Memórias do WTA 2019

XIII 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
2019

Ficha catalográfica

Workshop de Tecnologia Adaptativa (13: 2019: São Paulo)
Memórias do WTA 2019. – São Paulo; EPUSP, 2019. 19p.

ISBN 978-85-5338-007-7

ISBN 9788553380077

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 terceira edição do Workshop de Tecnologia Adaptativa realizou-se em São Paulo, Brasil, nos dias 31 de Janeiro e 1 de Fevereiro de 2019, 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

- Ricardo Luís de Azevedo da Rocha (São Paulo, SP, Brasil)
- Ricardo Nakamura (São Paulo, SP, Brasil)
- Rosalia Edith Caya Carhuanina (São Paulo, SP, Brasil)
- Sérgio Donizetti Zorzo (São Carlos, SP, Brasil)
- Sidney da Silva 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
- 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 2019

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 2019, o Décimo Terceiro Workshop de Tecnologia Adaptativa, realizado em São Paulo nos dias 31 de Janeiro e 1 de Fevereiro de 2019. A exemplo da edição de 2018, 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 apresentados no WTA 2019 – Décimo Terceiro Workshop de Tecnologia Adaptativa, realizado na Escola Politécnica da Universidade de São Paulo nos dias 31 de Janeiro e 1 de Fevereiro de 2019.

Esta edição contou com trabalhos e tutoriais relacionados à área de Tecnologia Adaptativa e aplicações nos mais diversos segmentos. 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 2019. A exemplo do que foi feito no ano anterior, todo o material referente aos trabalhos apresentados no evento estará acessível no portal do WTA 2019, 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 2020.

Agradecimentos

Às instituições que apoiaram o WTA 2019, 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
- Newton Kiyotaka Miura
- Nilton Araújo do Carmo
- Renata Luiza Stange
- Rodrigo Kavakama
- Rosalia Caya
- Suellen Alves

Divulgação

- Amanda Panteri
- Amanda Rabelo

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 segunda 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.

São Paulo, 1 de Fevereiro de 2019

João José Neto
Coordenador geral

Sumário

Lista de autores

Parte I

Submissões

Uso de macros na aprendizagem de autômatos adaptativos: ensaio de integração da técnica OC2-RD2 com a Taxonomia de Bloom Revisada

P. R. M. Cereda, I. S. Vega e F. S. Marcondes

***Abstract*—Este artigo promove a discussão da utilização do conceito de macros e sistemas de reescrita de termos na aprendizagem de teorias da computação em cursos de nível superior, viabilizada pela elaboração de ensaios narrativos integrando a técnica OC2-RD2 com a Taxonomia de Bloom Revisada.**

***Palavras-chave:*—autômato adaptativo, teoria da computação, macros, sistemas de reescrita, ensaio narrativo, OC2-RD2**

I. Introdução

A apresentação de teorias da computação em cursos de nível superior pode se beneficiar de técnicas instrucionais baseadas em narrativas e objetivos de aprendizagem. Este artigo descreve a elaboração de um ensaio narrativo OC2-RD2 [1], [2], integrado com a Taxonomia de Bloom [3] para apresentar os elementos centrais da Teoria dos Autômatos Adaptativos de acordo com Neto [4]. O ensaio explora os processos cognitivos de lembrança, entendimento e aplicação, proporcionando condições para a montagem de um ambiente no qual o aprendiz poderá desenvolver os níveis mais baixos de habilidades de pensamento.

Durante a elaboração de ambientes de aprendizagem, cuidados devem ser tomados em relação à aprendizagem por repetição [5]. Pode-se induzir o aprendiz a guiar-se por um processo de memorização no qual os conceitos são estuados por meio de alguma técnica de repetição, tipicamente a realização de uma lista de exercícios.

O artigo segue apresentando-se os objetivos de aprendizagem a serem alcançados pela narração da história da fábula OC2-RD2. Em seguida, caracterizam-se os tipos de cenas da fábula, de acordo com a técnica OC2-RD2.

A. Objetivos de aprendizagem

Durante o planejamento de material instrucional, recomenda-se a especificação de objetivos de aprendizagem como ponto de partida. De acordo com a Taxonomia de Bloom Revisada, duas importantes dimensões cognitivas devem ser contempladas: a dimensão do conhecimento e a dos processos cognitivos [3]. No ensaio-alvo deste artigo, apenas os processos cognitivos de nível mais baixo serão exercitados, uma vez que a intenção é de se criar um ambiente de aprendizagem no qual alguns conceitos fundamentais a respeito de autômatos adaptativos sejam introduzidos.

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: paulo.cereda@alumni.usp.br, italo@pucsp.br e yehaain@gmail.com.

A estrutura da dimensão de conhecimento inclui a categoria fatual (ou efetiva) e refere-se aos elementos básicos que devem ser conhecidos em uma disciplina, como a terminologia, por exemplo. Outra categoria, a conceitual, apresenta os interrelacionamentos entre os elementos básicos no contexto de uma estrutura mais abrangente, mostrando como devem funcionar em conjunto. O conhecimento da categoria procedural revela-se na maneira de se fazer coisas, principalmente os métodos, algoritmos e técnicas. Finalmente, a categoria metacognitiva, refere-se à cognição em geral e, em particular, ao processo de cognição do próprio aprendiz. No ensaio de elaboração da fábula deste artigo, apenas as duas primeiras categorias de conhecimento serão exploradas.

Quanto à dimensão dos processos cognitivos, a Taxonomia propõe seis categorias, hierarquicamente organizadas (Figura 1). Os processos de nível mais baixo remetem a pensamentos mais simples, iniciando pelo de lembrança. A ativação desta categoria de processo deve desencadear o reconhecimento e a recuperação de conhecimento no aprendiz. Em seguida, encontra-se o processo de entendimento que determina o significado das mensagens instrucionais. No caso de modelos de computação, tipicamente estas mensagens assumem a forma escrita e gráfica de comunicação. O processo cognitivo de aplicação completa o grupo de processos simples. A ativação desta categoria de processo leva o aprendiz a utilizar um procedimento em alguma situação. As três categorias de processos maiores, mais complexos, não serão exercitados pela fábula elaborada neste artigo.

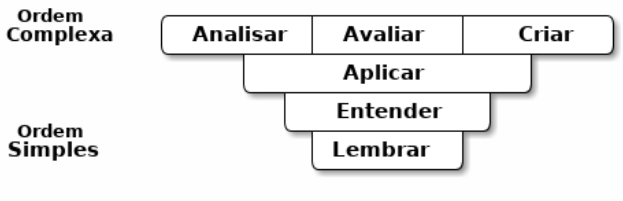


Figura 1. Níveis e processos cognitivos de Bloom.

A combinação destas duas dimensões conduz a um quadro no qual se declaram os objetivos de aprendizagem a serem alcançados. No presente caso, os objetivos são apresentados na Tabela I.

Visualizam-se estes objetivos, organizados nas dimensões de Bloom, conforme ilustrado na Figura 2. Cabe ao responsável pela construção do ambiente de aprendizagem, alocar

Tabela I
OBJETIVOS DE APRENDIZAGEM A SEREM ALCANÇADOS.

CA1	identificar os elementos da estrutura de um autômato sem transições adaptativas
CA2	descrever o funcionamento de um autômato sem transições adaptativas
CA3	listar os elementos que definem uma macro
CA4	nomear os tipos de ações adaptativas
CA5	combinar funções adaptativas com efeitos adaptativos
CA6	descrever a implementação de um autômato adaptativo por meio de macros
CA7	computar cadeias de símbolos reconhecidas por um autômato adaptativo

os objetivos às células desta matriz – preocupação que será refinada em um outro artigo.

B. Fábulas OC2-RD2

Vega [1] introduz uma estrutura narrativa para ser utilizada em ambientes de aprendizagem. O projeto das interações com o aprendiz é central à técnica. Pretende-se que a narração de uma história no ambiente desencadeie interações que estimulem o aprendiz a participar e a desenvolver o seu conhecimento por gradual ativação dos seus processos cognitivos.

A estrutura da fábula que origina as cenas de uma história segue um ritmo de sequências objetivas do tipo objetivo-contratempo-catástrofe, intercaladas com sequências subjetivas do tipo reação-dilema-decisão. Concentram-se nas passagens subjetivas as referidas interações.

As cenas-objetivo estabelecem um motivo para as ações dos personagens. No presente artigo, apenas o personagem Fubã será utilizado [6]. Ele representa o nível iniciante no trabalho realizado por Dreyfus e Dreyfus [7], comportando-se como uma pessoa interessada, curiosa, de acordo com a valoração de Kort et. al [8]. Considera-se este personagem apropriado para histórias que se propõem a apresentar conceitos introdutórios.

Cenas de contratempo apresentam um dilema decorrente de questões fechadas no sentido de Ragonis [9], mantendo-se o nível de lembrança, segundo a Taxonomia de Bloom. As cenas de catástrofe, por outro lado, alteram o ritmo narrativo estabelecido por uma sequência de contratempos. A quebra ocorre quando são acionados processos cognitivos de ordem maior. No ensaio, optou-se por acionar processos cognitivos de lembrança nas cenas de contratempo. Por conseguinte, as cenas de catástrofe apoiar-se-ão em processos cognitivos de entendimento. Tais cenas conduzirão a um dilema declarado na forma de uma pergunta aberta na linha de Ragonis [9].

Iniciando-se o projeto da fábula OC2-RD2, estabelece-se o objetivo a ser alcançado pelo personagem Fubã, quando este se propõe a estudar os fundamentos da Teoria dos Autômatos Adaptativos:

Cena 1 (objetivo) Em uma das páginas do seu livro-texto [10], Fubã encontra o problema de implementar um autômato adaptativo capaz de reconhecer a linguagem $a^n b^n c^n$, com $n > 0$. Ele conseguirá, caso utilize a noção de "macro"? ■

O projeto das cenas seguintes deve ser tal que, gradativamente, este objetivo seja alcançado. Cabe ao autor da fábula, estabelecer trechos subjetivos de complexidade cognitiva crescente, visando colaborar para que o aprendiz, assumindo o papel do personagem Fubã, consiga implementar o reconhecedor daquela linguagem. Na próxima sequência de cenas, a estrutura formal de um autômato torna-se o ponto focal.

