitions>
  <submachines> ... </submachines>
  <actions> ... </actions>
</adaptiveAutomaton>
```

Figura 2. Estrutura geral para a especificação XML de um autômato adaptativo. Os componentes `<transitions>` e `<submachines>` são obrigatórios.

É importante destacar que tais componentes são suficientemente expressivos para representar reconhecedores de linguagens regulares, livres de contexto e dependentes de contexto.

C. Transições

Uma transição do autômato adaptativo é representada através da etiqueta `<transition>` dentro da etiqueta ascendente `<transitions>`. Esta possui atributos que denotam qual o tipo de transição representado. Eventualmente, a etiqueta é identificada como vazia (isto é, não possui elementos internos), a menos da presença de funções adaptativas associadas. A Tabela I apresenta os atributos da etiqueta `<transition>` e seus significados correspondentes.

Tabela I
ATRIBUTOS DA ETIQUETA `<transition>` E SEUS SIGNIFICADOS CORRESPONDENTES.

Atributo	Significado
`from`	estado de origem
`to`	estado de destino
`symbol`	símbolo a ser consumido
`call`	chamada de submáquina

Conforme ilustra a Tabela I, os atributos `from` e `to` representam os estados de origem e destino, respectivamente, e são elementos obrigatórios em toda transição do modelo. Transições em vazio omitem o atributo `symbol`. Não é possível incluir atributos de consumo de símbolo e chamada de submáquina em uma mesma transição. Os estados são representados por identificadores inteiros positivos.

Exemplo 1. Considere o trecho de autômato apresentado na Figura 3, contendo transições com consumo de símbolos, em vazio e chamada de submáquina. A Figura 4 ilustra a especificação XML correspondente.

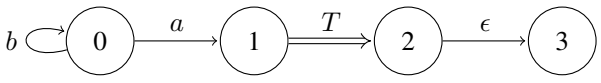


Figura 3. Trecho de autômato contendo transições com consumo de símbolos, em vazio e chamada de submáquina.

Observe que a etiqueta `<transition>` foi especificada em sua forma vazia. Entretanto, a forma longa da etiqueta é igualmente válida (e obrigatória quando existem funções adaptativas associadas). □

Uma transição pode conter funções adaptativas associadas. Para tal, utilizam-se as etiquetas internas `<preAdaptiveFunction>` e `<postAdaptiveFunction>`, denotando

```
<transitions>
  <transition from="0" symbol="a" to="1" />
  <transition from="0" symbol="b" to="0" />
  <transition from="1" call="T" to="2" />
  <transition from="2" to="3" />
</transitions>
```

Figura 4. Especificação XML das transições com consumo de símbolos, em vazio e chamada de submáquina do trecho de autômato da Figura 3.

as funções adaptativas anterior e posterior, respectivamente. Tais etiquetas requerem a presença do atributo `name` contendo o nome da função adaptativa a ser disparada. Eventualmente, as funções adaptativas possuem parâmetros associados, especificados através da sucessão de etiquetas `<parameter>` no corpo da etiqueta ascendente.

Exemplo 2. Considere o trecho de autômato apresentado na Figura 5, contendo transições com funções adaptativas associadas. A Figura 6 apresenta a especificação XML correspondente.

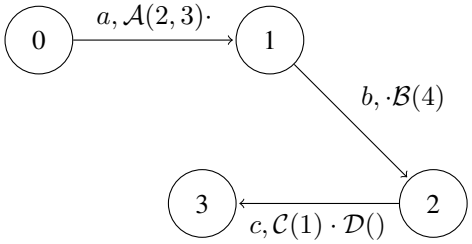


Figura 5. Trecho de autômato contendo transições com funções adaptativas associadas.

```
<transitions>

  <transition from="0" symbol="a" to="1">
    <preAdaptiveFunction name="A">
      <parameter>2</parameter>
      <parameter>3</parameter>
    </preAdaptiveFunction>
  </transition>

  <transition from="1" symbol="b" to="2">
    <postAdaptiveFunction name="B">
      <parameter>4</parameter>
    </postAdaptiveFunction>
  </transition>

  <transition from="2" symbol="c" to="3">
    <preAdaptiveFunction name="C">
      <parameter>1</parameter>
    </preAdaptiveFunction>
    <postAdaptiveFunction name="D" />
  </transition>

</transitions>
```

Figura 6. Especificação XML das transições com funções adaptativas associadas do trecho de autômato da Figura 5.

Observe que as etiquetas referentes às funções adaptativas

podem admitir formas vazias quando não existem parâmetros associados, isto é, nas chamadas de funções adaptativas que não sejam paramétricas. Cada transição admite, no máximo, duas funções adaptativas associadas. □

D. Submáquinas

Uma submáquina do autômato adaptativo é representada através da etiqueta `<submachine>` dentro da etiqueta ascendente `<submachines>`. Esta requer a presença do atributo `name` contendo o nome associado a tal submáquina, utilizado como identificador para eventuais chamadas, e um conjunto de estados correspondentes. O modelo do autômato requer que, ao menos, uma submáquina seja definida.

A especificação da submáquina pode conter ainda um atributo opcional `main`, indicando que esta é a submáquina principal do autômato adaptativo. Tal atributo, se presente, deve possuir o valor esperado `true`. Apenas uma submáquina pode conter o atributo `main`. A Tabela II ilustra os atributos da etiqueta `<submachine>`, seus significados correspondentes e valores esperados, quando aplicáveis.

Tabela II
ATRIBUTOS DA ETIQUETA `<submachine>`, SEUS SIGNIFICADOS CORRESPONDENTES E VALORES ESPERADOS, QUANDO APLICÁVEIS.

Atributo	Significado	Valor esperado
`name`	nome da submáquina	—
`main`	submáquina principal	`true`

Um estado é representado através da etiqueta vazia `<state>` e possui atributos que fornecem informações adicionais no contexto da submáquina na qual este está inserido. A Tabela III apresenta os atributos da etiqueta `<state>`, seus significados correspondentes e valores esperados, quando aplicáveis.

Tabela III
ATRIBUTOS DA ETIQUETA `<state>`, SEUS SIGNIFICADOS CORRESPONDENTES E VALORES ESPERADOS, QUANDO APLICÁVEIS.

Atributo	Significado	Valor esperado
`name`	nome do estado	—
`start`	estado inicial	`true`
`accepting`	estado de aceitação	`true`

De acordo com a Tabela III, um estado deve possuir um nome associado e, opcionalmente, conter indicações se este é o estado inicial ou um dos estados de aceitação da submáquina a qual está inserido. Os nomes dos estados são representados por identificadores inteiros positivos.

Exemplo 3. Considere as duas submáquinas T e U apresentadas na Figura 7, com T definida como submáquina principal. A Figura 8 apresenta a especificação XML correspondente.

Observe que, por uma questão de organização do modelo, os conjuntos de estados Q_T e Q_U das submáquinas T e U, respectivamente, são disjuntos, isto é, $Q_T \cap Q_U = \emptyset$. □

E. Ações

As ações adaptativas do autômato adaptativo são representadas através um conjunto de etiquetas `<adaptiveAction>` dentro

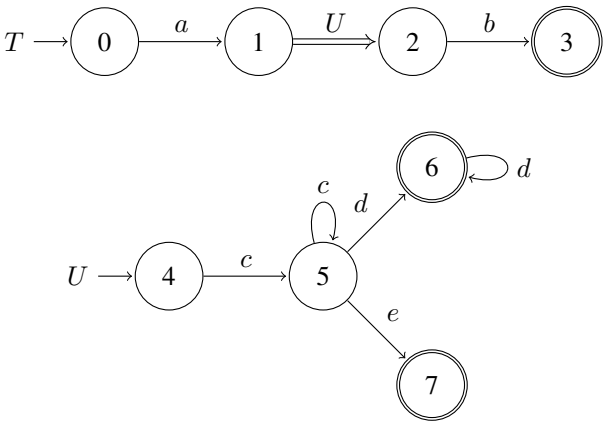


Figura 7. Submáquinas T e U. A submáquina T é definida como principal.

```
<submachines>

 <submachine name="T" main="true">
  <state name="0" start="true" />
  <state name="1" />
  <state name="2" />
  <state name="3" accepting="true" />
 </submachine>

 <submachine name="U">
  <state name="4" start="true" />
  <state name="5" />
  <state name="6" accepting="true" />
  <state name="7" accepting="true" />
 </submachine>

</submachines>
```

Figura 8. Especificação XML das submáquinas T e U da Figura 7. A submáquina T é definida como principal.

da etiqueta ascendente `<actions>`. Uma ação adaptativa requer a presença do atributo `name` contendo o nome associado a tal ação, utilizado como identificador para eventuais chamadas de funções adaptativas nas transições, e uma sequência não-vazia de ações adaptativas elementares.

Uma ação adaptativa elementar é representada através da etiqueta `<action>` e possui os mesmos atributos (Tabela I) e etiquetas descendentes associados a uma transição. Adicionalmente, uma ação adaptativa elementar requer o atributo `type`, que define o tipo de ação a ser aplicada, conforme ilustra a Tabela IV.

Tabela IV
VALORES PERMITIDOS PARA O ATRIBUTO `type` DA ETIQUETA `<action>`, REPRESENTANDO UMA AÇÃO ADAPTATIVA ELEMENTAR.

Valor	Significado
`query`	Realiza a inspeção do conjunto de regras
`remove`	Remove regras de acordo com um padrão
`add`	Insere regras que correspondem a um padrão

Exemplo 4. Considere o trecho de autômato da Figura 9, no qual uma função adaptativa $\mathcal{A}$ (Algoritmo 1) realiza alterações em sua topologia, removendo a transição $(1, b) \rightarrow 0$ e

inserindo uma nova transição $(0, a) \to 1$. A Figura 10 ilustra a especificação XML correspondente.

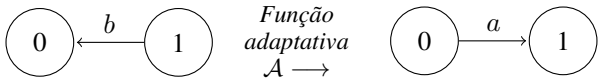


Figura 9. Trecho de autômato que ilustra uma função adaptativa removendo a transição $(1, b) \to 0$ e inserindo uma nova transição $(0, a) \to 1$.

Algoritmo 1 Função adaptativa $\mathcal{A}$

função adaptativa $\mathcal{A}$
$-(1, b) \to 0$
$+(0, a) \to 1$
fim da função adaptativa

```
<actions>

 <adaptiveAction name="A">
  <action type="remove" from="1" symbol="b" to="0" />
  <action type="add" from="0" symbol="a" to="1" />
 </adaptiveAction>

</actions>
```

Figura 10. Especificação XML da função adaptativa $\mathcal{A}$ do Algoritmo 1.

É importante observar que a especificação XML não admite uma sequência vazia de ações adaptativas elementares. □

Eventualmente, uma ação adaptativa pode conter parâmetros, variáveis e geradores. Tais elementos são disponibilizados de acordo com a especificação apresentada na Tabela V. Observe que as etiquetas não possuem atributos associados.

Tabela V
PARÂMETROS, VARIÁVEIS E GERADORES, DEFINIDOS COM SUAS ETIQUETAS E REGRAS DE FORMAÇÃO CORRESPONDENTES.

Etiqueta	Significado	Regra de formação
`<parameter>`	Parâmetro	Forma livre (não há restrição)
`<variable>`	Variável	Símbolo deve iniciar com `?`
`<generator>`	Gerador	Símbolo deve terminar com `*`

É importante observar que, ainda que não haja restrição quanto ao nome de um parâmetro na definição de uma ação adaptativa, é interessante evitar nomes que possam causar colisões potenciais com o alfabeto da linguagem (por exemplo, a colisão entre um símbolo `a` e um parâmetro `a` resultará na preferência pelo parâmetro, que possui maior prioridade).

Exemplo 5. Considere o trecho de autômato da Figura 11, no qual uma função adaptativa paramétrica $\mathcal{B}$ (Algoritmo 2) realiza alterações em sua topologia, invertendo os estados de origem e destino passados como parâmetros à função. A Figura 12 ilustra a especificação XML correspondente.

É importante destacar que as variáveis necessitam de preenchimento através de ações adaptativas elementares de inspeção. A utilização de uma variável não preenchida por uma ação adaptativa elementar de remoção ou inclusão resultará em uma exceção na biblioteca `AA4J` subjacente. □

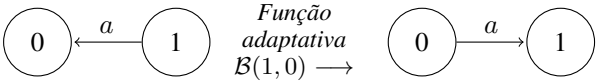


Figura 11. Trecho de autômato que ilustra uma função adaptativa invertendo os estados de origem e destino passados como parâmetros à função.

Algoritmo 2 Função adaptativa $\mathcal{B}(p_1, p_2)$

função adaptativa $\mathcal{B}(p_1, p_2)$
variáveis: $?x$
$?(p_1, ?x) \to p_2$
$-(p_1, ?x) \to p_2$
$+(p_2, ?x) \to p_1$
fim da função adaptativa

F. Modos de execução

A ferramenta `XML2AA` possui dois modos de operação. O primeiro é chamado *modo interativo*, no qual um arquivo XML é fornecido à ferramenta e um terminal de consulta para submissão de cadeias é disponibilizado ao usuário. A Figura 13 ilustra a organização do modo interativo. Observe que a biblioteca `AA4J` é uma dependência de execução.

De acordo com a Figura 13, a ferramenta `XML2AA` realiza o carregamento do arquivo XML e aplica transformações e validações na estrutura. Como resultado, uma instância do autômato adaptativo correspondente é gerada (utilizando a classe `AdaptiveAutomaton` da biblioteca `AA4J`) e um terminal de consulta para submissão de cadeias é disponibilizado ao usuário. A Figura 14 apresenta um exemplo de execução da ferramenta em modo interativo.

O terminal de consulta permite que o usuário submeta uma cadeia de símbolos ao autômato, através da ação `:check`, conforme ilustra a Figura 14. Como resultado, a ferramenta informa se a cadeia pertence à linguagem reconhecida pelo autômato, o tempo do processo de reconhecimento e o tipo de execução (determinística ou não-determinística). Adicionalmente, é possível obter uma representação gráfica da topologia do autômato, através da ação `:view` seguida de um índice positivo referente à topologia a ser visualizada. A ação `:quit` é utilizada para encerrar o terminal de consulta e o modo interativo da ferramenta.

O segundo modo de execução disponível é chamado *modo de biblioteca*, no qual a ferramenta é inserida no contexto

```
<actions>

 <adaptiveAction name="B">
  <parameter>p1</parameter>
  <parameter>p2</parameter>
  <variable>?x</variable>
  <action type="query" from="p1" symbol="?x" to="p2" />
  <action type="remove" from="p1" symbol="?x" to="p2" />
  <action type="add" from="p2" symbol="?x" to="p1" />
 </adaptiveAction>

</actions>
```

Figura 12. Especificação XML da função adaptativa $\mathcal{B}$ do Algoritmo 2.

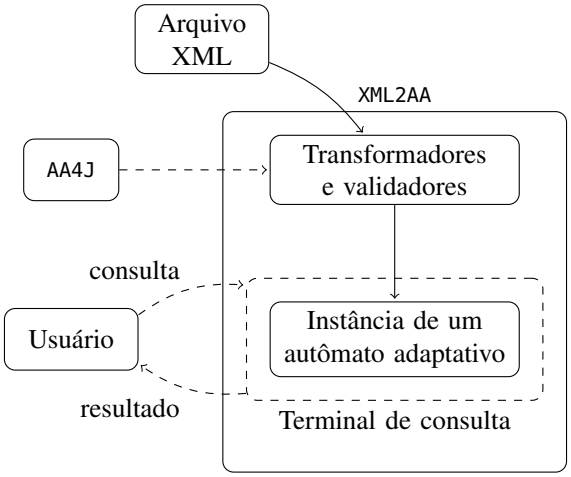


Figura 13. Organização da ferramenta XML2AA em modo interativo.

```
$ java -jar xml2aa.jar ape.xml
           _ ___
__ ___ __ | |_  )__ _ __ _
\ \ / '  \| |/ // _` / _` |
/_\_\_|_|_|_/___\__,_\__,_|

Laboratório de Linguagens e Técnicas Adaptativas
Escola Politécnica, Universidade de São Paulo
(versão 1.1)

O autômato possui 4 estados, 3 transições (2
transições com consumo de símbolo, 0 transições em
vazio, e 1 chamada de submáquina), 1 submáquina,
0 chamadas de funções adaptativas (0 funções
adaptativas anteriores, e 0 funções adaptativas
posteriores), e 0 ações adaptativas definidas.

Iniciando terminal, por favor, aguarde...
(pressione CTRL+C or digite ':quit' para sair do
    programa)

[1] consulta > :check ab
[1] resultado > cadeia aceita em 0:00:00.802
    (determinístico)

[2] consulta > :check aabb
[2] resultado > cadeia aceita em 0:00:00.024
    (determinístico)

[3] consulta > :quit

Isso é tudo, pessoal!
```

Figura 14. Exemplo de execução da ferramenta XML2AA em modo interativo.

de uma aplicação e atua como biblioteca propriamente dita, disponibilizando suas classes utilitárias para geração de uma instância de um autômato adaptativo (utilizando a classe `AdaptiveAutomaton` da biblioteca `AA4J`) a partir um arquivo XML contendo a especificação. A Figura 15 ilustra a organização do modo de biblioteca.

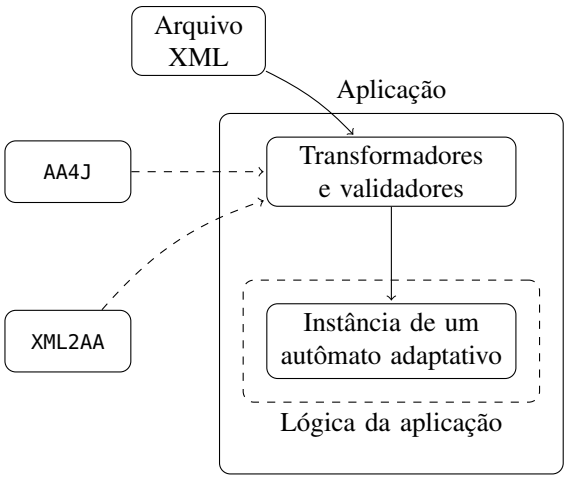


Figura 15. Organização da ferramenta XML2AA em modo de biblioteca.

De acordo com a Figura 15, a ferramenta XML2AA atua como biblioteca no contexto de uma aplicação e simplifica o processo de especificação de um autômato adaptativo diretamente em código Java (conforme exige a biblioteca AA4J), tornando-o transparente ao usuário, o qual deve apenas fornecer a especificação XML e tratar do modelo gerado em alto nível. A Figura 16 apresenta um exemplo da utilização da ferramenta em modo de biblioteca.

```
public class Main {
    public static void main(String[] args) {
        File file = new File("ape.xml");

        XMLTransformation transformation =
                                new XMLTransformation();
        XMLAdaptiveAutomaton xml = transformation.get(file);

        AutomatonValidator validator =
                                new AutomatonValidator(xml);
        validator.validate();

        AutomatonBuilder builder = new AutomatonBuilder();
        AdaptiveAutomaton automaton = builder.build(xml);
        ...
    }
}
```

Figura 16. Exemplo de utilização da ferramenta XML2AA em modo de biblioteca. Para fins didáticos, partes não essenciais do código Java foram omitidas.

A utilização da ferramenta em modo de biblioteca consiste, em linhas gerais, na transformação do arquivo XML em uma representação intermediária, validação dessa representação e construção efetiva da instância do autômato adaptativo correspondente. A biblioteca AA4J é uma dependência da ferramenta XML2AA e deve ser incluída no projeto da aplicação.

IV. Exemplos

Esta seção apresenta três exemplos de autômatos para ilustrar o funcionamento da ferramenta `XML2AA`. Todas as especificações XML correspondentes foram executadas em modo interativo.

Exemplo 6. Considere um autômato finito M_1 que reconhece cadeias pertencentes à linguagem regular $L_1 = \{w \in \{a,b\}^* \mid w = (ab)^+\}$, de acordo com a Figura 17. A especificação XML de tal autômato e a submissão das cadeias ab, $abab$ e aba são apresentadas nas Figuras 18 e 19, respectivamente. A janela de visualização da topologia do autômato finito M_1, obtida através da execução da ação `:view` da ferramenta, é apresentada na Figura 20.

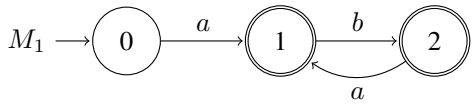


Figura 17. Autômato finito M_1 que reconhece cadeias pertencentes à linguagem regular $L = \{w \in \{a,b\}^* \mid w = (ab)^+\}$.

```
<adaptiveAutomaton>

 <transitions>
  <transition from="0" symbol="a" to="1" />
  <transition from="1" symbol="b" to="2" />
  <transition from="2" symbol="a" to="1" />
 </transitions>

 <submachines>
  <submachine name="M1" main="true">
   <state name="0" start="true" />
   <state name="1" />
   <state name="2" accepting="true"/>
  </submachine>
 </submachines>

</adaptiveAutomaton>
```

Figura 18. Especificação XML do autômato finito M_1 da Figura 17.

De acordo com a submissão de cadeias ilustrada na Figura 19, $ab, abab \in L(M_1)$ e $aba \notin L(M_1)$. □

Exemplo 7. Considere um autômato de pilha estruturado M_2 que reconhece cadeias pertencentes à linguagem livre de contexto $L_2 = \{w \in \{a,b\}^* \mid w = a^n b^n, n \in \mathbb{Z}\}$, de acordo com a Figura 21. A especificação XML de tal autômato e a submissão das cadeias ab, aab e $aabb$ são apresentadas nas Figuras 22 e 23, respectivamente. A janela de visualização da topologia do autômato de pilha estruturado M_2, obtida através da execução da ação `:view` da ferramenta, é apresentada na Figura 24.

De acordo com a submissão de cadeias ilustrada na Figura 23, $ab, aabb \in L(M_2)$ e $aab \notin L(M_2)$. □

Exemplo 8. Considere um autômato adaptativo M_3 que reconhece cadeias pertencentes à linguagem dependente de contexto $L_3 = \{w \in \{a,b,c\}^* \mid w = a^n b^n c^n, n \in \mathbb{Z}, n \geq 1\}$, de acordo com a Figura 25. A função adaptativa $\mathcal{A}$ é apresentada

```
[1] consulta > :check ab
[1] resultado > cadeia aceita em 0:00:00.461
    (determinístico)

[2] consulta > :check abab
[2] resultado > cadeia aceita em 0:00:00.013
    (determinístico)

[3] consulta > :check aba
[3] resultado > cadeia rejeitada em 0:00:00.011
    (determinístico)

[4] consulta > :view 1
[4] resultado > autômato visualizado em janela externa.
```

Figura 19. Submissão das cadeias ab, $abab$ e aba à instância do autômato finito M_1 gerado a partir da especificação XML da Figura 18.

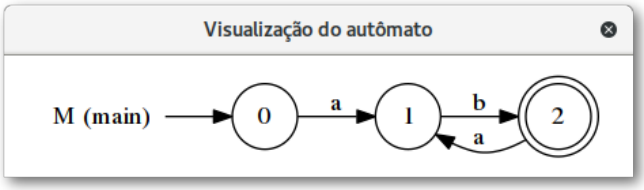


Figura 20. Visualização da topologia do autômato finito M_1.

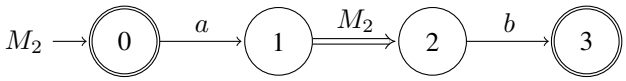


Figura 21. Autômato de pilha estruturado M_2 que reconhece cadeias pertencentes à linguagem livre de contexto $L_2 = \{w \in \{a,b\}^* \mid w = a^n b^n, n \in \mathbb{Z}\}$.

```
<adaptiveAutomaton>

 <transitions>
  <transition from="1" symbol="a" to="2" />
  <transition from="2" call="M2" to="3" />
  <transition from="3" symbol="b" to="4" />
 </transitions>

 <submachines>
  <submachine name="M2" main="true">
   <state name="1" start="true" accepting="true" />
   <state name="2" />
   <state name="3" />
   <state name="4" accepting="true"/>
  </submachine>
 </submachines>

</adaptiveAutomaton>
```

Figura 22. Especificação XML do autômato de pilha estruturado M_2 da Figura 21.

```
[1] consulta > :check ab
[1] resultado > cadeia aceita em 0:00:00.503
    (determinístico)

[2] consulta > :check aab
[2] resultado > cadeia rejeitada em 0:00:00.018
    (determinístico)

[3] consulta > :check aabb
[3] resultado > cadeia aceita em 0:00:00.016
    (determinístico)

[4] consulta > :view 1
[4] resultado > autômato visualizado em janela externa.
```

Figura 23. Submissão das cadeias ab, aab e $aabb$ à instância do autômato de pilha estruturado M_2 gerado a partir da especificação XML da Figura 22.

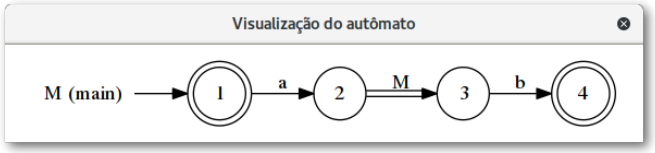


Figura 24. Visualização da topologia do autômato de pilha estruturado M_2.

no Algoritmo 3. A especificação de tal autômato e a submissão das cadeias $aabbc$, $aabbcc$ e $aaabbbccc$ são apresentadas nas Figuras 26 e 27, respectivamente. As janelas de visualização das topologias do autômato adaptativo M_3 referentes aos reconhecimentos das cadeias $aabbcc$ e $aaabbbccc$ são apresentadas nas Figuras 28 e 29, respectivamente.

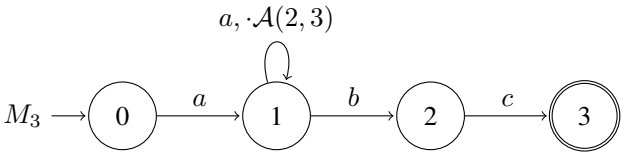


Figura 25. Autômato adaptativo M_3 que reconhece cadeias pertencentes à linguagem dependente de contexto $L_3 = \{w \in \{a,b,c\}^* \mid w = a^n b^n c^n, n \in \mathbb{Z}, n \geq 1\}$.

De acordo com a submissão de cadeias ilustrada na Figura 27, $aabbcc, aaabbbccc \in L(M_3)$ e $aabbc \notin L(M_3)$. □

V. Considerações finais

O código-fonte da ferramenta `XML2AA` está disponível para consulta e download em `https://github.com/cereda/xml2aa`, sob a licença GPLv3[1]. O bytecode gerado é compatível com Java 8 ou versões superiores. Espera-se que esta contribuição possa proporcionar subsídios para a implementação de programas que apresentem características adaptativas, através de uma especificação simplificada utilizando a notação XML, atuando como uma interface amigável à biblioteca `AA4J`.

[1]Disponível em `http://www.gnu.org/licenses/gpl-3.0.html`.

```
<adaptiveAutomaton>

 <transitions>
  <transition from="0" symbol="a" to="1"/>
  <transition from="1" symbol="b" to="2"/>
  <transition from="2" symbol="c" to="3"/>
  <transition from="1" symbol="a" to="1" >
   <postAdaptiveFunction name="A">
    <parameter>2</parameter>
    <parameter>3</parameter>
   </postAdaptiveFunction>
  </transition>
 </transitions>

 <submachines>
  <submachine name="M" main="true">
   <state name="0" start="true" />
   <state name="1" />
   <state name="2" />
   <state name="3" accepting="true"/>
  </submachine>
 </submachines>

 <actions>
  <adaptiveAction name="A">
   <parameter>p1</parameter>
   <parameter>p2</parameter>
   <variable>?x</variable>
   <variable>?y</variable>
   <generator>g1*</generator>
   <generator>g2*</generator>
   <action type="query" from="?x" symbol="b" to="p1"/>
   <action type="remove" from="?x" symbol="b" to="p1"/>
   <action type="query" from="?y" symbol="c" to="p2"/>
   <action type="remove" from="?y" symbol="c" to="p2"/>
   <action type="remove" from="1" symbol="a" to="1">
    <postAdaptiveFunction name="A">
     <parameter>p1</parameter>
     <parameter>p2</parameter>
    </postAdaptiveFunction>
   </action>
   <action type="add" from="?x" symbol="b" to="g1*"/>
   <action type="add" from="g1*" symbol="b" to="p1"/>
   <action type="add" from="?y" symbol="c" to="g2*"/>
   <action type="add" from="g2*" symbol="c" to="p2"/>
   <action type="add" from="1" symbol="a" to="1">
    <postAdaptiveFunction name="A">
     <parameter>g1*</parameter>
     <parameter>g2*</parameter>
    </postAdaptiveFunction>
   </action>
  </adaptiveAction>
 </actions>

</adaptiveAutomaton>
```

Figura 26. Especificação XML do autômato adaptativo M_3 da Figura 25.

Algoritmo 3 Função adaptativa $\mathcal{A}(p_1, p_2)$

função adaptativa $\mathcal{A}(p_1, p_2)$
variáveis: $?x, ?y$
geradores: $g_1^\star, g_2^\star$
$?(?x, b) \rightarrow p_1$
$-(?x, b) \rightarrow p_1$
$?(?y, c) \rightarrow p_2$
$-(?y, c) \rightarrow p_2$
$-(1, a) \rightarrow 1, \cdot\mathcal{A}(p_1, p_2)$
$+(?x, b) \rightarrow g_1^\star$
$+(g_1^\star, b) \rightarrow p_1$
$+(?y, c) \rightarrow g_2^\star$
$+(g_2^\star, c) \rightarrow p_2$
$+(1, a) \rightarrow 1, \cdot\mathcal{A}(g_1^\star, g_2^\star)$
fim da função adaptativa

```
[1] consulta > :check aabbc
[1] resultado > cadeia rejeitada em 0:00:00.463
    (determinístico)

[2] consulta > :check aabbcc
[2] resultado > cadeia aceita em 0:00:00.014
    (determinístico)

[3] consulta > :view 1
[3] resultado > autômato visualizado em janela externa.

[4] consulta > :view 2
[4] resultado > autômato visualizado em janela externa.

[5] consulta > :check aaabbbccc
[5] resultado > cadeia aceita em 0:00:00.030
    (determinístico)

[6] consulta > :view 1
[6] resultado > autômato visualizado em janela externa.

[7] consulta > :view 2
[7] resultado > autômato visualizado em janela externa.