II. APRESENTAÇÃO DE CONCEITOS FUNDAMENTAIS

Para alcançar o objetivo de projetar um reconhecedor da linguagem $a^n b^n c^n$ com o uso de autômatos adaptativos, criam-se cenas de contratempo de nível Bloom mais baixo: lembrar. Isto porquê se trata de uma apresentação introdutória ao assunto. Tais cenas deverão conduzir o personagem em uma trilha de apresentação de conhecimento fatual e conceitual a respeito da estrutura de um autômato. Com a intenção de explorar o conhecimento procedural, também será introduzida a noção do mecanismo de macro, considerada conveniente para as cenas de implementação neste ensaio.

A. Estrutura de um autômato adaptativo

A apresentação formal da estrutura algébrica de um autômato adaptativo encontra-se no trabalho de Neto [4]. Pressupondo-se a existência de algumas cenas de contratempo para introduzir a notação e os elementos estruturais, passa-se a uma cena que convida o personagem Fubã a recuperar este conhecimento (as siglas das cenas correspondem ao quadro de objetivos de aprendizagem).

Cena CA1 (autômato adaptativo, contratempo) Fubã encontra o diagrama de um autômato apresentado na Figura 3, especificado sem a presença de transições adaptativas. ■

> *Com base na representação da Figura 3, Fubã deve relacionar cada elemento da estrutura formal de um autômato com uma parte do exemplo, conforme ilustra a Figura 4.*

O aprendiz, atuando no papel do personagem Fubã, deverá ser capaz de produzir o seguinte resultado: (1, C), (2, D), (3, A) e (4, B) (Figura 5). Por quê? Ele identifica os elementos do diagrama, representando-os na forma simbólica de acordo com Neto. Além disso, pressupõe que os símbolos do alfabeto Σ restringem-se àqueles do diagrama e que $(q_0, a) \mapsto q_1 \in \delta$.

Na próxima cena CA2, averigua-se o grau de entendimento da representação e, por conseguinte, do autômato representado. Para isso, aciona-se o processo cognitivo de entendimento de Bloom e que deve corresponder a uma cena de catástrofe.

Cena CA2 (catástrofe) Fubã interpreta esta representação como "Após uma sequência de símbolos a, o autômato permanece no estado q_0. Na primeira ocorrência de b, ele passa ao estado q_1. Aí permanece na presença de uma subcadeia de símbolos b. Há reconhecimento caso seja q_1 o estado do autômato após o consumo completo da cadeia de entrada." (sim/não) ■

	Fatual	Conceitual	Procedural	Metacognitivo
Lembrar	CA1, CA3, CA4, CA5			
Entender		CA2, CA6		
Aplicar		CA7		

Figura 2. Plano dos objetivos de aprendizagem da fábula OC2-RD2.

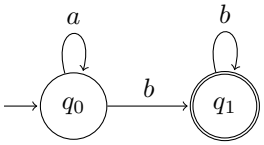


Figura 3. Diagrama de um autômato sem a presença de transições adaptativas.

Cada célula superior (estrutura formal) deve corresponder a uma única célula inferior (exemplo)

1. Q	2. δ	3. F	4. Σ

A. $\{q_1\}$	B. $\{a, b\}$	C. $\{q_0, q_1\}$	D. $(q_0, a) \mapsto q_1$

Figura 4. Relacionamento entre a estrutura formal de um autômato com a representação da Figura 3.

Caso o aprendiz concorde com esta interpretação de Fubã, ter-se-á um indício que o seu processo de entendimento está correto. E quanto a um modelo de implementação destes conceitos? No presente ensaio, o mecanismo de macros será utilizado para este fim.

B. Noção de macro

Macros constituem uma particular instância do fenômeno de reescrita de termos. Em linhas gerais, uma macro pode ser interpretada como uma abreviatura que remete a uma determinada entidade, convenientemente abstraída de características supérfluas ou irrelevantes no nível de observação corrente [11].

Pressupondo-se a existência de algumas cenas de contratempo para introduzir os conceitos de reescrita e macros, passa-se a uma cena que convida o personagem Fubã a recuperar este conhecimento.

Cena CA3 (macro, contratempo) Fubã encontra um diagrama contendo representações textuais (palavras em Português) e objetos (imagens estilizadas de animais), apresentado na Figura 6. As palavras e imagens representam macros e suas abstrações correspondentes, respectivamente. ■

Com base na representação da Figura 6, Fubã deve relacionar cada ocorrência de macro com sua abstração correspondente. Em síntese, associa-se a representação visual ao nome do animal.

Cada célula superior (estrutura formal) deve corresponder a uma única célula inferior (exemplo)

1. Q	2. δ	3. F	4. Σ

A. $\{q_1\}$	B. $\{a, b\}$	C. $\{q_0, q_1\}$	D. $(q_0, a) \mapsto q_1$

Figura 5. Resolução do relacionamento entre a estrutura formal de um autômato com a representação da Figura 3.

Cada célula superior (nome da macro) deve corresponder a uma única célula inferior (objeto)

rato	coelho	pato	galo

Figura 6. Relacionamento entre o nome da macro e o objeto a qual esta se remete.

O aprendiz, atuando no papel do personagem Fubã, deverá ser capaz de produzir as associações de acordo a Figura 7. Por que? Ele identifica os elementos do diagrama, representando-os na forma simbólica de acordo com critérios semióticos estabelecidos previamente (identificação do animal, abstração e recuperação do nome).

Como o escopo desta fábula inclui o estudo de autômatos adaptativos, ela prossegue na direção das transições adaptativas.

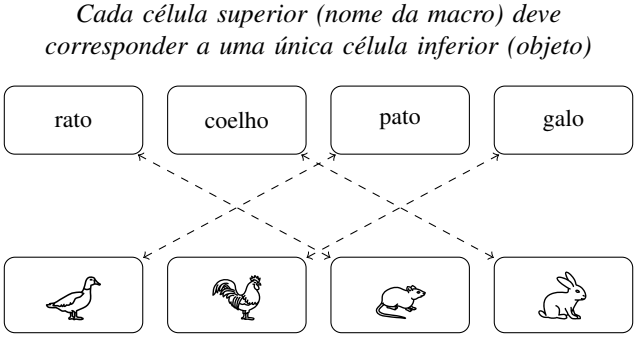


Figura 7. Resolução do relacionamento entre o nome da macro e o objeto a qual esta se remete.

III. Adaptatividade e macro expansão

O mecanismo adaptativo apoia-se em três tipos de ações adaptativas. Uma particular combinação de ações adaptativas origina uma função adaptativa, que pode ser posicionada antes ou depois do disparo de uma transição habilitada entre estados. A próxima cena revisa os tipos de ações adaptativas.

Cena CA4 (notação, contratempo) Fubã deve preencher as lacunas empregando as palavras *remoção*, *inserção* e *consulta*. ■

> Os símbolos ?, − e + denotam, respectivamente, (1) _____, (2) _____ e (3) _____.

O correto preenchimento associa ? com (1) *consulta*, − com (2) *remoção* e + com (3) *inserção*. Tais ações adaptativas foram projetadas para suportar operações básicas sobre a topologia de um particular autômato adaptativo. Essencialmente, uma consulta atribui estados e transições em variáveis locais, as quais podem ser utilizadas para remover e inserir elementos na estrutura do autômato. A próxima cena da fábula explora este conhecimento.

Cena CA5 (mecanismo adaptativo, contratempo) Fubã observa três momentos de alteração em um autômato com a transição adaptativa A_1, durante o processamento da cadeia $aabb$. No instante inicial, t_0, o autômato encontra-se no estado q_0, preparado para reconhecer o primeiro símbolo a da cadeia de entrada, conforme ilustra a Figura 8. ■

- Neste momento, após reconhecer o primeiro a (e antes de reconhecer o segundo), ele sofre uma transformação: indicação de q_1 como estado final e inserção da transição $(q_0, b) \mapsto q_1$ e, além disso, a transição adaptativa $(q_0, a), A_1 \mapsto q_0$ se altera para $(q_0, a), A_2 \mapsto q_0$.
- No instante t_1, a aceitação do segundo a provoca a execução da função A_2. Como resultado, duas novas transições são inseridas, $(q_0, b) \mapsto q_2$ e $(q_2, b) \mapsto q_1$, removendo-se $(q_0, b) \mapsto q_1$.
- A partir do instante t_2, a estrutura do autômato não mais se altera e o reconhecimento dos dois símbolos b restantes conduzem ao estado final q_1.

No papel do personagem, o aprendiz deveria associar as ações 1, 3, 5 e 6 à função adaptativa A_1. A execução da função

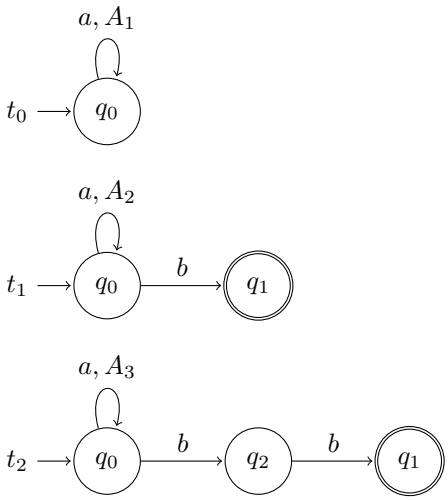


Figura 8. Autômato adaptativo, em três instantes. Observe que, em t_0, o estado de aceitação q_1 foi deliberadamente omitido para fins de construção narrativa.