[8] consulta > :view 3
[8] resultado > autômato visualizado em janela externa.
```

Figura 27. Submissão das cadeias $aabbc$, $aabbcc$ e $aaabbbccc$ à instância do autômato adaptativo M_3 gerado a partir da especificação XML da Figura 26.

A ferramenta apresentada neste artigo pode atuar como material didático no ensino de autômatos adaptativos, através de seu modo interativo, permitindo uma especificação em alto nível (utilizando a notação XML) e posterior submissão de cadeias no terminal de consulta. Adicionalmente, a ferramenta pode contribuir com a implementação de autômatos adaptativos utilizando a linguagem Java de forma consistente e aderente à teoria, através do modo de biblioteca.

AGRADECIMENTOS

Os autores agradecem ao professor Ítalo Santiago Vega, do Departamento de Ciência da Computação da Pontifícia Universidade Católica de São Paulo, pelas valiosas contribuições para o desenvolvimento da ferramenta e escrita do artigo.

REFERÊNCIAS

[1] J. José Neto, "Contribuições à metodologia de construção de compiladores," Tese de livre docência, Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

[2] J. José Neto and M. E. S. Magalhães, "Reconhecedores sintáticos: Uma alternativa didática para uso em cursos de engenharia," in *XIV Congresso Nacional de Informática*, 1981, pp. 171–181.

[3] J. José Neto, "Adaptive automata for context-sensitive languages," *SIGPLAN Notices*, vol. 29, no. 9, pp. 115–124, set 1994.

[4] ——, "Adaptive rule-driven devices: general formulation and case study," in *International Conference on Implementation and Application of Automata*, 2001.

[5] P. R. M. Cereda and J. José Neto, "AA4J: uma biblioteca para implementação de autômatos adaptativos," in *Memórias do X Workshop de Tecnologia Adaptativa – WTA 2016*, 2016, pp. 16–26.

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 doutorando do Programa de Pós-Graduação em Engenharia de Computação do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, atuando como aluno pesquisador no Laboratório de Linguagens e Técnicas Adaptativas do PCS. Tem experiência na área de Ciência da Computação, com ênfase em Teoria da Computação, atuando principalmente nos seguintes temas: tecnologia adaptativa, autômatos adaptativos, dispositivos adaptativos, linguagens de programação e construção de compiladores.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Técnicas Adaptativas 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.

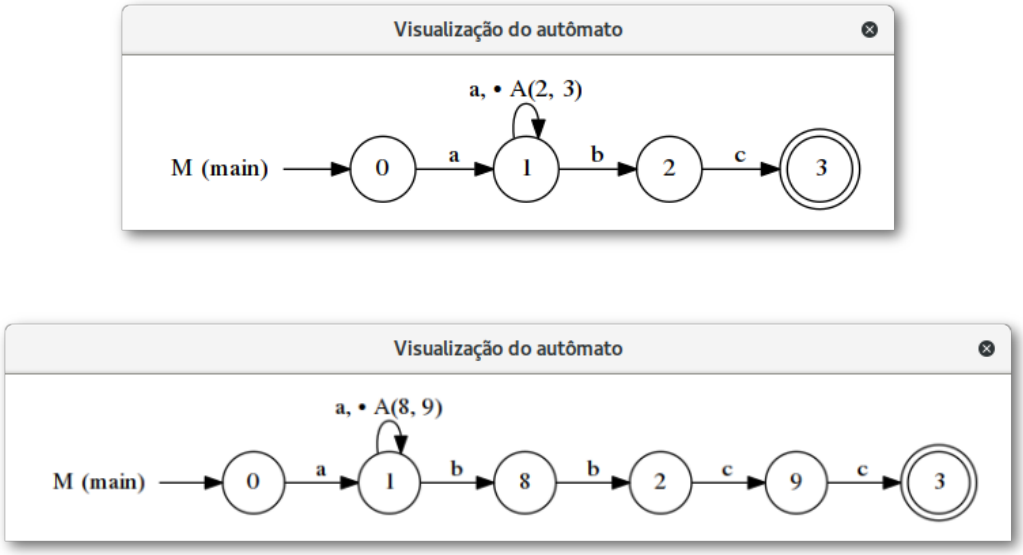


Figura 28. Visualização das topologias inicial e final do autômato adaptativo M_3, após o reconhecimento da cadeia $aabbcc$.

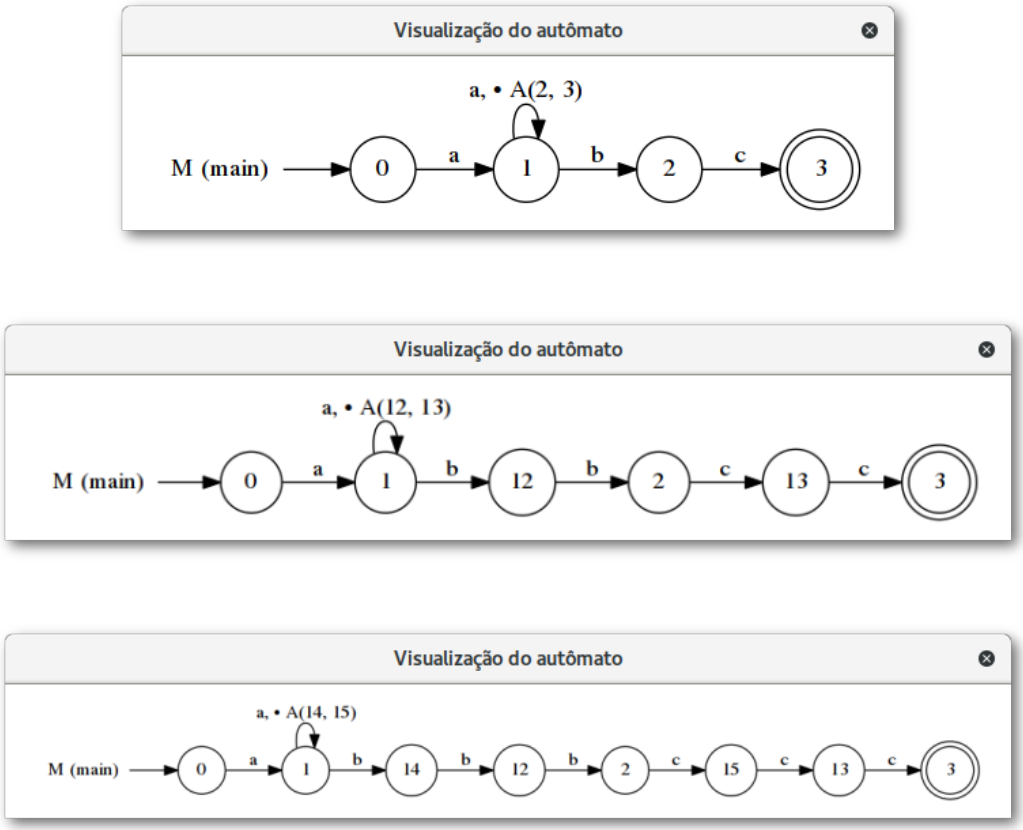


Figura 29. Visualização das topologias inicial, intermediária e final do autômato adaptativo M_3, após o reconhecimento da cadeia $aaabbbccc$.

Uso de técnicas adaptativas para manutenção da consistência de coleções baseada em igualdade

SOFIATO, B.*; ROCHA, R. L. A.*
*LTA – Laboratório de Linguagem e Técnicas Adaptativas - Escola Politécnica da USP
E-mail: bruno.sofiato@gmail.com, rlarocha@usp.br

***Resumo*— Linguagens modernas disponibilizam ao desenvolvedor um amplo leque de coleções e algoritmos altamente otimizados. Apesar de sua importância, algumas dessas bibliotecas de coleções apresentam uma falha importante: coleções baseadas em igualdade e coleções convencionais são intercambiáveis, o que potencialmente acarreta falhas, uma vez que a primeira não permite mutações de seus elementos. Devido ao seu grau de utilização, mudanças na hierarquia dessas coleções são inviáveis, uma vez que introduziriam incompatibilidades cuja correção seria custosa. Este artigo tem como objetivo o estudo do impacto da adoção de coleções reindexáveis (i.e. capazes de lidar com mutações em seus elementos) no desempenho geral de programas, bem como a influência da adoção de técnicas adaptativas na mitigação destes impactos. Para a medição do impacto no desempenho do uso dessas coleções, um experimento baseada na suíte de testes de desempenho DaCapo foi conduzido. Resultados obtidos apontam para uma queda expressiva de desempenho que é mitigada quando técnicas adaptativas são empregadas.**

I. Introdução

Um dos pilares da programação orientada a objetos é o conceito de polimorfismo, cuja etimologia remete ao grego *polimorphos* (muitas formas, em tradução livre). Através desse conceito é possível a substituição de uma implementação de um objeto por outra distinta, sem quaisquer alterações nos elementos que o utilizam.

Dentre as diversas formas de polimorfismo, encontra-se o chamado Polimorfismo de Subtipagem [1], com o qual é estabelecida uma relação entre tipo e subtipo. Uma das formas de se formalizar essa relação é através da noção de subtipo introduzida por Liskov e Wing [2] e seu princípio de substituição.

O Princípio da Substituição de Liskov estabelece algumas características que devem ser observadas. Falhas na sua manutenção podem acarretar mudanças inesperadas de comportamento quando existe uma substituição de instâncias de um tipo e por instâncias de um subtipo.

A biblioteca de coleções da linguagem Java [3] define um tipo básico (*Collection*), do qual todas as coleções são subtipos. Esse tipo base define algumas operações que podem ser invocadas em qualquer coleção. Entre essas operações, encontram-se as operações de inclusão de elementos. Apesar de parecer sólida à primeira vista, essa modelagem tem um problema: em razão de características de suas implementações, coleções baseadas em igualdade[1] (como por exemplo conjuntos) podem ter como elementos somente elementos imutáveis. Logo, a definição de um tipo de coleção básico englobando todas as coleções configura uma violação do Princípio da Substituição de Liskov.

Uma solução possível consiste na refatoração da biblioteca de coleções de modo a que coleções baseadas de igualdade tenham tipos distintos onde é permitida apenas a inserção de objetos imutáveis em coleções baseadas em igualdade. O acesso à elementos de coleções poderia ser realizado através de um tipo comum a todas coleções (por exemplo, através de Iteradores [4]) cujas operações se restringiriam a operações de acesso.

Essa solução se mostra impraticável, uma vez que acarreta na coexistência de duas bibliotecas distintas de coleções potencialmente incompatíveis. Outro desafio é a inexistência de suporte a qualificadores de tipos customizados[2] na linguagem Java. Informalmente, qualificadores de tipos permitem uma categorização adicional de tipos. Um exemplo de qualificador de tipos é a palavra-chave *const* [8], existente na linguagem C++: variáveis (ou parâmetros) qualificadas como *const* só podem ter seu valor atribuído por valores qualificado da mesma forma.

Uma outra solução é a utilização de coleções reindexáveis, onde os algoritmos convencionais são acrescido por algoritmos adicionais que exibam um comportamento alternativo quando elementos não são encontrados pelos meios tradicionais. Essa abordagem porém traz alguns desafios: os algoritmos auxiliares fazem uma varredura linear em todos os elementos a procura de elementos que não foram encontrado pelos algoritmos principais. Em outras palavras, a complexidade temporal [9] das operações de busca e remoção degradam, no melhor caso, de $\mathcal{O}(1)$ para $\mathcal{O}(n)$.

Uma forma de mitigação dos impactos na complexidade temporal consiste na adoção de técnicas adaptativas. De forma geral, o uso dessas técnicas consiste em se instanciar coleções

[1] Vale notar que a problemática aqui apresentada é também aplicável a coleções ordenadas. De modo geral, coleções ordenadas mantêm a ordem no momento da inserção de um elemento. Logo, mudanças subsequentes à inserção podem deixar a coleção em estado inconsistente.

[2] A linguagem Java incorpora, a partir de sua versão 1.8, o conceito de anotações de tipo. Esse recurso permite a implementação de sistemas de tipos plugáveis [5], que por sua vez podem dar suporte a qualificadores de tipos customizados. Um exemplo de sistema de tipos que utiliza essa abordagem é o *Checker Framework* [6], [7] (disponível em http://types.cs.washington.edu/checker-framework/).

reindexáveis apenas quando seus elemento sejam passíveis de mutação.

Tendo-se em vista esse cenário, tem-se como principal objetivo deste trabalho o estudo do impacto da adoção de coleções reindexáveis em programas de computador, bem como os ganhos relacionados ao uso de técnicas adaptativas no que tange a mitigação desses impactos. Para isso, é realizado um estudo empírico no qual o impacto da adoção de coleções reindexáveis (com e sem o uso de técnicas adaptativas) nos programas incluídos na suíte de testes de desempenho DaCapo [10] (versão 9.12-bach) é medido.

Segundo Nelson et al. [11], cerca de 40% dos elementos inseridos em coleções baseadas em igualdade são passíveis de mutação. Além disso, apenas 4% dos elementos sofrem mutações que acarretam inconsistências nas coleções onde estão incluídos. Intuitivamente, pode-se esperar um ganho sensível de desempenho quando técnicas adaptativas são utilizadas de modo a coleções reindexáveis só serem instanciadas quando necessário. Apesar da determinação da mutabilidade de um objeto (ou tipo) ser um problema indecidível (i.e. não pode ser resolvido através de algoritmos), é possível a criação de um procedimento que decida se um objeto é imutável. Nesse caso, coleções tradicionais (i.e. sem os algoritmos alternativos) são empregadas.

Este artigo é estruturado da seguinte maneira: uma introdução, descrições básicas sobre os conceitos fundamentais de igualdade, mutação e seus impactos em coleções (respectivamente, Seções II, III e IV). Na Seção V são descritas em detalhes as técnicas de reindexação utilizadas, sendo também apresentada uma análise da complexidade temporal e espacial de ambas as técnicas. A Seção expõe os detalhes de implementação do protótipo utilizado no experimento (Seção), cujo resultados são apresentados e discutidos na Seção VIII. Por fim, são discutidos trabalhos correlatos e futuros (Seções IX e X) e uma breve conclusão (Seção XI).

II. Igualdade

O conceito de igualdade é um conceito usado diariamente quando se deseja comparar e determinar a equivalência entre elementos. Segundo Tarski [12], a semântica da igualdade e comparação depende do domínio no qual esses elementos operam.

Para ilustrar o conceito de igual pode-se imaginar uma classe de objetos chamada $Coordenada$, cujas instâncias representam o conceito geográfico de coordenada. Instâncias dessa classe são compostas de dois atributos: a latitude e a longitude – posições relativa ao Equador (orientação norte-sul) e ao meridiano de *Greenwich* (orientação leste-oeste), respectivamente. Duas coordenadas são consideradas equivalentes quando têm os mesmos valores de latitude e longitude. Nesse caso, pode-se dizer que o estado de igualdade (ou equivalência) de uma coordenada é composto pela sua latitude e longitude.

Uma pequena digressão sobre a definição de estado de igualdade em questão se faz necessária. Nota-se o viés algébrico da definição (i.e. o estado é composto de partes que por sua vez também são compostas por partes menores). Pode-se porém se efetuar uma redução de um elemento (ou uma expressão) arbitrário à uma forma algébrica, conforme demonstrado por Gödel [13]. Apesar de uma procedimento geral ser incomputável, é possível a implementação de funções desse tipo para classes específicas de objetos. Por exemplo, pode-se descrever o estado de equivalência de um algoritmo de ordenação através de uma 1-upla cujo único elemento representa a estabilidade[3] do algoritmo em questão.

A. Coleções Baseadas em Igualdade

Coleções são estruturas de dados amplamente utilizadas em programas de computador, independentemente do paradigma empregado. Seu principal papel é o agrupamento de elementos relacionados. Pode-se citar como exemplos de coleções as listas ligadas, os vetores, e as tabelas de espalhamento [9].

Pode-se identificar entre as coleções um grupo de coleções que têm uma característica em comum: o uso do estado de igualdade de seus elementos. Essa categoria de coleções é denominada de coleções baseadas na igualdade e tem como membros coleções como conjuntos (que, de maneira análoga aos construtos matemáticos de mesmo nome, não permitem elementos duplicados) e as Tabelas de Espalhamento (também conhecidas como Tabelas *Hash* ou Dicionários).

Tabelas de espalhamento foram inventadas por Hans Peter Luhn em meados de 1953 enquanto trabalhava na IBM [14]. Elas têm um funcionamento simples e engenhoso. Uma tabela de espalhamento é composta por vetores auxiliares. No momento da inserção de um novo elemento, é decidido, através de um procedimento determinístico, em qual dos vetores internos o elemento será incluído. No momento de uma busca, esse mesmo procedimento é utilizado para a obtenção do vetor interno onde a busca será efetivamente realizada. Logo, a busca é restrita a apenas um subconjunto de todos os elementos incluídos na tabela.

O procedimento de escolha do vetor interno depende de uma função denominada função de espalhamento (ou função *hash*). Em termos informais, existe uma relação entre o estado de igualdade de um objeto e seu código *hash*: que objetos equivalentes deve ter o mesmo código *hash*. Caso valores diferentes fossem obtidos para objetos semelhantes, buscas subsequentes seriam impossíveis, uma vez que a busca seria realizada em um vetor distinto ao vetor onde o elemento fora incluso.

III. Mutação

Uma das principais diferenças entre os paradigmas de programação funcional e imperativo reside presença de efeitos colaterais (efeitos observáveis, secundários à obtenção de um resultado, que ocorrem durante a execução de uma função). No paradigma funcional, efeitos colaterais são evitados, enquanto que no paradigma imperativo, são abraçados. Essa característica é ainda mais pronunciada em linguagens orientadas à

[3]Um algoritmo de ordenação é considerado estável, quando não muda a ordem original dos elementos caso eles tenham o mesma classificação [9].

objetos, onde a própria definição de objeto embute o conceito de estado.

Vale ressaltar que, apesar da mutabilidade ser abraçada pelo paradigma orientado a objetos, nem todo objeto é obrigatoriamente mutável. Nelson et al. [11] define categorias de mutabilidade baseadas no grau de mutabilidade de um objeto. Segundo essa categorização, objetos podem ser: *imutáveis*, quando não existem mudanças no seu estado interno após a sua construção; terem seu *estado de igualdade imutável*, quando, apesar de existirem mudanças de estado, essas não afetam o estado de igualdade do objeto; e *mutáveis*, quando pode-se alterar indiscriminadamente os atributos de um objeto, sejam eles pertencentes ao seu estado de igualdade ou não.

Apesar de categorizar em sua completude os níveis de impacto de uma mutação no estado de igualdade de um objeto, essa categorização tem uma desvantagem importante: ela é indecidível. Logo, é impossível a escrita de um procedimento algorítmico que verifique o impacto de mutações no estado de igualdade dos objetos nas variadas ramificações de um programa.

Uma categorização menos abrangente, porém decidível, consiste em categorizar objetos em duas categorias distintas: *imutáveis* e *potencialmente mutáveis*. Objetos imutáveis consistem em objetos cuja própria declaração omite construtos que dão suporte à mutabilidade (i.e. é sintaticamente impossível a modificação do estado interno de um objeto). Objetos não categorizados como imutáveis, são, por definição, considerados potencialmente mutáveis.

IV. Mutações e Coleções Baseadas em Igualdade

Uma das possíveis consequências de mutações em um objeto é a alteração do seu estado de igualdade. Essas alterações têm um efeito negativo quando esses elementos estão inseridos em coleções baseadas em igualdade. Pode-se ilustrar esse impacto com o seguinte cenário: dada uma tabela de espalhamento cujo elementos são coordenadas. Mudanças na latitude (ou na longitude) desses elementos podem mudar o seu código *hash*, o que impossibilitaria sua busca, uma vez que a busca poderia ser realizada em um vetor interno distinto ao vetor onde a coordenada fora originalmente incluída.

Algumas técnicas de mitigação foram desenvolvidas no intuito de se mitigar esse problema. Essas técnicas diferem basicamente no grau de mutabilidade dos objetos passíveis de inserção em coleções baseadas em igualdade.

A. Estado de igualdade imutável

A primeira técnica foi proposta por Liskov e Guttag [15] e é usado por Vaziri et al. [16] na definição de tipos de relação – uma extensão da linguagem Java que permite a definição declarativa da igualdade – e consiste na proibição da alteração do estado de igualdade de objetos após a sua construção.

Apesar de ser extremamente restritiva, essa técnica tem como principal vantagem a decidibilidade de sua checagem. De fato, em linguagens como por exemplo C++ e Java, provêm construtos (*const* e *final*) que impedem a alteração de atributos de objetos após a sua inicialização. Nestes casos, tentativas de alteração do valor contido nesses atributos resultam em erros de compilação.

B. Estado de igualdade imutável após inserções

Esta técnica é consiste em só permitir a alteração de informações que impactem o estado de igualdade de um objeto até a sua inserção em uma coleção. Após sua inserção, mutações são proibidas. É usada por Grech et al. [17] em sua proposição de mecanismo para definição de semântica de igualdade em Java.

É interessante notar que essa técnica é suportada nativamente por algumas linguagens de programação, como por exemplo Ruby [18]. Nesta última, todos objetos têm a operação *freeze* – congelar, em inglês. Tentativas de mudança de estado de um objeto subsequentes à invocação dessa operação resulta em exceções em tempo de execução.

Apesar de ser a primeira vista mais flexível que a primeira técnica apresentada, essa técnica apresenta uma desvantagem importante – a sua checagem em tempo de compilação é um problema indecidível. Isso faz com que inconsistências no uso de coleções só sejam descobertas em tempo de execução.

V. Reindexação

Apesar de mitigar o problema em questão, técnicas que impedem a mutabilidade de objetos têm desvantagens importantes. Entre elas a impossibilidade da realização de certas modelagens válidas. Por exemplo, é uma modelagem plausível o armazenamento dos endereços de uma pessoa em um conjunto. Da mesma maneira alterações nesses endereços também são plausíveis. Caso a alteração do estado de igualdade seja proibida, endereços não poderiam ser alterados.

Uma forma de se mitigar essa consiste no emprego de coleções reindexáveis. Coleções reindexáveis diferem-se das coleções convencionais por incorporar uma série de algoritmos auxiliares empregados nos casos onde os elementos não são encontrados através dos procedimentos usuais. Com o uso de coleções reindexáveis pode-se manter a consistência de coleções baseadas na igualdade sem quaisquer restrições à mutabilidade em nenhuma etapa do ciclo de vida de um objeto.

A. Implementação Ingênua

Em termos gerais, a implementação ingênua de uma tabela de espalhamento re-indexável consiste na alteração das operações que façam buscas nos vetores internos da tabela (i.e. operações de busca, inserção e remoção), de modo que, nos casos onde um elemento não é encontrado dentro da tabela pela algoritmo habitual (i.e. baseado na escolha do vetor via código *hash*), seja realizada uma busca secundária, na qual todos os vetores internos da tabela são varridos. Caso o elemento seja encontrado nessa busca secundária, uma operação de reindexação é realizada seguida de uma nova execução da operação original. A reindexação de um elemento consiste na sua remoção seguida pela sua re-inserção. O Algoritmo 1 apresenta uma descrição deste procedimento em pseudo-código.

Algoritmo 1: Reindexação – Implementação ingênua

```
Data: Coleção-alvo c
Data: Operação k a ser executada contra c
Data: Elemento e a ser manipulado por k
Result: Resultado da invocação de k em c
início
    r ← invoca k em c
    se se k for mal-sucedida então
        para cada elemento x na coleção c faça
            se x e e são a mesma instância então
                remove x de c
                adiciona x à c
                retorna resultado de nova execução de k
            fim
        fim
    fim
    retorna r
fim
```

Algoritmo 2: Reindexação – Implementação adaptativa

```
Data: Coleção-alvo c
Data: Operação k a ser executada contra c
Data: Elemento e a ser manipulado por k
Result: Resultado da invocação de k em c
início
    r ← invoca k em c
    se se k for mal-sucedida e e mutável então
        para cada elemento x na coleção c faça
            se x e e são a mesma instância então
                remove x de c
                adiciona x à c
                retorna resultado de nova execução de k
            fim
        fim
    fim
    retorna r
fim
```

A lógica por trás dessa abordagem consiste na ideia de que, independentemente do estado de igualdade (e por consequência seu código *hash*) de um objeto no momento da sua inserção, esse objeto encontra-se em um dos vetores internos da tabela de espalhamento. Por exemplo, digamos que no momento da inserção tinha 1 (um) como valor de código *hash* cujo valor, após uma mutação, passou a ser 3 (três). Apesar da busca convencional ser incapaz de encontrar o objeto (o algoritmo usual apenas procuraria no terceiro vetor, onde não seria encontrado), uma busca linear em todos os vetores seria bem-sucedida, uma vez que ele seria encontrado na varredura do terceiro vetor. Após re-indexação o objeto seria incluído no terceiro vetor e sendo subsequentemente encontrado pelos algoritmos convencionais (a não ser que sofresse uma nova mutação, neste caso o ciclo se repetiria).

B. Implementação com uso de técnicas adaptativas

Segundo Nelson et al. [11], apenas 4% dos elementos têm seu estado de igualdade alterado após sua inclusão em coleções baseadas em igualdade. Sendo assim, conclui-se que o custo inerente à re-indexação muitas vezes não é necessário.

Com o intuito de diminuir esse impacto, pode-se adotar técnicas adaptativas. De acordo com Neto [19], um dispositivo é denominados adaptativo quando seu comportamento muda em função de sua entrada. Através do uso dessas técnicas pode-se incorporar duas otimizações distinta que têm como alvo tanto o algoritmo de reindexação quanto a instanciação condicional das coleções.

O algoritmo adaptativo de reindexação de coleções é semelhante à implementação nativa. Em ambos é realizada uma busca linear seguida por uma reindexação nos casos onde a busca convencional seja infrutífera. Porém, diferentemente da implementação ingênua, a reindexação só é executada quando o elemento sendo manipulado seja potencialmente mutável. Esse comportamento é descrito pelo Algoritmo 2.

Outra otimização compreende a instanciação condicional de coleções reindexáveis. Nesse caso coleções reindexáveis são instanciadas apenas quando forem passíveis de terem elementos potencialmente mutáveis. Para isso, utiliza-se de uma característica inerente às coleções incluídas em bibliotecas[4] encontradas em linguagem fortemente tipadas. Nessas bibliotecas é habitual que coleções possam ter seu tipo refinado através da definição do tipo de seus elementos. Através desse mecanismo, o compilador pode garantir que apenas instâncias de um determinado tipo possam ser incluídas em uma coleção. Dá-se a esse mecanismo o nome de Polimorfismo Paramétrico [1].

Existe porém uma restrição com relação à técnica descrita acima: os tipos usados para parametrizar os elementos de uma coleção podem ser tipos polimórficos (i.e. podem ser estendidos arbitrariamente). Algumas dessas extensões podem eventualmente ter um perfil de mutabilidade distinto. Por exemplo, um subtipo pode definir um novo atributo que porventura seja modificável. Sendo assim, só é possível se aplicar a instanciação condicional de coleções reindexáveis quando o tipo usado para sua parametrizar é imutável e não pode ser ser estendido.

C. Complexidade Algorítmica

Apesar de semelhantes em alguns aspectos, pode-se intuitivamente se inferir que as implementações – ingênua e adaptativa – têm perfis de execução distintos. Logo é desejável uma análise com o intuito de indicar o comportamento de ambos algoritmos, tanto no que tange o tempo de execução – complexidade temporal –, quanto o uso de memória – complexidade espacial. A Tabela I apresenta essas complexidades[5] categorizadas tanto por classe de operação (inclusão, remoção

[4] Pode-se citar, entre outras, o STL em C++, *Java Collections Framework* em Java.

[5] As complexidades referentes à implementação usual de tabela de espalhamento foram obtidas em Cormen et al. [9].

Tabela I: Complexidade Algorítmica – Tabela de Espalhamento

	Convencional[1]		Ingenua		Adaptativa	
Operação	Melhor	Pior	Melhor	Pior	Melhor	Pior
Elementos Mutáveis						
Inserção	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Remoção	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Busca	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$
Elementos Imutáveis						
Inserção	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(n)$
Remoção	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(n)$
Busca	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$
Espacial	$\mathcal{O}(n)$		$\mathcal{O}(n)$		$\mathcal{O}(n)$	

[1] Não trata mutações de elementos após sua inserção.

e busca) quanto pelo perfil dos elementos sendo incluídos (mutáveis e imutáveis). Para a expressão das complexidades foi adotada a notação Grande-O [9].

No que tange a complexidade espacial, não existem diferenças entre tabelas de espalhamentos convencionais e reindexáveis. Porém, no que tange a complexidade temporal, existem diferenças consideráveis. Pode-se notar uma degradação nas operações relativas à inserção e remoção nas coleções reindexáveis. Conforme discutido anteriormente, o processo de reindexação realiza uma busca linear na coleção nos casos onde os elementos não são encontrados. Buscas lineares têm a complexidade temporal $\mathcal{O}(n)$. Logo, a complexidade temporal das inserções e remoções degrada para $\mathcal{O}(n)$.

Notavelmente, a implementação baseada em técnicas adaptativas só apresenta degradação quando os elementos sendo inseridos (ou removidos) na coleção são mutáveis. No caso dos elementos imutáveis, essa implementação comporta-se de maneira semelhante à implementação convencional.

VI. Protótipo

Apesar da análise da complexidade algorítmica através da notação Grande-O ser capaz expressar do comportamento dos algoritmos com relação ao tamanho de sua entrada, não se pode através dela se inferir o tempo absoluto gasto por um algoritmo. Por exemplo, dois algoritmos que realizam uma operação para cada elemento de sua entrada têm a mesma complexidade temporal – $\mathcal{O}(n)$) –, independentemente do tempo gasto por cada um nessas operações. Tendo em vista esta limitação, foi elaborado um experimento para a medição dos impactos de ambas as técnicas – ingênua e adaptativa – no desempenho geral de programas. Para isso é necessária a implementação de um protótipo de ambas soluções.

Vale-se notar que não foi realizado nenhuma tentativa de otimização das estruturas e algoritmos aqui utilizadas, uma vez que o objetivo principal deste artigo não é a definição de uma implementação de coleções reindexáveis plenamente otimizadas e sim o estudo do impacto relativo do uso de coleções reindexáveis no desempenho. É importante salientar também que a implementação não é estruturalmente ótima: o uso de decoradores [4] para a inclusão das operações de reindexação levou ao uso de técnicas de programação reflexiva, que por sua vez acarretou a quebra do encapsulamento. O paradigma da linguagem de programação também influiu na estrutura adotada pelo protótipo. É possível, por exemplo, se realizar uma implementação mais elegante (e eficiente) em linguagens baseadas em protótipos (como por exemplo SELF e Javascript [20], [21]). Nessas linguagens é possível a manipulação da hierarquia de um objeto em tempo de execução. Logo, pode-se alterar a implementação de coleção utilizada no momento da inserção (ou remoção) de um elemento.

A. Detalhes de implementação

É desejável que o protótipo seja capaz de substituir as implementações de coleções por suas contrapartidas reindexáveis sem que seja necessária a alteração dos programas, uma vez que esta pode introduzir novos defeitos. Uma solução eficaz para esse problema reside na adoção de técnicas da programação orientada a aspectos [22]. Vislumbrado por Gregor Kiczales, esse paradigma tem como mote a decomposição de problemas em interesses transversais – funcionalidades que não são pertencem à um construto específico e são utilizadas de maneira dispersa por todo código. Através desse paradigma, pode-se definir comportamentos adicionais (denominados conselhos ou *advices*) a serem injetados em pontos específicos de um programa (chamados de pontos de corte ou *pointcuts*).

Pode-se aplicar essa capacidade de definição de comportamento alternativos para injetar as operações referentes à reindexação nas coleções baseadas em igualdade. Apesar da linguagem Java não ter suporte nativo à programação orientada a aspectos, é possível a incorporação desse paradigma através de bibliotecas como o AspectJ, essa última utilizada na modalidade de inclusão de aspectos em tempo de execução (*load-time weaving*). Essa modalidade tem como vantagem dispensar a recompilação dos programas usados pela suíte de testes, porém não permite a instrumentalização de classes pertencentes ao JDK (*Java Development Kit*). Em razão dessa limitação, optou-se pela interceptação de suas chamadas de instanciação ao contrário da modificação das classes de coleções propriamente ditas.

Ao invés de se realizar uma implementação uma implementação *clean room* (i.e. uma nova implementação, sem dependências), optou-se por adotar uma abordagem baseada em decoradores [4]. Nessa abordagem, as funcionalidades básicas são supridas pela implementação convencional das coleções e as operações passíveis de reindexação são decoradas de modo a realizá-la quando necessário. A principal razão para a adoção desse procedimento reside na maior fidelidade nas medições, uma vez que uma nova implementação poderia não ter o mesmo grau de otimização das coleções nativas.

B. Instanciação condicional de coleções

Conforme discutido anteriormente, a partir da versão 1.5, a biblioteca de coleções da linguagem Java dá suporte ao polimorfismo paramétrico. A primeira vista, pode-se usar essas informações para se determinar o grau de mutabilidade dos

elementos de uma coleção. E, a partir dessa informação, se determinar se é necessária uma implementação re-indexável.

Essa informação porém é removida do código-objeto durante a compilação através de um processo chamado apagamento de tipos (*Type erasure*, em inglês), cuja motivação é a manutenção da compatibilidade do código-objeto em diferentes versões da plataforma Java [3].

De modo a simular a presença das informações removidas através do processo de apagamento de tipos, optou-se por disponibilizar o código-fonte dos programas sendo testados no *classpath*[6] da suíte de testes. No momento da instanciação de uma coleção, a linha correspondente no código-fonte é analisada. Caso o seus elementos sejam instâncias de uma classe considerada imutável (i.e. todos seus atributos são declarados como finais e têm tipos imutáveis, e estende uma classe imutável; ou é uma das classes previamente assim categorizadas[7]), é utilizada a implementação de coleção convencional, caso contrário, é utilizada uma implementação re-indexável. O Algoritmo 3 descreve o procedimento de decisão sobre a imutabilidade de uma classe.

Algoritmo 3: Procedimento de Checagem de Mutabilidade

```
Data: Classe c, cuja mutabilidade se deseja verificar
Result: verdadeiro caso a classe c seja potencialmente
        mutável, falso caso contrário
início
    se se c for uma classes imutável por definição então
        return falso
    senão se a super-classe c for imutável então
        para cada campo a da classe c faça
            se classe de a for p. mutável então
                retorna verdadeiro
            fim
        fim
        retorna falso
    senão
        retorna verdadeiro
    fim
fim
```

Vale destacar que o mecanismo utilizado para contornar a limitação imposta pelo apagamento de tipos pode ser impraticável em ambiente produtivo. Além de ter um alto custo computacional, há a necessidade da inclusão do código-fonte, o que pode ser um empecilho nos casos onde, por razões de segurança, a ofuscação do código seja desejável.

VII. Experimento

Conforme previamente discutido, somente a análise de complexidade temporal de um algoritmo não é suficiente para a comparação do impacto no desempenho (e espaço) da adoção das técnicas apresentadas. Para isso é necessária a condução de um experimento, no qual o impacto da adoção das técnicas no desempenho de programas de variados segmentos é analisado.

No intuito de se obter resultados mais fidedignos aos resultados obtidos pelo uso das técnicas, foi utilizada a versão 9.12-bach da suíte de testes DaCapo [10] . Essa suíte de testes é composta por uma gama variada de programas de código-fonte aberto, entre os quais encontram-se compiladores, renderizadores e gerenciadores de banco de dados. Essa abrangência faz com que os resultados obtidos sejam próximos aos impactos esperados em cenários do mundo-real.

A. Métricas

Para a medição dos impactos no desempenho foram definidas duas métricas distintas: o tempo de execução e a pressão de memória. O tempo de execução (medido em segundos) é uma métrica de simples entendimento que consiste no tempo necessário para a execução de um número determinado de operações (cuja semântica depende do teste de desempenho sendo executado). Já pressão de memória tem como motivação indicar o grau de utilização da memória durante o cenário de testes.

Diferentemente de linguagens como C++, na linguagem Java a memória utilizada pelos programas é reciclada de forma automática, através de um componente denominado Coletor de Lixo [23]. De forma geral, um Coletor de Lixo varre a memória e libera blocos que não sejam mais utilizados. Conforme as características de uso de memória – taxa de alocação e tempo de vida de objetos – o coletor de lixo pode ser mais ou menos demandado.

$$\mathcal{P}_x = \frac{\tau_x}{\tau_0} \tag{1}$$

A Fórmula 1 calcula a pressão de memória $\mathcal{P}_x$ de uma execução x, onde τ_x e τ_0 são respectivamente a quantidade de memória manipulada pelo processo de coleta de lixo na execução x e da linha base. Por definição, a pressão de memória da linha base é 1. Notavelmente, a pressão de memória deve ser interpretada como uma métrica de comparação entre duas execuções distintas: quanto maior seu valor, mais o coletor de lixo foi exigido e consequente mais do subsistema de memória foi demandado.

B. Método

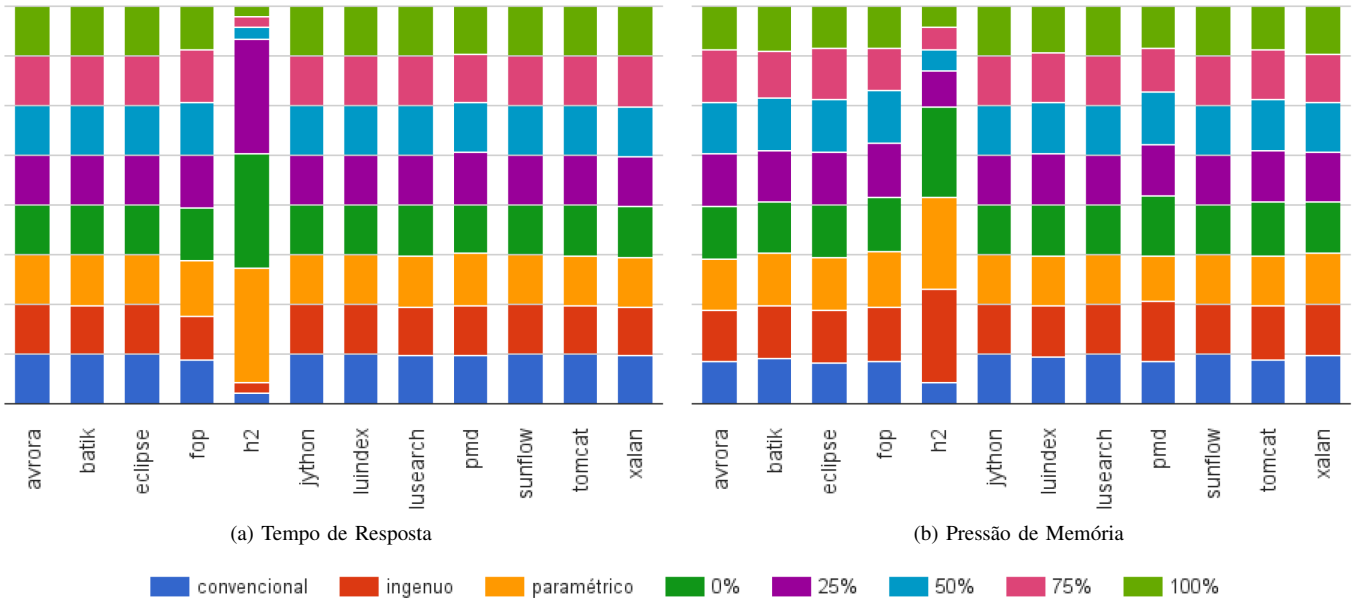
É importante porém ressaltar que a suíte de testes tivera que ser modificado[8] durante a realização do experimento. Essas alterações compreendem: a atualização de programas – *Apache Tomcat* – incompatíveis com a versão do JDK utilizada; a disponibilização do código-fonte dos programas analisados no *classpath*; e a inclusão do protótipo propriamente dito. Além disso, alguns programas – *tradebeans* e *tradesoap* – foram removidos do experimento. Esses programas são baseados do servidor de aplicação Apache Geronimo[9], que utiliza-se de

[6]Conjunto de recursos que podem ser carregados pela JVM. Uma classe deve estar incluída no *classpath* para ser utilizada.

[7]Enumerados; todos os tipos primitivos e suas classes correspondentes; e *String*, *BigInteger*, *BigDecimal* e *Object*.

[8]O pacote modificado, acrescido do código-fonte do protótipo, encontra-se disponível em https://drive.google.com/open?id=0B4-smhbrEJAaRGI1NXdJVUtKbGM.

[9]Disponível em http://geronimo.apache.org/.



(a) Tempo de Resposta

(b) Pressão de Memória

Figura 1: Resultados do Experimento

um mecanismo de carregamento de classe incompatível com instrumentação de código utilizado pelo AspectJ.

Inicialmente foram definidos três cenários de medição, referentes à linha-base, à implementação ingênua e a implementação adaptativa. Estudos realizados por Parnin et al. [24] porém mostraram um padrão de utilização de coleções parametrizadas que poderia comprometer o experimento. Apesar da utilização em código mais recentes de coleções parametrizadas, não existe um movimento de modernização de bases de código mais antigas para a sua adoção. Esse cenário invalidaria as medições referentes aos ganhos do uso de técnicas adaptativas, uma vez que nesse caso a última se comportaria de maneira semelhante à implementação ingênua. Tendo em vista essa conjuntura, foram definidos outros cinco cenários baseados na implementação adaptativa, nos quais a decisão sobre o uso de coleções reindexáveis é baseada em razões (0%, 25%, 50%, 75% e 100%) previamente definidas.

Notavelmente, alguns cenários executam procedimentos adicionais que pode ter um impacto considerável nas medições. Tendo em vista que o objetivo principal é o estudo do impacto das técnicas na execução de programas, se fez necessária uma normalização desses resultados. Para isso, todos os cenários executam as mesmas operações auxiliares (procedimentos de inferência sobre a mutabilidade de elementos de uma coleção e a obtenção de números randômicos para os cenários baseados na razão de objetos imutáveis são sempre realizados), independentemente da utilização de seus resultados. Um efeito colateral dessa abordagem é a disparidade entre as medidas obtidas pela linha base e pela execução da suíte de teste sem modificações.

Além das métricas de desempenho definidas anteriormente, também foram colhidos dados referente ao perfil de utilização de coleções em cada um dos programas utilizados. Esses dados englobam à implementação utilizadas – convencional ou reindexável – bem como a quantidade de instâncias criadas. A principal motivação para a obtenção desses dados auxiliares é se traçar um perfil de uso de coleções em cada um dos cenários estudados.

Tabela II: Perfil de Uso de Coleções

Programa	Instâncias Convencionais	Instâncias Reindexáveis	Instâncias Totais	%
avrora	0	469	469	100%
batik	0	2300	2300	100%
eclipse	0	2178	2178	100%
fop	0	11554	11554	100%
h2	0	8704171	8704171	100%
luindex	0	77	77	100%
lusearch	0	1156	1156	100%
pmd	203217	25195	228412	11,3%
tomcat	13309	10249	23684	43,51%
xalan	0	110338	110338	100%
jython[1]	–	–	–	–
sunflow[2]	–	–	–	–
tradebeans[3]	–	–	–	–
tradesoap[4]	–	–	–	–

[1, 2] Não foram criadas coleções durante a execução.
[3, 4] Baseados no Apache Geronimo, cujo mecanismo de carga de classe impediu a utilização do dispositivo de injeção de aspectos em tempo de execução.

VIII. Resultados

Tendo em vista o método e protótipo previamente descritos, foram executadas medições que encontram-se sumarizadas na

Tabela III: Resultados Obtidos

Programa	Linha Base	Ingenua		Paramétrica		Adaptativa 0%		25%		50%		75%		100%	
avrora	11,06s	11,05s	1,20	11,08s	1,20	11,08s	1,24	11,07s	1,22	11,08s	1,21	11,03s	1,24	11,05s	1,01
batik	2,04s	2,07s	1,14	2,07s	1,14	2,06s	1,14	2,06s	1,12	2,06s	1,12	2,05s	1,02	2,03s	1,00
eclipse	93,08s	93,33s	1,27	92,97s	1,26	92,64s	1,26	92,66s	1,27	92,67s	1,27	92,89s	1,27	93,36s	1,00
fop	0,27s	0,34s	1,32	0,33s	1,31	0,32s	1,31	0,33s	1,30	0,32s	1,28	0,27s	1,01	0,27s	1,00
h2	26,16s	274,00s	4,31	276,77s	4,27	275,01s	4,21	27,42s	1,70	26,07s	1,00	26,20s	1,00	26,31s	1,00
jython	8,58s	8,48s	1,00	8,56s	1,00	8,74s	1,00	8,55s	1,00	8,52s	1,00	8,55s	1,00	8,53s	1,00
luindex	0,90s	0,90s	1,09	0,90s	1,09	0,90s	1,09	0,89s	1,09	0,89s	1,09	0,90s	1,08	0,90s	1,00
lusearch	1,11s	1,15s	1,00	1,15s	1,00	1,14s	1,00	1,13s	1,00	1,13s	1,00	1,12s	1,00	1,08s	1,00
pmd	1,53s	1,58s	1,39	1,52s	1,06	1,61s	1,39	1,51s	1,22	1,51s	1,20	1,47s	1,02	1,50s	1,00
sunflow	3,77s	3,80s	1,00	3,78s	1,00	3,77s	1,00	3,78s	1,00	3,77s	1,00	3,78s	1,00	3,81s	1,00
tomcat	5,60s	5,72s	1,27	5,67s	1,11	5,73s	1,27	5,69s	1,19	5,69s	1,16	5,62s	1,15	5,58s	1,00
xalan	4,40s	4,57s	1,07	4,59s	1,07	4,59s	1,06	4,56s	1,05	4,60s	1,03	4,54s	1,01	4,42s	1,00
tradebeans[1]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
tradesoap[2]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Ambiente de execução: i7-4790K (3,6 Ghz) com 32GB de RAM rodando Windows 10. Foi usado a JDK 1.8.0_92 (64 bits), sem nenhum qualquer tipo de ajustes dos parâmetros de execução (i.e. tamanho da memória *heap* e parâmetros do coletor de lixo).
[1, 2] Baseados no Apache Geronimo, cujo mecanismo de carga de classe impediu a utilização do dispositivo de injeção de aspectos em tempo de execução.

Tabela III e no Gráfico 1 (resultados detalhados estão dispostos no Apêndice A). No intuito de se diminuir as desvios, cada cenário foi executado cem vezes com a obtenção da mediana dos valores.

Pode-se a primeira vista notar-se impactos mais pronunciados nas medidas obtidas pelo *H2*, onde os tempos de processamento decorrido na execução implementação ingênua foi cerca onze vezes maior que a linha-base. De maneira recíproca, alguns programas – *sunflow* e *jython* – não tiveram diferenças significativas tanto no tempo de processamento quanto na pressão memória através dos cenários. Essa discrepância está relacionada com o perfil de uso (disposto na Tabela II) das coleções em cada um dos programas analisados. Pode-se notar uma correlação entre o desempenho do programa e a quantidade de coleções reindexáveis criadas. Programas com maior quantidade de coleções criadas – *h2*, *pmd* e *tomcat* – estão entre os programas que tiveram maior impacto em seu desempenho (*h2*, *pmd*, *fop* e *tomcat*).

Apesar de comparativamente não criar muitas instâncias, o programa *fop*[10] obteve uma queda mais expressiva de desempenho que o *tomcat*, que ocupa o terceiro lugar na classificação de programas por número de coleções criadas. Pode-se intuitivamente se atribuir essa discrepância a dois fatores. O primeiro fator relaciona as diferenças no modo de execução dos dois cenários: enquanto o primeiro realiza suas operações basicamente dentro de um único processo, o último é baseado em requisições enviadas através do endereço de *loopback* por outro processo. Supõe-se que o tempo gasto em operações relativas à comunicação via rede dilua o impacto no desempenho no último cenário.

Já o segundo fator refere-se a presença de ruídos de medição: o tempo de execução do cenário baseado no *fop* é ordens de magnitude menor que o do baseado no *tomcat*, o que faz com que ruídos nas medições se tornem mais pronunciados.

Também pode-se notar que, em linhas gerais, a implementação adaptativa baseada nas informações de tipo das coleções não obteve um desempenho muito melhor da implementação ingênua. Essa observação corrobora a observação de Parnin et al. [24] de que programas mais antigos não sofreram alterações para o uso de coleções parametrizadas.

O estudo do perfil das coleções indicou que apenas dois programas – *tomcat* e *pmd* – se beneficiaram da técnica adaptativa. Nesses dois casos, os resultados obtidos através da instanciação condicional se equipararam aos resultados obtidos através dos cenários onde as razão entre coleções convencionais e reindexáveis eram pré-definidas (no caso do *tomcat*, os resultados obtidos pela instanciação condicional[11] e paramétrica foram 5,67s e 5,69s respectivamente, enquanto no *pmd* foram 1,52s e 1,50s).

Os resultados obtidos mostram que o emprego de técnicas adaptativa aliado ao uso de anotações de tipos em coleções pode mitigar os impactos no desempenho associados ao uso de coleções indexáveis. Sendo por meio destes possível a obtenção de resultados semelhantes aos obtidos através da implementação convencional (i.e. sem reindexação).

IX. Trabalhos Correlatos

Nelson et al. [11] analisa o impacto da mutabilidade de elementos em coleções. Suas conclusões indicam que a grande maioria dos objetos inseridos em coleções baseadas em igualdade não mudam seu estado de igualdade, mesmo quando são mutáveis. Esse cenário muda quando coleções não-baseadas na igualdade são levadas em conta. Nesse caso, mais da metade dos objetos apresentam mudanças após serem inseridos

[10] Um conversor da linguagem de marcação XSL-FO para PDF. Disponível em https://xmlgraphics.apache.org/fop/.

[11] Utilizada os cenários cujas razões são 50% e 100% para o*tomcat* e o *pmd*, respectivamente.

em coleções. Isto indica um potencial problema, uma vez que implementação dessas coleções pode ser substituída por implementações baseadas na igualdade sem qualquer tipo de aviso por parte do compilador.

Conforme previamente discutido, uma outra possibilidade de mitigação do problema apresentado consiste na refatoração das bibliotecas de coleções de modo que o sistema de tipos impeça a inclusão de objetos potencialmente mutáveis em coleções baseadas em igualdade. Para isso, se faz necessário o suporte a qualificadores de tipos. Existe na Literatura uma série de proposições [25], [26] de extensões da linguagem Java cujo sistema de tipos prevê suporte a qualificadores de tipos. Ainda nessa linha, uma abordagem menos intrusiva compreende a adoção de um sistema de tipos plugável [5] como o *Checker Framework* [6], [7]. Vale citar que que o último é disponibilizado com uma implementação de qualificadores de tipos que tratam da imutabilidade.

Pode-se também citar como trabalho correlato a ferramenta Chameleon [27], que tem como principal motivação a decisão da melhor implementação de coleções a ser utilizada. Porém, ao contrário do estudo aqui apresentado, essa ferramenta não altera as implementações em tempo de execução. Ao invés disso, apresenta um relatório onde sugere mudanças relativas às implementações utilizadas, ficando a cargo do desenvolvedor acatar as sugestões.

X. Trabalhos Futuros

Durante a execução desta pesquisa foram identificados alguns possíveis desdobramentos. Um desses desdobramentos consiste na realização de novos testes quando a reificação de tipos genéricos for suportada pelo Java. A inclusão do suporte à reificação de tipos está prevista para a versão 10 da linguagem Java, através do Projeto *Valhalla*[12].

Um outro possível desmembramento consiste num estudo exploratório sobre os impactos da adoção de uma biblioteca de coleções estruturada de modo a eliminar a possibilidade da inclusão de elementos mutáveis em coleções baseadas em igualdade é utilizada. Apesar da difusão do uso de uma biblioteca moldes ser improvável, os resultados obtidos através dessa pesquisa serão úteis no desenvolvimento de bibliotecas para novas linguagens.

Uma ramificação viável deste trabalho consiste em um estudo exploratório do impacto relativo ao uso instanciação condicional de coleções incorporando otimizações específicas para certos tipos de elementos no desempenho. Pode-se citar com exemplo de tipos que podem se beneficiar de implementações alternativa de coleções são os enumerados – tipo de dados representando um domínio de valores válidos. Uma característica importante dos enumerados é a possibilidade de serem mapeados para inteiros, o que permite a simulação de tabela de espalhamento através de um simples vetor[13]. Seguindo essa linha, existem implementações alternativas de coleções[14] já otimizadas para o armazenamento de tipos primitivos.

XI. Conclusão

Apesar de relativamente incomum, mutações em elementos de coleções baseadas em igualdade é problemática, uma vez que impossibilita a busca destes através dos algoritmos de busca convencionais.

A solução conceitualmente mais adequada para esse problema consiste na formalização de dois tipos distintos de coleções: coleções não-baseadas e baseadas em igualdade. O sistema de tipos deve garantir que apenas objetos imutáveis sejam inclusos na última. Essa solução porém se mostra impraticável no cenário atual, uma vez que sua adoção acarretaria uma série de incompatibilidades, tanto no nível de código-fonte quanto de código-objeto.

Outra forma de solucionar-se o problema consiste na criação de coleções reindexáveis (i.e. coleções passíveis cuja consistência é mantida mesmo quando um elemento sofre mutações). Uma implementação ingênua desse conceito consiste em implementações alternativas de coleções baseadas em igualdade, onde todos os objetos são tratados como mutáveis e buscas mal-sucedidas são seguidas de uma operação de reindexação.

Resultados obtidos experimentalmente mostram uma sensível queda de desempenho quando a técnica ingênua é empregada. Com adoção de técnicas adaptativas pode-se mitigar esses efeitos negativos, sendo possível a eliminação total das perdas quando todos os elementos inseridos em coleções baseadas em igualdade são imutáveis. Contudo, algumas questões impedem uso dessa técnica de maneira abrangente.

A primeira dificuldade é relativa a ausência de anotações referentes ao tipo de elementos de coleções na maioria dos programas analisados. Essa observação é corroborada por Parnin et al. [24], que constata que apesar de terem sido amplamente adotada desde a sua incepção, é relativamente rara a migração de código já existente unicamente para a inclusão dessas parametrizações.

Outra dificuldade é relacionada com a falta de um mecanismo de reificação das parametrizações de coleções na versão atual da linguagem Java. No experimento esse dispositivo foi simulado através de uma implementação baseada no processamento do código-fonte, que deve estar disponível em tempo de execução. Apesar de eficaz, essa implementação não é adequada no ambiente produtivo, uma vez o processo de análise do código fonte é relativamente custoso e a presença do código-fonte pode ser muitas vezes proibida por questões de segurança.

Sumarizando, técnicas adaptativas podem diminuir consideravelmente o custo do uso de coleções reindexáveis. Porém, para a construção de uma solução robusta, é necessário que tanto a Máquina Virtual Java de suporte a reificação de tipos de parâmetros e incorporação de coleções reindexáveis ao Kit de Desenvolvimento Java, uma vez que a instrumentalização das classes do último é problemática.

[12]Disponível em http://openjdk.java.net/projects/valhalla/.

[13]O JDK inclui uma implementação de tabela de espalhamento (*java.util.EnumMap*) seguindo esses moldes.

[14]Como por exemplo o GNU Trove. Disponível em http://trove.starlight-systems.com/.

REFERÊNCIAS

[1] PIERCE, B. C. *Types and Programming Languages*. 1st. ed. Cambridge, MA, USA: The MIT Press, 2002. ISBN 0262162091, 9780262162098.

[2] LISKOV, B. H.; WING, J. M. A behavioral notion of subtyping. *ACM Trans. Program. Lang. Syst.*, ACM, New York, NY, USA, v. 16, n. 6, p. 1811–1841, nov. 1994. ISSN 0164-0925. Disponível em: <http://doi.acm.org/10.1145/197320.197383>.

[3] NAFTALIN, M.; WADLER, P. *Java generics and collections*. Sebastopol, CA, EUA: O'Reilly Media, Inc., 2007.

[4] GAMMA, E. et al. *Design patterns: elements of*. Upper Saddle River, NJ, USA: Addison-Wesley, 1994.

[5] BRACHA, G. Pluggable type systems. In: CITESEER. *OOPSLA workshop on revival of dynamic languages*. Vancouver, BC, Canada, 2004. v. 1.

[6] PAPI, M. M. et al. Practical pluggable types for java. In: *Proceedings of the 2008 International Symposium on Software Testing and Analysis*. New York, NY, USA: ACM, 2008. (ISSTA '08), p. 201–212. ISBN 978-1-60558-050-0. Disponível em: <http://doi.acm.org/10.1145/1390630.1390656>.

[7] DIETL, W. et al. Building and using pluggable type-checkers. In: *Proceedings of the 33rd International Conference on Software Engineering*. New York, NY, USA: ACM, 2011. (ICSE '11), p. 681–690. ISBN 978-1-4503-0445-0. Disponível em: <http://doi.acm.org/10.1145/1985793.1985889>.

[8] DRAFT, W. Standard for programming language c++. *ISO/IEC N*, v. 4296, p. 2014, 2009.

[9] CORMEN, T. H. et al. *Introduction to algorithms*. Cambridge, MA, USA: MIT press Cambridge, 2001. v. 6.

[10] BLACKBURN, S. M. et al. The DaCapo benchmarks: Java benchmarking development and analysis. In: *OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-Oriented Programing, Systems, Languages, and Applications*. New York, NY, USA: ACM Press, 2006. p. 169–190.

[11] NELSON, S.; PEARCE, D. J.; NOBLE, J. Understanding the impact of collection contracts on design. In: VITEK, J. (Ed.). *Objects, Models, Components, Patterns: 48th International Conference, TOOLS 2010, Málaga, Spain, June 28–July 2, 2010. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 61–78. ISBN 978-3-642-13953-6. Disponível em: <http://dx.doi.org/10.1007/978-3-642-13953-6_4>.

[12] TARSKI, A. *Introduction to Logic and to the Methodology of the Deductive Sciences (Oxford Logic Guides)*. Oxford: Oxford University Press, 1994. ISBN 019504472X.

[13] GÖDEL, K. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für Mathematik und Physik*, v. 38, n. 1, p. 173–198, 1931. ISSN 1436-5081. Disponível em: <http://dx.doi.org/10.1007/BF01700692>.

[14] KONHEIM, A. G. *Hashing in Computer Science: Fifty Years of Slicing and Dicing*. Hoboken, NJ, USA: John Wiley & Sons, 2010.

[15] LISKOV, B.; GUTTAG, J. *Abstraction and specification in program development*. Cambridge, MA, USA: MIT press, 1986.

[16] VAZIRI, M. et al. Declarative object identity using relation types. In: ERNST, E. (Ed.). *ECOOP 2007 – Object-Oriented Programming: 21st European Conference, Berlin, Germany, July 30 - August 3, 2007. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 54–78. ISBN 978-3-540-73589-2. Disponível em: <http://dx.doi.org/10.1007/978-3-540-73589-2_4>.

[17] GRECH, N.; RATHKE, J.; FISCHER, B. Jequalitygen: Generating equality and hashing methods. In: *Proceedings of the Ninth International Conference on Generative Programming and Component Engineering*. New York, NY, USA: ACM, 2010. (GPCE '10), p. 177–186. ISBN 978-1-4503-0154-1. Disponível em: <http://doi.acm.org/10.1145/1868294.1868320>.

[18] FLANAGAN, D.; MATSUMOTO, Y. *The ruby programming language*. Sebastopol, CA, EUA: O'Reilly Media, Inc., 2008.

[19] NETO, J. J. Adaptive rule-driven devices - general formulation and case study. In: _____. *Implementation and Application of Automata: 6th International Conference, CIAA 2001 Pretoria, South Africa, July 23–25, 2001 Revised Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 234–250. ISBN 978-3-540-36390-3. Disponível em: <http://dx.doi.org/10.1007/3-540-36390-4_20>.

[20] UNGAR, D.; SMITH, R. B. Self: The power of simplicity. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 22, n. 12, p. 227–242, dez. 1987. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/38807.38828>.

[21] FLANAGAN, D. *JavaScript: the definitive guide*. Sebastopol, CA, EUA: O'Reilly Media, Inc., 2006.

[22] KICZALES, G. et al. Aspect-oriented programming. In: AKŞIT, M.; MATSUOKA, S. (Ed.). *ECOOP'97 — Object-Oriented Programming: 11th European Conference Jyväskylä, Finland, June 9–13, 1997 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. p. 220–242. ISBN 978-3-540-69127-3. Disponível em: <http://dx.doi.org/10.1007/BFb0053381>.

[23] JONES, R.; HOSKING, A.; MOSS, E. *The garbage collection handbook: the art of automatic memory management*. London: Chapman & Hall/CRC, 2011.

[24] PARNIN, C.; BIRD, C.; MURPHY-HILL, E. Adoption and use of java generics. *Empirical Software Engineering*, v. 18, n. 6, p. 1047–1089, 2013. ISSN 1573-7616. Disponível em: <http://dx.doi.org/10.1007/s10664-012-9236-6>.

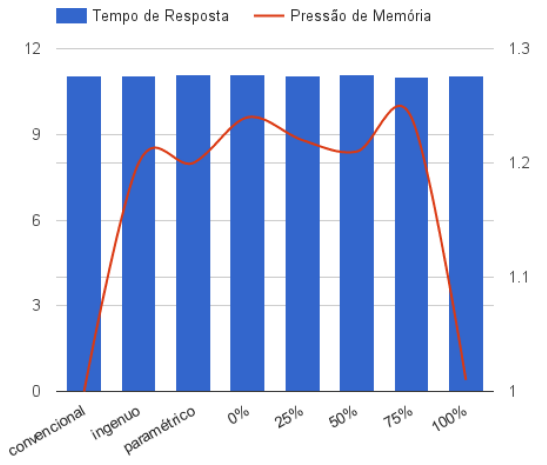
[25] ZIBIN, Y. et al. Object and reference immutability using java generics. In: *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. New York, NY, USA: ACM, 2007. (ESEC-FSE '07), p. 75–84. ISBN 978-1-59593-811-4. Disponível em: <http://doi.acm.org/10.1145/1287624.1287637>.

[26] ZIBIN, Y. et al. Ownership and immutability in generic java. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 45, n. 10, p. 598–617, out. 2010. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/1932682.1869509>.

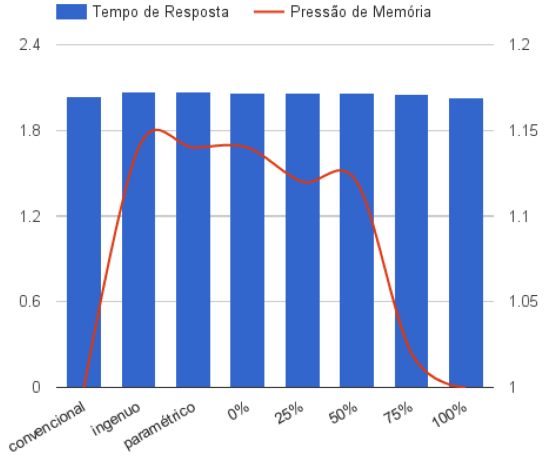
[27] SHACHAM, O.; VECHEV, M.; YAHAV, E. Chameleon: Adaptive selection of collections. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 44, n. 6, p. 408–418, jun. 2009. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/1543135.1542522>.

APÊNDICE A
DETALHAMENTO DOS RESULTADOS DO EXPERIMENTO

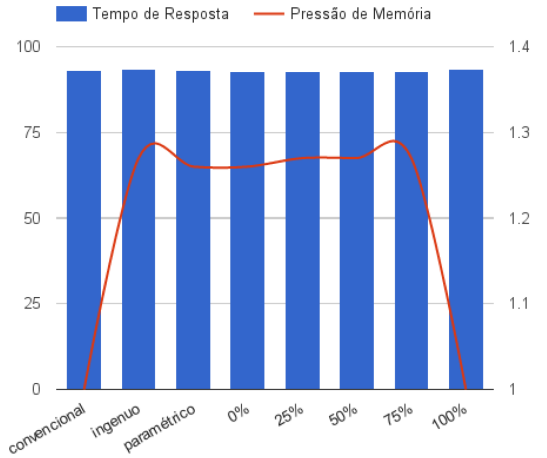
A. *avrora*



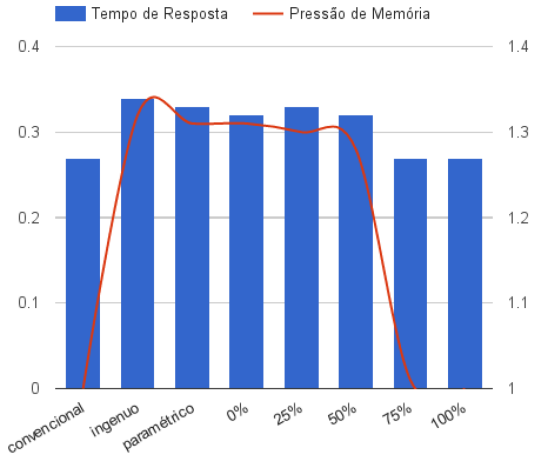
B. *batik*



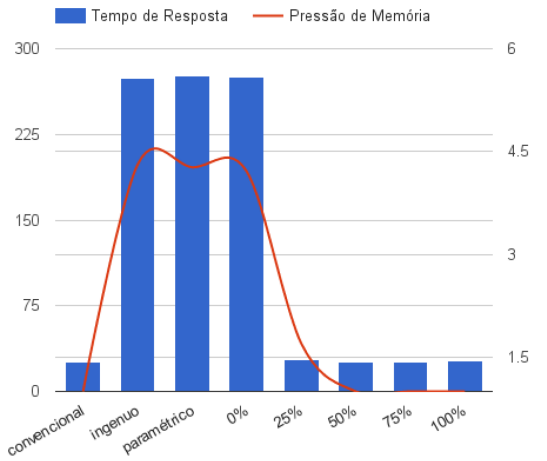
C. *eclipse*



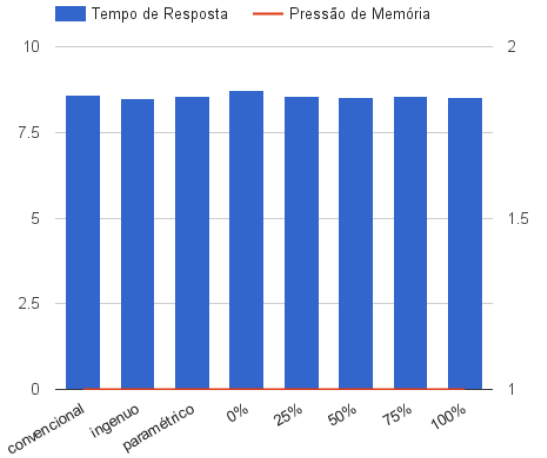
D. *fop*



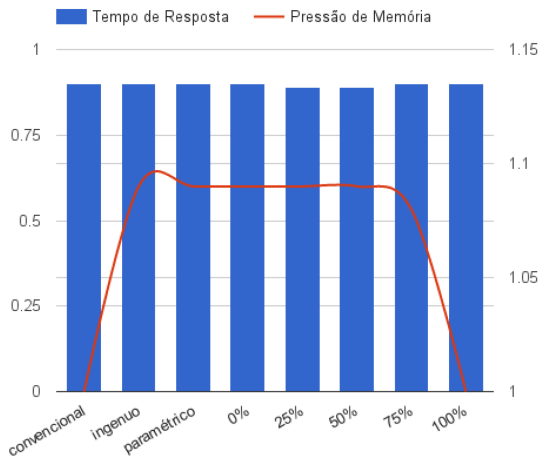
E. *h2*



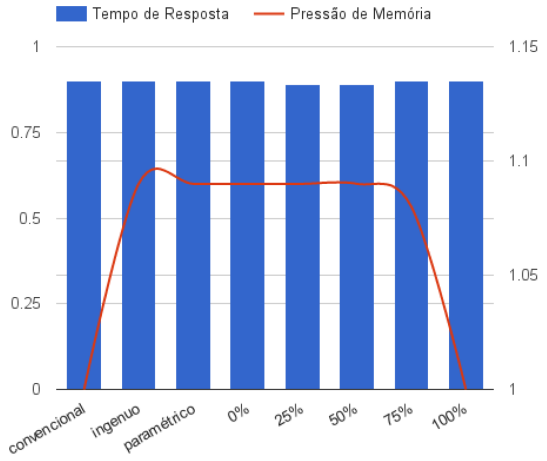
F. *jython*



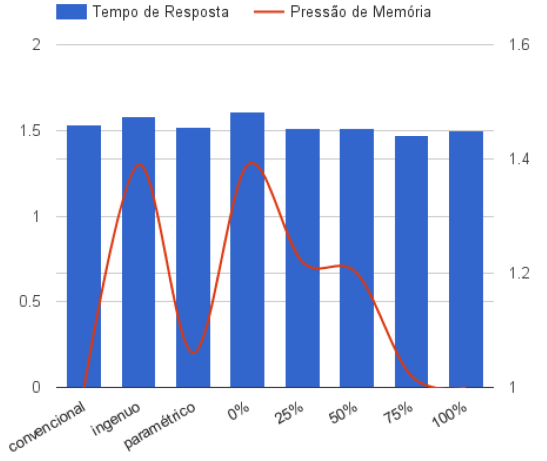
G. luindex



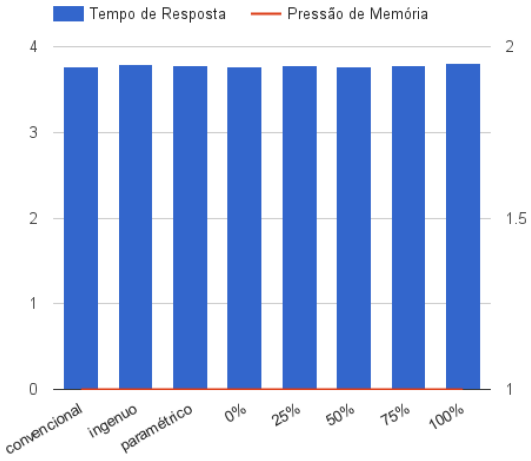
H. lusearch



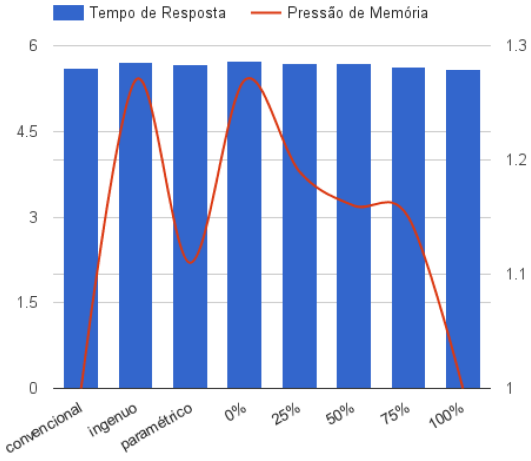
I. pmd



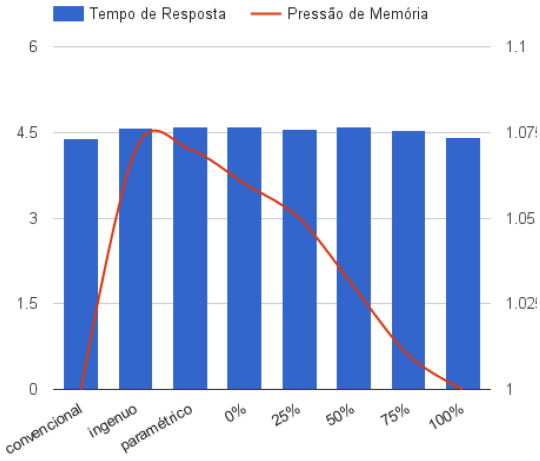
J. sunflow



K. tomcat



L. xalan



O Uso de Conceitos de Injeção de Dependência no Desenvolvimento de Aplicações Adaptativas

C. R. Rossini Junior, A. R. Camolesi

Resumo — ***A injeção de dependência é uma técnica que pretende inserir dependência em uma classe por meio de código externo. O presente trabalho descreve a utilização da injeção de dependência junto com a tecnologia adaptativa para o desenvolvimento de aplicações complexas. Para ilustrar o estudo, foi realizado a criação de um jogo de dominó adaptativo, o qual são introduzido os conceitos de injeção de dependência e da tecnologia adaptativa em sua criação, como forma de demonstrar o estudo de caso.***

Palavras Chaves — ***Tecnologia Adaptativa, Injeção de Dependência, Orientação a Objetos.***

I. INTRODUÇÃO

QUANDO se fala em sistemas, logo se vem a mente sistemas de computadores que automatizam tarefas diárias de uma empresa, mas sua atuação é mais ampla, para acompanharmos essa evolução é necessário entender a evolução do desenvolvimento científico e da inteligência humana. Além do desenvolvimento científico, parâmetros contidos em problemas crescem de uma forma assustadora e cada vez mais complexa. Impossibilitando que apenas uma área compreenda todas as informações e explicações desses fenômenos existentes.

Sistemas devem ser construídos para atender as demandas das empresas. Neste contexto os desenvolvedores de softwares a cada dia mais têm se deparado com aplicações complexas e que tem que ser adaptadas às exigências mais fundamentais segundo as características de cada empresa.

Aplicações complexas são caracterizadas por componentes e aspectos cuja estrutura e comportamento, comumente, podem modificar-se (Camolesi, 2004a). Tais aplicações possuem um comportamento inicial definido por um conjunto de ações que desempenham suas funções elementares, e durante a execução podem ter o seu comportamento modificado para dar suporte a novas funcionalidades. Tais modificações são decorrentes dos estímulos de entrada a que são submetidos no sistema e/ou da ocorrência de suas ações internas.

Uma técnica utilizada para auxiliar os projetistas na modelagem de aplicações com comportamento modificável é a tecnologia adaptativa (NETO, 1993). A tecnologia adaptativa envolve um dispositivo não-adaptativo (subjacente) já existente em uma camada adaptativa que permite realizar mudanças no comportamento da aplicação definida (Pistori, 2003).

Estudos já foram realizados com o objetivo de implementação de aplicações adaptativas. CASACHI (2011) realizou um estudo que teve por objetivo apresentar o uso da Programação Orientada a Aspectos (KICZALES, 1997) no desenvolvimento de aplicações que utilizam os conceitos de Tecnologia Adaptativa. Tal estudo permitiu a implantação da Tecnologia Adaptativa de uma forma fácil e segura, além de permitir que novas funcionalidades (aspectos) sejam adicionadas ao software sem a necessidade de modificar o código fonte já produzido. O programador deve apenas acrescentar novos aspectos ao software e o mesmo se adapta ao código já existente.

O objetivo deste trabalho foi estudar o conceito de desenvolvimento de aplicações complexas com foco no uso de Injeção de Dependência para implementação dos conceitos da Tecnologia Adaptativa com foco na resolução de tais problemas. Por fim, para ilustrar os estudos realizados foi desenvolvido um jogo, com base nas regras de um jogo de Dominó Adaptativo para demonstrar os conceitos estudados.

Este trabalho está organizado da seguinte forma, na Seção 2 são apresentados os principais conceitos de Tecnologia Adaptativa. A seguir, na Seção 3 são descritas as técnicas para o uso de injeção de dependência. Na Seção 4 é apresentado o estudo de caso fundamentado nos conceitos estudados e no jogo de Dominó Adaptativo. Por fim, na Seção 5 são apresentadas algumas conclusões e trabalhos futuros.

II. TECNOLOGIA ADAPTATIVA

Um dos primeiros trabalhos relacionado ao estudo da Tecnologia Adaptativa é o apresentado por NETO e MAGALHÃES (1981), que fornece uma visão de métodos de análise sintática e de geração de reconhecedores sintáticos. Esse trabalho foi resultado de um primeiro esforço em busca da inclusão dos conceitos de mecanismos adaptativos em um sistema para apoio à construção de compiladores.

Na sequencia, foram realizados estudos com objetivo de resolução de problemas relacionados a compilação e em (NETO, 1993) foi apresentado o autômato e o transdutor adaptativo como dispositivos de reconhecimento e transdução sintática.

Com base no trabalho de Neto (1993), foram realizados outros trabalhos nos quais foi aplicada a tecnologia adaptativa no projeto de sistemas reativos. Almeida (1995) apresentou uma evolução da notação de Statechart (HAREL et al., 1987), na qual foram acrescentadas características provenientes da teoria de Autômatos Adaptativos.

Um estudo que teve por objetivo melhorar a especificação de um conjunto de sistemas reativos complexos e sincronizados entre si pode ser encontrado em (NOVAES, 1997).

Um Ambiente de Desenvolvimento de Reconhecedores Sintáticos Baseado em Autômatos Adaptativos foi descrito em (PEREIRA, 1997). Este trabalho introduziu uma ferramenta de auxílio ao desenvolvimento de reconhecedores sintáticos denominada Reconhecedor Sintático para Windows (RSW) (PEREIRA, NETO, 1999).

Um formalismo paralelo ao autômato, o qual visava facilitar o desenvolvimento de linguagens complexas ou outras aplicações que necessitassem especificar linguagens dependentes de contexto na forma de gramáticas foi denominado como Gramática Adaptativa (IWAI 2000). Tal formalismo possui como característica principal a capacidade de se alterar a medida que vai sendo feita a geração da sentença pertencente à linguagem que é representada pela gramática adaptativa.

Rocha (2000) elaborou um estudo que visa o desenvolvimento de um método de construção de modelos e resolução de problemas complexos utilizando-se dispositivos adaptativos. Para tal, foram realizados estudos comparativos entre autômatos adaptativos, redes neurais, algoritmos genéticos e agentes, extraindo destes dispositivos, características que permitiram a composição do modelo denominado Busca de Soluções por Máquina Adaptável (BSMA) (ROCHA; NETO, 2000).

Neto (2001) busca características comuns presentes nos dispositivos adaptativos dirigidos por regras, o que permite a um especialista estender um dispositivo dirigido por regras para suportar tecnologia adaptativa. Com base nisso, Camolesi e Neto (2003) apresentaram uma proposta de extensão do dispositivo Interaction System Design Language (ISDL) (QUARTEL, 1997) definindo então o dispositivo ISDLAdp.

No trabalho desenvolvido por Pistori (2003) foi empregada a Tecnologia Adaptativa para facilitar a interação de jogadores que possuem deficiência motora. Foi desenvolvido um jogo tradicional, o "Jogo da Velha", que permite ao jogador escolher o local onde jogará utilizando para isto a direção do olhar por meio de uma câmera. Outros trabalhos relacionados ao uso da Tecnologia Adaptativa empregada a jogos também foram realizados, porém estes foram apenas exemplos em sala de aula e não foram realizadas publicações sobre os mesmos. Neste sentido espera que o desenvolvimento desta pesquisa possa contribuir com publicações para esta área. O principal diferencial na Tecnologia Adaptativa é a forma razoavelmente simples que podemos transformar teorias já existentes, bem como fazer um reaproveitamento e estruturação destas para melhorar sua capacidade de respostas e interação.

O desenvolvimento níveis de dificuldades com Tecnologia Adaptativas faz com que seja facilitada a interação com o usuário, sem a necessidade de uma base de dados contemplando todas as possíveis reações de um jogador, porque cada regra inicial pode ser modificada de acordo com o ambiente ao qual está sendo trabalhado de forma que não necessita de alguém programando isso a cada nova ocorrência encontrada.

Este conceito de auto-modificação da Tecnologia Adaptativa, torna a Inteligência Artificial (muito encontrada em jogos que exigem determinados tipos de respostas ao usuário), possa ser trabalhada de forma mais simples, e eficiente desde que seja feito de forma correta o seu desenvolvimento, seguindo passos que por mais simples que pareçam, mostrem uma complexidade no que diz a atenção dispensada para não ter um sistema com falhas futuras.

III. INJEÇÃO DE DEPENDÊNCIA

Nesta seção, serão apresentados conceitos ligados a injeção de dependência.

A. PADRÃO DE PROJETOS

Para Alexander (1978) um padrão de projeto descreve um problema o qual ocorre inúmeras vezes em um determinado contexto, e descreve ainda a solução para esse problema, de modo que essa solução possa ser reutilizada sistematicamente em distintas situações. Alexander idealizou esta ideia dentro do contexto da engenharia civil, porém anos depois estes conceitos foram trazidos dentro da engenharia de software.

Na projeção de um software, é muito comum em que o projetista encontre algum problema o qual possa ocorrer durante a implementação deste projeto, com isto, o projetista pode utilizar algum dos padrões de projetos que corresponde ao seu problema, para que consiga solucionar de uma maneira eficaz.

Muitos padrões de projetos estão ligado a qualidade do código, problemas de acoplamento e coesão são muito comuns em grandes projetos, o que dificulta novas implementações de funcionalidades após a finalização do projeto, com isto, existem diversos padrões de projetos os quais buscam minimizar estes problemas.

B. ACOPLAMENTO

O acoplamento significa o quanto uma classe depende de uma outra classe para realizar as suas funcionalidades.

Uma classe que possui um alto nível de acoplamento, irá ser responsável por instanciar suas dependências (objetos de outras classes). Em consequência, qualquer alteração nas classes das dependência, poderá ocasionar algum erro na classe que é responsável de instanciar essa dependência.

Já uma classe que possui um baixo nível de dependência, significa que esta classe não é responsável pela a criação de suas dependências.

O ideal seria buscar sempre um baixo acoplamento entre as classes, porém muitas das vezes para se buscar este baixo nível de acoplamento requer muito trabalho na projeção do projeto, entretanto o resultado de um projeto de software com um baixo nível de acoplamento é satisfatório, pelo o fato de uma manutenção facilitada e a possibilidade de alterações das funcionalidades deste software.

Devido a estes motivos, existem diversos padrões de projetos os quais buscam diminuir o acoplamento entre as classes ou módulos de um sistema, facilitando para o projetista o desenvolvimento deste projeto.

C. INVERSÃO DE CONTROLE

A inversão de controle é um padrão de projetos o qual visa diminuir o acoplamento de uma classe.

O principio básico da inversão de controle é fazer com que os controles de uma classe seja invertidos, no caso, ao se realizar a inversão de controle em uma classe, faz com que as responsabilidade que essa classe possui seja retirada dela, desta forma, a classe não será responsável por instanciar suas dependência, fazendo com que esta classe adquire um baixo nível de acoplamento.

Muitos frameworks utilizam do padrão de inversão de controle, devido ao fato das classes poderem ser reutilizáveis e de ser possível de criar novas funcionalidades sem precisar ter que alterar as já existentes, facilitando a maneira em que o usuário irá a utilizar e as possíveis futuras manutenções nessa framework.

D. INJEÇÃO DE DEPENDÊNCIA

A injeção de dependência é uma das maneiras de se realizar a inversão de controle. Ela permite com que seja retirado a responsabilidade de uma classe tenha que instanciar suas dependência, passando a tarefa de instanciar essas dependência para código externo.
Uma das vantagens de se utilizar o padrão de injeção de dependência é a facilidade de se aplicar ela, existem três maneiras de se aplicar das quais são as mais utilizadas: Injeção por Construtor, Injeção por Método Setter e Injeção por Interface.
Para exemplificar a maneira de se aplicar a injeção de dependência dentro de uma classe, será demonstrados alguns códigos os quais irão demonstrar o método injeção por construtor e a injeção utilizando o método setter escritos na linguagem de programação C#.

```
public class Nota
{
    private Disciplina disciplina;
    private Aluno aluno;
    private int valor;

    public Nota(Disciplina disciplina, Aluno aluno)
    {
        this.disciplina = disciplina;
        this.aluno = aluno;
    }
}
```

Figura 1. Código exemplificando a injeção por construtor.

No caso demonstrado, a classe Nota não será responsável de instanciar os objetos de suas dependências, devido ao fato desses objetos serem instanciados fora da classe e pelo fato deles serem passado através dos parâmetros do construtor.

```
public class Nota
{
    private Disciplina disciplina;
    private Aluno aluno;
    private int valor;

    public setNota(Aluno aluno)
    {
        this.aluno = aluno;
    }
    ...
}
```

Figura 2. Código exemplificando a injeção pelo método setter.

Neste caso, um objeto da dependência é entregue a classe a partir de um método dela, sendo assim, caso a classe necessite utilizar desta dependência para executar alguma ação a qual a dependência seja necessária, é necessário com que se utilize o método para receber o objeto desta dependência.

IV. ESTUDO DE CASO

Nesta seção inicialmente será apresentado os conceitos relacionado as regras de um jogo de Dominó Adaptativo. Tal jogo foi escolhido por ser um jogo que não demanda de muito tempo para o desenvolvimento da interface gráfica para o mesmo, com isso, o foco do estudo de caso seria a implementação dos conceitos de tecnologia adaptativa e de injeção de dependência para a a construção do jogo de dominó adaptativo.

A. DOMINÓ ADAPTATIVO

O dominó adaptativo é uma ideia apresentada por (NETO, 2007), o qual introduz fundamentos da tecnologia adaptativa dentro de um jogo de dominó.
Em um jogo de dominó comum, para se realizar uma jogada, é necessário com que a pedra a ser jogada, tenha ao menos uma das pontas com valores iguais a uma das duas extremidades das pontas do tabuleiro. Na proposta do dominó adaptativo, o jogador tem a possibilidade de alterar as regras de encaixe do jogo em tempo de execução de partida, em função disto, ao se fazer uma jogada, a pedra não irá necessariamente ter que possuir um dos lados iguais a alguma das extremidades das pontas do tabuleiro, devido as regras de encaixe terem sido modificadas.
No início de uma partida, são introduzidas aleatoriamente dentro do jogo, algumas pedras adaptativas, essas pedras permitem com que as regras de encaixe sejam alteradas na execução da jogada, uma vez que a regra seja modificada, essa modificação permanecerá até o fim do jogo, ao menos que alguma outra pedra adaptativa seja jogada e que ela tenha influência na regra modificada.

B. ARQUITETURA DO SISTEMA

Primeiramente foi desenvolvido um jogo de dominó tradicional (não adaptativo). Com base no jogo produzido foi implementado os conceitos da Tecnologia Adaptativa. Após a finalização da base inicial do jogo, foi realizada a implementação dos métodos que permite que uma função adaptativa modifique em tempo de execução as regras do jogo de dominó, por fim, para realizar o desenvolvimento dos conceitos da tecnologia adaptativa foi utilizadas os conceitos do padrão de injeção de dependências.
O Dominó Adaptativo teve suas classes modeladas pela a ferramenta Astash Communty e foi desenvolvido na plataforma .NET utilizando Visual Studio Community 2015.

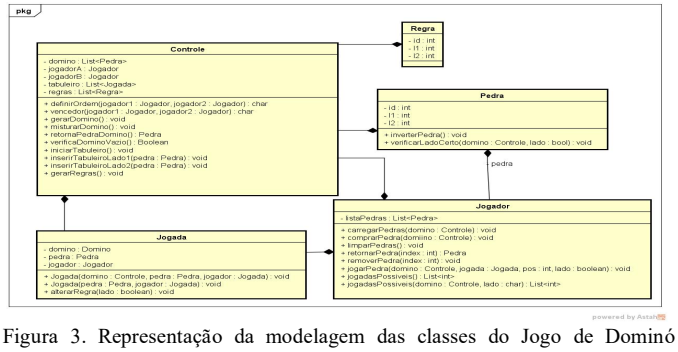


Figura 3. Representação da modelagem das classes do Jogo de Dominó Adapatativo.

A modelagem de classe foi estruturada de forma que pudesse criar as regras de encaixe, onde as jogadas só poderiam acontecer se estivesse de acordo com a regra vigente. Essas regras são definidas na classe Dominó no método de geração do dominó. Tal método tem por função gerar as 7 regras padrões e as 28 pedras do jogo.
Com as regras e pedras geradas, é preciso embaralhar essas pedras e sortear qual delas vão ser as pedras adaptativas, para que o jogo não se torne vicioso, utiliza-se de uma função aleatória com base nos valores da data e hora do embaralhamento como um índice aleatório, conforme demonstrado na Figura 4.