A_2, portanto, resulta das ações 2, 4, 7 e 8, conforme ilustra a Figura 9.

Considere uma representação alternativa das funções adaptativas A_1 e A_2 como ocorrências de macros. A notação algébrica tradicional pode ser substituída por padrões topológicos visuais e seus termos de reescrita correspondentes, de acordo com a Figura 10, inspirada em um ensaio preliminar realizado por Pistori [12]. Assim, o aprendiz poderá dispor de substituições puramente visuais, sem a necessidade de recorrer explicitamente às ações adaptativas elementares, encapsuladas na forma de funções adaptativas.

A representação através de macros com padrões visuais oferece subsídios para que o aprendiz consolide rapidamente o conhecimento adquirido, sem a necessidade imediata de manipulação de elementos da notação algébrica tradicional. O processo de macro expansão, portanto, corresponde operacionalmente ao mecanismo adaptativo.

Cena CA6 (mecanismo adaptativo e macros, catástrofe) Fubã interpreta esta representação como "Ao observar a topologia do autômato e encontrar partes que sejam idênticas aos padrões visuais das macros, substituí-las por suas representações correspondentes (macro expansão). Feitas as devidas substituições, terei como resultado o autômato correspondente ao instante seguinte, na linha do tempo." ■

Novamente, caso o aprendiz concorde com esta interpretação de Fubã, ter-se-á um indício que o seu processo de entendimento está correto e que os conceitos estão devidamente sedimentados.

IV. Exemplo de aplicação

A cena final desta fábula ilustra a utilização de macros para a implementação de um reconhecedor de linguagem, com base no mecanismo adaptativo. Ela convida o aprendiz a empregar o autômato das cenas anteriores como base para especificar o reconhecedor da linguagem $a^n b^n c^n$. Trata-se de uma cena que

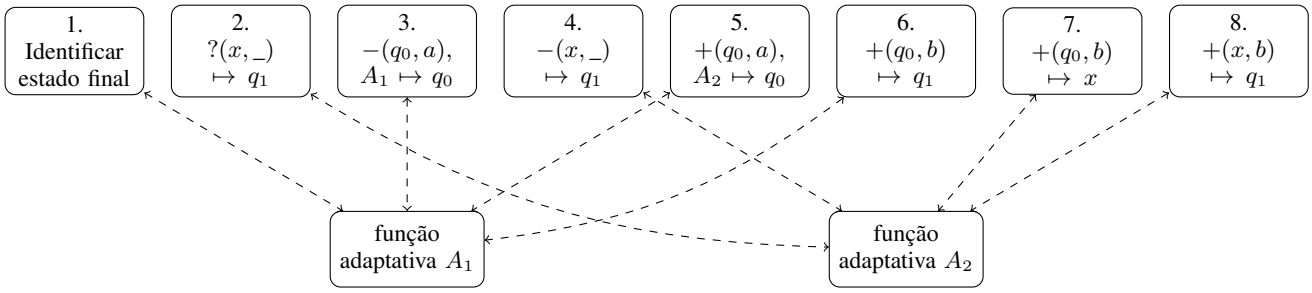


Figura 9. Relacionamento entre as ações adaptativas elementares e as funções adaptativas A_1 e A_2.

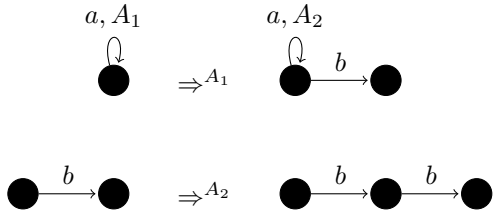


Figura 10. Funções adaptativas A_1 e A_2 da Figura 9, representadas como macros utilizando padrões topológicos visuais.

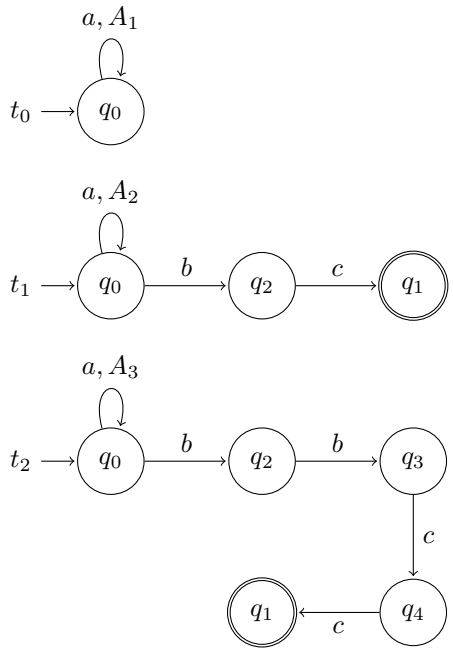


Figura 11. Autômato adaptativo que reconhece cadeias da linguagem $a^n b^n c^n$, evolução no tempo.

deve ser precedida pela apresentação da evolução no tempo apresentada na Figura 11.

A execução da função A_3, no instante t_0, introduzirá duas novas transições, bem como a definição do estado final q_1. Nos instantes seguintes de reconhecimento, transformações similares àquelas da cena CA5 deverão ocorrer, desde que seja corretamente projetada a função adaptativa A_4. Com esta hipótese, refina-se a cena final desta fábula, CA7.

Cena CA7 (reconhecedor, catástrofe) Fubã sente estar preparado para implementar um reconhecedor da linguagem $a^n b^n c^n$ usando macros. ■

A partir da proposta inicial das macros da Figura 10, é possível aprimorar ainda mais tal representação, efetivamente extraindo os padrões referentes às regras de formação de sentenças da linguagem $a^n b^n c^n$. Em síntese, na ocorrência de um símbolo a adicional, incrementam-se as ocorrências de b e c em uma unidade. Assim, é possível determinar o caso base e o passo indutivo de tais regras de formação, conforme ilustra a Figura 12.

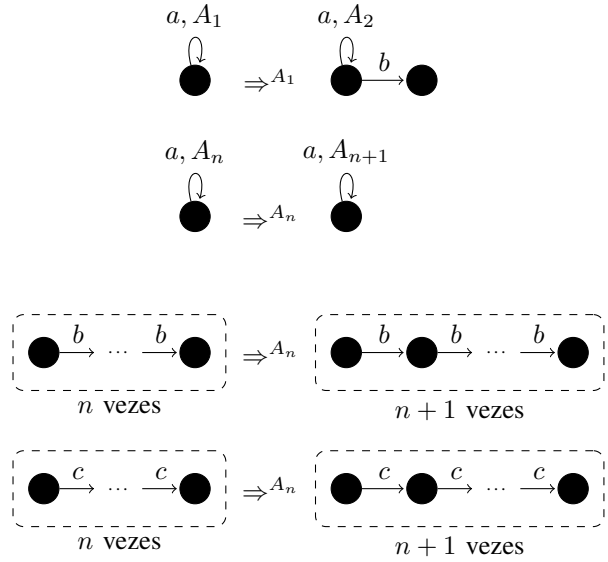


Figura 12. Extração dos padrões referentes às regras de formação da linguagem, determinando-se o caso base e o passo indutivo.

Nesta cena, sugere-se ao aprendiz o uso da linguagem de reescrita JIP [13] para que ele exercite os conceitos reforçados anteriormente e possa implementar um reconhecedor da linguagem estudada. A Figura 13 apresenta o cabeçalho de configuração de um reconhecedor baseado em máquinas de estados finitos em JIP, incluindo a definição da tabela de transições correspondente.

Além do cabeçalho que define a estrutura inicial do autômato adaptativo que reconhece cadeias pertencentes à linguagem $a^n b^n c^n$, o aprendiz dispõe das regras de reescrita correspondentes ao caso base e o passo indutivo (Figura 12), conforme ilustra a Figura 14. Entretanto, Fubã deve complementar a segunda regra (passo indutivo) com a adição do vínculo contextual referente ao símbolo c.

O correto preenchimento do vínculo contextual do símbolo c é apresentado na Figura 15. Assim, a implementação do

```
@fsm => {
  start => q0
  accepting => $seq [ q1 ]
  mapping => {
    (q0, a) -> (q0, 1)
  }
}
```

Figura 13. Cabeçalho de configuração de um reconhecedor baseado em máquinas de estados finitos na linguagem JIP.

```
# regra 1
(x, a) -> (x, 1) => {
  (x, a) -> (x, 1) => _
  _ => (x, a) -> (x, 2)
  _ => (x, b) -> (y, _)
}

# regra 2
(x, a) -> (x, y) => {
  (x, a) -> (x, y) => _
  _ => (x, a) -> (x, y + 1)
  $seq [ (x, b) -> (y, _), x <- y ] => {
    p1, _, p2, _ as $seq [ |$seq| ]
    (p1, _) -> (p2, _) => _
    _ => (p1, b) -> (z, _)
    _ => (z, b) -> (p2, _)
  }
  # preencher regra para consumo de c
}
```

Figura 14. Regras de reescrita correspondentes ao caso base e o passo indutivo (Figura 12), escritas na linguagem JIP.

reconhecedor da linguagem $a^n b^n c^n$ é concluída com sucesso. Ao executar `jip -i rec.jip aabbcc` na linha de comando, o interpretador retornará `true` como resultado, indicando que a cadeia $aabbcc$ pertence de fato à linguagem.

A cena final encerra-se, portanto, com a implementação bem-sucedida de um reconhecedor da linguagem $a^n b^n c^n$ usando macros a partir da Teoria dos Autômatos Adaptativos,

```
# regra para consumo de c
$seq [ (x, c) -> (y, _), x <- y ] => {
  p1, _, p2, _ as $seq [ |$seq| ]
  (p1, _) -> (p2, _) => _
  _ => (p1, c) -> (z, _)
  _ => (z, c) -> (p2, _)
}
```

Figura 15. Código-fonte inicial da implementação de um reconhecedor de sentenças da linguagem $a^n b^n c^n$ utilizando macros na linguagem JIP.

desde os conceitos preliminares até testes de pertinência.