```
public void misturarDomino()

   int seed = DateTime.Now.Millisecond *
DateTime.Now.Year * DateTime.Now.Second;
   Random rand = new Random(seed);
   for (int i = 0; i < 8; i++)
   {
       for (int j = 0; j < domino.Count; j++)
       {
            int iRand = rand.Next() % 28;
            Pedra aux = domino[j];
            domino[j] = domino[iRand];
            domino[iRand] = aux;
       }
   }
   for (int i = 0; i < 4; i++)
   {
       int iRand = rand.Next() % 28;
       domino[iRand].adap = true;
   }
}
```

Figura 4. Representação do método de embaralhamento das pedras do dominó

Para o jogo não se tornar injusto, é definido um limite de pedras que possam realizar jogadas adaptativas, as quais são escolhidas aleatoriamente dentro do tabuleiro já com as pedras embaralhadas.
O dominó adaptativo também possui um método de verificar as possíveis jogadas, o qual é um método polimórfico, o qual irá fazer a verificação de todas as possíveis jogadas do jogador, conforme as pedras que o jogador possua na mão e as regras de encaixe vigentes. Este é um método polimórfico devido ao fato de que caso o tabuleiro esteja vazio, a verificação das jogadas possíveis é diferente da verificação de quando estiver com uma ou mais pedras no tabuleiro. O método foi utilizado para verificar se uma jogada é realmente válida, de acordo com as regras vigente, as quais podem ser alteradas por uma jogada adaptativa tanto do jogador como do computador.
Para ser possível implementar a tecnologia adaptativa, foi utilizado dos conceitos do padrão de injeção de dependência, desta forma, foi possível realizar a implementação da tecnologia adaptativa sem precisar ter que alterar a estrutura do jogo de dominó.
As regras do dominó foi utilizado como o mecanismo subjacente desse sistema, conforme foi definido na projeção inicial do dominó adaptativo, as regras irão se modificar apenas no momento de uma jogada, sendo ela uma jogada adaptativa, contanto a classe jogada precisava deste mecanismo subjacente para realizar as alterações das regras, o qual estava contido na classe Controle, porém não cabe a classe Jogada instanciar essa dependência, portanto foi utilizado dos conceito da injeção de dependência para retirar a responsabilidade da classe Jogada de instanciar um objeto da classe Controle, passando esta responsabilidade para um código externo.
A técnica utilizada para a injeção de dependência na classe jogada foi por via Construtor.

```
public Jogada(Controle domino, Pedra pedra, Jogador
jogador)
   {
        this.domino = domino;
        this.pedra = pedra;
        this.jogador = jogador;
   }
```

Figura 5. Utilização da Injeção de Dependência na classe Jogada.

Com o objetivo de demonstrar o uso da tecnologia adaptativa para alterar as regras de encaixe do jogo de dominó é apresentado no Figura 6, o qual é um método que é utilizado quando alguma pedra adaptativa é usada no jogo. Devido ao fato que este método depende do lado em que a peça foi jogada, o código precisou ser dividido em duas partes para se fazer as devidas verificações, no caso, o código apresentado corresponde a peça que foi jogada no inicio do tabuleiro.