V. Discussão

As ideias centrais que levaram à concepção de uma fábula OC2-RD2 integrada com a Taxonomia de Bloom Revisada foram apresentadas neste artigo. Procurou-se iluminar um caminho para que tal integração produzisse um veículo narrativo de uma introdução à Teoria dos Autômatos Adaptativos. Como suporte ao conhecimento procedural, não apenas os cálculos formais podem ser exercitados, mas, também, a narrativa contempla cenas que exploram um modelo de execução baseado em macros.

O texto narrativo elaborado neste artigo destaca uma particular sequência de cenas para a apresentação de conceitos, definições, terminologia e instruções executáveis a respeito de autômatos e macros. Como ponto de partida, a concepção da sequência de cenas apoiou-se em objetivos de aprendizagem claramente estabelecidos em termos de Bloom. A Técnica OC2-RD2 contribuiu para estabelecer um ritmo narrativo com oportunidades de interação mestre-aprendiz projetadas nas cenas de contratempo e de catástrofe. Neste sentido, o ambiente de aprendizagem construído com a dimensão narrativa caminha na direção das metodologias ativas: *instructional activities involving students in doing things and thinking about what they are doing* [14]. Durante as cenas do tipo reação-dilema-decisão, a dinâmica de aprendizagem focaliza-se no engajamento do aprendiz, ao invés de atividades realizadas pelo mestre. O exemplo de construção de textos narrativos voltados para cursos de nível superior aqui apresentado dificilmente mostra-se-á adequado para todas as realidades educacionais, mas espera-se que sirva como fonte de inspiração para a construção de outros ambientes de aprendizagem da Teoria de Autômatos Adaptativos.

A identificação de padrões sintáticos contextuais através de macros constitui um recurso interessante para a implementação de reconhecedores utilizando técnicas adaptativas [11]. Do ponto de vista da construção de textos narrativos, macros viabilizam representações conceituais mais palatáveis ao aprendiz, dispensando a necessidade imediata e premente da compreensão de notações algébricas convencionalmente empregadas na Teoria de Autômatos Adaptativos [15].

Em continuidade a este artigo, pretende-se elaborar cenas envolvendo a linguagem DInAton, proposta por Vega [16], bem como o refinamento da linguagem de reescrita JIP.

Agradecimentos

Ao prof. Dr. João José Neto pelo apoio e incentivo na elaboração deste artigo, bem como aos colegas do Grupo de Estudos em Modelagem de Software da PUCSP que colaboraram no refinamento da técnica OC2-RD2. De modo especial, uma menção honrosa aos personagens Ocara, Spec, Feh – estes, companheiros de Fubã – bem como Lum e Caô, que foram deliberadamente omitidos na escrita deste artigo.

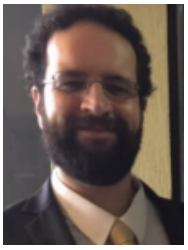
Referências

[1] I. S. Vega, "Fábulas OCC-RDD: histórias didáticas para ambientes interativos híbridos e presenciais de aprendizagem," *Revista da Associação Brasileira de Tecnologia Educacional*, vol. 31, pp. 105–118, 2016.

[2] ——, "Elaboração de histórias OC2-RD2," Pontifícia Universidade Católica de São Paulo – PUCSP, São Paulo, Relatório técnico 5095, Feb. 2018.
[3] D. R. Krathwohl, "A revision of Bloom's taxonomy: an overview," *Theory in practice*, vol. 41, no. 4, pp. 212–218, 2002.
[4] J. J. Neto, "Adaptive rule-driven devices: general formulation and case study," in *International conference on implementation and application of automata*, 2001.
[5] H. M. Bush, J. Daddysman, and R. Charnigo, "Improving outcomes with Bloom's taxonomy: from statistics education to research partnerships," *Journal of Biometrics and Biostatistics*, vol. 4, no. 1, 2014.
[6] I. S. Vega, "Adaptive specifications and emotions: model of narrative generation for interactive learning environments," *IEEE Latin America Transactions*, vol. 13, no. 3, pp. 753–761, Mar. 2015.
[7] S. E. Dreyfus and H. L. Dreyfus, "A five-stage model of the mental activities involved in directed skill acquisition," *Distribution*, 1980.
[8] B. Kort, R. Reilly, and R. W. Picard, "An affective model of interplay between emotions and learning: reengineering educational pedagogy building a learning companion," *ICAT*, pp. 43–48, 2001.
[9] N. Ragonis, "Type of questions: the case of computer science," *Olympiads in Informatics*, vol. 6, pp. 115–132, 2012.
[10] M. V. M. Ramos, J. J. Neto, and I. S. Vega, *Linguagens formais: teoria, modelagem e implementação*. Bookman, 2009.
[11] P. R. M. Cereda, "Macros como mecanismos de abstração em transformações textuais," Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, 2018.
[12] H. Pistori, "Tecnologia em engenharia de computação: estado da arte e aplicações," Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, 2003.
[13] P. R. M. Cereda, "JIP: a modern, type-safe term rewriting language." Escola Politécnica, Universidade de São Paulo, São Paulo, Technical report, Jan. 2019.
[14] C. C. Bonwell and J. A. Eison, "Active learning: creating excitement in the classroom," ASHE-ERIC Higher Education Report, Washington, D.C, Tech. Rep., 1991.
[15] P. R. M. Cereda, N. K. Miura, and J. J. Neto, "Syntactic analysis of natural language sentences based on rewriting systems and adaptivity," *Procedia Computer Science*, vol. 130, pp. 1102–1107, 2018.
[16] I. S. Vega, J. J. Neto, and F. S. Marcondes, "DInAton: a didactic and interative language for learning adaptive automata by construction," *Procedia Computer Science*, pp. 1176—1181, 2017.

Paulo Roberto Massa Cereda é graduado em Ciência da Computação pelo Centro Universitário Central Paulista (2005), mestre em Ciência da Computação pela Universidade Federal de São Carlos (2008) e doutor em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (2018). 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, sistemas de reescrita, cálculo lambda, complexidade computacional, linguagens de programação e construção de compiladores.

Ítalo Santiago Vega possui graduação em Engenharia Elétrica pela Universidade de São Paulo (1986), mestrado em Engenharia Elétrica pela Universidade de São Paulo (1993) e doutorado em Engenharia Elétrica pela Universidade de São Paulo (1998). Atualmente é professor da Pontifícia Universidade Católica de São Paulo e professor das Faculdades Integradas Rio Branco. Tem experiência na área de Engenharia Elétrica, com ênfase em modelagem de sistemas de software, atuando principalmente nos seguintes temas: paradigma de objetos, tecnologia de objetos, processos de desenvolvimento de software, modelos formais e modelagem de sistemas discretos.



Francisco Supino Marcondes possui graduação em Sistemas de Informação pela Faculdades Integradas Rio Branco (2004), especialização Em Análise e Projeto de Sistemas pelo Centro Estadual de Educação Tecnológica Paula Souza (2007), mestrado em Tecnologias da Inteligência e Design Digital pela Pontifícia Universidade Católica de São Paulo (2015), mestrado em Engenharia Eletrônica e Computação pelo Instituto Tecnológico de Aeronáutica (2011), curso técnico profissionalizante em Processamento de Dados pelo Colégio Mário de Andrade (1999) e aperfeicoamento em Formação Pedagógica de Docentes da Educação Profissional em nível Médio pelo Instituto Federal de São Paulo (2017). Atualmente é Professor do Instituto Federal de Ciências e Tecnologia de SP, Revisor de periódico da Innovations in Systems and Software Engineering e Membro de corpo editorial da Acta Informaticae Brasiliensis. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Computação.

Formalização da expansão de submáquinas do autômato de pilha estruturado

P. R. M. Cereda e J. José Neto

***Abstract*—Uma chamada de submáquina tradicional em um autômato de pilha estruturado pode ser substituída por uma expansão desta, tal que uma cópia da submáquina que originalmente seria chamada é incorporada à topologia corrente, com as devidas ligações. Neste caso, a expansão de submáquinas pode ser entendida como uma forma de adaptatividade mais restrita. Este artigo apresenta uma formalização da expansão de submáquinas do autômato de pilha estruturado, através de uma extensão do dispositivo original. A pilha sintática é preservada para chamadas convencionais. Adicionalmente, são apresentadas discussões acerca da semântica operacional.**

***Palavras-chave:*—Autômato de pilha estruturado, sistema de reescrita, expansão de submáquinas, adaptatividade.**

I. Introdução

Macros constituem um mecanismo significativo para representação de artefatos em um determinado nível de abstração, sem a necessidade da exposição excessiva de detalhes ou características particulares, viabilizando estruturas mais convenientes e aderentes às necessidades do usuário. Transformações simbólicas e algorítmicas conferem a tal mecanismo expressividade e poder computacional [1].

É importante destacar que o conceito de macro extrapola sua vertente textual. Em [2], Moraes substitui a chamada de submáquina tradicional em um autômato por uma expansão desta, dispensando o uso da pilha sintática. Em linhas gerais, uma cópia da submáquina que originalmente seria chamada é incorporada à topologia corrente, com as devidas ligações (por exemplo, transições em vazio dos estados de aceitação para o estado de destino da chamada). Neste caso, o conceito de macro pode ser entendido como uma forma de adaptatividade mais restrita [1]. Este artigo apresenta uma formalização da expansão de submáquinas do autômato de pilha estruturado, através de uma extensão do dispositivo original. A pilha sintática é preservada para chamadas convencionais.

A organização deste artigo é a seguinte: a Seção II apresenta uma breve revisão bibliográfica da teoria. A Seção III introduz formalmente um autômato de pilha estruturado com suporte a expansão de submáquinas. Discussões sobre a semântica operacional são contempladas na Seção IV. As considerações finais são apresentadas na Seção V.

II. Conceitos iniciais

Esta seção apresenta os conceitos relevantes à formalização da expansão de submáquinas do autômato de pilha estruturado através de um sistema de reescrita de propósito geral.

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

A. Autômato de pilha estruturado

O autômato de pilha estruturado [3], [4] é um tipo de autômato de pilha formado por um conjunto de autômatos, também chamados *submáquinas*, a cada um dos quais cabe a tarefa de efetuar o reconhecimento de uma das diferentes classes de subcadeias que compõem uma cadeia de entrada em análise [5]. Diferentemente do autômato de pilha tradicional, a pilha tem a finalidade exclusiva de armazenar estados de retorno a cada chamada de uma submáquina. As chamadas e retornos consistem em transferir o controle entre uma submáquina e outra; essa transição consiste em utilizar o símbolo de entrada apenas para a tomada de decisão do autômato em relação a qual transição executar, sendo o tal símbolo consumido na transição subsequente [4], [6].

Definição 1 (autômato de pilha estruturado)**.** Um *autômato de pilha estruturado* M é definido como $M = (Q, A, \Sigma, \Gamma, P, Z_0, q_0, F)$, em que Q é um conjunto finito não-vazio de estados, A é um conjunto de submáquinas (introduzidas formalmente na Definição 2), Σ é o alfabeto do autômato, correspondendo ao conjunto finito não-vazio dos símbolos de entrada, Γ é o conjunto finito não-vazio dos símbolos de pilha, a serem 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, e $F \subseteq Q$ é o conjunto dos estados de aceitação do autômato (da primeira submáquina) [3], [4]. □

Definição 2 (submáquina do autômato de pilha estruturado)**.** Uma *submáquina* $a_i \in A$ é definida como um autômato finito tradicional, da forma $a_i = (Q_i, \Sigma_i, P_i, q_{i,0}, F_i)$, no qual $Q_i \subseteq Q$ é o conjunto de estados de a_i, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada de a_i, $q_{i,0} \in Q_i$ é o estado de entrada da submáquina a_i, $P_i \subseteq P$ é a relação de transição de estados de a_i, e $F_i \subseteq Q_i$ é o conjunto de estados de retorno da submáquina. □

Definição 3 (relação de transição do autômato de pilha estruturado)**.** A *relação de transição* P é definida como $P \subseteq \Gamma \times Q \times \Sigma \times \Gamma \times Q$, na forma $(\gamma g, q, s\alpha) \rightarrow (\gamma g', q', \alpha)$, na qual q, q' são os estados corrente e de destino, respectivamente, 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, e γ é o restante da pilha. Uma configuração é um elemento de $Q \times \Sigma^* \times \Gamma^*$, e uma relação entre configurações sucessivas $\vdash$ é definida como:

- *Consumo de símbolo:* $(q, \sigma w, uv) \vdash (q', w, xv)$, com $q, q' \in Q$, $u, x \in \Gamma$, $v \in \Gamma^*$, $\sigma \in \Sigma \cup \{\epsilon\}$, $w \in \Sigma^*$, se σ foi consumido pelo autômato, $x = u$, e $(\gamma, q, \sigma\alpha) \rightarrow (\gamma, q', \alpha) \in P$.
- *Chamada de submáquina:* $(q, w, uv) \vdash (q', w, xv)$, com

$q, q' \in Q$, $u \in \Gamma$, $v, x \in \Gamma^*$, $w \in \Sigma^*$, $x = pu$, com chamada da submáquina R, estado inicial q', retorno em p, e $(\gamma, q, \alpha) \rightarrow (\gamma p, q', \alpha) \in P$.
– *Retorno de submáquina:* $(q, w, uv) \vdash (q', w, v)$, com $q, q' \in Q$, $u, x \in \Gamma$, $v \in \Gamma^*$, $w \in \Sigma^*$, $u = q'$, com retorno de submáquina para q', e $(\gamma g, q, \alpha) \rightarrow (\gamma, g, \alpha) \in P$. □

Definição 4 (linguagem reconhecida por um autômato de pilha estruturado)**.** A *linguagem reconhecida por um autômato de pilha estruturado* M é dada por $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (f, \epsilon, Z_0), f \in F\}$. □

Notação 1 (representação gráfica de chamada de submáquina)**.** Uma chamada de submáquina pode ser representada graficamente através de uma transição com linhas duplas, conforme ilustra a Figura 1. Observe que, a partir do estado $q_{i,m}$ da submáquina a_i, a execução é transferida para a submáquina a_j e o endereço referente ao estado de retorno $q_{i,n}$ é inserido no topo da pilha. No exemplo, o estado corrente passa a ser $q_{i,0}$, que é o estado inicial da submáquina a_j. □

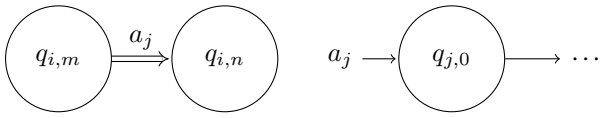


Figura 1. Exemplo de chamada da submáquina a_j.

É importante notar que, por uma questão de organização do modelo, admite-se que $a_i, a_j \in A$, $a_i = (Q_i, \Sigma_i, P_i, q_{i,0}, F_i)$, $a_j = (Q_j, \Sigma_j, P_j, q_{j,0}, F_j)$, $Q_i \cap Q_j = \emptyset$ e $P_i \cap P_j = \emptyset$, isto é, os conjuntos de estados e mapeamentos das submáquinas são disjuntos entre si.

B. Sistema abstrato de redução

Um sistema de reescrita (também chamado *sistema de redução* em sua forma geral) realiza transformações entre termos de acordo com um conjunto de *regras de substituição* (também chamadas *regras de reescrita*) [7]. Aplicações de sistemas de reescrita incluem especificações de tipos abstratos de dados (propriedades de consistência), teoria de computabilidade, decidibilidade de problemas de palavra, prova de teoremas, implementações de linguagens de programação funcionais e dedução automática [1].

Definição 5 (sistema abstrato de redução)**.** Um *sistema abstrato de redução* R é definido como $R = (A, I)$, no qual A é um conjunto de elementos e I é uma sequência de relações binárias $\rightarrow_\alpha$ sobre A, também chamadas de relações de redução ou reescrita [8]. Um sistema abstrato de redução com apenas uma relação de redução é chamado *sistema de substituição* [9] ou *sistema de transformação* [10]. Se $a, b \in A$ e $(a, b) \in \rightarrow_\alpha$, tal relação de redução pode ser escrita como $a \rightarrow_\alpha b$ e b é dita uma α-redução (de um passo) de a. Analogamente, $a \rightarrow_\alpha^* b$, sendo $\rightarrow_\alpha^*$ o fecho transitivo e reflexivo de $\rightarrow_\alpha$, se existe uma sequência finita, potencialmente vazia, de passos de redução $a \equiv a_0 \rightarrow_\alpha a_1 \rightarrow_\alpha \ldots \rightarrow_\alpha a_n \equiv b$, no qual $\equiv$ denota a identidade de elementos de A [11]. □

Definição 6 (forma normal)**.** Um termo $a \in A$ é dito estar em sua *forma normal* quando este não pode ser reescrito, ou seja, $\nexists (a, b) \in I$, $a, b \in A$, tal que $a \rightarrow b$. □

A terminação de reescrita de termos (isto é, quando não há mais regras de substituição que possam ser aplicadas na sequência de termos) é, em geral, um problema indecidível [12]. Entretanto, existem estudos no desenvolvimento de condições para garantia de terminação através de ordenações de redução [13], [14], [15], [16], [17]. Um sistema de reescrita é dito *noetheriano* (isto é, possui garantias de terminar a reescrita de termos) se cada termo possui uma forma normal [12].

III. Expansão de submáquinas

Esta seção apresenta a formalização da expansão de submáquinas do autômato de pilha estruturado, através de uma extensão do dispositivo original.

Definição 7 (autômato de pilha estruturado com suporte a expansão)**.** Um *autômato de pilha estruturado com suporte a expansão* M é definido como $M = (Q, A, \Sigma, \Gamma, P, Z_0, q_0, F, \Phi)$, em que $Q \subset H$ é um conjunto finito não-vazio de estados, H é o conjunto enumerável de todos os estados possíveis, A é um conjunto de submáquinas (introduzidas formalmente na Definição 8), Σ é o alfabeto do autômato, correspondendo ao conjunto finito não-vazio dos símbolos de entrada, Γ é o conjunto finito não-vazio dos símbolos de pilha, a serem 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 Φ é o sistema de reescrita associado (introduzido formalmente na Definição 9). □

Definição 8 (submáquina do autômato de pilha estruturado com suporte a expansão)**.** Uma *submáquina* $a_i \in A$ é definida como um autômato finito tradicional, da forma $a_i = (Q_i, \Sigma_i, P_i, q_{i,0}, F_i)$, no qual $Q_i \subseteq Q$ é o conjunto de estados de a_i, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada de a_i, $q_{i,0} \in Q_i$ é o estado de entrada da submáquina a_i, $P_i \subseteq P$ é a relação de transição de estados de a_i, e $F_i \subseteq Q_i$ é o conjunto de estados de retorno da submáquina. □