```
public void alterarRegra(bool lado)
...
for (int i = 0; i < domino.regras.Count; i++)
{
    if (domino.regras[i].l1 == pedra.l2)
    {
        domino.regras[i].l2 =
                        domino.tabuleiro[0].pedra.l1;
        i = domino.regras.Count;
    }
}
...
```

Figura 6. Método Adaptativo.

O método inicia realizando uma busca nas regras, para que assim localize a regra que será modificada. Ao ser localizada é realizado uma alteração na regra, para que as regras fiquem condizentes a ultima jogada adaptativa.

C. EXECUÇÃO DA APLICAÇÃO

Para fim de experimento da aplicação, uma interface com poucos componentes foi criada.

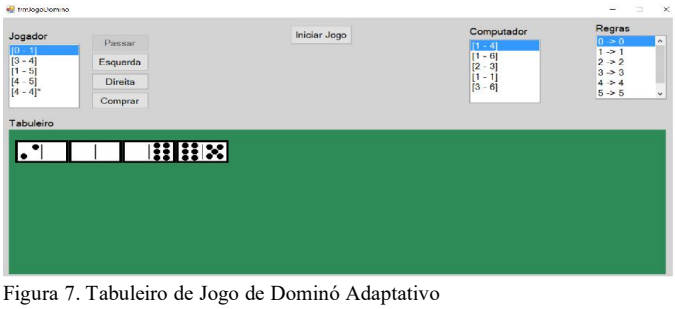


Figura 7. Tabuleiro de Jogo de Dominó Adaptativo

Para realização de testes, as pedras do computador são visíveis, as quais as jogadas são realizadas automaticamente de acordo com as regras vigentes.
As pedras que possuírem um asterisco,significa que ela é uma pedra adaptativa, com isso, o jogador tem a possibilidade de realizar uma jogada com ela em qualquer lado do tabuleiro, independentemente se é uma combinação válida ou não, caso

a combinação não for válida, o método adaptativo é utilizado e modifica as regras do jogo, as quais irão ficar em vigor até o fim do jogo.

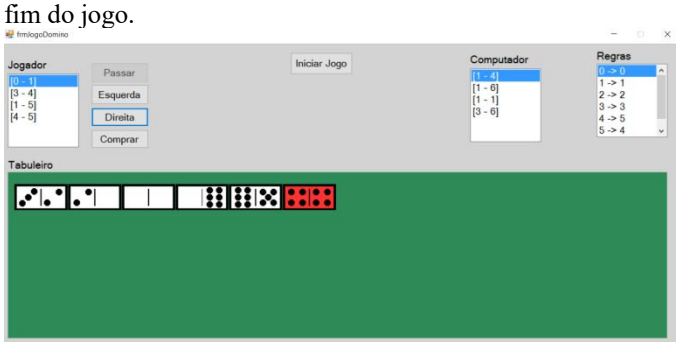


Figura 8. Realização de uma jogada adaptativa.

A figura 8 demonstrou como ficam as regras de encaixe após acontecer uma jogada adaptativa, no caso referente, as pedra que tiverem lado com valor igual a 4 só fará combinação com pedras que tiverem valor igual a 5. Após a jogada adaptativa, o computador já identificou as novas regras e realizou uma nova jogada utilizando delas.

O jogo acaba quando as peças do jogador ou do computador acabarem, caso não haver mais pedras para comprar e não existir mais nenhuma possibilidade de jogada, é realizado uma soma utilizando como base os valores das pedras, o jogador que tiver mais pontos será o vencedor.

V. CONCLUSÕES

Considerando-se os dados apresentados sobre a tecnologia adaptativa e a injeção de dependência, é possível constatar que a utilização de injeção de dependência pode contribuir positivamente na implementação da tecnologia adaptativa em um sistema, pois ao se possuir um sistema em que suas classes possuem um baixo acoplamento, o qual a injeção de dependência contribuiu para isto, a implementação de uma nova funcionalidade, como no caso, a tecnologia adaptativa, é realizada de uma maneira simples, ao contrário de um sistema que possuem suas classes com muito acoplamento, o que irá dificulta o processo de implementação da tecnologia adaptativa, devido a provável necessidade de realizar mudanças em várias partes do sistema.

Após a implementação da tecnologia adaptativa no jogo de dominó, foi possível perceber que mudou significativamente a maneira de se jogar, pois o jogador sempre irá possuir regras diferente, fazendo com que o jogo tenha que prestar mais atenção nas regras que mudaram constantemente.

Existem diversos tipos de padrão de projetos além do padrão de injeção de dependência, fica como trabalhos futuros, o estudo da aplicações de outros padrões de projeto dentro de um contexto que utiliza a tecnologia adaptativa, o desenvolvimento de uma interface com mais recursos para o Dominó Adaptativo e a implementação de um novo modo de jogo, o qual irá possibilitar jogos online por redes.

REFERÊNCIAS

[1] ALMEIDA, J.R. STAD - Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos. Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 1995.

[2] CAMOLESI, A.R.; NETO, J.J. An adaptive model for specification of distributed systems. IX Congreso Argentino de Ciencias de la Computación, La Plata, Argentina, 6-10 de Outubro, 2003.

[3] CAMOLESI, A.R.; NETO, J.J. Modelagem Adaptativa de Aplicções Complexas. XXX Conferencia Latinoamericana de Informática - CLEI'04. Arequipa - Peru, Setiembre 27 - Octubre 1, 2004a.

[4] CAMOLESI, A.R.; NETO, J.J. Representação Intermediária para Dispositivos Adaptativos Dirigidos por Regras. 3rd International Information and Telecommunication Technologies Symposium, UFSCar, São Carlos, Brasil, 2004b.

[5] CASACHI, R. A. Aplicação do Paradigma de Programação Orientada a Aspectos no Desenvolvimento de Software. Trabalho de Conclusão de Curso, Bacharelado em Ciência da Computação, FEMA-IMESA, 2011.

FOWLER, Martin. Inversion of Control Containers and the Dependency Injection pattern. Disponível em: <http://martinfowler.com/articles/injection.html>. Acesso em 20 set. 2016.

[6] ALEXANDER, C. A Pattern Language (Estados Unidos: Oxford University Press), 1977.

[7] FREEMAN Eric.; FREEMAN Elizabeth. Use a Cabeça! Padrão de Projetos. Rio de Janeiro: Alta Books. 496 p.

[8] HAREL D. et al. On the formal semantics of statecharts. In: Symposyum on logic in Computer Science, 2°, Ithaca, Proceedings, IEEE Press, pp. 54-64, New York, 1987.

[9] IWAI; M.K. Um formalismo gramatical adaptativo para linguagens dependentes de contexto. Tese de Doutorado, USP, São Paulo, 2000.

[10] KICZALES, Gregor; LAMPING, John; MENDHEKAR, Anurag; MAEDA, Chris; LOPES, Cristina Videira; LOINGTIER, Jean-Marc. Aspect-Oriented Programming. In: European Conference on Object-Oriented Programming (ECOOP), 06,1997. Finlândia. Anais Springer-Verlag LNCS 1241, 06, 1997.

[11] MACORATTI, Jose Carlos. .NET - Inversão de Controle (IoC) e Injeção de Dependência (DI). Disponível em: <http://www.macoratti.net/11/07/ioc_di1.htm>. Acesso em 20 set. 2016.

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

[13] NETO, J.J. Contribuições à metodologia de construção de compiladores. Tese de Livre Docência, USP, São Paulo, 1993.

[14] NETO, J.J. e MAGALHÃES, M.E.S. Um Gerador Automático de Reconhecedores Sintáticos para o SPD. VIII SEMISH - Seminário de Software e Hardware, pp. 213-228, Florianópolis, 1981.

[15] NETO, J.J.; SILVA, P.S.M. An adaptive framework for the design of software specification language. Proceedings of the International Conferenceon Adaptive and Natural ComputINing Algorithms (ICANNGA), Springer Verlag editor, pp. 349-352, Coimbra University, Coimbra, Portugal, 21 - 23 march , 2005.

[16] NOVAES, J.M. Um formalismo adaptativo com mecanismo de sincronização para aplicações concorrentes. Dissertação de Mestrado, USP, São Paulo, 1997.

[17] PEREIRA, J.C.D.; NETO, J.J. Um Ambiente de Desenvolvimento de Reconhecedores Sintáticos Baseado em Autômatos Adaptativos. II Simpósio Brasileiro de Linguagens de Programação - SBLP97, pp. 139-150, Campinas, 1997.

[18] PEREIRA, J.C.D. Ambiente integrado de desenvolvimento de reconhecedores sintáticos, baseado em autômatos adaptativos. Dissertação de Mestrado, USP, São Paulo, 1999.

[19] PISTORI, H. Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações. Tese de Doutorado, USP, São Paulo, 2003.

[20] QUARTEL, D. Actions Relations – Basic design concepts for behaviour modelling and refinement. Ph.D Thesis Twente University, Netherlands, 1997.

[21] ROCHA, R.L.A. Um método de escolha automática de soluções usando tecnologia

Carlos Roberto Rossini Junior Estudante de Bacharel em Ciência da Computação na Fundação Educacional do Município de Assis (FEMA) em Assis, São Paulo, Brasil. Entre os anos de 2015 a 2018. Suas principais áreas de pesquisas são: Orientação a Objetos e desenvolvimento de aplicação .NET.

Almir Rogério Camolesi possui graduação em Processamento de Dados pela Fundação Educacional do Município de Assis (1992), mestrado em Ciência da Computação pela Universidade Federal de São Carlos (2000) e doutorado em Engenharia de Computação e Sistemas Digitais pela Universidade de São

Paulo (2007). Atualmente é professor titular da Fundação Educacional do Município de Assis. Tem experiência na área de Ciência da Computação, com ênfase em Tecnologias Adaptativas, atuando principalmente nos seguintes temas:. tecnologia adaptativa, computação distribuída, teoria da computação, modelagem abstrata, ensino a distância, algoritmos e programação para web.

Aplicação de tabelas de decisão adaptativas em sistemas de controle de crescimento de plantas

L. P. Moreira, *Ms. Student, Escola Politécnica da Universidade de São Paulo*

M. R. Pereira-Barretto, *Prof. Dr., Escola Politécnica da Universidade de São Paulo*

***Abstract* — A automação agrícola tem sido desenvolvida de forma significativa no mundo. Para contribuir com este desenvolvimento, o presente trabalho descreve uma aplicação de abordagens metodológicas de tabelas de tomadas de decisão adaptativas, tendo em vista um uso futuro em sistemas de controle, juntamente com aplicações e integração nos sistemas existentes na área do crescimento de plantas em ambientes controlados.**

***Keywords* — TDA, Automação Agrícola, Adaptatividade, Sistemas de Controle.**

I. INTRODUÇÃO

O estudo sistemático do crescimento dinâmico de plantas tem seu modelamento como um desafio real para pesquisadores, dada a interdisciplinaridade dos aspectos levados em consideração para sua elaboração, como o formalismo matemático, o conhecimento sobre botânica e fisiologia, agronomia, engenharia e ciência da computação, a fim de criar possibilidades de operação mais ricas e produtivas[1].

A automação agrícola vem se desenvolvendo significativamente no mundo. É de extrema importância em produção de alta tecnologia, garantindo precisão e confiabilidade. Com o avanço em sistemas de produção agrícolas com alto grau de controle e possibilidade de automação, esta automação a atenção para sua alta produtividade e baixo consumo de recursos, a exemplo da produção de minitubérculos básicos de batata (*Solanum tuberosum*) com técnica de produção sem solo, aeroponia[2].

Nesse sentido, destaca-se que o sistema aeropônico tem necessidade de elevado nível de controle[2]. O processo de crescimento desse sistema elimina influências indesejadas, como a presença do solo, o que permite avaliar o crescimento de uma planta por inteiro, bem como facilita a elaboração de modelos de crescimento dinâmico baseados em observação com condições de parâmetros relevantes altamente controláveis.

O uso de tecnologia de informação (TI) em agricultura de precisão ajuda a dar suporte a decisões complexas, com grande número de dados e variáveis, em aplicações de tecnologia adaptativa, por exemplo, em solução preliminar do problema de identificação de unidades de gerenciamento diferenciado[3], como potencial tecnologia voltada para o campo.

O papel da internet e de seus sistemas correlatos tomam força com a mudança de relação de comunicação homem-homem e homem-máquina para a relação de comunicação máquina-máquina de forma promissora e revolucionária. Há ainda a tendência da exploração da Internet das Coisas (*Internet of Things IoT*), que favorece uma ampla rede de sensores em combinação para diversas aplicações em sistemas de controle[4]. Isso vem sendo utilizado gradualmente em países como Coreia do Sul em setores simples, como em sistemas de pagamento de varejo até aos mais complexos, como em assistência médica, com grade destaque para aplicações em fazendas inteligentes (*Smart Farm*) com sensoriamento expandido[5].

Este trabalho discute aplicações da tecnologia adaptativa na automação agrícola, com a ferramenta de tabelas de decisão adaptativas em um exemplo de aplicação de controle de manejo preventivo de requeima (*Phytophthora infestans*) em cultivos de batata. Trata-se, portanto de alternativa de substituição de tomada de decisão humana por tomada de decisão adaptativa.

II. REVISÃO BIBLIOGRÁFICA

Dentro dos novos ambientes de produção agrícola, destacam-se os ambientes de cultivo protegido com sistemas de controle. Estes buscam maximizar a produção e minimizar o consumo de energia, água e insumos, considerados células de crescimento autônomas de plantas. Trata-se de uma categoria recente chamada de Fábrica de Plantas (*plant factory*), que em seu design e requerimentos de engenharia analisam diversos parâmetros observáveis e controláveis, de alta relevância para o crescimento otimizado das plantas em controle de malha fechada, tratando desde o sistema de irrigação até o de iluminação artificial por LED[6].

Fora de um sistema convencional de produção agrícola, os novos requisitos de engenharia surgem como desafios a ser vencidos e oportunidades para surgimento de sistemas auxiliares de desenvolvimento de iluminação artificial com LED[7]. Isso permite o uso de sistemas inteligentes de visão com análise de imagens para irrigação de precisão e fornecimento de iluminação artificial de precisão[8].

Um exemplo de caso de uso de novos sensores é o que avalia a dinâmica da espessura das folhas de plantas. Ele pode ser utilizado como novo parâmetro de controle para sistemas de irrigação ou aspersão em sistemas aeropônicos, cuja medida fornece informações fisiológicas importantes e confiáveis na aplicação de irrigação de precisão. Isso permite

fornecer às plantas água somente quando elas necessitam, economizando o máximo de água-doce possível[9].

Outra abordagem agronômica são os chamados sistemas de previsão. São meios alternativos para controle eficiente de doenças, com ações preventivas que visam à redução os custos de produção e de poluição ambiental. Isso é importante para o manejo de doenças limitantes para algumas culturas, como a da batata[10], que podem ser integradas a outros métodos de controle.

Os manejos de soluções nutritivas em sistemas que utilizam fertirrigação, como hidroponia e aeroponia, aparecem como parâmetros controláveis das unidades autônomas de produção. Têm o manejo químico de correção de solução nutritiva de forma manual por produtores analisando a condutividade salina da solução, com possível desequilíbrio nutricional, dando espaço para controles mais elaborados, como de nutrientes individualizados[11].

III. Tabela de decisão adaptativa

O estudo dos diversos métodos de controle, parâmetros observáveis e variáveis controláveis, exemplos de casos de uso e sugestões para manejos, faz-nos inferir que há falta de formalização de regras de controle para sistemas de crescimento de plantas. Há apenas uma sucessão de pequenas malhas de controles, com um direcionamento desconexo de ações humanas e decisões humanas associadas a controles diversos.

Uma vez definida a escolha por produções em sistemas autônomos inteligentes de crescimentos de plantas, com diversas áreas e parâmetros controláveis, deve-se reduzir as interferências de decisão humana e ações humanas diretas diversas, a fim de tornar as unidades produtivas o mais próximo possível de serem totalmente autônomas com sistemas supervisórios reduzidos a ações emergenciais.

No tocante à tecnologia adaptativa, seu papel é fundamental quanto a interpretação de desejos humanos representados por conjunto de decisões a serem tomadas dado um conjunto de condições base para estabelecer uma sequência naturalmente mutável e com multiplicidade de cenários possíveis e ordenamento.

Dentre várias alternativas possíveis encontradas, uma decisão é a escolha de uma dentre as possíveis seguido critérios preestabelecidos, sendo a decisão um processo intrinsecamente lógico, complexo, dinâmico, de puro raciocínio. Tabelas de decisão figuram como ferramenta importante na solução do cenário conflitante. Permitem encontrar a solução para um problema de decisão, favorecendo efetivamente uma conclusão decisiva. A escolha de uma alternativa implica renúncia das demais possíveis, exigindo um maior número de informações possíveis, a fim de reduzir riscos associados à decisão a depender do nível de complexidade[12].

A tecnologia adaptativa é, dentro da computação, uma área de concentração de estudos relacionados às técnicas adaptativas que exploram a capacidade de automodificação dos dispositivos. É apresentada como característica principal, com sua estrutura autorreprogramável dinamicamente, bem como seu comportamento, baseado no movimento das ações durante a operação e desenvolvimento[13].

A característica de modificação dinâmica, adaptando-se ao cenário modificado em resposta a estímulos com reações adequadas ao longo de todo processo, confere ao sistema características de recursos de aprendizagem e inteligência.

O dispositivo adaptativo[13], cuja operação é definida por um conjunto finito de regras que se automodificam dinamicamente, é formado por duas camadas: uma conhecida não adaptativa, que representa o núcleo do sistema; e outra adaptativa, externa, que atua sobre o núcleo, modificando-o, conferindo novo conjunto de regras, dada uma ação de interação com meio externo, ocorrendo de forma autônoma em decorrência das ações adaptativas.

Dentre os dispositivos adaptativos disponíveis, a facilidade de definição e emprego de formalismos já conhecidos como os de tabelas de decisão tradicionais, acrescidas de camada adaptativa, gera a Tabela de Decisão Adaptativa, ferramenta poderosa na descrição de problemas complexos, tornando um cenário difuso em formulação de problema em formato sucinto de apresentação, conferindo ao observador um meio de análise claro e objetivo.

Na camada não adaptativa, há uma organização das informações relativas a condições e ações específicas, e um conjunto de regras regendo a interação entre os possíveis cenários, definidos de forma tradicional (Figura 1). Já a camada adaptativa conserva a mesma forma convencional com a adição das ações a serem executadas, com ações antes da regra ser executada (*before action*) e ações tomadas depois da regra aplicada (*after action*) (Figura 2).

			Regras			
			r1	r2	...	rn
TDC	Condições	c1	T	T		-
		c2	-	T		T
		...				
		cm	-	-		-
	Ação	a1	F	T		T
		a2	F	F		T
		...				
		ap	F	F		T

Figura 1. Estrutura de uma tabela de decisão convencional.

			Regras			
			r1	r2	...	rn
TDC	Condições	c1	T	T		-
		c2	-	T		T
		...				
		cm	-	-		-
	Ação	a1	F	T		T
		a2	F	F		T
		...				
		ap	F	F		T
CAMADA ADAPTATIVA	Before Action	ba1	F	T		T
		ba2	F	F		T
		...				
		baq	F	F		T
	After Action	aa1	F	T		T
		aa2	F	F		T
		...				
		aao	F	F		T

Figura 2. Estrutura de uma tabela de decisão convencional acrescida de camada adaptativa.

A estrutura geral de uma Tabela de Decisão Adaptativa pode ser encontrada com pequenas varrições de representação, como visto[14,12], e segue a formulação base[13] como sendo definida pela dupla TDA = (TDC, CA), em que TDC representa a tabela de decisão convencional e CA o mecanismo adaptativo.

A tabela de decisão não adaptativa é composta pelos elementos TDC = (CT, t_0, R, C, A), em que:

• CT é o conjunto de todas as configurações possíveis da tabela de decisão.

• $t_0 \in$ CT e é a configuração inicial da tabela de decisão.

• R é o conjunto de regras de decisão da tabela: R = $\{r_j , 1 \leq j \leq n\}$.

• C é o conjunto finito das condições (ou critérios) do problema: C = $\{C_i , 1 \leq i \leq m\}$.

• A é o conjunto finito de ações possíveis (ou alternativas) do problema: A = $\{A_k , 1 \leq k \leq p\}$.

Cada regra $r_j \in$ R é formada por um conjunto de valores das condições C_i na regra r_j, podendo assumir três valores, sendo reles T, F ou -:

• T: corresponde à valor verdadeiro.

• F: corresponde à valor falso.

• -: corresponde à valor indiferente.

O conjunto de valores assinalados para as ações A_k a serem executadas na regra r_j, quando satisfeitas as condições C_i para ativar a execução das ações A_k.

O mecanismo adaptativo CA para a TDA é definido pelo conjunto de todas as ações adaptativas.

As ações adaptativas podem ser executadas antes ou depois das regras não adaptativas da tabela, e de acordo com os parâmetros de entrada das funções. Cada ação adaptativa é definida por, pelo menos, uma das ações elementares: consulta às regras da tabela, inclusão de novas regras e exclusão de regras existentes, podendo existir mais de uma ação elementar, também possuindo conjunto de regras como na TDC, mas regras adaptativas RA.

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

Cada regra adaptativa ra_j é definida por (r_j, ba_q, aa_o), em que r_j é a regra não adaptativa e:

- $ba_q \in$ RA: se houver, indica ação ou ações adaptativas a serem executadas antes da regra r_j.
- $aa_o \in$ RA: se houver, indica ação ou ações adaptativas a serem executadas depois da regra r_j.

A ordem de execução segue a sequência:

1. Regra r_j é ativada quando as condições impostas na coluna são satisfeitas, isto é, todos os valores dos critérios na coluna r_j para C_i conferem com os valores predeterminados na TDC.
2. Nessa situação, se houver, executa-se a ação ou ações adaptativas ba correspondentes válidas (pode provavelmente acarretar mudança do conjunto de regras do dispositivo).
3. Aplica-se a ação ou ações da regra subjacente, executadas como na TDC clássica não adaptativa.
4. Nessa situação, se houver, executa-se a ação ou ações adaptativas aa correspondentes válidas (pode provavelmente acarretar novamente na mudança do conjunto de regras do dispositivo).

IV. Modelo proposto

Um estudo de caso é apresentado neste artigo como uma proposta de uso de TDA para gerenciamento de ações de automação e formalização de regras de controle aplicada ao processo de tomada de decisão. Trata-se de uma forma de substituição da tomada de decisão humana no manejo de controle de patógenos limitantes à cultura de batata, especificamente para requeima, em uma unidade autônoma de produção (*plant factory*) em aeroponia, destacando o papel da TDH no diagrama de processo como visto na Figura 3.

A cultura de batata é afetada por vários fatores que causam queda de produtividade com a requeima. Ocupam papel de destaque preocupante ao produtor as aplicações de fungicida em geral, que são excessivas por serem realizadas de forma empírica, sem levar em consideração condições meteorológicas e do ciclo de vida do patógeno, levando ao seu uso indiscriminado, o que eleva o custo de produção, sem uma devida formalização das regras de controle de patógenos[10].

Formulamos a questão como um clássico problema de otimização recursos, reduzindo a uma solução de decisão de único critério, que trata de realizar o menor número de aplicações de fungicidas ao longo do ciclo de produção, mantendo controladas as ocorrências de requeima, sem afetar a produtividade, para aplicação e uso futuro em uma instalação real de produção de minitubérculos de batata semente com técnica de Aeroponia, localizado na cidade de Divinolândia (SP).

O ambiente controlado, objeto de futura implementação da proposta apresentada neste trabalho, é constituído de uma casa de vegetação recoberta por tela antiafídica nas laterais e com filme agrícola no teto; possui também abertura superior para melhor circulação de ar quente e proteção da abertura com tela antiafídica, com estrutura de bancadas elevadas de produção divididas em 8 bancadas de 18 m X 1,5 m, com corredores de 0,80 m entre bancadas, trabalhando com adensamento de plantas de 90 plantas por m², como visto nas Figuras 4 e 5.

Figura 4. Disposição das bancadas com alto adensamento de plantas.

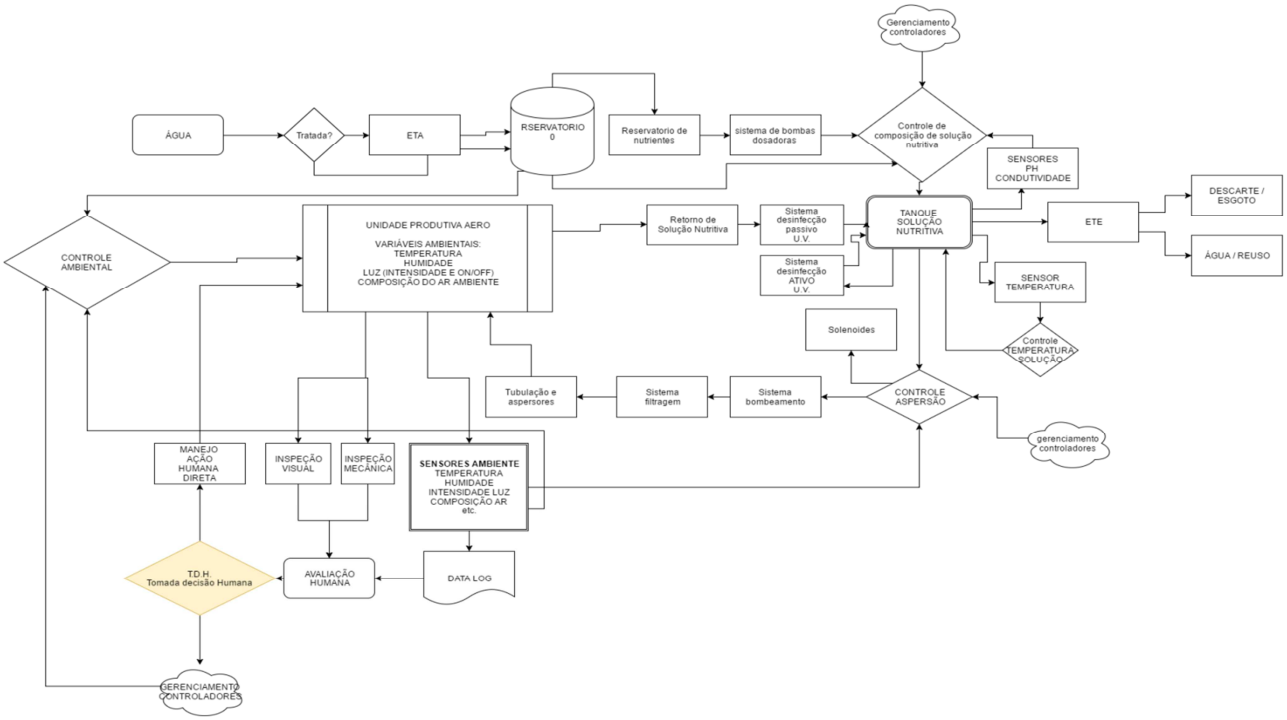


Figura 3. Diagrama de processos de uma unidade autônoma de produção (*plant factory*) em aeroponia.

Figura 5. Casa de vegetação como ambiente controlado.

O alto adensamento de plantas e o ambiente não perfeitamente isolado tornam o ambiente suscetível a contaminações por patógenos, como os causadores da requeima, e são ambientes propícios para alta disseminação e multiplicação, caso manejo não seja feito de maneira correta, interferindo drasticamente na produtividade.

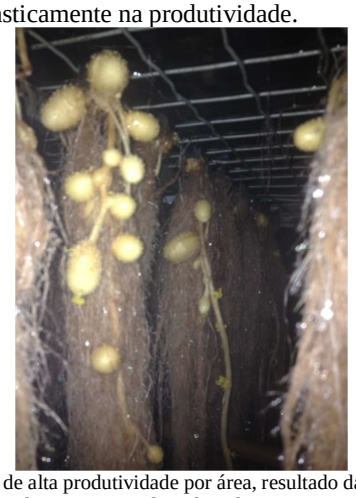


Figura 6 Exemplo de alta produtividade por área, resultado da tecnologia de produção empregada e alto adensamento.

O manejo atualmente é realizado manualmente com bomba costal de 20 L de forma empírica, seguindo recomendações de fabricantes e orientações do agrônomo responsável técnico de forma preventiva, com alto custo de produção envolvido, sendo ótima oportunidade para automatização de processo de manejo.

Figura 7. Exemplo de manejo de aplicação de fungicida convencional.

Os critérios escolhidos, comparando a escolha de método de sistema de previsão entre os sistemas Blitecast e Prophy, são resultados vistos em alguns trabalhos[10], que utilizam as indicações de sensoriamento e registro de valores de severidade diários, descrevendo a TDA para decidir sobre a ação de manejo de acordo com o melhor desempenho de cada método, além das ações emergenciais envolvidas e que alteram a dinâmica.

Os fungicidas a serem utilizados para controle da requeima são mancozebe (Dithane NT – 3,0 kg ha^{-1} do produto comercial), oxicloreto de cobre (Cuprogarb 350 – 4g L^{-1} de água do produto comercial), piraclostrobina + metiram (Cabrio Top – 3,0 kg ha^{-1} do produto comercial), conforme recomendação do ministério da agricultura[15], aplicado em concentração 50% menor indicado para aeroponia, com aplicações alternadas combinadas (Dithane NT e Cuprogarb) com ação de contato e (Cabrio TOP) com ação sistêmica[10].

Optou-se pela montagem da TDA como processo de escolha entre os resultados de métodos de previsão de requeima mais satisfatórios encontrados[10], MBLI24 e MPRO25, com a informação adicional de eventos emergenciais, em cenário de combinação otimizada entre os resultados dos métodos. Nesse sentido, a proposta para o processo decisório segue descrita a seguir:

i. Realizar a leitura dos sensores para obtenção dos valores de severidade acumulados por dia.
ii. Realizar inspeção de campo (local ou remotamente por imagem) para verificar ocorrência de infestação emergencial.
iii. Verificar falha no isolamento da unidade de produção protegida.
iv. Verificar a contagem de dias após transplante (DAT) como contagem do tempo de ciclo de desenvolvimento da planta.
v. Levantar os requisitos de cada critério para execução das ações não adaptativas.
vi. Levantar os requisitos de cada combinação de ações para execução das ações adaptativas e mudanças no conjunto de regras do dispositivo.
vii. Criar regras que são solução do problema e que continuem sendo após mudanças em seu conjunto após ação adaptativa.
viii. Criar a TDC.
ix. Criar a camada CA.
x. Mostrar a dinâmica adaptativa de funcionamento em dois passos.

A TDA do modelo proposto segue na Figura 8, em sua condição t_0, sem necessidade verificada de ações adaptativas a serem executadas depois da regra r_j.

				Regras			
				r1	r2	r3	r4
TDC	Condições	c1	VS > 24	-	T	-	-
		c2	DAT > 10	T	-	-	-
		c3	Infestação emergencial?	F	F	T	F
		c4	Falha no isolamento?	F	F	F	T
	Ação	a1	Manejo Fungicida A	F	F	T	T
		a2	Manejo Fungicida B	T	T	F	F
CAMADA ADAPTATIVA	Before Action	ba1	Ação de choque inicial	T	T	T	T
		ba2	Ação preventiva intermediária	T	T	T	T
		ba3	Ação preventiva final	F	F	F	F

Figura 8 TDA tem t_0.

As condições da TDC são:

C1: Valor de severidade VS > 24, respeitando a metodologia para estágio inicial/intermediário MBLI24.
C2: Valor de dias após o transplante DAT > 10.
C3: Existência de infestação por patógeno encontrada.
C4: Existência de falha no isolamento verificada.

Nas condições iniciais, DAT = 0, VS = 0, sem infestações emergenciais e sem falhas no isolamento. Imediatamente após o transplante, inicia-se a contagem de tempo e de VS, e as observações de falhas de isolamento e infestações emergenciais por patógenos.

As ações são:

A1: Manejo com a utilização de fungicida de ação de contato (Dithane NT e Cuprogarb).
A2: Manejo com a utilização de fungicida de ação sistêmica (Cabrio TOP).

A camada adaptativa é composta somente por ações a serem realizadas antes da regra r_j., sendo elas:

Ba1: Ação de choque inicial altera as condições eliminando a componente inicial da lista de condições e suprimindo a regra correspondente, inclusive ela mesma.
Ba2: Ação preventiva intermediária altera as condições para uma contagem de tempo intermediária e zera VS, altera os valores das ações adaptativas restantes.
Ba3: Ação preventiva final altera as condições para uma contagem de tempo final do ciclo, suprime condições intermediárias, cria nova ação adaptativa de fim de ciclo.

Passo 1, ativadas as regras r_1 ou r_2, a TDA caminha para t_1, como visto na Figura 9.

t1

				Regras			
				r1	r2	r3	r4
TDC	Condições	c1	VS > 24	-	T	-	-
		c2	DAT > 65	T	-	-	-
		c3	Infestação emergencial?	-	F	T	F
		c4	Falha no isolamento?	-	F	F	T
	Ação	a1	Manejo Fungicida A	F	T	F	F
		a2	Manejo Fungicida B	F	F	T	T
		ba2	Ação preventiva intermediária	F	T	T	T
		ba3	Ação preventiva final	T	F	F	F

Figura 9 TDA em t_1

Passo 1', ativadas as regras r3 ou r4, a TDA caminha para t1, como visto na Figura 10.

t1'

				Regras			
				r1	r2	r3	r4
TDC	Condições	c1	VS > 24	-	T	-	-
		c2	DAT > 65	T	-	-	-
		c3	Infestação emergencial?	-	F	T	F
		c4	Falha no isolamento?	-	F	F	T
	Ação	a1	Manejo Fungicida A	F	F	T	T
		a2	Manejo Fungicida B	F	T	F	F
		ba2	Ação preventiva intermediária	F	T	T	T
		ba3	Ação preventiva final	T	F	F	F

Figura 10. TDA em t_1'

Para os passos intermediários, as ações são executadas quando satisfeitas as condições para cada regra, mantendo os manejos, alternando entre com a ação adaptativa modificando a1 e a2 de forma alternada, a exemplo do que seria o Passo 2, como mostrado na Figura 11.

t2

				Regras			
				r1	r2	r3	r4
TDC	Condições	c1	VS > 24	-	T	-	-
		c2	DAT > 65	T	-	-	-
		c3	Infestação emergencial?	-	F	T	F
		c4	Falha no isolamento?	-	F	F	T
	Ação	a1	Manejo Fungicida A	F	F	T	T
		a2	Manejo Fungicida B	F	T	F	F
		ba2	Ação preventiva intermediária	F	T	T	T
		ba3	Ação preventiva final	T	F	F	F

Figura 11. TDA em t_2.

Para o passo final e última modificação autônoma realizada pelo mecanismo, satisfeita a condição para ação ba3, ficamos com a condição final da TDA até o último DAT culminando na colheita, conforme visto na Figura 12.

tf

				Regras	
				r1	r2
TDC	Condições	c1	VS > 25	T	-
		c2	DAT = 90	F	T
	Ação	a1	Manejo Fungicida A	F	F
		a2	Manejo Fungicida B	F	F
CAMADA ADAPTATIVA	Before Action	ba3	Ação preventiva final	T	F
		ba4	Ação de fim de ciclo	F	T

Figura 12. TDA em configuração final, aguardando ativação da última ação adaptativa.

Ao fim do passo final, é ativada a última ação adaptativa, que declara encerrada as atividades do ciclo e determina que nenhuma ação mais seja tomada até que um novo ciclo passe a existir.

V. CONCLUSÃO

Foi apresentada uma proposta de controle adaptativo que permite que a estrutura de controle fique bem caracterizada, tornando fácil sua manutenção. Ou seja, trata-se de proposta prática de implementação na instalação de produção de minitubérculos de batata semente com técnica de aeroponia. Tal aplicação foi feita na cidade de Divinolândia (SP) e visava

ao controle de patógenos de forma automatizada em substituição ao sistema de aplicação convencional empírico.

A montagem do sistema de controle de forma estruturada, com aplicação do conhecimento em prática de alternar métodos de controle preventivo de manejo de combate à requeima da batata em seus momentos ótimos de atuação ao longo do ciclo, associada às ações emergenciais, em uma abordagem direcionada em uma linguagem específica ao especialista, colaboram para uma aplicação real em sistema de controle.

Um complemento ao sistema de controle existente, reduzindo a necessidade de tomada de decisão humana, em nível mais elevado substituindo a ação humana por sistema de barra de aspersão automatizada, permite aumentar os níveis de automação, caminhando para melhorar a condição de autonomia da unidade de produção.

A ferramenta adaptativa pode ser explorada aumentando seu nível de complexidade tanto quanto for necessário, sem deixar de ser clara e objetiva, sendo considerada uma ferramenta poderosa em todo o processo.

Como trabalhos futuros, tem-se o comprometimento em dar continuidade, explorando a aplicação em campo do projeto proposto, garantindo publicações futuras buscando excelência na aplicação em casos reais de tecnologias adaptativas em controle de crescimento de plantas em ambiente protegido, explorando ao máximo as possibilidades das unidades da categoria recente chamada de Fábrica de Plantas (*plant factory*).

VI. REFERÊNCIAS BIBLIOGRÁFICAS

[1] REFFYE, P. DE., B. G. HU. Relevant Qualitative and Quantitative choices for building an efficient dynamic plant growth model: Greenlab Case. International Symposium on Plant Growth Modeling, Simulation, Visualization and their Applications. PMA'03, 2003, pp. pp.87-107.

[2] FACTOR, T. L., J. A. C. ARAUJO, F.P.C KAWAKAMI, V. IUNCK. Produção de minitubérculos básicos de batata em três sistemas hidropônicos. Hortic. Bras. jan-mar, 2007.

[3] FABIANA S. SANTANA, ANTONIO M. SARAIVA, RENATA L. STANGE. Oportunidades de Aplicação das Tecnologias Adaptativas para a Modelagem em Agricultura de Precisão. 5º Workshop de Tecnologia Adaptativa WTA2011. 2011.

[4] M.U. FAROOQ, MUHAMMAD WASEEM, SADIA MAZHAR, ANJUM KHAIRI, TALHA KAMAL. A Review on Internet of Things (IoT). International Journal of Computer Applications. March de 2015, Vol. 113, Nº 1.

[5] YOUNG-MO KANG, MI-RAN HAN, KYEONG-SEOK HAN, JONG-BAE KIM. A Study on the Internet of Things (IoT) Applications. International Journal of Software Engineering and Its Applications. N.9, 2015, Vol. vol.9, pp. pp. 117-126.

[6] IOANNIS TSITSIMPELIS, IAN WOLFENDEN, C. JAMES TAYLOR. Development of a grow-cell test facility for research into sustainable controlled-environment agriculture. biosystems engineering. I50, 2016, pp. pp. 40-53.

[7] CAMILA C. ALMEIDA, PEDRO. S. ALMEIDA, NICOLAS R. C. MONTEIRO, MILENA F. PINTO AND HENRIQUE A. C. BRAGA. LED-Based Electronic System to Support Plant Physiology Experiments. IEEE 23rd International Symposium on Industrial Electronics (ISIE). 23, 2014.

[8] MURASE, YUSUF HENDRAWAN. DIMAS FIRMANDA AL RIZA. HARUHIKO. Applications of Intelligent Machine Vision in Plant Factory. Proceedings of the 19th World Congress The International Federation of Automatic Control. 24-29 de August de 2014, pp. pp.8122-8127.

[9] SEELING, H.D, STONER, R.J. & LINDEN, J.C. Irrigation control of cowpea plants using the measurement of leaf thickness under greenhouse conditions. Irri. Sci. DOI 10.007/s00271-011-0268-2, 2011.

[10] LEOSANE CRISTINA BOSCO, ARNO BERNARDO HELDWEIN, ELENA BLUME, GUSTAVO TRENTIN, EDENIR LUIS GRIMM, DIONÉIA DAIANE PITOL LUCAS, LUIS HENRIQUE LOOSE, SIDINEI ZWICK RADONS. SISTEMAS DE PREVISÃO DE REQUEIMA EM CULTIVOS DE BATATA EM SANTA MARIA, RS. Bragantia,Campinas. n. 3, 2010, Vol. v. 69, pp. p649-660.

[11] SALVADOR, CONAM AYADE. Desenvolvimento e avaliação de um sistema de controle de nitrato em soluções nutritivas. Tese de Doutorado. 2014.

[12] TCHEMRA, ANGELA. Aplicação de Tecnologia adaptativa em sistemas de tomada de decisão. Revista IEEE América Latina. 2007, Vol. Vol. 5, Num. 7.

[13] NETO, JOÃO JOSÉ. Adaptive rule-driven devices - general formulation and case study. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001. CIAA 2001, 23-25 de July de 2001, Vol. Vol.2494, pp. pp. 234-250.

[14] A. H. TCHEMRA, R. CAMARGO. Tabela de decisão adaptativa, simulação de um autômato adaptativo. In: Segundo Workshop de Tecnologia Adaptativa - WTA 2008. 2008.

[15] Coordenação-Geral de Agrotóxicos e Afins/DFIA/DAS. MAPA (Ministério da Agricultura, Pecuária e Abastecimento). Disponível em: < http://www.agricultura.gov.br/ >. Acesso em: 10 set. 2016.

Lucas Pladevall Moreira é graduado em Engenharia Mecatrônica (2011) pela Escola Politécnica da Universidade de São Paulo (POLI-USP), mestrando em Engenharia Mecânica pelo PPGEM da POLI-USP. É empresário de agronegócio atuando em culturas através de tecnologia de produção em aeroponia. Atua nas seguintes áreas de pesquisa: desenvolvimento de sensor agrícola, sistemas de controle para crescimento de plantas em ambiente protegido, sistemas de automação processos de produção agrícola.

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 (POLI-USP). É professor da POLI-USP desde 1986. Atua nas seguintes áreas de pesquisa: robôs sociáveis, computação afetiva e arquitetura de sistemas para aplicações críticas como Automação Industrial.

Grafo Adaptativo: uma Proposta de Topologia Adaptativa

Francisco Supino Marcondes[1] e Miguel Ângelo Tancredi Molina[1]

***Abstract*—This paper presents an adaptive version to "normal" graph called adaptive graph. An adaptive graph is graph whose vertices and edges can be modified in order to perform topological variation on it. As concept proof it was choosen the VNE problem, an NP-hard routing problem. The adaptive graph provided an way to reduce computation time aiding the practical use of such technology. Since some edges can be 'hidden' on substract network they cannot be considered by routing algorithms decreasing computation effort.**

Keywords: graph, topoly, VNE, adaptivity

I. INTRODUÇÃO

Embora o formalismo adaptativo tenha sido proposto para reconhecedores de linguagem, o objeto de interesse neste artigo é estudar sua estrutura topológica na forma de uma grafo. Assim, este estudo trata da adaptatividade sobre grafos a partir de uma adaptação do autômato adaptativo. Os grafos são objeto de interesse em diversas áreas da computação como estrutura de dados, inteligência artificial e redes de computadores.

O objetivo principal deste estudo é apresentar uma proposta de topologia adaptativa com base em um grafo chamado *grafo adaptativo*.

A aplicação escolhida para exemplificar o uso do grafo adaptativo proposto é o mapeamento de enlaces de redes virtuais em sua respectiva estrutura física também chamada de rede de substrato (*substract network*) esse problema é conhecido como VNE (ou *virtual network embedding*). O grafo adaptativo proposto será então aplicado como estrutura de descrição para redes virtuais cuja topologia possa ser alterada visando atender diferentes necessidades ou restrições. Naturalmente tal alteração topológica influi nos percursos sobre o grafo, isto é, nos caminhos de roteamento na rede de substrato. Por isso, a prova de conceito será realizada tendo como base a álgebra de caminhos.

II. TOPOLOGIA ADAPTATIVA

O termo *topologia* remete à estrutura de um grafo (ou grafo estático). Portanto, o termo *topologia adaptativa* remete ao grafo capaz de alterar sua estrutura. Em outros termos um *grafo adaptativo* consiste em um grafo onde se pode incluir e remover vértices e arestas.

Convém diferenciar a especificação matemática de grafo de sua implementação computacional. Matematicamente um grafo é uma estrutura estática onde não se permite a inclusão ou remoção de vértices e arestas. No entanto, computacionalmente antes de utilizar o grafo é preciso constituí-lo, para isso é necessário associar vértices e arestas à endereços na memória do computador. Deste tipo de necessidade surgem operações do tipo *addEdge* [8]. Isso não deve ser confundido com a possibilidade de alteração topológica, isto é, não se deve confundir uma necessidade de implementação com uma característica de especificação.

O que se propõe neste estudo é justamente um grafo cuja natureza seja adaptativa, isto é, uma estrutura diversa da proposta pelo grafo estático embora derivada dela.

A. Breve Revisão da Teoria de Grafos

O grafo é uma forma de representação comum, utilizada para expressar diferentes tipos de autômatos. Formalmente:

> Um grafo G consiste em uma tripla ordenada $(V(G), E(G), \psi_G)$ onde $V(G)$ é um conjunto não vazio de vértices, $E(G)$ um conjunto de arestas disjunto de $V(G)$ e ψ_G é uma função de incidência que associa cada aresta com um par não ordenado e não necessariamente distinto de arestas. Se a é uma aresta e u e v são vértices, a função de incidência toma a forma $\psi_G(a) = uv$, ou seja, se diz que a conecta u e v; os vértices u e v são chamados extremidades de a. [1, tradução do autor]

Existem ao menos duas formas equivalentes para descrever a topologia de um grafo: diagramático (círculos e retas) e matricial (adjacências ou incidência). A figura 1 apresenta os dois tipos de representação. Na figura, G representa a forma diagramática, $M(G)$ a matriz de incidência e $A(G)$ a matriz de adjacências. O valor de cada elemento a_{ij} na matriz de incidência ($M(G)$) representa o número de vezes que uma aresta incidiu sobre um vértice; os na matriz de adjacências ($A(G)$) representa o número de arestas que conecta dois vértices (caso o valor seja *0* significa que os dois vértices não estão conectados).

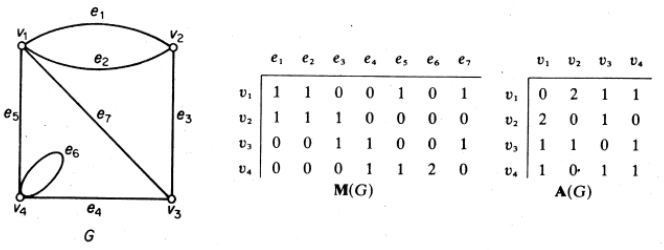


Figura 1. Maneiras para descrever a topologia de um grafo *cf.* [1].

Quando dois vértices estão ligados por uma aresta são chamados de vértices adjacentes e esta aresta pode ser interpretada como uma relação binária denominada adjacência; a matriz de adjacências é sempre uma matriz simétrica de dimensão $n \times n$.

Para [1], é possível transitar por um grafo de três formas: percurso (*walk*), trilha (*trail*) e caminho (*path*). Um percurso

[1] Instituto Federal de Educação, Ciência e Tecnologia - São Paulo {supino|miguel.molina}@ifsp.edu.br

consiste em uma sequência qualquer, porém não vazia, de vértices e arestas alternados, por exemplo, $v_1e_1v_2e_3v_3...e_iv_i$. Uma trilha consiste em um percurso onde não é permitido a repetição de vértices. Um caminho é uma trilha onde não se admite a repetição de arestas. Portanto, um *caminho* consiste em uma sequência não vazia e sem repetição de vértices e arestas alternados.

Um subgrafo consiste em um subconjunto de vértices e arestas do grafo original. Formalmente, o grafo H é um subgrafo de G ($H \subseteq G$) se $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ e se ψ_H restringir ψ_G. Se H é subgrafo de G, então G é super-grafo de H.

B. Aresta Adaptativa

Um autômato adaptativo consiste em um dispositivo capaz de alterar seu comportamento dinamicamente conforme o estado corrente e o símbolo consumido na cadeia de entrada [7], ou seja, se trata de um formalismo que especifica máquinas de estado auto-modificáveis. Pode-se dividir a operação de um autômato adaptativo em dois recursos distintos. Um deles consiste no reconhecedor subjacente e o outro no dispositivo adaptativo [9]. Em outros termos, o dispositivo adaptativo realiza modificações na estrutura do reconhecedor subjacente. O dispositivo adaptativo pode operar sobre reconhecedores dos diversos tipos de linguagem, por exemplo, pode-se utilizar um autômato finito como dispositivo subjacente ao dispositivo adaptativo [5].

> Todo autômato adaptativo pode ser visto como uma máquina de estados que, ao início de sua operação, apresenta uma topologia fixa, predeterminada, denominada máquina de estados inicial E_0. Na topologia dada por qualquer máquina de estados E_i, o autômato adaptativo opera como um autômato usual, efetuando uma seqüência de transições não-adaptativas, sendo que tal seqüência é finalizada ou pelo término do reconhecimento da sentença, ou então pela execução de alguma transição adaptativa, responsável por alterar sua topologia. Neste caso, pode-se dizer que o autômato se modifica, assumindo uma nova topologia, representada por alguma outra máquina de estados E_{i+1}, cujo estado inicial de operação deverá ser determinado pela transição executada. [5]

Autômatos e grafos são conceitos diferentes. No entanto, o fato de ser possível utilizar o grafo para representar o autômato implica na possibilidade de se abstrair características convenientes à proposta do grafo adaptativo.

Uma função de transição adaptativa, segundo [5], possui a forma: $(\gamma g, q, s\alpha) : A, \rightarrow (\gamma g', q', s'\alpha), B$ onde g e g' representam o conteúdo no topo da pilha, respectivamente, antes e depois da realização da transição; q e q' representam respectivamente o estado do autômato antes e depois da ocorrência da transição; s é o símbolo da cadeia consumido durante a transição e s' o símbolo inserido na cadeia; A e B representam transições adaptativas que ocorrem, respectivamente, antes ou depois da transição.

Nota-se da função de transição adaptativa que o foco de um autômato está no percurso sobre o grafo conforme símbolos da cadeia de entrada são consumidos. Quando se trata de um grafo propriamente dito, não existem símbolos a serem consumidos, isto significa que os elementos s e s' podem ser removidos. Além disso, para explicitar a semântica bidirecional da aresta no grafo convém trocar a seta unidirecional na transição adaptativa por uma bidirecional. Ainda neste sentido, o termo estado se mostra inadequado sendo melhor representado pelo termo vértice (troca-se q por v na notação da transição adaptativa). O termo *transição adaptativa* também, por isso pode ser renomeado para *aresta adaptativa*. Como a aresta adaptativa não será utilizada para percorrer o grafo, não faz sentido manter elementos que marquem transições de pilha (remove-se o γg e $\gamma g'$) e nem associar ações adaptativas à eventos do percurso (remove-se A e B).

A forma final da aresta adaptativa fica: $(v) \leftrightarrow (v')$.

Dado que o comportamento do autômato adaptativo é passível de representação por um grafo e que todo grafo á passível pode ser representado de forma matricial, logo, é possível representar qualquer autômato adaptativo por meio de um estrutura matricial.

C. Ações Adaptativas sobre Grafos

Diferente do que ocorre com o reconhecedor, pelo fato do controle adaptativo não se dar por meio de percurso, as ações adaptativas são disparadas *ad hoc*. As ações adaptativas aplicáveis à autômatos adaptativos e se mantém no grafo adaptativo: inclusão ($+[(v) \leftrightarrow (v')]$) e remoção ($-[(v) \leftrightarrow (v')]$) [5].

Considere como exemplo o grafo G_0 na tabela I. Aplicando sobre ele a ação adaptativa $+[(v_1) \leftrightarrow (v_3)]$ se criará no grafo subjacente uma aresta que conecta v_1 e v_3; caso v_3 não exista ele é criado (ver G_1). Aplicando sobre G_1 a ação adaptativa $-[(v_1) \leftrightarrow (v_2)]$ remover-se-a a aresta entre v_1 e v_2; caso algum vértice fique desconexo ele é removido (ver G_2).

Tabela I
AÇÕES ADAPTATIVAS SOBRE UM GRAFO SUBJACENTE.

matriz inicial	$+[(v_1) \leftrightarrow (v_3)]$	$-[(v_1) \leftrightarrow (v_2)]$
$G_0 = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{pmatrix} v_1 & v_2 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$	$G_1 = \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} \begin{pmatrix} v_1 & v_2 & v_3 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	$G_2 = \begin{matrix} v_1 \\ v_3 \end{matrix} \begin{pmatrix} v_1 & v_3 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$

D. Grafo Adaptativo

Se diz que se trata de um “grafo adaptativo"se ele puder ser descrito por meio de arestas adaptativas e possuir um conjunto não vazio de *scripts* de adaptação. Os *scripts* de adaptação são também compostos por um conjunto não vazio de ações adaptativas que fazem o grafo assumir diferentes configurações topológicas. A figura 2 apresenta a proposta de estrutura em nível de metamodelo para o Grafo Adaptativo.

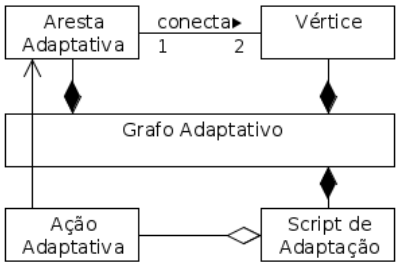


Figura 2. Metamodelo proposto para o Grafo Adaptativo.

E. Script de Adaptação

O *script* de adaptação pode operar no grafo subjacente de duas maneiras distintas: invariável e complexa. Um *script* de adaptação invariável garante que qualquer que seja a configuração topológica do grafo subjacente num dado momento, sua aplicação resulta sempre em uma mesma configuração topológica. Por outro lado, um *script* de adaptação complexo aplica de maneira incremental seu conjunto de ações sobre a estrutura topológica do grafo subjacente no instante da aplicação, isto significa que a aplicação de um mesmo *script* de adaptação complexo deve resultar em topologias diferentes e eventualmente desconexas. Por isso o *script* de adaptação complexo exige por parte do idealizador certo esforço para evitar este tipo de situação. Naturalmente, ao aplicar os algoritmos de percurso no grafo adaptativo, o resultado irá variar conforme a topologia do grafo no momento.

A figura 3 mostra como a topologia do grafo G pode variar conforme são aplicados diferentes *scripts* de adaptação. A tabela II apresenta um exemplo de aplicação de um *script* de adaptação invariável e a tabela III apresenta um exemplo de aplicação de um *script* de adaptação complexo. Pode-se dizer também que as adaptações sejam subgrafos de um super-grafo G ideal.

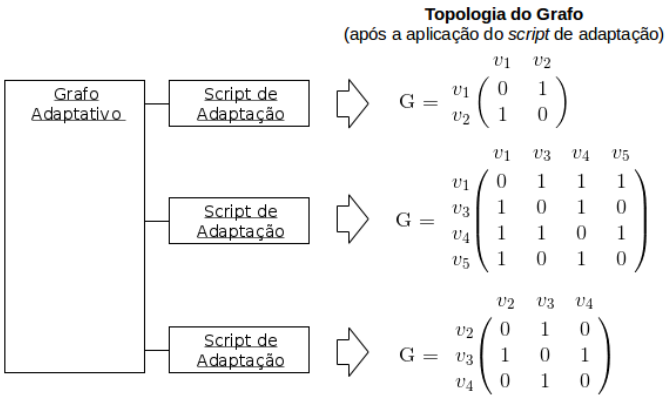


Figura 3. Adaptação topológica do grafo G.

III. *VIRTUAL NETWORK EMBEDDING* ADAPTATIVO

Ambientes de rede heterogêneos são aqueles compostos por uma diversidade de tecnologias de rede interconectadas [3].

Tabela II
EXEMPLO DE APLICAÇÃO DE UM MESMO *script* INVARIÁVEL.

	Exemplo 1	Exemplo 2
grafo inicial	$G = \begin{matrix} v_1 \\ v_3 \\ v_4 \end{matrix} \begin{pmatrix} v_1 & v_3 & v_4 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	$H = \begin{matrix} v_8 \\ v_9 \end{matrix} \begin{pmatrix} v_8 & v_9 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$
grafo resultante	$G = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{pmatrix} v_1 & v_2 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$	$H = \begin{matrix} v_1 \\ v_2 \end{matrix} \begin{pmatrix} v_1 & v_2 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$

Tabela III
EXEMPLO DE APLICAÇÃO DE UM MESMO *script* COMPLEXO.

	Exemplo 1	Exemplo 2
grafo inicial	$G = \begin{matrix} v_1 \\ v_3 \\ v_4 \end{matrix} \begin{pmatrix} v_1 & v_3 & v_4 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$	$H = \begin{matrix} v_8 \\ v_9 \end{matrix} \begin{pmatrix} v_8 & v_9 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$
grafo resultante	$G = \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} \begin{pmatrix} v_1 & v_2 & v_3 & v_4 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	$H = \begin{matrix} v_1 \\ v_2 \\ v_8 \\ v_9 \end{matrix} \begin{pmatrix} v_1 & v_2 & v_8 & v_9 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Quando se toma rede como serviço se torna necessário atender à diversas propriedades de qualidade do serviço (*Quality of Service* - QoS) – percepção subjetiva de um grupo de *stakeholders* em relação à qualidade do serviço oferecido por uma rede. Dado que uma mesma infraestrutura de rede pode atender diferentes *stakeholders*, o comum e que tais propriedades sejam conflitantes. Esse problema costuma ser administrado por meio das chamadas redes virtuais [3]. Em ambientes de redes virtuais diferentes provedores podem criar redes virtuais heterogêneas e assim prestar serviço de forma personalizada.

Assim, a tecnologia de virtualização de redes é um paradigma que propicia à múltiplas redes virtuais (*Virtual Networks* - VNs) compartilharem de uma mesma infraestrutura denominada rede de substrato (*Substract Network* - SN). Talvez o maior desafio na implementação deste conceito seja justamente embutir a rede virtual na rede de substrato (*Virtual Network Embedding* - VNE) [6].

O VNE costuma ser realizado aplicando algoritmos sobre a topologia da rede de substrato o qual considera as restrições durante a computação. Neste estudo se propõe que a topologia apresentada ao algoritmo seja um grafo adaptativo, com isso é possível remover antecipadamente os enlaces que ferem determinado conjunto de restrições do *stakeholder*.

A. Virtualização de Redes

Uma rede virtual é uma coleção de enlaces virtuais mapeados em diferentes enlaces físico interconectados por caminhos na rede de substrato. Como diversas redes virtuais podem depender de um mesmo conjunto de recursos físicos, é preciso que o *embedding* (mapeamento ou atribuição) de cada rede

virtual *on-line* seja eficiente visando o uso apropriado dos recursos da rede [3]. A figura 4 apresenta o problema de VNE por meio de duas redes virtuais e uma de substrato.

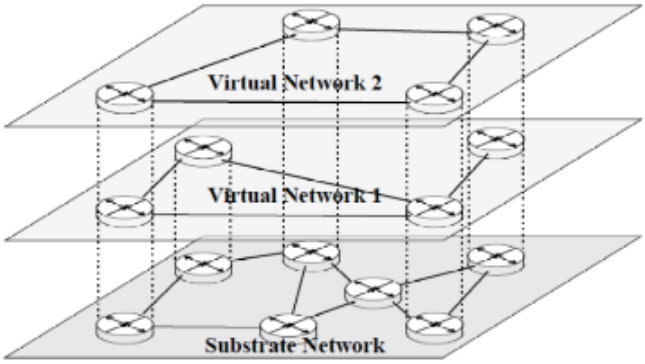


Figura 4. Ilustração de *Virtual Network Embedding* [2].

Os algoritmos ou heurísticas que propiciam o VNE devem garantir a convergência e eliminação de laços de roteamento nessas redes, obedecendo a tipos diferentes de métricas de QoS. Se diz desafio pois o VNE nesses termos é considerado multi-restritivo e por isso computacionalmente, um problema *NP-hard*. Isto, por sua vez, torna inevitável o uso de algoritmos heurísticos e aproximados para mapear nós e enlaces virtuais nos dos estágios (virtual e substrato) de forma independente ou coordenada [6].

Uma requisição de rede virtual (*Virtual Network Requisition* - VNR) se traduz em uma rede virtual com nós e enlaces virtuais com certas necessidades de rede, garantia de QoS e segurança. Os elementos da rede virtual (links e nós) devem ser mapeados sobre elementos concretos da rede de substrato com a preocupação de se obter uma solução otimizada no sentido de atender a demanda da requisição e que consuma a menor quantidade de recursos físicos da rede de substrato.

B. Álgebra de Caminhos

Uma alternativa à abordagem heurística geralmente utilizada para tratar problemas de roteamento em redes virtuais, tem se estudado abordagens algébricas como a "Álgebra de Caminhos" [6]. A álgebra de caminhos, por exemplo, como proposta por [4] consiste na quadrupla $\langle M, F, \bar{S}, \preceq_{ML} \rangle$, onde M é o conjunto de métricas de roteamento; F é o conjunto de combinação de métricas; $\bar{S}$ é o conjunto de operações binárias denominadas sínteses; e $\preceq_{ML}$ é a relação de ordenação léxica multi-dimensional.

Grosso modo, as métricas (M) enumeram as dimensões de interesse para a escolha do caminho, os quais podem ser combinados por meio de uma função (F). Um exemplo é a função $QoS = f(THR, PDT, PDV, PLR)$ com $QoS \in F$, onde *THR* é o *throughput* da rede; *PDT* o *packet delay transfer*; *PDV* o *packet delay variation*; e o *PLR* o *packet loss rate*. A síntese consiste em, dado os vértices de origem e destino, reduzir os caminhos possíveis à um valor com base em um conjunto de funções. Quando o caminho é analisado por uma única função se diz que ele é mono-restritivo, em caso contrário multi-restritivo.

Nesta álgebra existem quatro sínteses: a) *minimizativa* ou $\bar{S}_1$, o menor valor atribuído às aresta de um caminho; b) *maximitativa* ou $\bar{S}_2$, o maior valor atribuído às arestas de um caminho; c) *aditiva* ou $\bar{S}_3$, soma dos valores atribuídos às arestas do caminho; e d) *multiplicativa* ou $\bar{S}_4$, multiplicação dos valores atribuídos às arestas do caminho. Caso a síntese resultante de um caminho hipotético α seja maior do que a síntese resultante de outro caminho hipotético β se diz que $\bar{S}[\bar{C}(\alpha_{O,D})] \preceq_{ML} \bar{S}[\bar{C}(\beta_{O,D})]$, isto é, o caminho β é melhor que o α; desta comparação se obtém a ordenação léxica multi-dimensional ($\preceq_{ML}$).

1) Dígrafo de Rede Virtual: A álgebra de caminhos opera sobre uma grafos orientados (ou dígrafos) com propriedades específicas, neste estudo chamado de Dígrafo de Rede Virtual. Por ser um dígrafo, a matriz de adjacências, embora se mantenha quadrada e de dimensão $n \times n$, não é necessariamente simétrica (pode existir $\psi_G = uv$ mas não $\psi_G = vu$). O Dígrafo de Rede Virtual aceita apenas uma aresta na conexão de dois vértices ($\exists \psi_G(a) = uv | \neg \psi_G(\delta) = uv$ onde δ é o conjunto de todas as outras arestas do grafo). Além disso, não é permitida a existência de arestas que liguem um mesmo vértice ($\psi_G(a) = uu$); isso força que a diagonal da matriz de adjacências possua sempre valor *0*.

2) Dígrafo de Rede Virtual Adaptativo: Um dígrafo adaptativo é um grafo adaptativo orientado. De maneira geral se mantém as propriedades do grafo adaptativo com a diferença de que as ações adaptativas são unidirecionais, ou seja, inclusão ($+[(v) \rightarrow (v')]$) e remoção ($-[(v) \rightarrow (v')]$). A ação adaptativa altera a aresta adaptativa no sentido da seta mas não no sentido oposto. A equivalência com a ação sobre o grafo adaptativo se dá na forma: $+[(v) \leftrightarrow (v')] \Leftrightarrow (+[(v) \rightarrow (v')]) \wedge (+[(v') \rightarrow (v)])$ e $-[(v) \leftrightarrow (v')] \Leftrightarrow (-[(v) \rightarrow (v')]) \wedge (-[(v') \rightarrow (v)])$. Somase à esta propriedades as restrições impostas pelo Dígrafo de Rede Virtual, isto é, aceita apenas uma aresta que incide sobre dois vértices e rejeita areactas que incidem sobre um mesmo vértice.

C. Prova de Conceito

Considere como exemplo a topologia da SN apresentada na figura 5 e sua representação matricial na tabela IV. Supondo que se queira ir do vértice F ao vértice D, se obtém os caminhos *F,E,B,D* e *F,E,C,A,B,D*. Para esses caminhos os valores das sínteses *F,E,B,D* são: $\bar{S}_1[\alpha_{F,D}] = 3$, $\bar{S}_2[\alpha_{F,D}] = 65$, $\bar{S}_3[\alpha_{F,D}] = 50$ e $\bar{S}_4[\alpha_{F,D}] = 1700$. Os valores para *F,E,C,A,B,D* são: $\bar{S}_1[\beta_{F,D}] = 5$, $\bar{S}_2[\beta_{F,D}] = 60$, $\bar{S}_3[\beta_{F,D}] = 50$ e $\bar{S}_4[\beta_{F,D}] = 2400$. Portanto, $\bar{S}[\bar{F}(\beta_{F,D})] \preceq_{ML} \bar{S}[\bar{F}(\alpha_{F,D})]$, isto é, o caminho α é mais eficiente que o β.

Suponha agora que por uma restrição qualquer do *stakeholder* determinadas tecnologias não possam ser utilizadas no serviço de rede a ser prestado. Naturalmente não se espera, por exemplo, que tais tecnologias sejam removidas da rede de substrato mas apenas que a rede virtual não as utilize. Isto significa que o caminho que passa por um determinado enlace

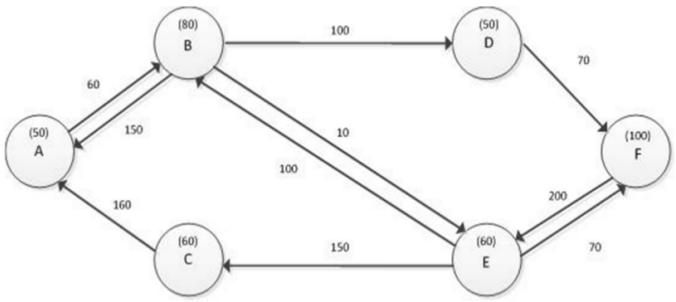


Figura 5. Exemplo de topologia de rede [6].

Tabela IV
REPRESENTAÇÃO MATRICIAL DO EXEMPLO DE TOPOLOGIA DE REDE.

$$R_0 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \qquad R_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

deverá ser eliminado, no entanto, isso costuma ser efetuado após a listagem de todos os caminhos.

Aplicando o grafo adaptativo pode-se remover tais enlaces antes de se listar cada um dos caminhos possíveis e assim reduzir o esforço computacional envolvido. Suponha, por exemplo, que a tecnologia do enlace *E-C* da figura 5 seja considerado indesejável pelo stakeholder; aplicando sobre R_0 na tabela IV a ação adaptativa $-[(E) \rightarrow (C)]$ resulta em R_1. Esta modificação faria com que o caminho *F,E,C,A,B,D* não pudesse mais ser encontrado pela rede virtual.

Naturalmente se o enlace *F-E* fosse removido não seria possível encontrar nenhum caminho; este tipo de preocupação deve ser parte da especificação do *script* de adaptação.

Com esta estratégia é possível ter diferentes *scripts* de adaptação os quais ajudam a atender diferentes especificações de QoS.

D. Uma consideração sobre Complexidade

Dado que álgebra de caminhos trata cada caminho possível entre o vértice de origem e o de destino, se pode afirmar que a complexidade algorítmica envolvida é $O(n^n)$ [6], ou seja, se trata de uma problema *NP-hard*. Naturalmente a tecnologia adaptativa não altera a complexidade intrínseca do problema mas ao propiciar a redução no número vértices e/ou arestas assegura a diminuição no tempo efetivo de computação. O algoritmo PAViLiM (Paths Algebra Virtualization Link Mapping) proposto por [6] consiste em uma implementação computacional da álgebra de caminhos orientada a solução do problema VNE.

Considere como exemplo, a execução deste algoritmo em uma máquina padrão tendo como base um grafo hipotético com nove vértices requer cinco horas e meia de computação [6]; a execução deste mesmo algoritmo na mesma máquina mas tendo como base um grafo hipotético com dez vértices requer quase dois dias de computação [6]. Por outro lado, a execução deste mesmo algoritmo na mesma máquina mas tendo como base um grafo hipotético com oito vértices requer cerca de trinta minutos [6]. Isso sugere que a escolha qualitativa dos vértices e arestas a serem considerados durante cada VNE pode reduzir significativamente o tempo de computação necessário. A escolha qualitativa implica em alterar a topologia existente no grafo que representa a rede de substrato, daí a necessidade do grafo adaptativo.

IV. CONCLUSÃO

Este estudo propôs uma estrutura de topologia adaptativa chamada *grafo adaptativo*. A apresentação foi informal e teve a intenção de sugerir suas principais propriedades. Como trabalho futuro sugere-se propor a formalização da estrutura proposta.

Embora informal, é possível vislumbrar o potencial da estrutura proposta tanto em seu aspecto teórico quanto aplicado ao contexto de VNE proposto. Também como trabalho futuro sugere-se aprofundar na investigação do uso desta estrutura junto com algoritmos de roteamento e em estruturas topológicas mais próximas do real. Sugere-se também investigar quais outras aplicações de grafo podem se beneficiar de uma estrutura adaptativa, entre outras, estruturas de dados, complexidade e redes neurais artificiais.

REFERÊNCIAS

[1] Bondy, J. A. Murty U. S. R. (1982). *Graph Theory with Applications*. North-Holland.

[2] Botero, J. F. Molina, M. A. T., Serra, X. H. Amazonas, J. R. (2013). A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems. *Journal of Network and Computer Applications v. 36, i. 6, Elsevier.*

[3] Chowdhury, N. M. M. K. Rahman, M. R. Boutaba, R. (2009). Virtual Network Embedding with Coordinated Node and Link Mapping. *IEEE INFOCOM*.

[4] Herman, W. P. (2008). Formulação algébrica para a modelagem de algoritmos de roteamento multi-restritivo hop-by-hop. *Tese - Universidade de São Paulo*.

[5] Neto, J. J. (2003). Autômatos em engenharia de Computação - uma visão unificada. *Primera Semana de Ciencia y Tecnología de la Sociedad Chotana de Ciencias y la Red Mundial de Científicos Peruanos*.

[6] Molina, M. A. T. (2013). Utilização da álgebra de caminhos para realizar o mapeamento de requisições virtuais sobre redes de substrato. *Tese - Universidade de São Paulo*.

[7] Ramos, M. V. M. Neto, J. J. Vega, I. S. (2009). *Linguagens Formais: teoria, modelagem e implementação*. Editora Bookman.

[8] Sedgewick, R. Wayne, K. (2011). *Algorithms*. Addison-Wesley Professional, 4th edition.

[9] Vega, Í. S. (2008). An adaptive automata operational semantic. *IEEE Latin America Transactions*, v. 6, n. 5

Programação Dirigida por Contratos Adaptativos — Especificação de Propriedades Comportamentais Dinâmicas

Italo S. Vega

***Abstract*—The design of adaptive automata involves the development of rules that, when applied, can transform the topology of states and transitions during a computation. Adaptive specifications were introduced by Vega as a way to represent the different transformations in the automaton state space. In this article, we present a technique that helps in designing adaptive transitions that interconnect the states in the adaptive state space of an adaptive automaton.**

***Index Terms*—adaptive specifications, adaptive contracts, object-oriented modeling, software design, state machine, UML.**

I. Introdução

A LÓGICA proposta por Hoare [1] introduziu uma forma de raciocínio sobre o correto funcionamento de programas de computador na forma de triplas $\{P\}C\{Q\}$. Caso a pré-condição P seja verdadeira, estabelece-se a pós-condição Q pela execução do comando P. Posteriormente, a lógica de Hoare foi estendida para lidar com ponteiros para estruturas de dados por Reynolds [2]. A *lógica da separação* lida com porções de memória apenas, ao invés do espaço global de estados do sistema. Mas, como seria isso útil no processo de desenvolvimento de software? Meyer propôs uma técnica de projeto de software baseada na noção de contrato — os componentes de um sistema respeitam uma relação de obrigação-benefício durante uma particular colaboração [3], [4]. Componentes de software, tipicamente implementados como objetos no paradigma de objetos, executam comportamentos computacionais [5]chamados de métodos. Estes são concebidos, portanto, a partir do que se deseja que o componente faça em um determinado contexto computacional. *Contratos* formalizam as propriedades que devem ser exibidas por comportamentos ao serem executados: quando violados, acusam problemas de execução comportamental.

Por outro lado, Bock propõe uma representação das ocorrências de estados de um comportamento usando *estados-objeto*. Esta técnica trata estados como se fossem classes dinamicamente instanciadas e destruídas na forma de objetos. Sob tal perspectiva de modelagem, estados-objeto possuem atributos estruturais e podem ser ligados com outros estados-objeto. Como consequência, obtém-se uma representação de comportamentos mais expressiva durante a etapa de modelagem baseada em estados.

Especificações adaptativas foram introduzidas por Vega com a intenção de organizar a descrição de modelos comportamentais de autômatos adaptativos [6]. Representações parciais de máquinas de estados foram sugeridas com base em autômatos adaptativos. Entretanto, a relação entre as diferentes configurações de um particular autômato adaptativo ficaram implícitas na especificação. Este artigo propõe uma maneira de se lidar com esta questão, combinando as técnicas de programação por contratos e representação de estados por meio de estados-objeto. *Contratos adaptativos*, introduzidos neste artigo, formalizam as propriedades dinâmicas das operações modeladas por autômatos adaptativos vistos como objetos. Elas formalizam a conexão entre as diferentes configurações de um particular autômato adaptativo ao longo da sua existência na dimensão temporal.

Na Seção II apresentam-se os conceitos de estado-objeto e de contratos de operação. Em seguida, na Seção III conecta-se a noção de adaptatividade com objeto e estado-objeto. A Seção IV ilustra a utilização da técnica de programação baseada em contratos adaptativos. Os efeitos da programação dirigida por contratos adaptativos são discutidos também naquela Seção. Finalmente, na Seção V apresentam-se projetos que pretendem utilizar os contratos adaptativos durante a sua realização.

II. Estado-Objeto e Contrato de Operação

Estados-objeto e contratos de operações, juntamente com a noção de especificação adaptativa, formam a base das ideias que originaram os contratos adaptativos.

A. Estado-Objeto

Em uma série de artigos a respeito de modelagem de comportamentos no contexto da orientação a objetos, Bock discute três maneiras de se modelar estados: (i) estado-comportamento, (ii) estado-condição e (iii) estado-objeto [7]. Utiliza-se a primeiras delas em conjunção com transições de estado para especificar reação a eventos. Uma máquina pode apresentar diversas reações a um determinado tipo de evento, dependendo do seu estado corrente. Esta é a visão tradicional na ciência da computação e, em particular, no estudo de autômatos finitos. A

Italo S. Vega trabalha no Departamento de Computação da Faculdade de Ciências Exatas e Tecnologia da Pontifícia Universidade Católica de São Paulo, São Paulo, capital, Brasil e-mail: `italo pucsp.br`.

Manuscrito recebido no dia xx/xx/2016.

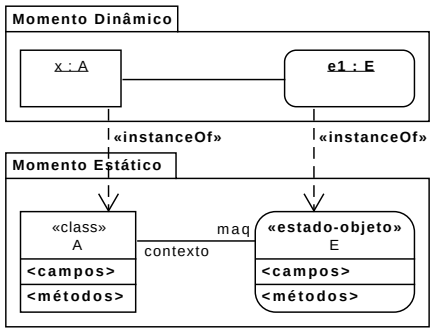


Figure 1: Diagrama UML de um estado-objeto de acordo com Bock

segunda maneira interpreta um estado como restrições que determinados valores devem obedecer em um instante de tempo. Estados-condição serviram de base, por exemplo, para a linguagem TLA+ de Lamport, que suporta a especificação de comportamentos com fórmulas da *Temporal Logic of Actions* [8]. Diferentes condições situacionais ao longo da execução de um comportamento originam diferentes estados computacionais.

Na técnica de estado visto como objeto, Bock sugere que estados sejam percebidos como instâncias de classes — construídos e destruídos ao longo do momento dinâmico de execução do comportamento [9]. Objetos tanto podem ser usados para capturar a perspectiva da primeira forma (estado-comportamento), quanto da forma estado-condição. Isto porque eles são especificados por classes constituídas por campos e métodos [5]. Na Fig. 1 ilustra-se a ideia por meio de um diagrama UML de estrutura. No momento estático, existe a classe A associada a um estado-classe E contendo eventuais campos e métodos que especificam estados-objeto desta classe. Em um momento dinâmico, instâncias da classe A (como o objeto x) encontram-se ligadas a objetos do estado-classe M (como o objeto e1). Condições e ocorrências comportamentais podem ser modelados por valores de campos e ocorrências de execução de métodos invocadas pelo estado-objeto e1.

Ainda na Fig. 1, os papéis de objetos maq e contexto suportam a combinação de campos e métodos que se referem a variáveis de estado com campos que se referem a outras estruturas de informação. Se x for um campo de A, então ele pode ser manipulado no contexto E por meio da expressão contexto.x. De maneira similar, se y for um campo de E, ele pode ser acessado por meio da expressão maq.y no contexto A. Estas expressões serão posteriormente utilizadas nos contratos adaptativos. Na próxima seção destacam-se os elementos centrais da programação por contratos de Meyer.

B. Contrato de Operação

A noção de comportamento em sistemas de software remete a procedimentos e métodos, relacionados ao ciclo chamada-invocação-execução conforme apresentado na

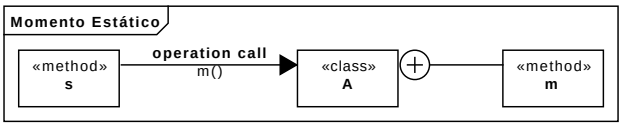


Figure 2: Chamada síncrona da operação **m**

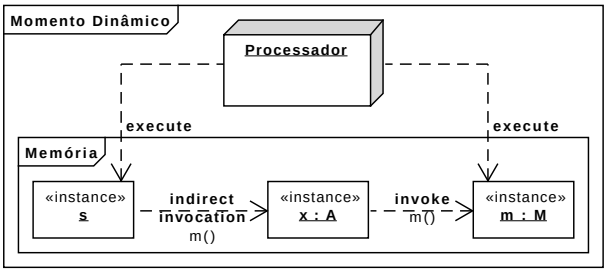


Figure 3: Invocação indireta e execução do comportamento m

especificação UML [5]. No momento estático, programa-se uma chamada de comportamento que, no caso do paradigma de objetos, traduz-se como uma *chamada de operação*. Na Fig. 2, no contexto do método s programou-se uma chamada síncrona de operação m() por meio da classe A. Esta, por sua vez, contém (por aninhamento) o método m.

No momento dinâmico, chamadas de operações originam *mensagens* (síncronas) e *eventos* os quais produzem *invocações indiretas* de métodos. Indiretas, pois o comportamento a ser executado depende do objeto que recebeu a mensagem. Considere-se um objeto s programado para enviar a mensagem chamando a operação m sobre o objeto x (Fig. 3). Esta invocação indireta de comportamento é resolvida pelo objeto x — invocação do comportamento m. Como resultado, o processador passa a *executar* este comportamento. A interação entre os objetos s e x é conhecida por *colaboração*.

Na utilização da técnica de programação por contratos explicita-se a condição a ser estabelecida para que um comportamento seja invocado; a [pré-condição]. Também deve-se explicitar a condição que deverá ser estabelecida pela execução do comportamento; a [pós-condição]. Na proposta de Meyer, o processador verifica a pré-condição no contexto de envio da mensagem, enquanto a pós-condição é verificada no final da execução do comportamento. No caso da Fig. 3, a pré-condição deve ser verdadeira no contexto de envio da mensagem m de s para x. A pós-condição deve ser verificada pelo processador no final da execução do comportamento m.

Na Fig. 4, representam-se os elementos centrais de um contrato comportamental de acordo com os estados-objeto de Bock. No momento estático, especifica-se o contrato de execução de comportamentos com base no estado-classe E como segue. Na ocorrência de um evento no qual se verifica

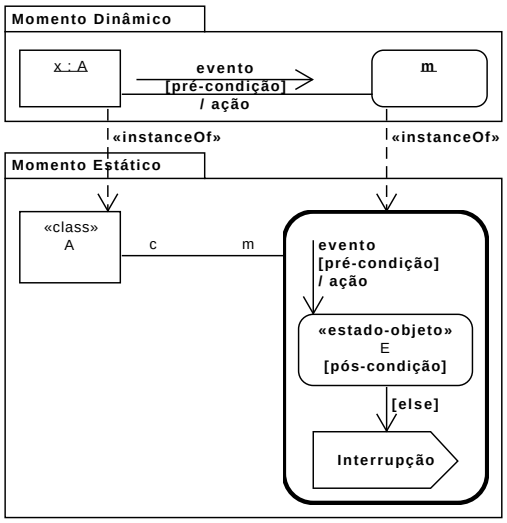


Figure 4: Representação de contratos em estados-objeto

a [pré-condição], a invocação indireta da **ação** provoca a sua execução. (Uma ação é uma espécie de método cuja execução não pode ser interrompida uma vez iniciada). Ao término da execução da ação, a [pós-condição] deve estar satisfeita. Caso contrário, interrompe-se o processo computacional devido à violação do contrato de execução da ação em um estado-objeto da classe E.

A ideia de estados-objeto suporta análises mais expressivas para a programação de um sistema computacional, enquanto a programação por contratos introduz a possibilidade de se observar violações de contratos — indícios de falhas computacionais. A combinação destas duas técnicas será explorado na concepção de contratos adaptativos a seguir.

III. Adaptatividade e Contratos Adaptativos

No caso da tecnologia adaptativa, a camada adaptativa proporciona um mecanismo para modificar a topologia de um dispositivo subjacente. Caso este seja um autômato, as modificações topológicas afetam os movimentos por ele realizados. Sabe-se que as ações adaptativas antes e depois, se realizadas, contribuem para alterar a topologia de um autômato adaptativo em razão de algum propósito de modelagem. Modelados como execuções comportamentais, os movimentos de um autômato podem ser projetados com o auxílio dos contratos adaptativos e estados-objeto como será mostrado a seguir.

A. Autômatos e Mecanismo Adaptativo

Seguindo a formulação de Neto e Bravo [10], um autômato de estados finito é composto por um conjunto de estados Q, um alfabeto finito e não vazio de símbolos Σ, uma função de transição δ, um estado inicial $q_0 \in Q$ e um conjunto de estados finais $F \subseteq Q$. Em particular, a função δ mapeia pares ordenados que especificam o estado corrente $q \in Q$ e a entrada corrente $\alpha \in \Sigma$ em um novo estado $q \in Q$: $(p, \alpha) \mapsto q$.

Um autômato de pilha estruturado possui uma estrutura semelhante àquela do autômato de estados finitos, além de um alfabeto de pilha Γ e elementos que suportam transições externas, responsáveis pelo esquema de chamada e de retorno. As transições de chamada possuem a forma $(p, \epsilon) \mapsto (\downarrow\ q_b, q_x)$. Interpreta-se p como o estado de chamada de uma submáquina com estado inicial q_x. No final da sua execução, a submáquina retorna o controle para o estado q_b. Cabe ao autômato de pilha preservar o estado q_b na sua memória durante a ocorrência da transição. As transições de retorno possuem a forma $(p, \epsilon) \mapsto (\uparrow\ q_r, q_r)$, para algum p no conjunto de estados finais da submáquina corrente. O estado q_r representa qualquer um dos estados anteriormente empilhados na memória do autômato pela submáquina chamada a partir daquela corrente. Esse estado é removido da memória e a submáquina chamadora continua a sua execução a partir de q_r.

Finalmente, Neto e Bravo referem-se ao mecanismo adaptativo. Ações adaptativas modificam o comportamento do autômato adaptativo, alterando o conjunto de regras que o definem. Define-se o mecanismo adaptativo pela vinculação de ações adaptativas anteriores e posteriores às regras não-adaptativas que definem as transições do autômato adaptativo. As transições adaptativas assumem a forma $(p, \alpha) \mapsto q[\mathcal{B} \bullet \mathcal{A}]$. Na ocorrência de α, estando o autômato no estado p, executa-se a ação adaptativa $\mathcal{B}$, realiza-se a transição para q e executa-se a ação adaptativa $\mathcal{A}$, nesta ordem.

Na técnica de programação proposta neste artigo, estados-objeto, chamadas de operações e contratos adaptativos são arquétipos dos elementos formais utilizados para a definição de autômatos finitos, autômatos de pilha e autômatos adaptativos. Assim, estados-objeto possuem alguma relação com os estados em Q. De modo semelhante relacionam-se chamadas de operações com chamadas e retornos de submáquinas e as ações adaptativas $\mathcal{B}$ e $\mathcal{A}$ com os métodos dos contratos adaptativos. A próxima seção define o elemento central da técnica: os contratos adaptativos.

B. Contratos Adaptativos

Partindo-se do modelo de contratos representado na Fig. 4, se o estado-objeto m for interpretado como um mecanismo adaptativo, então ele assumiria a responsabilidade de modificar o comportamento de x: ao longo do seu ciclo de vida, x teria o seu comportamento dinamicamente modificado por m. Em um contrato adaptativo, operações pré-adaptativas e pós-adaptativas (inspiradas nas funções adaptativas do mecanismo adaptativo) são vinculadas às pré- e pós-condições de um contrato: ao serem chamadas, provocam a modificação do comportamento dos objetos da classe A.

Os elementos estruturais de um contrato adaptativo são especificados conforme ilustrado na Fig. 5. Na ocorrência

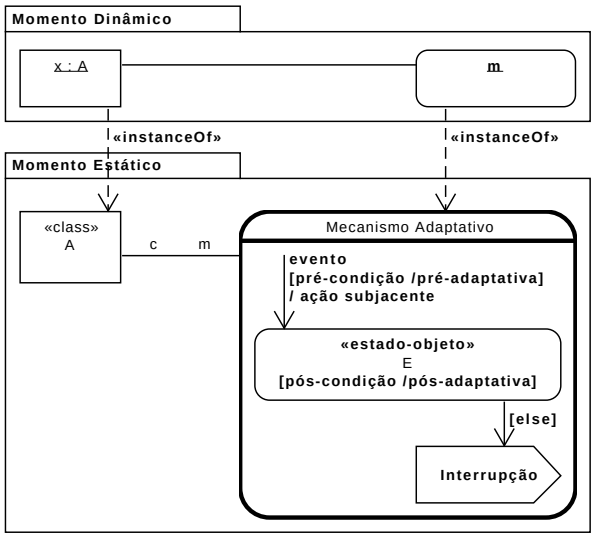


Figure 5: Elementos estruturais de um contrato adaptativo

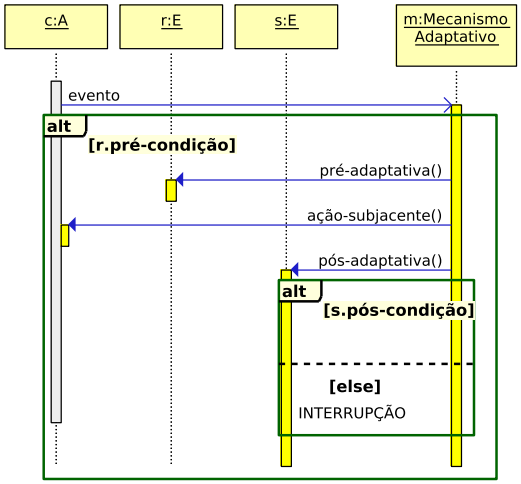


Figure 6: Vista do momento dinâmico de um contrato adaptativo

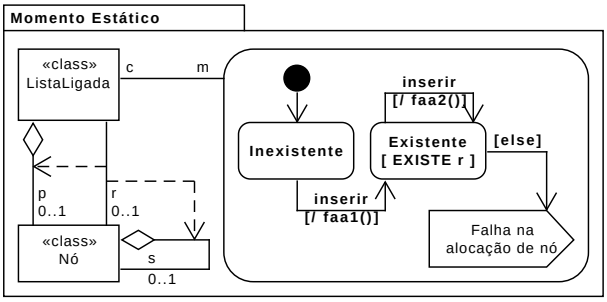


Figure 7: Contrato adaptativo da operação **inserir** em uma lista-ligada

de um evento do tipo **evento**, o estado-objeto corrente do mecanismo adaptativo m verifica a [pré-condição]. Se ela for verdadeira, ele dispara a transição, provocando a chamada da operação **pré-adaptativa**. Em seguida, ocorre a chamada da operação ação-subjacente envolvendo o contexto-objeto c. Finalmente, ocorre a chamada da operação **pós-adaptativa** e o estado-objeto seguinte do mecanismo adaptativo verifica a [pós-condição]. Se ela for falsa, ele emite um sinal de interrupção do comportamento computacional do objeto c.

Em uma vista de diagrama de sequências de mensagens, o momento dinâmico de um contrato adaptativo apresenta-se como na Fig. 6. O tratamento do **evento** é encaminhado para o mecanismo m. Este objeto, em colaboração com o estado-objeto corrente r, verifica a [pré-condição]. Sendo verdadeira, inicia-se o comportamento de reação ao evento propriamente dita. O mecanismo m envia uma mensagem para r chamando a operação **pré-adaptativa**. Em seguida, ocorre a chamada da operação **ação-subjacente**. Por último, m chama a operação **pós-adaptativa** e, juntamente com o estado-objeto seguinte s, verifica a [pós-condição]. Ocorrerá interrupção do comportamento de c se tal condição for falsa.

Contratos adaptativos especificam propriedades relacionadas com adaptatividade estrutural e comportamental. Uma *adaptatividade estrutural* é uma mudança nos campos de um objeto. Uma *adaptatividade comportamental* é uma mudança nos métodos de uma classe. Em ambos os casos, a adaptatividade decorre da chamada das operações **pré-adaptativa** e **pós-adaptativa**. Esta observação sugere os seguintes modelos de computação:

- computação inspirada em procedimento — o comportamento computacional é estabelecido pela ordem dos passos de um procedimento;
- computação inspirada em estrutura adaptativa — o comportamento computacional depende da evolução dinâmica dos campos de um objeto;
- computação inspirada em comportamento adaptativo — o comportamento computacional depende da evolução dinâmica dos campos e dos métodos de um objeto.

Computações inspiradas em procedimento são tradicionalmente estudadas na literatura ([11], [12]). Computações inspiradas em comportamento adaptativo são mais facilmente suportados pelos mecanismos de programação funcional ([13], [14]). O exemplo da próxima seção ilustra a noção de contratos adaptativos e de um modelo inspirado em estrutura adaptativa.

IV. Exemplo Ilustrativo

Uma estrutura de dados tradicional para a implementação do tipo de dado *lista* é a *lista ligada simples* [12]. Trata-se de um modelo de implementação cuja estrutura se constrói via alocação dinâmica de memória. Neste exemplo, enfatiza-se apenas a evolução da estrutura ao longo de uma série de eventos de inserção: ignora-se a perspectiva lógica de armazenagem dos valores.

A especificação estática (Fig. 7) introduz a classe ListaLigada composta por um objeto opcional p da classe Nó. O

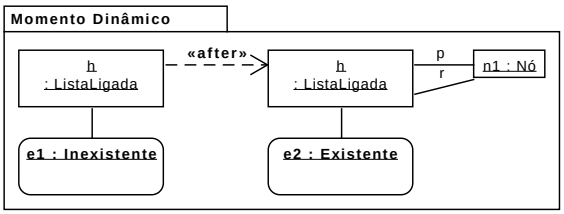


Figure 8: Estrutura de objetos depois da primeira transição de estado (exemplo-20)

campo r é uma referência para o objeto denominado p (no contexto de um objeto da classe ListaLigada) ou para o objeto s (no contexto de um objeto da classe Nó). A linha tracejada indica uma dependência existencial no momento dinâmico: a existência de r depende da existência de p ou de s. Na especificação da máquina de estados m observa-se que o estado inicial de um objeto ListaLigada é **Inexistente**, indicando que não existe o objeto p. De acordo com o contrato adaptativo, diante de uma ocorrência de evento do tipo inserir realiza-se uma chamada da operação pré-adaptativa **faa1**:

Method IV.1: faa1