Definição 9 (sistema de reescrita do autômato de pilha estruturado com suporte a expansão)**.** O *sistema de reescrita* Φ *do autômato de pilha estruturado com suporte a expansão* é definido como $\Phi = (K, \phi_0, H, J)$, em que K é um conjunto enumerável de funções de mapeamento entre identificadores de estados, $K = \{\phi_i \mid \phi_i \colon H \mapsto H\}$, $\phi_0 \in K$ é a função de mapeamento inicial, H é o conjunto enumerável de todos os estados possíveis do autômato de pilha estruturado, e J é uma relação de reescrita $J \colon H \times \Sigma^* \times \Gamma^* \mapsto H \times \Sigma^* \times \Gamma^*$. A cada nova expansão de submáquina, o índice i da função $\phi_i \in K$ é adicionado em uma unidade, $i \in \mathbb{N}$, isto é, a i-ésima expansão está associada à função ϕ_i correspondente. □

Definição 10 (relação de transição do autômato de pilha estruturado com suporte a expansão)**.** A *relação de transição* P é definida como $P \subseteq \Gamma \times H \times \Sigma \times \Gamma \times H$, na forma $(\gamma g, q, s\alpha) \rightarrow (\gamma g', q', \alpha)$, na qual q, q' são os estados corrente

e de destino, respectivamente, 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, e γ é o restante da pilha. Uma configuração é um elemento de $H \times \Sigma^* \times \Gamma^*$, e uma relação entre configurações sucessivas $\vdash$ é definida como:

- *Consumo de símbolo:* $(q, \sigma w, uv) \vdash (q', w, xv)$, com $q, q' \in H$, $u, x \in \Gamma$, $v \in \Gamma^*$, $\sigma \in \Sigma \cup \{\epsilon\}$, $w \in \Sigma^*$, se σ foi consumido pelo autômato, $x = u$, e $(\gamma, q, \sigma\alpha) \to (\gamma, q', \alpha) \in P$.
- *Chamada de submáquina:* $(q, w, uv) \vdash (q', w, xv)$, com $q, q' \in H$, $u \in \Gamma$, $v, x \in \Gamma^*$, $w \in \Sigma^*$, $x = pu$, com chamada da submáquina R, estado inicial q', retorno em p, e $(\gamma, q, \alpha) \to (\gamma p, q', \alpha) \in P$.
- *Retorno de submáquina:* $(q, w, uv) \vdash (q', w, v)$, com $q, q' \in H$, $u, x \in \Gamma$, $v \in \Gamma^*$, $w \in \Sigma^*$, $u = q'$, com retorno de submáquina para q', e $(\gamma g, q, \alpha) \to (\gamma, g, \alpha) \in P$.
- *Expansão de submáquina:* $(q, w, uv) \vdash (q', w, xv)$, com $q, q' \in H$, $u, x \in \Gamma$, $v \in \Gamma^*$, $w \in \Sigma^*$, $x = u$, com expansão da submáquina R, estado inicial r, saída em t, $\phi_i(r) = q'$, e $\forall p_i \in P_R, (\gamma g, e, s\alpha, \gamma g', e', \alpha) \to (\gamma\phi_i(g), \phi_i(e), s\alpha, \gamma\phi_i(e'), \phi_i(e'), \alpha)$, e $P = P \cup \{(uv, q, w, xv, q', w)\} \cup \{p \mid p \in F_R, (uv, \phi_i(p), w, xv, t, w)\}$, e i é adicionado em uma unidade após a expansão. □

Definição 11 (linguagem reconhecida por um autômato de pilha estruturado com suporte a expansão)**.** A *linguagem reconhecida por um autômato de pilha estruturado com suporte a expansão* M é dada por $L(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (f, \epsilon, Z_0), f \in F\}$. □

Notação 2 (representação gráfica de expansão de submáquina)**.** Uma expansão de submáquina pode ser representada graficamente através de uma transição com linha ondulada, conforme ilustra a Figura 2. Observe que, a partir do estado $q_{i,m}$ da submáquina a_i, uma cópia da submáquina a_j é inserida na topologia corrente, com as devidas transições de entrada e saída. A execução do dispositivo, portanto, prossegue como em um autômato finito tradicional. O índice k da função ϕ_k indica que esta é a k-ésima expansão de submáquina. □

Observe que a representação gráfica de chamada de submáquina do autômato de pilha estruturado apresentada na Notação 1 mantém-se inalterada para o dispositivo estendido apresentado nesta seção.

IV. Semântica operacional

Esta seção contempla e discute alguns aspectos da semântica operacional envolvida na expansão de submáquinas do autômato de pilha estruturado.

A. Protótipo de submáquina

A expansão de submáquinas utiliza o conceito de *protótipos* estabelecidos em tempo de definição do autômato de pilha estruturado. Tal resolução determina que as expansões procedam somente nas definições de submáquinas e não em suas instâncias particulares potencialmente modificáveis em tempo de reconhecimento de cadeia. A Figura 3, a seguir, ilustra o conceito de protótipo de submáquina.

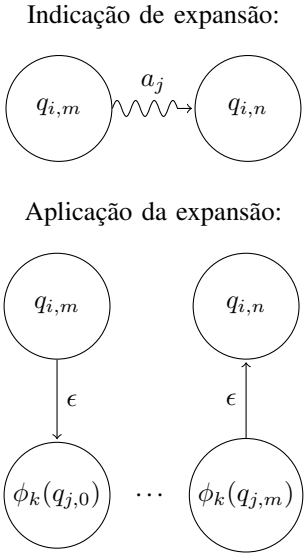


Figura 2. Exemplo de expansão da submáquina a_j.

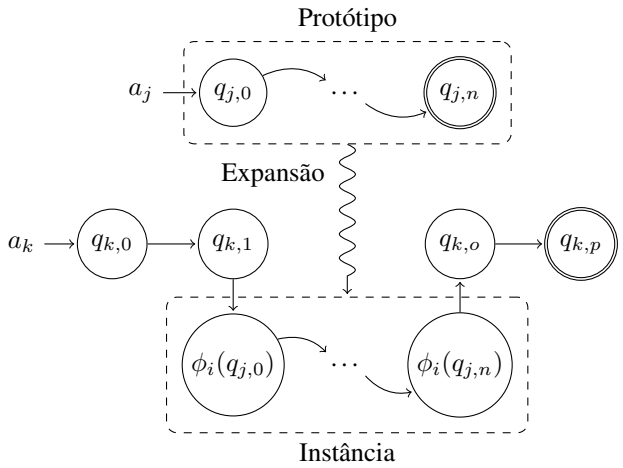


Figura 3. Exemplo de resolução de expansão fundamentada em protótipos de submáquinas. Uma vez expandida, a instância torna-se potencialmente modificável ao longo do tempo, não refletindo em seu protótipo correspondente.

Observe que, de acordo com a Figura 3, o estado de retorno $q_{j,n}$ da submáquina a_j perde tal propriedade quando representado na instância expandida. A razão para tal comportamento deve-se ao fato de que, por definição, o conjunto de estados finais do dispositivo é imutável. De acordo com a Definição 10, os estados de retorno da submáquina recebem transições explícitas em vazio ao estado de destino, tal qual o retorno implícito no formalismo original.

Exemplo 1. Considere um autômato de pilha estruturado M_1 que reconhece cadeias pertencentes à linguagem livre de contexto $L_1 = \{w \in \{a, +, (,)\}^* \mid w$ representa uma soma de termos com suporte a parênteses aninhados $\}$, de acordo com a Figura 4.

O reconhecimento da cadeia $a + a \in L_1(M_1)$ resulta em três expansões de submáquina, incorporando tais instâncias na topologia do autômato de pilha estruturado da Figura 4. A

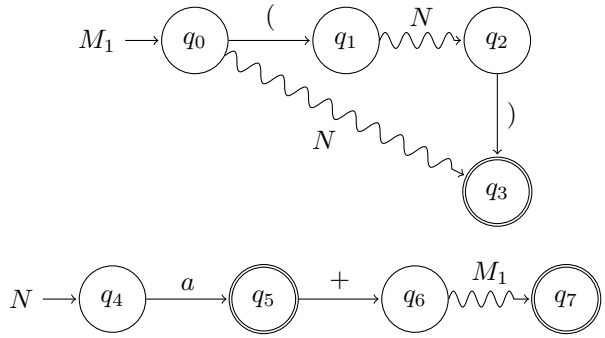


Figura 4. Autômato de pilha estruturado M_1 que reconhece cadeias pertencentes à linguagem livre de contexto $L_1 = \{w \in \{a, +, (,)\}^* \mid w$ representa uma soma de termos com suporte a parênteses aninhados $\}$.

topologia resultante é apresentada na Figura 5.

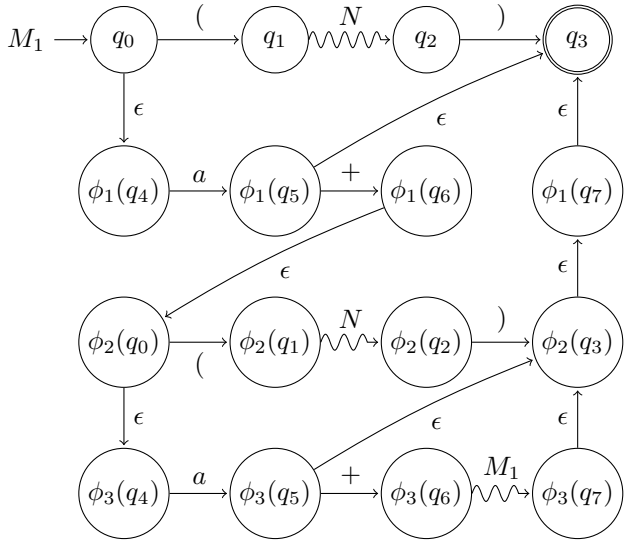


Figura 5. foo

Observe que, de acordo com a Figura 5, a expansão das submáquinas M_1 (nível 2) e N (níveis 1 e 3) procedeu de acordo com suas respectivas definições (protótipos) e não conforme suas instâncias particulares. □

É importante destacar que esta resolução evita que submáquinas recursivas expandam suas instâncias ao invés de seus protótipos. Observe que cada instância particular de submáquina efetua o reconhecimento de uma das diferentes classes de subcadeias que compõem a cadeia de entrada e, portanto, pode não representar uma classe arbitrária em um momento subsequente durante o reconhecimento.

B. Referência ao ponto de expansão

Sob a perspectiva teórica, a função de mapeamento ϕ_i referencia novos identificadores de estados para garantir a unicidade dos estados criados durante a i-ésima expansão. Entretanto, operacionalmente, a função ϕ_i pode manter referência explícita ao ponto de expansão da submáquina, conforme ilustra a Equação 1.

$$\phi_i(a_{j,0}) = \langle s, a_{j,0} \rangle \mid (s, w, uv) \vdash (\phi_i(a_{j,0}), w, uv) \qquad (1)$$

De acordo com a Equação 1, uma expansão indexada por i e iniciada no estado s o especifica como prefixo dos identificadores do mapeamento de ϕ_i para a renomeação de estados na submáquina a_i sendo expandida. A Equação 2 apresenta a forma geral, considerando p como prefixo de mapeamento e k como índice da expansão corrente.

$$\phi_k(a_{i,j}) = \langle p, a_{i,j} \rangle \mid (p, w, uv) \vdash (\phi_k(a_{i,j}), w, uv) \qquad (2)$$

É importante observar que identificadores de mapeamento podem conter prefixos aninhados. Tal característica permite o reconhecimento facilitado de trechos expandidos na topologia do autômato, incluindo seus pontos de origem.

Exemplo 2. Considere um autômato de pilha estruturado M_2 que reconhece cadeias pertencentes à linguagem livre de contexto $L_2 = \{w \in \{a, b\}^* \mid w = a^n b^n, n \in \mathbb{N}\}$, de acordo com a Figura 6.

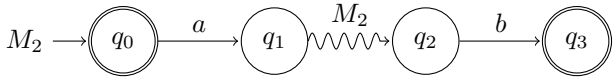


Figura 6. Autômato de pilha estruturado M que reconhece cadeias pertencentes à linguagem livre de contexto $L = \{w \in \{a, b\}^* \mid w = a^n b^n, n \in \mathbb{N}\}$.

O reconhecimento da cadeia $aaabbb \in L_2(M_2)$ resulta em três expansões de submáquina, incorporando tais instâncias na topologia do autômato de pilha estruturado da Figura 6. A topologia resultante é apresentada na Figura 7.

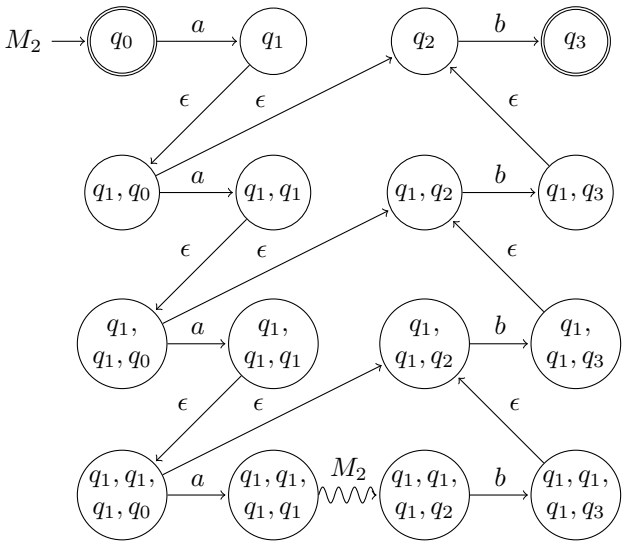


Figura 7. Reconhecimento da cadeia $aaabbb \in L(M)$ pelo autômato de pilha estruturado da Figura 6, resultando em três expansões de submáquina.

Observe que, de acordo com a Figura 7, é possível identificar os pontos de origem de expansão da submáquina M_2 na topologia do autômato e as instâncias propriamente ditas. □

A referência ao ponto de expansão nos identificadores de estados das instâncias de submáquinas oferece conveniências

para análise dos trechos expandidos e, consequentemente, do reconhecimento das classes de subcadeias que compõem a cadeia de entrada. É possível, entretanto, usar outras estruturas de representação conforme o domínio de aplicação, desde que a unicidade dos identificadores de estados seja preservada.

V. Considerações finais

Este artigo apresentou uma formalização da expansão de submáquinas do autômato de pilha estruturado através de uma extensão do dispositivo original, como alternativa a desvios de controle entre autômatos mutuamente recursivos para reconhecimento da cadeia de entrada.

É importante observar que chamadas e expansões de submáquinas, de acordo com a relação de transição apresentada na Definição 10, representam operacionalmente os mesmos passos computacionais necessários para reconhecimento das classes particulares de subcadeias componentes da cadeia de entrada em análise. A distinção reside na substituição do desvio de controle convencional entre autômatos mutuamente recursivos, auxiliado por uma pilha sintática para armazenamento exclusivo do estado de retorno, por transições e estados explicitamente incorporados à topologia da submáquina corrente. A extensão apresentada neste artigo mantém a possibilidade de chamadas de submáquinas e preserva a pilha sintática. Por esta perspectiva, o formalismo pode ser simplificado para admitir apenas expansões de submáquinas.

Pragmaticamente, o conceito de expansão de submáquina dispensa o uso da pilha sintática, mas pode aumentar significativamente o espaço de estados e transições do autômato de pilha estruturado. É necessário considerar tais particularidades ao adotar um tratamento específico para submáquinas, conforme o domínio da aplicação. Uma solução híbrida (equilíbrio entre chamada e expansão) pode oferecer vantagens operacionais. Do ponto de vista de implementação, a expansão de submáquina permite que o dispositivo em execução atue como um autômato finito tradicional, sem manipulação de pilha.

A expansão de submáquina pode ser entendida como uma forma de adaptatividade mais restrita, dado que o autômato de pilha estruturado sofre modificações em sua topologia em tempo de reconhecimento de cadeia sem, entretanto, alterar a classe de linguagens que identifica.

Espera-se que tal formalização contribua para a especificação de autômatos com características de automodificação seletiva através da expansão controlada de submáquinas, conforme a conveniência e as características do domínio de aplicação.

Referências

[1] P. R. M. Cereda, "Macros como mecanismos de abstração em transformações textuais," Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 2018.

[2] M. Moraes, "Alguns aspectos de tratamento de dependências de contexto em linguagem natural empregando tecnologia adaptativa," Tese de Doutorado, Escola Politécnica, Universidade de são Paulo, São Paulo, 2006.

[3] J. José Neto and M. E. S. Magalhães, "Reconhecedores sintáticos: Uma alternativa didática para uso em cursos de engenharia," in *Congresso Nacional de Informática*, 1981, pp. 171–181.

[4] J. José Neto, "Contribuições à metodologia de construção de compiladores," Tese de Livre docência, Escola Politécnica, Universidade de São Paulo, São Paulo, 1993.

[5] M. V. M. Ramos, J. José Neto, and Í. S. Vega, *Linguagens formais: teoria, modelagem e implementação*. Porto Alegre: Bookman, 2009.

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

[7] F. Baader and T. Nipkow, *Term rewriting and all that*. New York: Cambridge University, 1998.

[8] J. W. Klop, "Term rewriting systems," Tech. Rep., 1992.

[9] J. Staples, "Church-Rosser theorem for replacement systems," in *Algebra and Logic*, ser. Lecture Notes in Mathematics, J. Crosley, Ed. Springer-Verlag, 1975, no. 450.

[10] M. Jantzen, "Confluent string rewriting and congruences," in *EATCS (European Association for Theoretical Computer Science) Monographs on Theoretical Computer Science*. Berlin: Springer-Verlag, 1988, vol. 14.

[11] G. Huet and D. C. Oppen, "Equations and rewrite rules: A survey," Stanford University, Stanford, Technical report, 1980.

[12] M. Hermann, C. Kirchner, and H. Kirchner, "Implementations of term rewriting systems," *The Computer Journal*, vol. 34, no. 1, pp. 20–33, feb 1991.

[13] N. Dershowitz, "Orderings for term-rewriting systems," *Theoretical Computer Science*, no. 17, pp. 279–301, 1982.

[14] ——, "Termination of rewriting," *Journal of Symbolic Computation*, vol. 3, no. 1–2, pp. 69–116, 1987.

[15] J.-P. Jouannaud, P. Lescanne, and F. Reinig, "Recursive decomposition ordering," in *Formal Description of Programming Concepts 2*, D. Bjørner, Ed. Garmisch-Partenkirchen, Germany: North Holland, 1982, pp. 331–348.

[16] P. Lescanne, "Uniform termination of term rewriting systems," in *Colloque les Arbres en Algebre et en Programmation*, B. Courcelle, Ed. Bordeaux: Cambridge University, 1984, pp. 182–194.

[17] G. Huet, "Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems," *Journal of the ACM*, vol. 27, no. 4, pp. 797–821, oct 1980.

Paulo Roberto Massa Cereda é graduado em Ciência da Computação pelo Centro Universitário Central Paulista (2005), mestre em Ciência da Computação pela Universidade Federal de São Carlos (2008) e doutor em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (2018). 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, sistemas de reescrita, cálculo lambda, complexidade computacional, 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.