```
1  input: ListaLigada c
2  begin
3     // modificação da estrutura de "c":
4     c.p ← Nó.new()
5     c.r ← c.p
6     // modificação da máquina de "c":
7     c.m ← Existente.new()
8     c.m.pósCondição()
9  end
```

No diagrama de instâncias mostrado na Fig. 8, observa-se o objeto h da classe ListaLigada em um momento de recém instanciação. Aí, ele encontra-se ligado ao estado-objeto e1. Quando h trata um evento do tipo inserir, a sua máquina de estados m realiza uma transição, chamando a operação **faa1**. Por conseguinte, o objeto h passa de uma configuração na qual ele se encontrava ligado ao estado-objeto e1 para outro, agora ligado ao estado-objeto e2. Além disso, devido ao atendimento à chamada da **faa1**, o objeto h conecta-se ao objeto p dinamicamente instanciado, ajustando, também, a referência para o nó corrente r.

Em uma vista dinâmica com destaque da transição do estado **Inexistente** para **Existente**, o diagrama de sequências de mensagens apresentado na Fig. 9 observa-se que um dos efeitos da operação pré-adaptativa é a instanciação de um novo nó n1 da lista ligada h. Também ocorre a construção do estado-objeto seguinte e2, responsável por verificar a [pós-condição] de existência de n1.

Retornando ao momento estático (Fig. 7), uma nova ocorrência de evento do tipo inserir, mas a partir do estado **Existente**, provoca uma chamada da operação pré-adaptativa **faa2** do contrato:

Method IV.2: faa2

```
1  input: ListaLigada c
2  begin
```

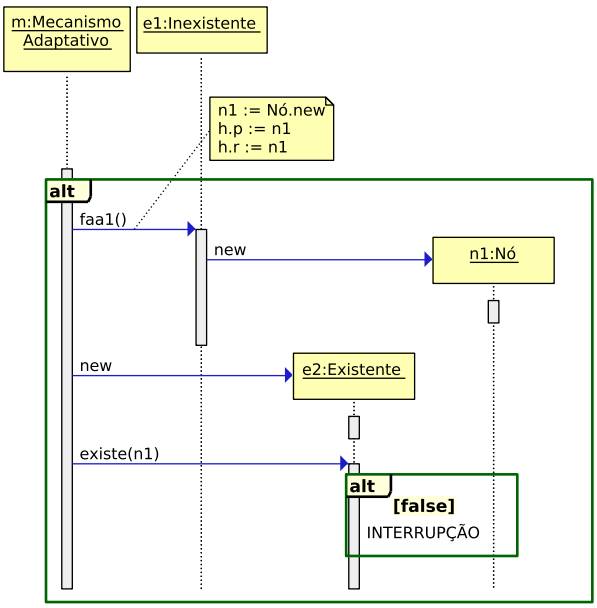


Figure 9: Ordem temporal de construção dos objetos devido à adaptação **faa1**

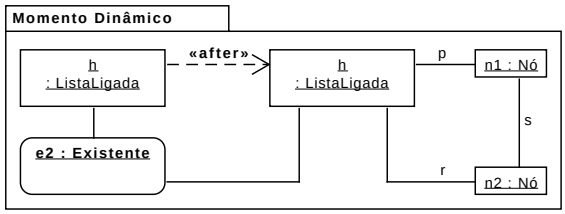


Figure 10: Estrutura de objetos depois da segunda transição de estado

```
3     // modificação da estrutura de "c":
4     c.r.s ← Nó.new()
5     c.r ← c.r.s
6     // modificação da máquina de "c":
7     c.m.pósCondição()
8  end
```

Em uma vista da estrutura dinâmica (Fig. 10), observa-se que a lista ligada h continuou conectada ao estado e2 depois de atender ao evento inserir pela segunda vez. Complementarmente, a sua estrutura de suporte à armazenagem dos elementos da lista foi modificada: existem dois nós encadeados.

A ordem temporal de modificação da estrutura de h revela que é durante o tratamento da chamada da operação pré-adaptativa **faa2** que se constrói o segundo nó da lista (Fig. 11). Como mostrado na Fig. 9, cabe ao estado-objeto e2 confirmar a pós-condição de existência do novo da estrutura ligada.

V. Consideração Final

A noção de contratos para formalizar propriedades de operações com pré-, pós-condições e invariantes ajuda na identificação de problemas de execução [4]. No entanto,

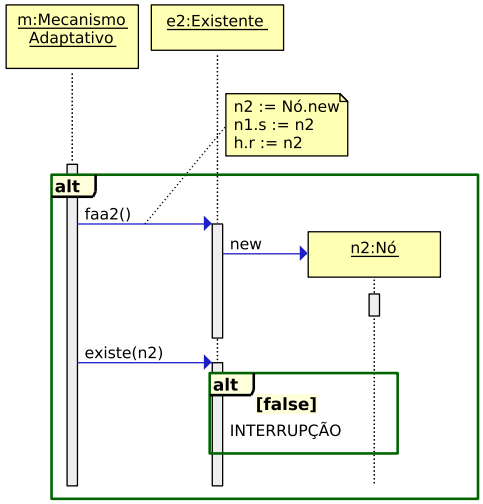


Figure 11: Ordem temporal de construção dos objetos devido à adaptação **faa2**

outras preocupações escapam do alcance de uma análise baseada em contratos. Por exemplo, em determinadas situações, espera-se que a execução de um comportamento estabeleça a pós-condição antes do seu término, mas e se a execução não terminar? Tais contratos meramente especificam correção parcial decorrente do tratamento das chamadas das operações. O requisito para que uma operação eventualmente também termine conduz a uma situação de correção total, o que pode ser feito incorporando-se uma convenção de término a todos os contratos [15].

A programação por contratos adaptativos oferece uma alternativa à convenção. Em um contrato adaptativo, a existência de operações **pré-adaptativa** e **pós-adaptativa** cria a possibilidade de se modificar o comportamento de um objeto de software em um momento dinâmico de funcionamento. Ajustes realizados nos seus campos e métodos são verificados sistematicamente pelas pré- e pós-condições, emitindo-se um sinal de interrupção caso alguma falha contratual tenha ocorrido.

Como se pode observar no exemplo ilustrativo, a preocupação com o projeto das modificações de uma estrutura de dados dinâmica foi separada de uma outra, referente à interface de funcionamento desejado em termos do tipo de dado abstrato. Neste caso, o contrato adaptativo foi utilizado para lidar com o modelo de evolução da estrutura em separado ao suporte da lógica de armazenagem dos dados do tipo abstrato.

Em uma situação mais concreta, espera-se empregar os contratos adaptativos para projetar um protótipo de ferramenta que sugira configurações de ambulâncias de emergência para atendimento de chamadas de gestantes [16]. Aqui, contratos adaptativos podem considerar que o diálogo entre o atendente e a gestante gradativamente estabelece a estrutura do modelo da gestante até que seja possível calcular a sugestão de configuração de uma ambulância de atendimento.

Há, também, um projeto de pesquisa em andamento cujo objetivo principal é analisar determinadas propriedades de encadeamento de cenas em narrativas de ambientes de aprendizagem. Após a narração de uma cena, a transição para a cena seguinte pode modificar a continuidade da história, considerando eventos de interação proporcionado pelos aprendizes. Contratos adaptativos podem ser empregados para administrar o espaço das cenas seguintes em função das interações dos aprendizes de modo que sejam evitadas configurações com ciclos ou ilhas de narração, por exemplo.

Este artigo também introduziu uma extensão da UML para a representação de contratos adaptativos, bem como dos conceitos de suporte (chamada, invocação indireta e execução). Espera-se que ela auxilie na interpretação dos modelos estáticos e dinâmicos de software projetado com o paradigma de objetos.

Agradecimentos

O autor agradece aos membros do GEMS-PUCSP pelas diversas discussões a respeito do refinamento das ideias referentes aos contratos adaptativos: Carla, Karina, Daniel, Eduardo, Francisco, Giorno e Mário.

Referências

[1] C. Hoare, “An axiomatic basis for computer programming,” *Communications of the ACM*, 1969.

[2] J.C. Reynolds, “Separation logic: A logic for shared mutable data structures,” *Proceedings of the 17th annual ieee symposium on logic in computer science*, Washington, DC, USA: IEEE Computer Society, 2002, pp. 55–74.

[3] B. Meyer, “Design by contract,” *IEEE*pp. 40–51.

[4] B. Meyer, *Object-oriented software construction*, Paperback; Prentice Hall PTR, 2000.

[5] *OMG Unified Modeling Language*, Object Management Group, 2015.

[6] I.S. Vega, “Especificações adaptativas e objetos, uma técnica de design de software a partir de statecharts com métodos adaptativos,” 2012.

[7] C. Bock, “A more object-oriented state machine,” *Journal Of Object-Oriented Programming*, vol. 12, January. 2000.

[8] L. Lamport, *Specifying systems: The tla+ language and tools for hardware and software engineers*, Addison, 2002.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Hardcover; Addison-Wesley Professional, 1994.

[10] J.J. Neto and C. Bravo, “Adaptive automata - a reduced complexity proposal,” 2002, pp. 161–170.

[11] B.W. Kernighan and R. Pike, *The practice of programming*, 1999.

[12] A.V. Aho, J.D. Ullman, and J.E. Hopcroft, *Data structures and algorithms*, Addison-Wesley, Reading, Mass., 1983.

[13] D.S. Touretzky, *COMMON lisp: A gentle introduction to symbolic computation*, The Benjamin/Cummings

Publishing Company, Inc., 1990.

[14] B.C. Pierce, *Types and programming languages*, The MIT Press, 2002.

[15] C.A. Szyperski, *Component software: Beyond object-oriented programming*, AddisonWesley, 1999.

[16] D.F. Lourenço, "A inteligência computacional no auxílio de atendimento a emergências médicas: Atendimentos emergenciais as mulheres gestantes," Dissertação de Mestrado, Pontifícia Universidade Católica de São Paulo, 2016.

Italo S. Vega recebeu os títulos de Mestre e Doutor em Engenharia de Software na Escola Politécnica da Universidade de São Paulo, Brasil, em 1993 e 1998, respectivamente. Professor Associado do Departamento de Computação da Pontifícia Universidade Católica de São Paulo e líder do Grupo de Estudos em Modelagem de Software (GEMS) do programa de pós-graduação em Tecnologia da Inteligência e *Design* Digital (TIDD).

Literature review of formal testing on Adaptive Systems

R. Caya, J.Neto

***Abstract*— This paper presents a review of the literature about the work on the model-based testing area to generate test cases for adaptive systems. We present a brief definition of the testing software activity, its importance as general parameter of software quality, and the particular need for having proper methods to perform it the domain of adaptive systems. We describe as well, the core principle bellow adaptive technology and present the reasoning for the need performing automatic testing for these systems. As result of our literature review we recover the most used formalisms and testing strategies used for adaptive technologies. Finally we present some insights and conclusions derived from the information we gather about the matter of model-based testing adaptive systems.**

***Keywords*— review, adaptivity, testing, state machine, test cases, software testing tools.**

I. INTRODUCTION

Testing is a significant part of software engineering and with the evolution of computer languages, methodologies and techniques that speed up the implementation of complex systems the task for testing and guaranteeing the quality has become increasingly demand [1]. There are several definitions for what is testing and what it aims according with different approaches and methodologies. Highly recommended documentation and specification about the testing process and its taxonomy are provided at [2] [3]. Even though, a in [4] Miller defines the goal of testing as:

"to affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances."

In Miller's approach, a good test is one that has a high probability of finding an as yet undiscovered error, and a successful test is one that uncovers an as yet undiscovered error. As critical as it is, this process can be very time and resource consuming, being applied in different parts of the software developing cycle, from specification to maintenance. According to [1], by 2001, testing typically consume from 40% up to 50% of the software development effort.

Unfortunately, even being as critical as it is, the carry out of testing as formal and strict process is not usually integrated within the software development process [5]. As consequence, some of practitioners on the software industry consider it too expensive, case involves the production of artifacts useful for testing, but not considered at the beginning, i.e. clear or formal requirement specification. However, the guarantee of the quality of a software is a feature that different enterprises demand, some because it is the attribute that identify the trademark, and other because the critical nature of the system, in terms of reliability and robustness, established to be so, i.e. high-confidence medical cyber-physical systems [6].

Particularly, those technologies that manifest a mechanism that allows them to self-managing the decision over changes in its own configuration on run time, technologies that implement adaptive features, have being increasingly research in the past decade [7]. This interest might be consequence of the new needs exposed by the users of the, so called, "Information Society" that claim for flexible systems that can respond to the constant changes in their environments. The whole work at the Laboratório de Tecnologias Adaptativas (LTA) at the University of São Paulo focuses in the concepts, theories and mechanisms that model and implement adaptive behavior. Systems and devices that are complex, heterogeneous and perform dynamic changing behavior can be found in a very wide spectrum of areas, some examples are [8]: Virtual Reality applications, Multi-agent Systems [9], Simulations, Natural Language Processing, Human-Computer Interaction, Computing Theory, Organization and Orchestration of service-oriented software, Pattern Recognition strategies, and many more. An additional care must be taken when looking for adaptive systems and that is the proliferation of terminology [10] that tries to comprise the phenomenon that characterizes adaptivity: self-managing the changes in the configuration of the system (reflected by its behavior) at run time and as consequence of external stimuli.

The motivation for this work is to identify which elements from the testing field can assist developers to examine quality parameters in adaptive systems.

Our objective is to review literature to retrieve the main approaches followed in the area of testing to convey formalism, techniques, strategies that allow the specification of adaptive systems, followed by the generation of test cases that are adequate to guarantee the quality of the system.

The next sections in this document are organize as follows: Section II describe the main concepts related to this work, so it present testing as activity in software development, its classical approaches and formalisms. In the same way, we present the definition some of the well know strategies to conduct testing and to guarantee the quality of a software: Model-Based testing and Property-based testing. Section III we present the definition of Adaptive Technology and its basic features. Section IV describes the formal languages and models used in the literature to perform software testing for adaptive systems. Section V describes the results and insights

R. Caya, Universidade de São Paulo (USP), São Paulo, São Paulo, Brasil, rosalia.caya@usp.br

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

we were able to reach after performing the literature review. Finally, Section VI state the conclusions of the review, some of the difficulties founded and possible future works in the endeavor of applying MBT to adaptive systems.

II. TESTING: DEFINITIONS AND APPROACHES

In this section we define testing as activity in the software development life cycle, the diffent approaches followed by researchers from several fields and some of the formalisms used for work in this area.

A. *Testing as activity*

The task of test a program is at least as old as computing itself. However, Software industry have suffer tremendous growth over the past three decades. Software applications have proliferated from the original task-oriented scientific computing domains into our daily lives in such a way that sometimes we do not realize that some kind of computation is performed when we do everyday tasks. Of course, very highly specialized and complex systems are coming to the rescue in different arenas, such as cyber-physical systems, virtual technology, and medical and military applications, as many others. This proximity implies that software is required to fulfill some quality attributes(to be usable, dependable, and safe) because of its impacts in our daily lives, such as economy, personal and national security, health, and safety. Three decades ago, testing accounted for about 50% of the total time and more than 50% of the total money expended in a software development project—and, actually the percentage is sort of the same today, with a tendency to grow. Actually, global competition, outsourcing, off-shoring, and increasing customer expectations have brought the concept of quality to the forefront. Developing quality products on tighter schedules is critical for a company to be successful in the new global economy. Traditionally, efforts to improve quality have centered around the end of the product development cycle by emphasizing the detection and correction of defects. On the contrary, the new approach to enhancing quality encompasses all phases of a product development process, from a requirements analysis to the final delivery of the product to the customer. Every step in the development process must be performed to the highest possible standard. As consquence, a software system goes through four stages of testing before it is actually deployed. These four stages, as mentioned in [11] and shown in Figure 1, are known as unit, integration, system, and acceptance level testing.

The objective of testing is to deliver a reasonably stable, roboust system that can satisfy the needs of the user and at the same time meeting a tight schedule, to discovering most of the faults, and to verifying that fixes are working and will not result in new faults. In general, software testing is a highly labor intensive task, it comprises a number of distinct activities, and it consumes a lot of resources: time and human. This high cost is one of the reasons why test automation is very attractive. With the help of proper tools the durations of those tasks can be shortened, and the skills of est engineers can be focus in designing and developing better and refined strategies to generate automated test cases instead of manual and extenuating work.

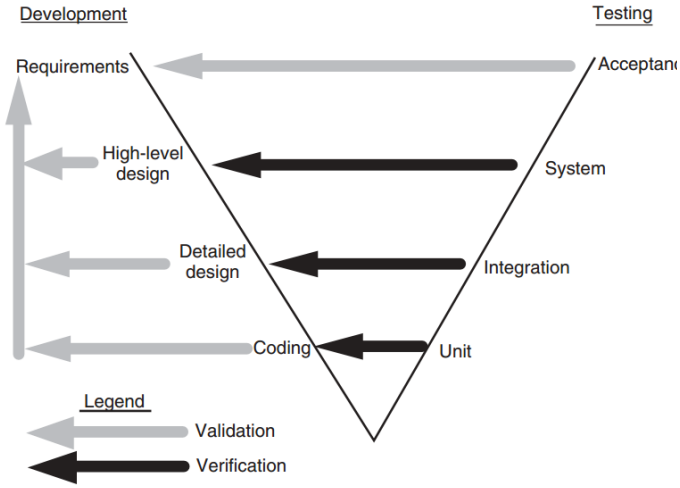


Figure 1. Software Testing Four Stages presented at [11].

B. *Classical Approaches*

As stated before, seems that test automation may be of great help, but it comes with a cost: sufficient time and resources need to be allocated for the development and maintaining of an automated test suite. Different approaches consider the sources from which it will be designed, and others consider the moment in the software life-cycle in which testing will be applied.

In the first case, test cases need to be designed by considering information from several sources, either from the requirements specification of a system (functional or black box testing) or from source code (structural or white box testing). In specification-based testing a key element is that the requirements had been specified in a formal manner, so they are a precise and detailed description of the system's functionality and constraints so the can be a the point of reference for test data selection and adequacy.

In the second case, it is necessary to distinguish between two similar concepts related to software testing: validation and verification. In case of validation, it establishes the correspondence between a system and users' expectations. In this type of testing a key element is the specification of what functionality is needed and which additional properties the system should posses. The objetive of validation is to determine if the entire system meets the customer's needs and expectations. In case of verification, a key element is building a high-level description (abstraction) of the functionality the system is supposed to have. Then, the whole testing process is about making sure of the correspondence of an implementation of the software with its specification.

In case of this work we are interested in study the automation of functional test cases towards verification of a system. This can be done by property extraction or model-

based testing.

C. *Model-based testing:*

The goal of model based testing is to show that the implementation of the system behaves compliant with this model. Model based testing uses a concise behavioral model of the system under test, and automatically generates test cases from the model [12].

As mention before, the goal of testing is to identify differences between the behavior of the implementation and the intended behavior of a system under test (SUT), as expressed by its requirements [13]. Usually, the model of the SUT is given as a black box that accepts inputs and produces outputs, but also exists testing processes that work with white and gray box approaches [3]. Model-Based Testing (MBT) is an area of testing in which the work relies on explicit abstract formal behavior models that encode the behavior of the SUT or its execution environment or sometimes both of them [14]. In this context, abstraction is not only beneficial, but also necessary for hiding unnecessary programming details as well as reducing the complexity of the system. However, it is also essential for the test model to be detail enough in order to generate effective test cases. The model must describe possible input/output sequences, and is linked to the implementation by a conformance relation. Finding the right level of abstraction for the test model is one of the challenges in MBT and is for now regarding to the test team.

The basic idea, and main practical advantage, of MBT is that from a formal or semi-formal model complete test cases can be generated. Another expected advantage of MBT is that it is hope to mitigate some well know problems of deriving tests "by hand", most of all because of its dependency on the approach of single engineers [3]. The ideas of MBT date back to 70's with studies about specification-based testing [15] [4]. Based on these studies, software system now can be specified in more rigorous, understandable, clear ways, which has brought great chances to improve automated functional testing techniques. Meanwhile, software development has evolve from standalone systems to complex, heterogeneous, distributed, real-time, and component based systems. This evolution joined by the advances reached in the area of formal verification have led to an increased level of interest from the academic field as well as from the industry.

However, the methods and tools used in MBT are dependent of the creation of the formal model for the system under test. This way, MBT does not replace traditional test design, but supports them in a way that allows testers to improve their understanding of the domain and to perform tests more effectively and efficiently.

Nowadays, a variety of models and different notations are available to present the system from different perspectives. Some formal notations use Finite State Machines, grammars and set theory to present precise semantics. Some other prefer more visual representations, such as diagrams and tables, to facilitate communication with different audiences. Some of the most known models are:

- UML
- Statecharts and Extended Finite State Machines
- Petri nets
- Tasks models
- Z, Larch, VDM, Pre/Post Conditions, Estelle, and λ-calculus, and π-calculus

The multitude of specification models and its semantics allow different degrees of conduciveness for analysis and impose a difficulty for people looking to develop encompassing technology for model-based testing. In [16] some of the typical misleading expectations, misunderstandings and pitfalls are point out as:

- MBT solves all problems;
- MBT is just a matter of tooling;
- Models are always correct Test case explosion will occur.

Nowadays, the research in this field has ignite. Several international conferences (ACM TAAS, IEEE ICCAC, SEAMS, Adaptive) have been created on the specific topic, aiming to explore, develop, implement and compare different techniques according to the features of the SUT. In the same way, some of the most important events of the area of Software Development have incorporated the MBT and formal verification as a track of interest.

III. ADAPTIVE SYSTEMS

Adaptation is a promising approach to manage the complexity embedded in contemporary systems [12] [7]. An adaptive system is an entity able to adapt autonomously, reconfigure at run-time, to internal dynamics according to sensed conditions in the environment to achieve a particular goal. [12] [17]. The increasingly complexity and heterogeneity in nowadays systems make difficult to apply static strategies to reflect dynamic behavior, so adaptive methods and technology has become an important research topic in many diverse application areas [18]. One of the more recently growing fields in research related with adaptivity is in Software Engineering [18] under the terminology of *SaS* (Self-adaptive Systems). This way has become necessary to include adaptivity in the research agenda of technological community.

Basically, an adaptive system, as shown in Figure 2, comprises two parts: the managed system (also called system layer, managed resources, core function, underlying subsystem as presented in [7]) and the managing system (or architecture layer, autonomic manager, adaptation engine, reflective subsystem, adaptive layer, as presented in [7]) .

The underlying subsystem deals with the domain functionality and the adaptive layer deals with the adaptations of the managed system to achieve particular quality objectives. The key underlying principle of adaptive systems is complexity management through separation of concerns.

A. *Adaptive Technologies at LTA*

The Laboratório de Tecnologias Adaptativas (LTA) at the Politechnical School of the University of São Paulo have develop several researches studying adaptive technology since the late 90s. As result several formalisms, techniques, methods, tools and applications were develop based on the incorporation on the features of adaptive behavior. In these researches, adaptivity is applied in different areas, such as formal languages [19], human-computer interaction [20], formal models [21] [22], data mining, robotics [23], and so on, proving the potential and versatility that the basic and intuitive concept of adaptivity has. In the context of the works at the LTA, adaptivity is understood as the ability a system has to respond to external stimuli, deciding and applying dynamic self-reconfigurations in its structure [8] and behavior [22], taking into account the historical information stores from previous executions [19]. This way the term adaptive technology refers to the application of adaptivity to practical and concrete purposes, meaning that an adaptive device is a rule-driven device able in an independent way of applying changes in that same ruleset and processing the corresponding actions.

One interesting feature about adapting entities is that even when most of the time different research laboratories do not have close collaborations the approach about the foundations of adaptivity seem concomitant. For example, the core formulation developed and currently use to define adaptive devices at LTA is in close concordance with the general one presented at the begging of this section. In the formalism it is possible to identify two major components [22]: the underlying device (usually a non-adaptive one), and the adaptive mechanism, responsible for the incorporation and well performance of adaptivity. The fact that exist a major concordance about the core formalism below adaptive technology is very important because allow us to gather several researches that otherwise could remain as disseminated efforts. This work is actually one of several efforts aiming at helping in the integrations of the theory of adaptive technologies and other fields in computing.

However, the importance of study adaptivity does not hold only in creating adaptive devices, it also covers the understanding of the different features and properties implied by it. For example, to correctly incorporate and create adaptive devices on-the-fly, through strategies as negotiation in Multi-Agent Systems or orchestration in Service-Oriented Software, it will be necessary to identify the properties this dynamic entities should satisfy. Furthermore, researching and exploring literature about formal representation allows developing appropriate tools to apply technology available for non-adaptive software.

It also allow us to distinguish very closely related but different terms by understanding its differences, as is the case with adaptability and adaptivity.

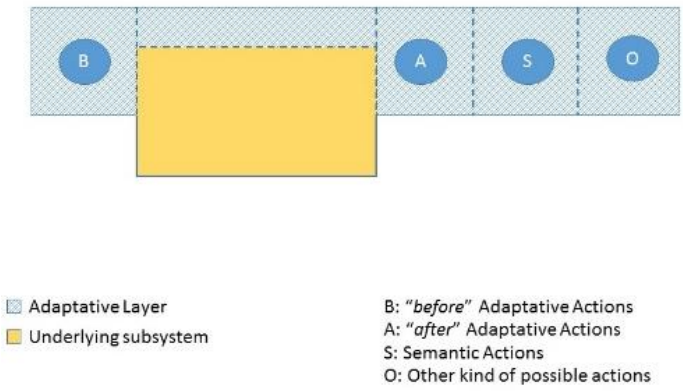


Figure 2. Basic formulation of an adaptive device at LTA

IV. TESTING ON ADAPTIVE SYSTEMS

As we have seen before, the complexity and heterogeneity in nowadays systems is requiring new technologies to be develop and to measure the quality of such systems. In this context, one major challenge in self-adaptive systems is to assure the required quality properties [7]. This is the kind of task in which the work developed in Testing (Validation and verification) can assist. However, before applying the algorithms ad strategies from testing it is necessary to satisfy its requirements and avoid understand its constraints. As an emerging partnership testing adaptive software needs to face some intrinsic challenges, some of them are:

- Handle its natural complexity, uncertainty and incomplete information
- The expansion of decision space
- Handling error propagation

We need a formal model to support the specification of such systems [12], we need to understand how the adaptive features are encoded in such models, we need to choose the proper testing criteria to check those properties [12], and we need to use some kind of formalism to process the model and generate the test cases. Adaptive features are, in formal terms, behavioral properties [12] in the system.

The necessity of testing adaptive systems has been recognized not only by the academic community researching in adaptive behavior, or the software development arena, it has being pointed out as a necessity in the testing community [24].Because this systems are increasingly entering the industry venue and there is need for certification from costumers [17]. It is necessary to provide validation and verification methods to motivate the adoption of adaptive technology by industry and unlock the potential of adaptivity beyond academic arena [25]. Notwithstanding, as we have stated before the task of

testing is by itself very extenuating even in the case of software with classical behavior, so in the case of adaptive, mostly changing, partly undefined behavior the complexity rises. This way, is not possible to perform the activities required for testing by hand because the amount of information that needs to be consider by testers, for designing the tests, incorporate into the test strategy, monitoring, and evaluate, is huge and make the task a herculean mission for engineers.

To design and develop automatic test case generation it is fundamental to describe the system's behavior in a precise way. This is the main objective of formal specification techniques.

Formal specification differs from other type of specification in its high level of detail, precision and correctness. A formal specification language is compose by a clear syntax, a precise semantics within the domain and a proof theory [26]. Whereas other notation aim for more visual or user-friendly presentations, formal languages usually have a syntax based in mathematical techniques that allows analyzing and applying reasoning over it. The branch of mathematics used is discrete mathematics and the mathematical concepts are drawn from set theory, logic and algebra. The use of formal methods is increasing in the area of critical systems development, where emergent system properties such as safety, reliability and security are very important. In such areas, the possibility of automatic derivation over an specification is one of the main advantages of formal languages over informal ones. Two fundamental approaches to formal specification have been used to write detailed specifications for industrial software systems. These are: algebraic and model-based. A basic description of the most used techniques in each case will be given in the following sections.

B. Languages for formal specification of Adaptive Systems

The principle in this approach is to specify a system as a structured collection of mathematical functions. The can be grouped by defining algebraic structures, logical theories or as a structured collection of processes.

Some of the formal languages used in studies to specify systems with dynamic reconfigurations, such as adaptive systems, are:

- **Process calculi:** is a family of approaches for formally modelling concurrent systems that provide a high level description of interactions, communication strategies and synchronization between a collection of individual processes. The most used are π-calculus [27] [9] and LOTOS.
- **Timed automaton** [7] [28]**:** is a theory for modeling and verification of real time systems. It is basically a finite automaton extended with real-valued variables.

C. Models for formal specification of Adaptive Systems

Model-based specification exposes the system state and defines the operations in terms of changes to that state. Some of the used models to describe adaptive systems are:

- **Petri nets** [29]: is a state-transition system mainly used to work with distributed systems. It offers an exact mathematical definition of their semantics, a well-developed mathematical theory for process analysis and a graphic representation. Petri nets are well suited for modeling the concurrent behavior of distributed systems. Several extensions of the original formalism created by Carl Adam Petri at 1939 to meet some other features [30].
- **Z**: The Z notation is based upon set theory and mathematical logic. The set theory used includes standard set operators, set comprehensions, Cartesian products, and power sets. The mathematical logic is a first-order predicate calculus. A characteristic feature of Z is the use of types. A variation called uSZ [13] has combine statecharts with Z and temporal logic.

- **Different kinds of EFSM (**Extended Finite State Machine**) and hybrid approaches:** Characterize the required transitions from state to state. The properties of interest are specified by a set of transition functions in the state machine transitions [26]. Statecharts, are one of the most used formalisms to specify dynamic systems behavior because it offers hierarchy, concurrency, broadcasting strategy and history states.

The main observation made at [7] is that regular algebra is one of the most used modeling languages. One of the reasons for that besides the strong community of formal verification in computer science, is that the majority of studies working in formal specification of adaptive systems formulate the properties of interest in the modeling language they use for modeling.

The use of traditional formal modeling languages such as transition systems, automata, state machines and EFSM, Markov models, graphs, and process algebras (*μSZ* [13], λ-calculus, and π-calculus) is about equally distributed and most of the time some type of logic (frst order, deontic, modal and temporal) as property specification language is combined with other modeling languages.

D. Models for testing adaptive systems

Some of the most used models in literature to design testing strategies for systems that possess adaptive features are the following:

- **Linear Temporal Logic**: convenient formalism for specifying and verifying properties of reactive systems [31]. The strategy for testing (by theorem proving or model checking) is to express desired properties using LTL operators and actually check if the model satisfies this property. This model provides a particularly useful set of

operators for constructing LT properties without specifying sets.

- **Computation Tree Logic CTL*** [7]: is a temporal logic where full computation tree logic has no syntactic restrictions on the applications of temporal operators and path quantifiers, allowing explicit quantification over all possible futures starting at some initial state.

- **Communicating Sequential Processes**: [32] formal language for describing patterns of interaction in concurrent systems [29]. CSP has been successfully applied in industry as a tool for specifying and verifying the concurrent aspects of a variety of different systems.

V. RESULTS FROM SYSTEMATIC ANALYSIS OF LITERATURE

From the performed literature review, it is possible to obtain some key insights about the state of the art of testing adaptive software. They are presented grouped by concerns in an intent for clarity.

A. About systematic reviews

There is a clear need for performing literature review, in particular systematic review that allow incoming researchers get organized, clear and synthetized information about specific endeavors to solve current challenges. Following the same logic it is also important to give continuity to work already available to keep the information up to date [7], especially for uses in industry.

B. About Testing criteria for testing adaptive systems

Some well-known classic testing criteria have been applied in adaptive systems, such as: simple, statement coverage, pair-wise, dependency-based and path coverage giving some results about quality. However, the need for exploring and formulate proper criteria to specifically test adaption properties remains a challenge [17] [24].

In the same manner, some requirements for testing adaptive system have been developed in resent research projects [17]. A new approach is the one presented at [24] called Veritas, in which a technique for generating test cases that evolves and customize the set of test cases, and/or parameters, to correspond the current conditions of the adaptive system.

C. About testing models for adaptive systems

Some models have been used to specify adaptive systems, most of them were recommended by literature because they consider properties closely related to adaptivity such as: concurrency, distributive components, communication strategies, and historical records. Most existing formal approaches for self-adaptive systems assume a central point of control to realize adaptations [12].

A particular innovative proposal for organizing the testing activity is the one made in [12] and presented at Figure 3 called zones in the state space for different behaviors of adaptive systems. In the *normal behavior zone*, the system is performing its canonical functionality. In the *undesired behavior zone*, the system has entered in a state where adaptation is required. In the *adaptive behavior zone*, the system is adapting itself to manage the undesired behavior, if the adaptation succeed the adjusted system will return to a normal behavior. In case the adaptation fails or generate consequences that can no be properly handled by the system it will enter in the last zone. Finally, the *invalid behavior zone* corresponds to states where the system should never be. One of the most interesting contributions of this mapping model is the possibility to map properties of interest for testing with respect to the system model using transitions between different zones.

Another interesting proposal is one presented at [33]. Their approach uses evolutionary computation to adjust requirements-based test cases at run time so the test suite can handle different system and environmental conditions.

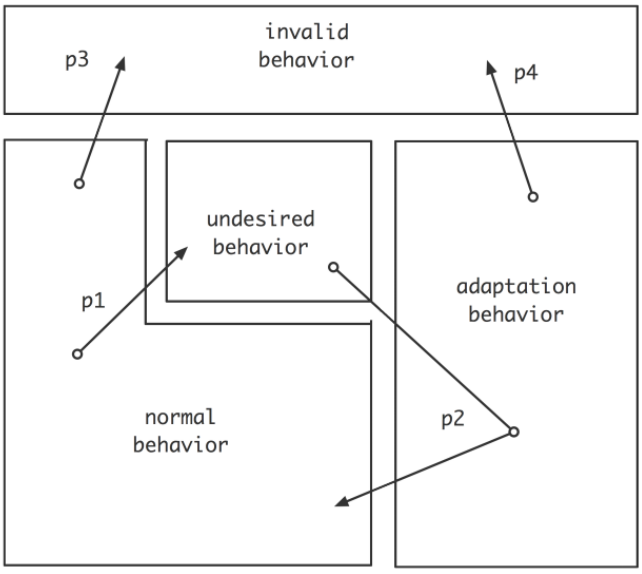


Figure 3. Zones in the space state of an adaptive system.

D. About properties of interest in adaptive systems

From the studies reviewed [7] [25] [24] [33] [27] we can report that the top concerns of interest in adaptation are the following:

- efficiency/performance,
- reliability,
- resilience [25]
- guaranteeing functionality, and
- flexibility [12].

Other properties measure through model testing are: robustness [25] [12], and correctness of adaptations [12].

Traditional properties, including safety, liveness and reachability [12] and deadlock are tested too, but in less frequency.

VI. CONCLUSION

After performing the review we arrive to some conclusions about the challenges that remain open and some of the issues in this research field that require attention from the academic community.

First, we do not found models developed to specify the particularities of adaptive systems behavior. There are formalisms that provide some tools to translate adaptive to some of its properties, but some of them do not allow an entire mapping. Moreover, the very use of formal methods as evidence of system properties remains limited. The development of environment models are an essential condition for model based testing of runtime qualities, which is central to self-adaptation [12]. It allows an engineer to specify in a precise manner the failure scenarios of interest and the conditions under which the failures happens.

Second, no standard tools have been emerged for formal modeling and verification of adaptive systems. [7].

Third, a particular difficult step in the testing activities becomes extremely complex in case of adaptive systems: identify the properties or concerns it is important to test. Several factors need to be consider: the domain of the system, the focus of the test, the strategy of the testing criteria, and the processing of the results. In the literature review we observed that most of the self-* properties (self-healing, self-configuring, and self-optimizing, self-organizing) which are able to form emergent behavior are not properly specified in testing studies. Actually, only self-organizing property has being point out as flexibility in some studies [24].

Four, once we know the properties we need to test selecting the right testing criteria according to them becomes a cumbersome task. The test engineer must take into account controlling the combinatory explosion product of either state space, decision space, path or propagation at executing the strategy. Few information about this concern was found in the literature review, which raises the question about how to deal with these dangers in an efficient way.

Five, managing the testing activity requires not only good technology, but also well prepared managers. This way, Software testing engineers need to be properly train in the use of formal methods to be able to design suites of tests, evaluate the adequacy of testing criteria and analyze the results with mathematical methods. Most of the time testing theory is not included in the curricula of several courses of either software engineering or Computer Science.

Six, there is a need for optimizing test strategies to reduce the consuming of resources, high costs, tenuous manual efforts, and processing of failure at test design activities.

Finally, we considered that performing a systematic revision help us to assemble a consistent vision about adaptivity at two levels: an inner level and an outside level. At inner level we could verify that adaptivity consist on a simple and intuitive concept, a feature, that can be applied to solve or empower different areas. At the outside level, we could gather information in a systematic and organized way and verify the different potential areas in which adaptivity can be beneficial.

This way we consider that systematic reviews help to deeply understand the area of interest and the key information about our particular problem. It is our thought that motivating researchers at early stages of academic life is very fruitful to strength and clarify the emerging area of adaptive technology.

VII. REFERENCES

[1] L. Luo, "Software testing techniques - technology maturation and research strategy," Institute for Software Research International, Carnegie Mellon University, Pittsburgh, 2001.

[2] M. Y. Shafique, "A systematic review of state-based test tools," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 1, p. 59–76, February 2015.

[3] M. Utting, A. Pretschner and B. Legeard, "A taxonomy of model-based testing approaches," *SOFTWARE TESTING, VERIFICATION AND RELIABILITY*, vol. 22, p. 297–312, 12 April 2012.

[4] E. F. Miller, "Introduction to Software Testing Technology," in *Tutorial: Software Testing & Validation Techniques*, Second ed., Vols. IEEE Catalog No. EHO 180-0, IEEE Computer Society, 1980, pp. 4-16.

[5] A. C. Dias Neto, R. V. M. Subramanyan and G. H. Travassos, "A survey on model-based testing approaches: a systematic review," in *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, Atlanta, Georgia, 2007.

[6] E. A. Lee, "Cyber physical systems: Design challenges," in *11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, Orlando, Florida, 2008.

[7] D. a. I. M. U. a. d. l. I. D. G. a. A. T. Weyns, "A Survey of Formal Methods in Self-adaptive Systems," in *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering, C3S2E '12*,

Montreal, Quebec, Canada, 2012.

[8] J. J. Neto, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa," vol. 5, no. 7, p. 496–505, 2007.

[9] W. Jiao, M. Zhou and Q. Wang, "Formal framework for adaptive multi-agent systems," in *IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003*, 2003.

[10] K. Compton and S. Hauck, "An introduction to reconfigurable computing," *IEEE Computer*, April 2000.

[11] K. a. T. P. Naik, "Chapter 10: Test Generation from FSM Models," in *Software Testing and Quality Assurance: Theory and Practice*, New Jersey, John Wiley & Sons, Inc, 2008.

[12] M. U. a. D. W. Iftikhar, "A case study on formal verification of self-adaptive behaviors in a decentralized system," in *11th International Workshop on Foundations of Coordination Languages and Self Adaptation (FOCLASA'12)*, 2012.

[13] K. Bogdanov, "Atomated testing of Harel's statecharts (PhD. Thesis)," University of Sheffield, Department of Computer Science, Sheffield, 2000.

[14] F. Siavashi and D. Truscan, "Environment modeling in model-based testing: concepts, prospects and research challenges: a systematic literature review," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, New York, 2015.

[15] T. Chow, "Testing Software Design Modeled by Finite-State Machines," *IEEE Transactions on Software Engineering*, Vols. SE-4, no. 3, pp. 178 - 187, May 1978.

[16] The International Software Testing Qualifications Board, "Model-Based Tester Extension Syllabus," The International Software Testing Qualifications Board, 2015.

[17] G. Püschel, S. Götz and C. a. A. Wilke, "Towards Systematic Model-based Testing of Self-adaptive Software," in *ADAPTIVE 2013, The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications*, n Valencia, Spain, 2013.

[18] R. de Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, D. Weyns, L. Baresi, B. Becker, N. Bencomo, Y. Brun, B. Cukic and R. Desmarais, "Software Engineering for Self-Adaptive Systems: A Second Research Roadmap," in *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, 2013.

[19] J. J. Neto and A. V. Freitas, "Using Adaptive Automata in a Multi-Paradigm Programming Environment," in *International Conference on Applied Simulation and Modelling, ASM2001 - IASTED*, Marbella, Espanha, 2001.

[20] J. J. Neto, "Um Glosário sobre Adaptatividade , Laboratório de Linguagens e Técnicas Adaptativas," in *Memórias do WTA 2009 - Terceiro Workshop de Tecnologias Adaptativas*, São Paulo, Brasil, 2009.

[21] J. Neto and C. Pariente, "Adaptive Automata - a reduced complexity," in *Proceedings of the Conference on Implementation and Application of Automata–CIAA*, Tours, France, 2002.

[22] A. H. Tchemra, "Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão," *IEEE America Latina*, pp. 552-556, 2007.

[23] M. de Sousa, A. Hirakawa and J. Neto, "Adaptive Automata for Mapping Unknown Environments by Mobile Robots," in *Proceedings of the Ibero-American Conference on Artificial Intelligence*, 2004.

[24] B. a. S. H. a. K. A. a. R. W. Eberhardinger, "Towards Testing Self-organizing, Adaptive Systems," in *Proceedings of International Conference on Testing Software and Systems: 26th IFIP WG 6.1 , ICTSS 2014*, Madrid, Spain, 2014.

[25] J. Cámara, R. de Lemos, N. Laranjeiro, R. Ventura and M. Vieira, "Testing the robustness of controllers for self-adaptive systems," *Journal of the Brazilian Computer Society*, vol. 20, no. 1, pp. 1-14, 23 January 2014.

[26] A. v. Lamsweerde, "Formal Specification: A Roadmap," in *Proceedings of the Conference on The Future of Software Engineering, ICSE '00*, Limerick, Ireland, 2000.

[27] A. M. a. A. K. Misra, "Formal Aspects of Specification and Validation of Dynamic Adaptive System by Analyzing Execution Traces," in *2011 Eighth IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems*, Las Vegas, Nevada, 2011.

[28] Uppsala Universitet; Aalborg University, "Uppaal," [Online]. Available: http://www.uppaal.org/.

[29] J. a. C. B. H. C. Zhang, "Model-based Development of Dynamically Adaptive Software," in *Proceedings of the 28th International Conference on Software Engineering, ICSE '06*, Shanghai, China, 2006.

[30] F. D. a. A. A. D. M. Zhou, "A hybrid methodology for synthesis of Petri net models for manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 350-361, June 1992.

[31] Stanford University, "Temporal Logic," Stanford Encyclopedia of Philosophy, [Online]. Available: http://plato.stanford.edu/entries/logic-temporal/#LinTimTemLogLTL.

[32] C. A. R. Hoare, "Communicating Sequential Processes," *Commun. ACM*, vol. 21, no. 8, pp. 666-677, August 1978.

[33] E. Fredericks, B. DeVries and B. Cheng, "Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2014.

Rosalia Edith Caya Carhuanina was born in Lima, Peru. She received the degree of BsC (Bachelor in Science and Engineering) with major in Computer Engineer, and *the* professional degree of Computer and Systems Engineer from the Pontifical Catholic University of Peru (PUCP, Pontificia Universidad Católica del Perú) at Lima in 2009. Between the years 2007 and 2011, she participated of the activities at the Artificial Intelligence Research Group at the Computer Engineering Department of the same university. She holds a MsC degree in Electric Engineering, with major in the field of Computer Engineering from the University of São Paulo (USP) at São Paulo, Brazil. She is currently a PhD student developing her research in the Laboratory of Adaptive Languages and Techniques (LTA). Her main areas of research are: Artificial Intelligence, Natural Language Processing, Human-Machine Interaction and Adaptive Technology.

João José Neto graduated in Electrical Engineering (1971), Master in Electrical Engineering (1975), PhD in Electrical Engineering (1980) and Professor (1993) at the Polytechnic School of the University of São Paulo. Currently, he is an associate professor at the Polytechnic School of the University of São Paulo and coordinates the LTA - Laboratory of Languages and Adaptive Technology of PCS - Department of Computer Engineering and Digital Systems of EPUSP. He has experience in the area of Computer Science, with emphasis on the Fundamentals of Computer Engineering. He works mainly on the following topics: adaptive devices, adaptive technology, adaptive automata, and in their applications to Computer Engineering, particularly in adaptive decision making systems, analysis and processing of natural languages, construction of compilers, robotics, computer-aided teaching, modeling of intelligent systems, automatic learning processes and inferences based on adaptive technology

Incorporação de novas regras em uma Rede de Petri Colorida Adaptativa

H. I. Guibu e J. José Neto

***Abstract*— Este artigo apresenta a utilização de Rede de Petri Colorida Adaptativa (RdPCA) que incorpora o conhecimento de especialistas de diferentes áreas. A RdPCA é uma extensão da Rede de Petri Colorida (RdPC) que permite a alteração de sua topologia de forma dinâmica. Anteriormente, Camolesi definiu a Rede de Petri Adaptativa (RdPA) a partir de uma Rede de Petri (RdP) ordinária[2],utilizando o esquema se dispositivo subjacente envolto por uma camada adaptativa. Normalmente, os especialistas em uma determinada área conseguem estabelecer um conjunto de regras para o bom funcionamento de um empreendimento, de um negócio, ou mesmo de um processo de fabricação. Por outro lado, é possível que este mesmo especialista tenha dificuldade em incorporar este conjunto de regras a uma RdPC que descreva e acompanhe a operação do empreendimento e, ao mesmo tempo, seja aderente às regras de bom desempenho. Para incorporar as regras do especialista a uma RdPCA, transforma-se o conjunto de regras do formato IF – THEN para o formato de tabela de decisão adaptativa (TDA) e, em seguida, funções adaptativas da RdPCA incorporam estas regras à RdPCA.**

***Keywords*— Redes de Petri Adaptativas, Redes de Petri Coloridas, Redes de Petri Adaptativas Coloridas.**

I. INTRODUÇÃO

As Redes de Petri Coloridas são um aperfeiçoamento das Redes de Petri introduzidas por Carl Petri na década de 60. Em função de sua capacidade na descrição de problemas complexos, sua utilização se espalhou tanto na área de engenharia como também na área administrativa.

Na área de sistemas de apoio à tomada de decisão, as ferramentas mais utilizadas pelos especialistas são as tabelas de decisão, que deram origem a diversos métodos com o objetivo de auxiliar os gestores em suas escolhas.

Entre os métodos desenvolvidos para apoio à decisão estão os denominados métodos multicritério, que envolvem a adoção de múltiplas tabelas de decisão encadeadas de forma hierárquica.

No processo de aperfeiçoamento das tabelas de decisão, observa-se o surgimento de novas características que, apesar de mais complexas, proporcionam aos especialistas a capacidade de descrever de forma mais realista o seu modelo de trabalho.

Neste artigo estão descritos o modo de operação das tabelas de decisão e o modo de transcrever as regras das tabelas para funções estendidas das redes de Petri.

Ao embutir uma tabela de decisão em uma rede de Petri, as ferramentas de simulação e análise disponíveis nos ambientes de desenvolvimento das redes de Petri podem ser utilizadas, o que gerar um aumento da confiança nos critérios de decisão adotados.

II. TABELAS DE DECISÃO

A Tabela de Decisão é uma ferramenta auxiliar na descrição de procedimentos de problemas complexos [8].

Uma Tabela de Decisão Convencional, apresentada na Tabela 1, pode ser considerada como um problema composto por condições, ações e regras onde as condições são variáveis que devem ser avaliadas para a tomada de decisão, as ações são o conjunto de operações a serem executadas dependendo das condições neste instante, e as regras são o conjunto de situações que são verificadas em resposta às condições [8].

TABELA I
Tabela de Decisão Convencional

	Colunas de Regras
Linhas das Condições	Valores das Condições
Linhas das Ações	Ações a serem tomadas

Uma regra é constituída pela associação das condições e das ações em uma dada coluna. O conjunto das colunas de regras devem cobrir todas as possibilidades que podem ocorrer em função das condições observadas e das ações a serem tomadas.

Em função das condições atuais de um problema, procuram-se quais regras da tabela satisfazem estas condições:

- Se nenhuma regra satisfaz as condições impostas, nenhuma ação é tomada;
- Se apenas uma regra se aplica, então as ações correspondentes à regra são executadas;
- Se mais de uma regra satisfaz as condições, então as ações correspondentes às regras são aplicadas em paralelo.
- Uma vez aplicadas as regras, a tabela pode ser utilizada novamente.
- As regras de uma tabela de decisão são previamente definidas e novas regras só podem ser acrescentadas ou suprimidas através de uma revisão da tabela.

II - 1 -TABELAS DE DECISÃO ADAPTATIVAS.

Em 2001 Neto [7] introduz a Tabela de Decisão Adaptativa (TDA) a partir de um dispositivo adaptativo dirigido por regras.

Além da consulta às regras, uma TDA permite que se inclua ou que se exclua uma regra do conjunto de regras durante a operação do dispositivo.

Como exemplo de seu potencial, Neto simula um autômato adaptativo para reconhecer sentenças de linguagens dependentes de contexto.

Na TDA uma tabela de decisão convencional é o dispositivo subjacente ao qual será acrescentado um conjunto de linhas para definição das funções adaptativas.

As funções adaptativas constituem a camada adaptativa do dispositivo adaptativo dirigido por regras. A modificação do conjunto de regras implica no aumento do número de colunas no caso de inserção de regras, ou na diminuição do número de colunas, no caso de exclusão de regras. Em ambos os casos a quantidade de linhas permanece fixa.

A Tabela de Decisão Adaptativa (TDA) é capaz de alterar seu conjunto de regras como resposta a um estímulo externo através da ação de funções adaptativas [8]. Entretanto, a implementação de ações mais complexas não é uma tarefa simples em virtude da limitação das três operações elementares [9].

Quando um dispositivo adaptativo típico está operando e não encontra regras aplicáveis, ele interrompe a sua execução, indicando que esta situação não era prevista.

Para dispositivos de operação contínua, que não possuem estados terminais de aceitação ou rejeição, parar sua execução seria reconhecer uma situação não prevista e constitui um erro.

II - 2 - TABELAS DE DECISÃO ADAPTATIVAS ESTENDIDAS.

Para superar este problema enfrentado pelos dispositivos de operação contínua, Tchemra [8] criou uma variante da TDA e a denominou Tabela de Decisão Adaptativa Estendida (TDAE).

Na TDAE, a adaptatividade não se aplica apenas durante a aplicação de uma regra, mas também na ausência de regras aplicáveis.

Um dispositivo auxiliar modificador é consultado e a solução produzida pelo modificador é incorporada à tabela na forma de uma nova regra, ou seja, na repetição das condições que provocaram a chamada do dispositivo auxiliar modificador, a nova regra será executada e o dispositivo modificador não precisará ser chamado.

III. REDES DE PETRI

As redes de Petri (RdP) foram criadas por Carl Petri na década de 60 para modelar a comunicação entre autômatos.

Formalmente, uma rede de Petri é uma quádrupla RdP = [P,T,I,O] onde

P é um conjunto finito de lugares;

T é um conjunto finito de transições;

I:(P x T) → N é a aplicação de entrada, onde N é o conjunto dos números naturais;

O:(T x P) → N é a aplicação de saída, onde N é o conjunto dos números naturais.

Uma rede marcada é uma dupla RM= [RdP,M] , onde RdP é uma rede de Petri e M é um conjunto com a mesma dimensão de P tal que M(p) contém o número de marcas ou fichas do lugar p.

No instante inicial, M representa a marcação inicial da RM e vai variando ao longo do tempo na medida em que as transições vão se sucedendo.

Além da forma matricial indicada na definição formal as redes de Petri, é possível interpretar as redes de Petri como um grafo com dois tipos de nós interligados por arcos que apresenta um comportamento dinâmico, e também como um sistema de regras do tipo "condição → ação" que representam uma base de conhecimento.

A Figura 1 apresenta uma rede de Petri em forma de um grafo, no qual os círculos são os "Lugares", os retângulos são as "Transições". Os "Lugares" e as "Transições" constituem os nós do grafo e eles são interligados através dos arcos orientados.

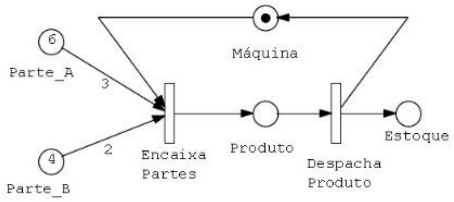


Figura 1. Exemplo de uma rede de Petri.

III – 1 - REDE DE PETRI COLORIDA

A rede de Petri colorida é uma extensão da rede de Petri ordinária na qual os tipos das fichas não são apenas indicações do estado do sistema, mas sim todos os tipos de variáveis que uma linguagem de programação moderna possui [5].

A rede de Petri colorida está para uma rede de Petri ordinária da mesma forma que uma linguagem de programação de alto nível está para uma linguagem assembler.

Formalmente, uma rede de Petri colorida é definida por uma 9-tupla:

RdPC = (P, T, A, Σ, V, C, G, E, I) onde

P é o conjunto finito de lugares,

T é o conjunto finito de transições,

A é o conjunto de arcos direcionados,

Σ é o conjunto não vazio de cores,

V é o conjunto de variáveis,

C é o conjunto de funções de guarda,

E é o conjunto de expressões dos arcos,

I é a função de inicialização.

As cores da RdPC são diferentes tipos de ficha que, no caso da RdP ordinária, é de um único tipo que corresponde aos números naturais. Além disso, as transições podem depender de expressões

III - 1 - REDES DE PETRI RECONFIGURÁVEIS

Diversas extensões das Redes de Petri foram desenvolvidas com o objetivo de simplificar a modelagem de Sistemas a Eventos Discretos, inclusive para ambientes distribuídos.

Entretanto, a maioria das extensões não foi projetada para modelar sistemas que se modificam durante sua operação.

Uma vertente das extensões das Redes de Petri que procura atacar o problema da modelagem de sistemas que se modificam durante sua operação é constituída pelas Redes de Petri Auto-modificáveis [10], pelas Redes de Petri Reconfiguráveis via reescrita de grafos [7], pelas Redes de Petri Reconfiguráveis via mecanismo modificador[4], pelas Redes de Petri Adaptativas [1], pelas Redes de Petri Difusas e Adaptativas [6].
Cada uma destas extensões possui características próprias, mas compartilham o fato de poderem modificar, durante a execução, as regras de disparo das transições ou a topologia da rede.

Com o mesmo nome, a mesma sigla, mas de origens diversas, encontramos na literatura as Redes de Petri Reconfiguráveis (Reconfigurable Petri Nets - RPN) introduzidos em [4] e em [7].

O trabalho de Llorens e Oliver é uma evolução do trabalho de Badouel e Oliver [[1] e combina as técnicas das gramáticas de grafos com a ideia de Rede de Petri Auto-Modificável de Valk, criando um sistema de reescrita da rede. Neste trabalho, Llorens e Oliver demonstraram a equivalência entre as RPN e as PN em termos de propriedades e também que as RPN são equivalentes às máquinas de Turing quanto ao poder de expressão.

Na Figura 2 temos esquematizado uma Rede de Petri Reconfigurável conforme Guan. Existem duas camadas interdependentes, a camada de controle e a camada de apresentação. Os lugares da camada de controle são diferentes em sua natureza dos lugares da camada de apresentação.

Cada lugar da camada de controle possui associado um conjunto de funções que são capazes de alterar a topologia da camada de apresentação, ou seja, reconfiguram a camada de apresentação.

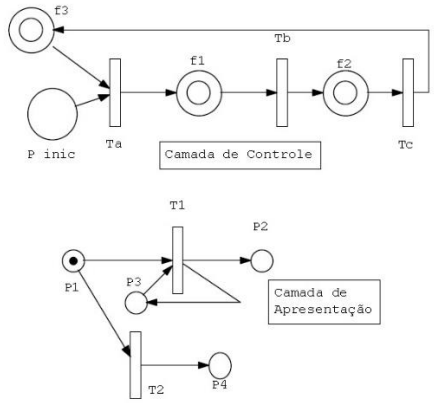


Figura 2. Rede de Petri Reconfigurável conforme Guan.

III - 2 - REDES DE PETRI ADAPTATIVAS

Uma rede de Petri adaptativa foi definida por Camolesi [2] a partir do esquema dispositivo subjacente mais camada adaptativa da seguinte forma:

$RdPA = (C_0, AR_0, \Sigma, c_0, A, NA, BA, AA)$ na configuração inicial c_0.

Estímulos de entrada movimentam a RdPA para a próxima configuração se, e somente se, uma ação adaptativa não-vazia for executada.

No k-ésimo passo temos:

$RdPA_k = (C_k, AR_k, \Sigma, c_k, A, NA, BA, AA)$, onde

$RdPA = (RdP_0, AM)$ é formado por um dispositivo inicial subjacente (RdP_0) e um mecanismo adaptativo AM;

RdP é o dispositivo Rede de Petri no passo k. RdP_0 é o dispositivo subjacente inicial e o conjunto CR_0 representa o comportamento não-adaptativo inicial;

C_k é o conjunto de todos os possíveis comportamentos de RdP no passo k e $c_k \epsilon C_k$ é o seu comportamento inicial no passo k;

ε ("cadeia vazia") denota ausência de elemento válido;

Σ é o conjunto de todos os possíveis eventos de que se compõem a cadeia de entrada;

$A \subseteq C$ é o subconjunto de configurações de aceitação de RP;

$F = C - A$ é o conjunto das configurações de rejeição de RP;

BA e AA são conjuntos de ações adaptativas, que incluem a ação vazia;

$w = w_1 w_2 \ldots w_n$ é a cadeia de entrada;

NA é um conjunto finito de todos os símbolos que podem ser gerados como saídas por RdPA em resposta à aplicação de regras adaptativas;

AR_k é o conjunto das regras adaptativas que definem o comportamento adaptativo de RdPA no passo k e é dado por uma relação $AR_k \subseteq BA \times \Sigma \times C \times RP \times AA$.

AR_0 define o comportamento inicial da RdPA e as ações adaptativas de inserção ou eliminação de lugares e transições vão transformando o conjunto de regras.

As regras $reg \in AR_k$ são da forma (<ba>,(P,T,I,O),<aa>) e operam da seguinte forma:

Um símbolo $\sigma \epsilon \Sigma$ faz *reg* executar a ação $ba \epsilon BA$. Se a ação de ba eliminar *reg* de AR_k, a execução de *reg* é abortada, caso contrário, aplica-se a regra subjacente de *reg* = (P, T, I, O). Finalmente, executa-se a ação adaptativa $aa \epsilon AA$.

III – 4 – REDE DE PETRI COLORIDA ADAPTATIVA

A rede de Petri Colorida Adaptativa utiliza o mesmo esquema de envolver o dispositivo subjacente (RdPC) com uma camada adaptativa (CA).

RdPCA = (RdPC, CA) onde

RdPC é a rede de Petri colorida convencional,

CA = (FA, RA) é a camada adaptativa.

Por sua vez, a camada adaptativa é composta pelo conjunto de funções adaptativas (FA) e pelo conjunto de regras tipo IF – THEN (RA).

FA é o conjunto de funções adaptativas e está embutida na rede de Petri Colorida Adaptativa.

RA é o conjunto de regras que deverá ser inserido na rede de Petri Colorida Adaptativa através da execução das funções adaptativas.

As funções adaptativas básicas são de inspeção, inserção ou incorporação e exclusão de uma regra.

Na Tabela II temos esquematizado um exemplo de conjunto de regras a serem inseridas em uma RdPCA.

TABELA II
Regras para serem transformadas em funções da RdPA

		Regras				
		R1	R2	R3	R4	R5
Condições	A	A = 4	A > 4	A=4	A=6	A=1
	B	B= TRUE	B= FALSE	B	B	B
	C	C >6	C ≠6	C=6	C	C
Ações/ Decisões	D	A+2	5	4	A-3	2
	E	TRUE	FALSE	TRUE	TRUE	FALSE

A regra R1 possui três condições (A<4, B=ON, C=X) e duas ações (D=A+2, E=ON).

Para transformar esta regra em uma parte de uma rede de Petri, é utilizado um modelo (Lugar,Transição,Lugar) esboçado na Figura 3.

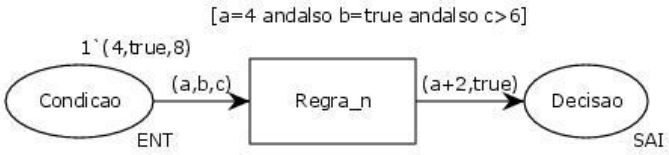


Figura 3- Modelo básico para transcrever uma regra da Tabela 3

Conforme as regras vão sendo inseridas, elas vão sendo agrupadas hierarquicamente, ou seja, a execução das regras vai depender da sequência de decisões que forem ocorrendo na rede de Petri.

Na Figura 4 está esquematizado o resultado da inserção de três regras que compõem um certo subconjunto que compõem o total de regras.

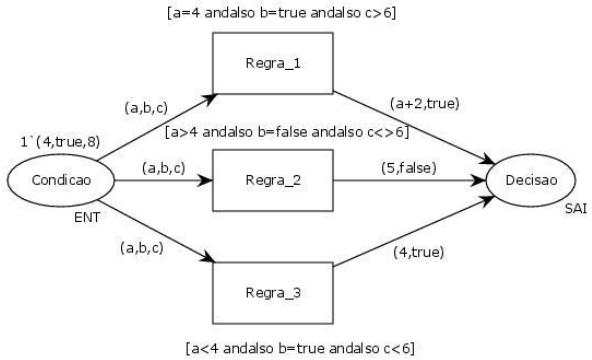


Figura 4- Subconjunto de regras em uma hierarquia

Ao inserirmos estas regras na rede de Petri, o conhecimento gerado pelo especialista passa a ser incorporado na rotina de acompanhamento do empreendimento, alertando o gestor em caso de alguma decisão implicar em não conformidade com as regras.

IV. EXEMPLO DE INSERÇÃO DE REGRA

Na Figura 5 está esquematizado o processo de fabricação de um produto a partir de 4 matérias primas, utilizando-se três máquinas.

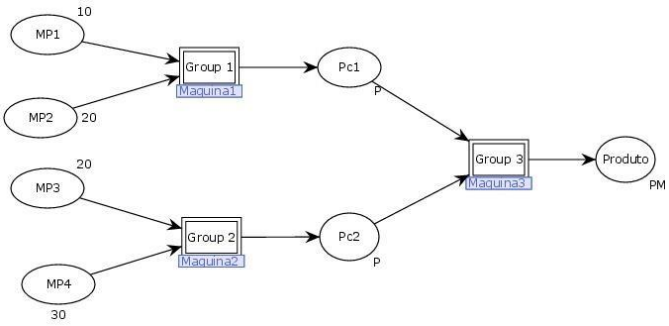


Figura 5- Rede de Petri de fabricação de um produto a partir de 4 matérias primas

A máquina 1 transforma as matérias primas MP1 e MP2 na peça 1, a máquina 2 transforma as matérias primas MP3 e MP4 na peça 2, e a máquina 3 utiliza as duas peças Pc1 e Pc2 para gerar o Produto.

Na Figura 6 está esquematizada a sub-rede correspondente ao funcionamento da máquina 1 e que é o mesmo para as máquinas 2 e 3.

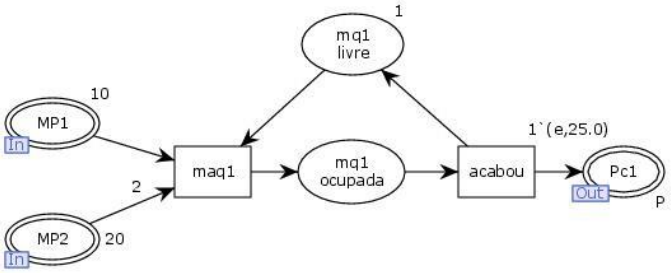


Figura 6- Rede de Petri de operação da Máquina 1

O funcionamento das máquinas especificado pela rede de Petri da Figura 5 foi definido pelo setor de fabricação.

Posteriormente, o setor de controle de qualidade detectou que o produto final não estava adequado, apresentando uma regra simples para melhorar a qualidade do produto.

Se x1 de Pc1 >= x2 de Pc2, então completar o produto, caso contrário, encaminhar Pc1 e Pc2 para reciclagem, onde x1 é um parâmetro da Peça 1 e x2 é um parâmetro da Peça 2.

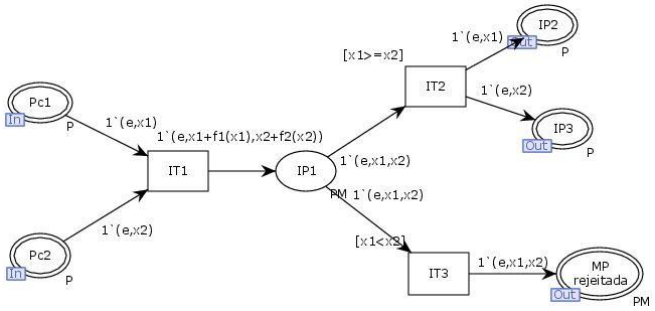


Figura 7- Sub-rede de Petri correspondente à regra de controle de qualidade inserida

A Figura 7 detalha a transformação da regra acrescentada na Rede de Petri de fabricação a partir de uma nova regra tipo IF-THEN, produzindo a nova Rede de Petri esquematizada na Figura 8.

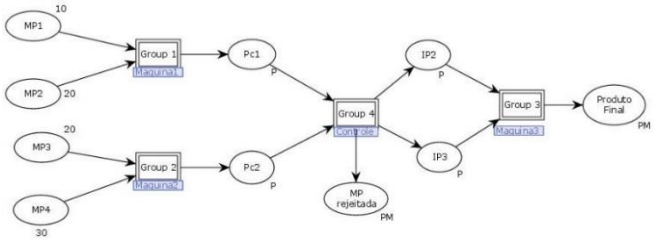


Figura 8- Rede de Petri modificada após a inserção de nova regra de controle

A transformação do formato de regras é útil quando o formato destino é mais adequado a uma aplicação que o formato original.

No exemplo fornecido, as Redes de Petri são mais utilizadas nos processos de fabricação que as Tabelas de Decisão. Além disso, diversas metodologias desenvolvidas para as Redes de Petri possibilitam detectar a existência de deadlocks e também a aplicação de esquemas que evitam seu aparecimento em redes de fabricação.

V. CONCLUSÃO

Neste trabalho mostramos como prover adaptatividade às redes de Petri coloridas, definindo as redes de Petri adaptativas coloridas (RdPCA) que são capazes de incorporar tabelas de decisão em redes de Petri.

Esta característica tem implicação prática, pois possibilita a união de dois ou mais conjuntos de saberes que normalmente convivem em paralelo, interagem, mas muitas vezes sem um sincronismo adequado.

Como exemplo, as boas práticas de gestão muitas vezes são desconsideradas no dia-a-dia de um empreendimento em virtude do desconhecimento das consequências de certas decisões, nem sempre aparentes, ou seja, a ferramenta de acompanhamento de um empreendimento feita por um especialista não contempla os cuidados e recomendações de outro especialista.

Em outras áreas onde a utilização das redes de Petri está muito disseminada, também é possível obter um ganho como, por exemplo, nos sistemas de manufatura flexível.

As regras para operação em caso de falhas podem ser incorporadas à rede padrão, possibilitando maior agilidade na operação sem a necessidade de parar o processo até que os especialistas nas falhas assumam o controle.

As recomendações destes especialistas já seriam incorporadas à rede de Petri padrão utilizada no monitoramento da operação.

REFERÊNCIAS

[1] Badouel E. ; Oliver, J. Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes wiwith Workflow Systems. [S.l.],1998.

[2] Camolesi, A. R. Proposta de um Gerador de Ambientes para a Modelagem de Aplicações usando Tecnologia Adaptativa. Tese (Doutorado) —Escola Politécnica da Universidade de São Paulo, 2007.

[3] Guan S. U. ; Yu, H. Y.; Yang, J.-S. A prioritized petri net model and its application in distributed multimedia systems. IEEE TRANSACTIONS ON COMPUTERS, VOL. 47, NO. 4, APRIL 1998, 1998.

[4] Guan S. U. ; Lim, S. S. Modeling adaptable multimedia and self-modifying protocol execution. Future Generation Computing Systems, vol.20, no 1, pp. 123-143, 2004.

[5] K. Jensen and L. M. Kristensen, Coloured petri nets: modelling and validation of concurrent systems, 1st ed., Springer Publishing Company, Incorporated., 2009.

[6] Little T. D. C.; Ghafoor, A. Synchronization and storage model for multimedia objects. IEEE J. Selected Areas Comm., pp. 413-427, Apr.1990., 1990.

[7] Llorens, M. ; Oliver, J. Structural and dynamic changes in concurrent systems: Reconfigurable petri nets. IEEE Trans. Comput., vol. 53, no.9, pp. 11471158, 2004.

[8] Neto, J. J. Adaptive rule-driven devices - general formulation and case study. Proc. 2001 Lecture Notes in Computer Science. Watson, B.W.and Wood, D. (Eds.): Implementation and Proc. 2001 Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conf., Springer-Verlag,Vol.2494, pp. 234-250., 2001.

[9] Tchemra, A. H. Tabela de decisão adaptativa na tomada de decisões multicritério. Tese (Doutorado), Escola Politécnica da USP, São Paulo, 2009.

[10] Valk, R. Self-modifying nets, a natural extension of petri nets. Lecture NotesLecture Notes in Com, vol. 62, pp. 464-476, 1978.

Haroldo Issao Guibu é graduado em Engenharia de Eletricidade e mestre em Engenharia Elétrica (1998) pela Escola Politécnica da Universidade de São Paulo. Atualmente é professor do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo das áreas de Eletrônica e Automação. Tem experiência na área de Automação com CLPs e Sistemas de Operação baseados em falhas seguras.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Técnicas Adaptativas 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 da 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.

Software para Interpretação de Medições Baseado em Tecnologia Adaptativa

S. Nicoli, J. A. Jardini and R. G. Lins

***Abstract* – The interpretation of measurements have been evolving nowadays. In this paper, a software based on the concepts of adaptivity has been proposed and developed with the goal to perform signal interpretation. The software receives signals acquired in the field through the sensors. From the processing carried out by the software, the system must be able to detect anomalies in the measured point and emits alarms or even actions. The experimental results validate the application of the adaptive method for monitoring and interpretation of signals.**

***Keywords*— Adaptive decision table, Adaptive device, Monitoring, Alarms, Measurement Interpretation.**

I. INTRODUÇÃO

Um sistema de monitoramento da condição de uma estrutura tem como principal objetivo reconhecer o surgimento de falhas incipientes, devendo ser capaz de fornecer informações úteis para funções de operação e manutenção da estrutura monitorada. O monitoramento da condição da operação envolve a medição de variáveis do processo e de operação das estruturas, condicionamento e processamento dos sinais e aplicação de técnicas especialistas para detectar e classificar padrões. A arte do monitoramento de condição consiste em extrair o máximo de informações de um conjunto mínimo de medições a fim de se obter um bom diagnóstico do estado da estrutura. A complexidade do sistema de monitoramento depende do nível de informação desejada. Sistemas mais simples podem indicar apenas se a estrutura está em condição de operação adequada ou não. Sistemas mais complexos são capazes de mostrar a evolução da condição da estrutura, de um estado adequado de operação continuada até um estado crítico. Alguns, mais avançados, detectam e identificam falhas que levaram a estrutura a uma determinada condição.

No caso da segurança estrutural de barragens, cada estrutura é considerada uma estrutura única. Para seu monitoramento, instrumentos específicos são instalados em locais adequados a fim de medir características específicas, sendo as principais grandezas a serem monitoradas são [1]:

- Deslocamentos;
- Deformações e tensões;
- Temperatura;
- Níveis Piezométricos em fundações;
- Pressões de água; e
- Vazões.

Os principais fatores que influenciam as grandezas monitoradas [2]:

- Carga direta: forças exercidas pelos contatos com a barragem de terra e pelos níveis d'água a montante e jusante;
- Subpressões na fundação: devido à percolação ou infiltração de água pela rocha de fundação, durante e após o enchimento do reservatório;
- Pressão intersticial do concreto: pressão exercida pela água que infiltra pelos interstícios do concreto, juntas de construção e falhas de construção durante a concretagem;
- Calor de hidratação do cimento: calor gerado pela hidratação do cimento, ficando armazenado no interior de um bloco, provocando tensão de compressão no concreto. E o posterior resfriamento da estrutura, provocando tensões de tração;
- Sismos: naturais causados pelo deslocamento de placas tectônicas e atividades vulcânicas. Induzidos causados pela criação de um reservatório, que altera as condições estáticas das formações geológicas, do ponto de vista mecânico (peso da massa d' água) e do ponto de vista hidráulico (a infiltração de fluidos pode causar pressões internas nas camadas rochosas profundas). É um fenômeno dinâmico, resultante das novas forças induzidas, e que passam a interferir sobre o regime das forças pré-existentes;

Desta forma, o presente trabalho tem como objetivo o estudo e aplicação de um *software* baseado em tecnologia adaptativa para Interpretação de medições, sendo que para o presente trabalho, o monitoramento de sinais oriundos de uma barragem será utilizado como estudo de caso. Este estudo se propõe a desenvolver um *software*, inspirado nas tabelas de decisão convencionais e nos dispositivos dirigidos por regras adaptativas para a detecção de anomalias em uma série de sinais elétricos. Embora o estudo de caso seja baseado na aplicação de um *software* adaptativo para interpretação de sinais para monitoramento de barragens, o algoritmo proposto pode ser utilizado para interpretação de sinais de outros sistemas, tais como máquinas, plantas industrias, dentre outras que utilizam instrumentação para o monitoramento de segurança, manutenção e outras tarefas que demandem a interpretação de sinais instalados em campo.

O presente trabalho começa pela descrição dos trabalhos relacionados na Seção II. A Seção III discute sistema e o algoritmo proposto. O experimento e os resultados são mostrados na seção IV. Finalmente, a Seção V apresenta a conclusão.

S. Nicoli, Universidade de São Paulo (USP), São Paulo/SP, Brasil, sidnei.nicoli@gmail.com

J. A. Jardini, Universidade de São Paulo (USP), São Paulo/SP, Brasil, jose.jardini@gmail.com

R. G. Lins, Universidade Federal do ABC (UFABC), Santo André/SP, Brasil, romulo.lins@ufabc.edu.br

II. TRABALHOS RELACIONADOS

Como apresentado em [3], em termos gerais adaptabilidade define que uma entidade pode ser adaptada por um agente externo que tem o poder de decisão sobre as mudanças a serem feitas, as quais são aplicadas pela entidade a ser adaptada, ou, num caso particular, ela mesma pode governar o processo de decisão e aplicação das mudanças em sua própria estrutura. Com relação à adaptatividade, de maneira geral pode ser definida como a capacidade que possui alguma coisa para efetuar uma mudança a fim de se adequar a uma nova situação. Considerando o *software* proposto neste trabalho, de maneira geral esses são os termos mais adequados aplicados ao desenvolvimento do presente trabalho.

Aplicar o conceito de adaptatividade para resolução eficiente de certos problemas é a referência da Tecnologia Adaptativa, pois ela possibilita a adaptação, de forma autônoma, de um sistema ou dispositivo adaptativo ao detectar situações que exijam mudanças nas reações em resposta aos estímulos de entrada [5], com o uso de ações adaptativas, que permitem modificar o conjunto de regras, removendo regras existentes e incluindo novas regras dinamicamente, sem a interferência de qualquer agente externo [4].

É interessante ressaltar que duas instâncias idênticas de um mesmo sistema adaptativo, por exemplo, podem atingir estados finais diferentes de acordo com a diversidade dos estímulos a que forem submetidas em suas operações [4].

Essas ações adaptativas não acrescentam poder adicional à computação, mas melhoram o poder de expressão de atividades computacionais intrincadas.

Algumas áreas de aplicações da Tecnologia Adaptativa são [6]:

- Robótica – utilizados autômatos adaptativos em automação, planejamento de rotas para trânsito urbano, etc;
- Segurança – utilizados autômatos adaptativos em criptografia e mecanismos de controle de acesso;
- Arte – exploração da composição musical automática, busca de padrões musicais, etc;
- Otimização – otimização de parâmetros, em função da particularidade de cada caso, otimização de rotas em tempo real, etc;
- Tomada de decisão – é um alvo interessante para o emprego da adaptatividade, pois possui um grande leque de alternativas.

Desta forma, o desenvolvimento de aplicações de *software* baseado em tecnologia adaptativa está ganhando grande destaque nas referidas áreas. No campo de automação robótica, o uso da adaptatividade permite que robôs consigam navegar em um ambiente de forma autônoma através de autômatos concisos e de baixa complexidade computacional [7]. O estudo de processamento de linguagens naturais também possibilita a construção de robôs capazes de gerenciar diálogos homem-máquina, demonstrando um comportamento supostamente criativo [8].

Na área de mineração de dados, muitas pesquisas também estão sendo desenvolvidas e a adaptatividade é usada como forma de simplificar o problema de tratar um volume elevado de informações, como exemplo, na identificação de indivíduos através de suas características [9]. Também se torna possível identificar padrões através de operações de convolução, onde a escolha do núcleo da operação é feita baseada em soluções anteriores [10]. Foi também idealizado um sistema capaz de recomendar novos locais para um indivíduo visitar, baseado no contexto em que encontra (e.g., locais diferentes podem ser recomendados, dependendo se o indivíduo está trabalhando, ou então, de férias) [11]. Também é possível extrair um comportamento preditivo, baseado em informações passadas, conforme mostra o sistema adaptativo de previsão de tempo descrito em [12].

Todos os trabalhos apresentados nesta seção indicam que os conceitos de adaptatividade podem ser aplicados para o desenvolvimento do *software* proposto.

III. ALGORITMO PROPOSTO

Alarmes constituem um elemento fundamental nos sistemas de monitoramento e tradicionalmente eles têm como finalidade a apresentação de anormalidades através da representação gráfica e/ou lista de alarmes para um operador humano. Assim, para atender o objetivo proposto neste trabalho, a Fig. 1 mostra o fluxo de dados do algoritmo proposto.

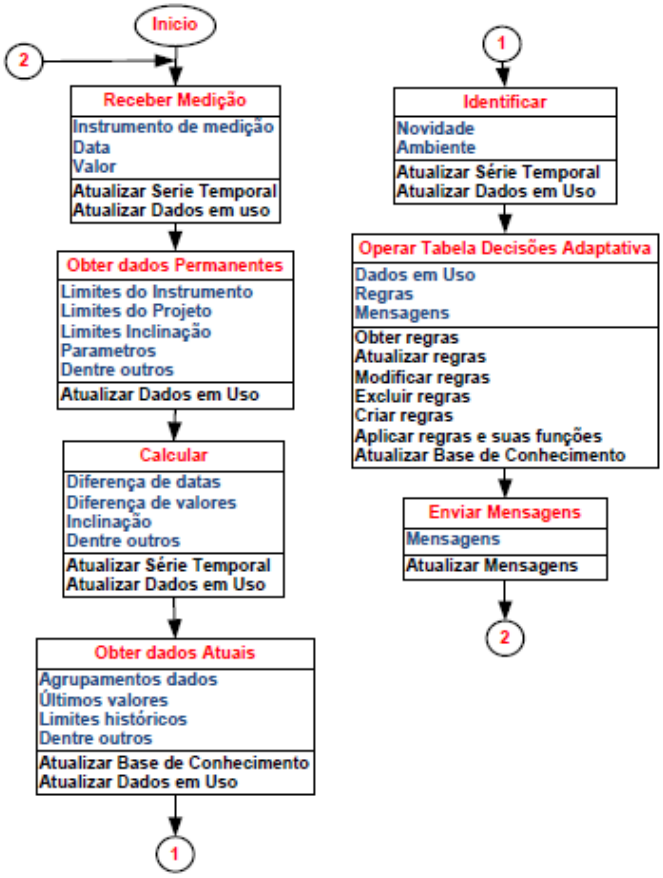


Figura 1. Fluxo de dados do Algoritmo proposto.

A Fig. 1 mostra os principais elementos e operações realizadas pelo sistema proposto. Em cada operação indicada é mostrada a principal atividade do bloco, os dados e as funções envolvidas em cada um deles.

Os sinais são aquisitados pelos sensores em campo e os valores de medição são registrados junto com a data de aquisição. Após a recepção da medição e sua inclusão na base de conhecimentos, os dados permanentes relacionados a um

instrumento de medição são obtidos a partir da base de conhecimento. Em seguida, os cálculos são realizados com o valor atual e dados históricos de forma a obter novos dados que são atualizados na base de conhecimento. Finalmente são identificados o ambiente e novidade se necessário. Finalizando o processamento das regras existentes na tabela de decisão adaptativa.

Para o processamento dos dados, a metodologia proposta estabelece a existência de períodos de tamanho variado correspondendo a ambientes característicos (meio ambiente, contexto, etc.). Também foi considerada a existência de novidades, que correspondem a valores que aparecem pela primeira vez em um ambiente.

O método considera que um ambiente está em evolução quando aparecem constantemente novidades durante o tempo, passando para um ambiente estável quando não aparecerem mais novidades a partir de um momento até o instante em que uma novidade é detectada, passando assim o ambiente a ser considerado novo. Assim um ambiente possui os seguintes estados: em evolução, estável e novo.

A análise de cada valor dentro do método inclui os estados do ambiente e as novidades, além de considerar as diversas composições possíveis de um dado atual com os dados históricos, dentre as quais estão:

- Diferença entre as datas de medição atual e anterior;
- Diferença entre o dado atual com o anterior;
- Ângulo obtido entre o dado atual e o anterior;
- Agrupamentos do dado atual com dados anteriores, podendo envolver grupos composto por um elemento até cinco elementos (ou mais);
- O menor valor da série temporal (de toda ela, ou somente do ambiente);
- O maior valor da série temporal (de toda ela, ou somente do ambiente);
- Os limites informados pelo fabricante do instrumento de medição e/ou sensor, desde que informados;
- Os limites informados pelo projeto, desde que informados;
- A média dos dados de toda a série e/ou do ambiente atual.

Cabe destacar que as considerações para a utilização de novidade, ambiente, agrupamento, dentre outras, foram utilizadas após a realização de provas de conceitos.

Além do apresentado até agora no desenvolvimento do método de detecção de anomalias também foram considerados os seguintes requisitos para o sistema computacional:

- Operar em ambiente *on-line*;
- Não ser supervisionado (portanto sem treinamento);
- Ser parametrizável (incluindo precisão, tolerância, quantidade de valores sem aparecimento de novidade, dentre outros);
- Capaz de se adaptar as condições externas desconhecidas;
- Não utilização de curvas com previsões estatísticas;
- Analisar a valor imediatamente após o seu recebimento e incorporação na série temporal;
- Não permitir a inclusão de dados externos durante a sua operação;
- Enviar diversos tipos de mensagens inclusive de alertas;
- Dentre outros.

Finalmente, a busca de um valor candidato a anomalia ocorre após a recepção *on-line* da leitura, sua transformação e processamento. O algoritmo 1 mostra a sequência de tarefas a serem realizadas.

Algoritmo 1. Algoritmo de Monitoramento implementado.

Algoritmo 1 Entrada: valor de medição obtida a partir de um sensor; Saída: Mensagem aos interessados, em geral indicando medição candidata à anomalia.
1. Utilizar a última medição realizada (uma por vez); 2. Registrar na série temporal instrumento de medição, o valor da medição e a data; 3. Obter da base de conhecimento os Parâmetros relacionados ao instrumento de medição; 4. Obter limites do instrumento de medição (inferior e superior, se existirem); 5. Obter limites de projeto (inferior e superior, se existirem); 6. Obter limites operacionais adotados de inclinação (inferior e superior, se existirem); 7. Obter a diferença das datas entre a medição atual e a anterior; 8. Obter a diferença entre os valores atual e anterior; 9. Obter a inclinação entre valor atual com o valor anterior; 10. Registrar na série temporal (atual com referência ao anterior): diferença entre datas, diferença entre valores, e inclinação entre os valores; 11. Obter o tipo de ambiente anterior; 12. Atualizar agrupamento de dados com o valor atual (para 1, 2 3, 4 e 5, atualizar quantidade somando um, quando existir o grupo e criar quando não inexistente), atualizar grau de confiança e suporte dos agrupamentos (relacionados a regras de associação) e identificar novidade (que corresponde a um agrupamento que aparece pela primeira vez); 13. Atualizar os últimos valores, modificando quando necessário os limites operacionais (inferior e superior); 14. Operar a Tabela de Decisões Adaptativa; 15. Enviar mensagens e alertas aos interessados, caso necessário; 16. Retornar para etapa 1.

Cabe ressaltar as considerações baseadas em [13] com relação à utilização da Tabela de Decisões Adaptativa (T.D.A) considerada um dispositivo adaptativo, que no caso deste *software* tem como base uma Tabela de Decisões, (T.D) considerada um dispositivo subjacente.

As operações do sistema seguem um módulo baseado na T.D.A. No esquema mostrado na Fig.2, a T.D é representada basicamente pelas regras compostas por condições e ações, complementada pela camada adaptativa que possibilita as ações adaptativas de consulta, exclusão e criação de novas regras. São vários os elementos adaptativos como funções adaptativas, parâmetros, variáveis, geradores, e chamadas de funções que

podem ser executadas antes e chamadas que podem ser executadas depois de uma regra. A maioria desses elementos é opcional, de tal forma que se nenhum for utilizado é executada uma T.D caso contrário é executada uma T.D.A.

		Cabeçalho das colunas da declaração das funções adaptativas que podem conter colunas H, +, -, ?	Cabeçalhos das colunas com Regras que podem conter S, R, E (o número de colunas é variável)
Conjunto de linhas de condições	Nomes / descrições das Condições		Celulas com os conteudos das condições
Conjunto de linhas de ações	Nomes / descrições das ações	Declaração das funções adaptativas (protótipo)	Celulas indicando a aplicação da ação
Conjunto de linhas das funções adaptativas	Nome das funções adaptativas	As colunas com "?" indica consulta a com "-" exclusão e a com "+" inclusão de regras	Células indicando a execução da função adaptativa
Conjunto de linhas com os parâmetros das funções adaptativas	Nome dos Parâmetros das funções	Na coluna com H em suas células são indicadas se uma função é executada antes (B) ou depois (A)	Células indicando conteudos dos parâmetros
Conjunto de linhas com as variáveis das funções adaptativas	Nome das variáveis das variáveis	e com (P) os parâmetros, e com (V) as variáveis e com (G) os geradores.	Células indicando conteudo das variáveis
Conjunto de linhas com os geradores das funções daptativas	Nome dos geradores		Células indicando conteudo dos geradores

Figura 2. Esquema Básico de uma T.D.A (Adaptado de [14]).

O item 14 do algoritmo 1 será melhor explicado e detalhado conforme o algoritmo 2, sendo que corresponde a um conjunto de operações.

Algoritmo 2. Algoritmo para processamento da uma T.D.A

```
Procedimentos envolvendo T.D.A
Entrada: Regras (do tipo Se condições então Ações), e conteúdos necessários para utilização das regras;
Saída: Mensagens
Pode ser executada mais de uma regra aumentando o tamanho da Mensagem.
Busca das Regras Pertinentes
Executar enquanto existir regras Pertinentes
        Verificar se a regra possui as variáveis disponíveis para a operação
                Se sim
                        Executar o lado das condições
                        Se verdadeiro
                                Executar as ações e as funções relacionadas às ações da regra (que podem ser do tipo adaptativo ou não)
                                Compor a Mensagem
                        Se Falso
                                Não executar ações
Continuar a executar enquanto existir regras Pertinentes
Atualizar a base de conhecimento
Ir para próxima operação (no caso é enviar mensagem).
```

Nota-se no Algoritmo 2, que ao executar as ações da regra apesar de não ser indicado é verificado se a regra possui relacionada a ela a uma função a ser executada antes, além disso após a execução da regra também é importante verificar se existe a execução de uma função após a regra. Também pode ocorrer a exclusão de uma regra e a criação de uma regra seguindo um protótipo determinado.

Outras questões sobre o *software* desenvolvido é que ele permite a interpretação de mais de uma regra apesar da perda de desempenho no processamento, sendo que isso pode ser desabilitado, além disso na implantação do sistema é necessário informar as regras iniciais.

IV. EXPERIMENTO E RESULTADOS

Conforme citado, para o experimento serão utilizadas medições oriundas de instrumento de medição civil do tipo piezômetro.

A partir das leituras obtidas no ponto de medição (de agosto a outubro de 2014) pelo piezômetro instalado com energia fornecida por painel solar fotovoltaico. Uma série temporal contendo 2289 amostras foi armazenada e traçada. O período de aquisição de sinal é 15 minutos de 29/08/2014 16h07min à 01/10/2014 10h04min. A Fig.3 mostra a série temporal que representa a pressão medida pelo piezômetro instalado, onde o eixo Y representa os valores medidos (kgf/cm^2) e o eixo x representa o tempo. Note que na parte inferior do traçado do gráfico existem patamares que iniciam com leitura de fundo relacionada à diminuição de energia para o instrumento.

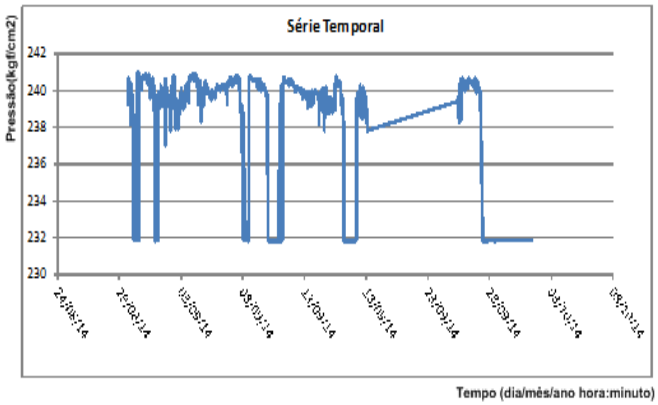


Figura 3. Série de sinais aquisitados pelo piezômetro instalado em campo.

Considerando a Fig.3, nota-se que existe uma sequência das medições da série (armazenada nos registros), a data/hora e o valor inteiro (utilizado assim para facilitar o processamento), o valor real (em kgf./cm2) corresponde ao valor inteiro dividido por 1000000, assim um valor inteiro da medição é 231852345 corresponde a um valor real de 231,852345 que por sua vez corresponde a uma leitura com um valor negativo (sendo considerado um valor de fundo) antes de ser transformada em uma medição. Com a finalidade de facilitar o entendimento do processamento dois pontos serão utilizados para processamento: 945 e 946, sendo que as informações a serem processadas no software são mostradas na tabela 1.

TABELA I
DADOS LIDOS PELO PIEZÔMETRO PARA PROCESSAMENTO

Sequência	944	945	946
Dia	08/09/2014 12:13:00	08/09/2014 12:28:00	08/09/2014 12:43:00
Valor	240085295	240485988	231852345
Períodos Após Anterior	1	1	1
Diferença atual anterior	-344416	400693	-8633643
Inclinação com anterior	-1,57	1,57	-1,57
Inclinação com ante grau	-89,99	89,99	-89,99
Indicação	N5	N5	N5
Ambiente	3	3	3
Estado Ambiente	EVOLUÇÃO	EVOLUÇÃO	EVOLUÇÃO
INDICAÇÃO GRUPO 1		N1	

A partir do processamento dos dados com o *software* proposto, baseado nos conceitos de adaptividade, os resultados são apresentados na tabela 2.

TABELA II
RESULTADO DO PROCESSAMENTO

Sequência 944	Sequência 945	Sequência 946
Dados reunidos para a inferência	Dados reunidos para a inferência	Dados reunidos para a inferência
Valor = 240085295	Valor = 240485988	Valor = 231852345
Período entre os registros = 1	Período entre os registros = 1	Período entre os registros = 1
Ambiente = 3	Ambiente = 3	Ambiente = 3
Novidade = não	Novidade = sim (surgimento de um novo grupo)	Novidade = não
Tipo de ambiente = EVOLUÇÃO	Tipo de ambiente = EVOLUÇÃO	Tipo de ambiente = EVOLUÇÃO
Limite operacional anterior Menor = 231838088 e Maior = 241057012 (da sequência 943)	Limite operacional anterior Menor = 231838088 e Maior = 241057012 (da sequência 944)	Limite operacional anterior Menor = 231838088 e Maior = 241057012 (da sequência 945)
Limite Operacional Angular adotado = 85,94º ou 1,5 rad.	Limite Operacional Angular adotado = 85,94º ou 1,5 rad.	Limite Operacional Angular adotado = 85,94º ou 1,5 rad.
Inclinação = -1,57 rd	Inclinação = 1,57rd	-1,57rd
Inclinação graus = -89,99º	Inclinação graus = 89,99º	Inclinação graus = -89,99º
Suporte = não foi necessário, só se incluir previsão imediata.	Suporte = não foi necessário, só se incluir previsão imediata.	Suporte = não foi necessário, só se incluir previsão imediata.
A Inferência Resultante	A Inferência Resultante	A Inferência Resultante
Aplicação da regra R3 limite angular maior	Aplicação da regra R3 limite angular maior. Aplicação da regra R10 com novidade e limite angular maior.	Aplicação da regra R3 limite angular maior

Como resultado do processamento, a mensagem a ser emitida na medição 945 da série temporal é composta pelas seguintes informações:

- Candidato a anomalia;
- Identificação do Instrumento de Medição Piezômetro PZ1762;
- Sequência na série temporal original: 945;
- Data/hora da ocorrência: 08/09/2014 12:28:00;
- Valor medição: 240485988;
- Inclinação crescente crítica;
- Valor novo no ambiente / série temporal;
- Texto: Inclinação gráfica abrupta com novidade.

Assim está constatada a identificação de regras e sua aplicação no envio de mensagens indicando que um ponto é candidato a anomalia.

V. CONCLUSÃO

Este trabalho apresentou os resultados do desenvolvimento de um *software* baseado em técnicas adaptativas para interpretação de medições. O sistema computacional foi desenvolvido baseado na necessidade de que medições podem ser interpretadas por um método não supervisionado trabalhando com um fluxo continuo. Para isso, foi desenvolvido um sistema computacional com a finalidade de identificar anomalias importantes e enviar alarmes parametrizados aos interessados.

O *software* processa os dados e devolve como resultado análises e/ou inferências sob uma nova perspectiva que classifica o ambiente monitorado em: novo, evolução e estabilidade. A partir dos experimentos realizados com um instrumento real, o sistema desempenhou sua função de interpretação e emissão de alarmes, melhorando assim a qualidade da informação disponível para tomada de decisão.

AGRADECIMENTOS

Os autores agradecem a EDP Bandeirante e a CESP energia pelo suporte financeiro e pelo suporte técnico de seus profissionais, agradecemos a ANEEL (Agência Nacional de Energia Elétrica) pela aprovação deste projeto de pesquisa e ao corpo técnico da EPUSP, pelo suporte durante o desenvolvimento do projeto. Também agradecemos de forma especial à Diogo. L. T. Alves e Alex. L. de Oliveira pelas colaborações técnicas.

REFERÊNCIAS

[1] MACHADO, William Gladstone de Freitas. Monitoramento de barragens de contenção de rejeitos da mineração. 2007. Tese de Doutorado. Universidade de São Paulo.

[2] LYRA, J. S. et al. Consolidação de pastas cimentícias contendo policarboxilatos um estudo calorimétrico e reológico (Consolidation of policarboxilates containing cement pastes: a calorimetric and rheological study). Cerâmica, v. 58, p. 137-143, 2012.

[3] Rosalia Edith Caya Carhuanina, João José Neto. Personalização, "customização", adaptabilidade e adaptatividade. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, pp. 52-59. 28 e 29 de Janeiro, 2016.

[4] NETO, J. José. Um levantamento da evolução da adaptatividade e da tecnologia adaptativa. Revista IEEE América Latina, v. 5, n. 7, p. 1548-0992, 2007.

[5] TCHEMRA, Angela Hum. Application of adaptive technology in decision-making systems. IEEE Latin America Transactions, v. 5, n. 7, p. 552-556, 2007.

[6] PISTORI, Hemerson. Tecnologia adaptativa em engenharia de computação: Estado da arte e aplicações. Universidade de São Paulo (USP), São Paulo, 2003.

[7] HIRAKAWA, Andre Riyuiti; SARAIVA, Antonio Mauro; CUGNASCA, Carlos Eduardo. Adaptive Automata Applied on Automation and Robotics (A4R). IEEE Latin America Transactions, v. 5, n. 7, p. 539-543, 2007.

[8] OKADA, Rodrigo Suzuki; JOSE, Joao. Cascade-Structured Classifier Based on Adaptive Devices. IEEE Latin America Transactions, v. 12, n. 7, p. 1307-1324, 2014.

[9] CEREDA, P. R. M.; NETO, J. José. Mineração adaptativa de dados: Aplicação à identificação de indivíduos. In: Memórias do Sexto Workshop de Tecnologia Adaptativa, Sao Paulo. 2012.

[10] OKADA, Rodrigo Suzuki. Tecnologia adaptativa aplicada a sistemas híbridos de apoio à decisão. Tese de Doutorado. Universidade de São Paulo.

[11] CRIVELARO, Celso Vital; BARTH, Fabrício J.; ROCHA, Ricardo Luis Azevedo. Proposta de Modelo Adaptativo para Geração de Contextos na Recomendação de Locais. Revista de Sistemas e Computação-RSC, v. 2, n. 2, 2012.

[12] ROCHA, Felipe Barbalho et al. Sistema Automatizado utilizando Tecnologias de Baixo Custo para Controle e Monitoramento de Irrigações no Vale do Açu. In: IX Congresso de Iniciação Científica do IFRN. 2013.

[13] NETO, João José. Adaptive rule-driven devices-general formulation and case study. In: International Conference on Implementation and Application of Automata. Springer Berlin Heidelberg, 2001. p. 234-250.

[14] TCHEMRA, Angela Hum. Tabela de decisão adaptativa na tomada de decisão multicritério. 2009. Tese de Doutorado. Universidade de São Paulo.

Sidnei Nicoli é Doutorando em Engenharia Elétrica de Potência e Mestre em Engenharia Elétrica de Sistemas Digitais pela Escola Politécnica da USP (2004) e Graduado em Engenharia Mecânica pela Fundação Armando Álvares Penteado (1981). Além de atuar como consultor na área de sistemas e engenharia.

José Antonio Jardini é nascido em São Paulo/Brasil. Jardini recebeu seus títilos de mestre, doutor, professor associado e professor titular da Escola Politécnica da Universidade de São Paulo em 1963, 1971, 1973 e 1991, respectivamente. Ele também trabalhou na Themag Eng. Ltda, empres lider em consultoria, onde realizou muitos estudos de sistemas de energia e participou de grandes projetos de sistemas de energia, como a usina hidroelétrica de Itaipu. Também representou o Brasil no International Council on Large Eletric Systems (CIGRÈ), além de ter recebido o prêmio de notável professor do IAS/IEEE.

Romulo Gonçalves Lins é Professor Adjunto do Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas da Universidade Federal do ABC desde 2015. Romulo recebeu o título de doutor em engenharia mecânica pela UNICAMP em 2013, além de ter estágio pós-doutorado no Royal Military College of Canada em 2015. Suas áreas de pesquisa são processamento de sinais e imagens, automação industrial e robótica.

Parte III

Transcrição da mesa redonda

A transcrição da mesa redonda está em fase de revisão. Em breve, esta seção será substituída pelo texto adequado.