Controle Adaptativo de Intersecções em Trânsito Urbano Baseado em Parâmetros Reais de Vias

Nelson Murcia García, André R. Hirakawa
Departamento de Engenharia de Computação
Universidade de São Paulo, USP
São Paulo, Brasil
nelson.murcia@usp.br

***Abstract*— O desenvolvimento econômico mundial tem como desvantagem o aumento da população nos grandes centros urbanos, isto resulta em maior trânsito de veículos nas cidades, o que aumenta a possibilidade de engarrafamentos no trânsito. Inúmeras pesquisas são desenvolvidas com objetivo de fornecer soluções na área do trânsito urbano. Este trabalho visa como solução para o problema do engarrafamento de trânsito, aplicar um controle adaptativo das intersecções dependendo de parâmetros reais da rede de vias. Para atingir essa meta, é utilizado um modelo de Redes Neurais Bio-Inspiradas e três métodos diferentes para calcular o coeficiente de relação entre interseções. Posteriormente são analisados os resultados obtidos com cada método e comparados com um método de controle de tempos fixo atualmente utilizado na cidade de São Paulo. O cenário escolhido para as simulações e analises apresenta características reais de uma região da cidade de São Paulo como: distribuição das vias, fluxos de veículos e tempos de ativação das fases dos semáforos. Para cada método foram realizados dois tipos de simulações, compreendendo demandas baixas e altas de veículos. Os resultados mostram que o método de Redes Neurais para o cálculo do coeficiente de relação entre as intersecções do modelo de Redes Neurais Bio-Inspiradas obteve os melhores resultados.**

Keywords—Bio-Inspired Neural Networks; Traffic Lights Control; Urban Traffic Control; Travel Average Time; Level Occupation Roads.

I. INTRODUÇÃO

O desenvolvimento e o crescimento das grandes cidades hoje é regido pela economia e o desenvolvimento social. Isso explica como a população urbana tem aumentado significativamente nos últimos anos, de acordo com [1] em 2007, a população urbana excedeu à população rural, sugerindo que para o ano 2050, a população urbana vai representar o 70% da população mundial. Consequentemente, os pesquisadores apresentam soluções que procuram melhorar o problema da superlotação nos grandes centros urbanos. Por esta razão as pesquisas são conduzidas para resolver questões fundamentais, em temas tais como: energia, desenvolvimento sustentável, segurança, habitação, saúde e transporte [2]–[4].

O transporte é um elemento-chave para o desenvolvimento urbano nas cidades, soluções para problemas relacionados com esta área são ainda um desafio para os pesquisadores. O aumento do número de pessoas em um espaço de tamanho semelhante, provoca o aumento do trânsito, sendo um problema sério enfrentado hoje nas grandes cidades [5], [6]. O aumento da infraestrutura de estradas e rodovias, do transporte público e o controle inteligente nas intersecções das vias são algumas das ações a serem realizadas para resolver problemas na área do transporte. Os semáforos são dispositivos comumente usados para controlar as intersecções, e o ajuste do tempo de cada fase do semáforo pode ajudar consideravelmente evitando engarrafamentos nas estradas [7]–[13].

Muitos estudos foram realizados para controlar os tempos de cada fase em um semáforo. A natureza imprevisível e estocástica da demanda de trânsito torna a tarefa de otimizar um controle adequado mais difícil. Como consequência encontramos na literatura os algoritmos e métodos mais variados, embora atualmente o controle de trânsito urbano moderno pode ser dividido em dois grupos: teoria do Controle Ótimo e a Inteligência Artificial [8].

Um controlador de semáforo adaptativo para o controle de semáforos é proposto por Taranjeet Kaur et al. em [13]. O controlador utiliza redes neurais e algoritmos genéticos para adaptar os horários do sinal de trânsito de acordo com o congestionamento de cada intersecção. A rede neural obtém os tempos do sinal como a entrada e fornece o comprimento da fila como a saída. Outro exemplo da utilização das redes neurais é apresentado em [10], a proposta dos autores é baseada em um ajuste ótimo nos tempos do semáforo, concluindo que a maior parte do tempo, os desempenhos dos dois algoritmos propostos têm um comportamento semelhante, mas categoricamente superior ao controlador de tempo fixo, uma observação deste é que o algoritmo é proposto e testado apenas para um único cruzamento de semáforo.

Finalmente em [7], é desenvolvida uma rede neural de inspiração biológica (BiNN, "Bio-Inspired Neural Network") que é capaz de monitorar continuamente o status do sistema e tomar decisões, estabelecendo um sistema multiagente e permitindo o controle coordenado de várias intersecções. Considerando como variável de entrada do sistema o nível de ocupação dos veículos nas intersecções das ruas. Além disso propõe um método para a determinação de parâmetros de acordo com o comportamento desejado e proporciona um método para a análise de estabilidade. O algoritmo é validado utilizando um simulador de mobilidade urbana e comparado com um controlador iterativo convencional, obtivendo melhores resultados em simulações para demandas de trânsito baixa, moderada e pesada. Este modelo estende-se a um modelo multiagente, no qual cada agente controla uma única intersecção e interage com os agentes vizinhos para conseguir o controle coordenado de várias intersecções. A proposta evita a saturação

das ruas e coordena as atividades dos agentes vizinhos causando ondas de luzes verdes nas fases comuns de intersecções vizinhas.

A seguir exibe-se a aplicação de uma variante do algoritmo BiNN para o controle de interseções em um cenário com características reais. A segunda seção apresenta a estrutura do modelo para o Controle Multiagente. Posteriormente, apresenta-se o conceito de Coeficiente de Relação e os três métodos utilizados para o cálculo de deste coeficiente nas Seções III e IV. A seguir apresenta-se o cenário utilizado e as simulações desenvolvidas nas seções V e VI. Posteriormente, na seção VII apresentam-se e analisam-se os resultados obtidos na simulação. Finalmente, apresentam-se as observações finais na Seção VIII.

II. ESTRUTURA DO MODELO PARA O CONTROLE MULTIAGENTE

Para o desenvolvimento deste trabalho foi tomado como algoritmo de controle as Redes Neurais Bio-Inspirada, com base nos resultados obtidos em [7] para sistemas de controle dinâmicos, como a estabilidade e adaptabilidade. Outra vantagem deste método é o monitoramento continuo do status do sistema e a toma das decisões. Além disso, adota características biológicas, tais como sinapses inibitórias e mecanismos de adaptação neuronais de redes neurais. Este modelo de acordo com [8], [14], possui um baixo custo computacional, um mecanismo de plasticidade em curto prazo e um comportamento oscilatório o que provoca as mudanças nas fases semafóricas.

A proposta analisada neste artigo visa estender o modelo BiNN proposto por Castro, Martini e Hirakawa em [7] para conseguir diferenciar as relações existentes entre as interseções, adicionando informações ao modelo tais como: a distância entre as interseções, a velocidade dos veículos, o tempo de demora dos veículos até a próxima interseção e o número de faixas da via comum entre essas interseções.

Para adicionar essas características ao modelo e diferenciar a relação de coordenação entre interseções propõe-se realizar um estudo para determinar quanto influi cada característica no atraso médio do fluxo de veículo entre duas interseções. Como são características invariantes no tempo, pode-se realizar um cálculo off-line para obter esse valor. Portanto, este parâmetro, vai se utilizar para obter um coeficiente de relação entre interseções que relaciona o nível de ocupação da via de uma interseção (variável de entrada) com a ativação da fase comum da próxima interseção ou da interseção anterior. Portanto, nessa variante do algoritmo BiNN apenas vamos adicionar informação na rede BiNN encarregada pela coordenação entre intersecções, a rede encarregada pelo controle de uma intersecção [7] vai permanecer intacta.

O modelo para o controle multiagente realizar duas funções imprescindíveis para um controle coordenado efetivo entre os agentes que controlam as interseções. A primeira função centra-se em armazenar a informação quando uma fase de um agente vizinho for ativada até que a fase comum do seguinte agente correspondente esteja pronta para se tornar ativa, provocando ondas de luzes verdes em fases comuns de agentes vizinhos. A segunda função do controle multiagente inibe a fase ativa atual de um agente nos casos em que o nível de ocupação dos veículos na via comum da próxima interseção esteja alta, o que significa que essa via comum não está pronta para receber um fluxo de veículo porque está lotada, evitando engarrafamentos e permitindo que o agente em questão possa ativar uma outra fase com possibilidade de se tornar ativa. A seguir apresenta-se a estrutura do modelo proposto.

O modelo apresentado na Figura 1 reflete a coordenação entre dois agentes de controle de duas interseções com uma via comum entre eles e, portanto, uma fase comum (e.g. Figura 2). Um modelo como o explicado a seguir é utilizado para a coordenação entre todas as fases comuns dos agentes de controle de cada intersecção vizinha.

A Figura 1 mostra o modelo para a coordenação entre as interseções - exemplos A e B do sistema de trânsito apresentado na Figura 2. No modelo, o neurônio $p_{1,A}$ representa o neurônio p (do modelo de controle para agentes individuais) da fase 1 do agente de controle da intersecção A, enquanto o neurônio $p_{1,B}$ representa o neurônio p da fase 1 do agente de controle da interseção B, Figura 11. Desta forma, as fases comuns 1 das interseções vizinhas (com vias comuns) são coordenadas. Além disso, $q_{a,A}$ é o receptor sensorial q_a da intersecção A e $q_{a,B}$ é o receptor sensorial q_a da intersecção B. Todos os outros neurônios ($a_{A\text{-}B}$, $b_{A\text{-}B}$, $c_{A\text{-}B}$ e $h_{A\text{-}B}$) não fazem parte do modelo responsável pelo controle individual das interseções, apresentado no capítulo anterior, e serão explicados logo.

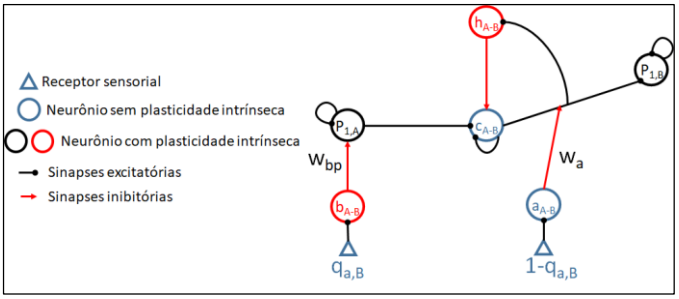


Figura 1. Estrutura do modelo BiNN para a coordenação de intersecções vizinhas

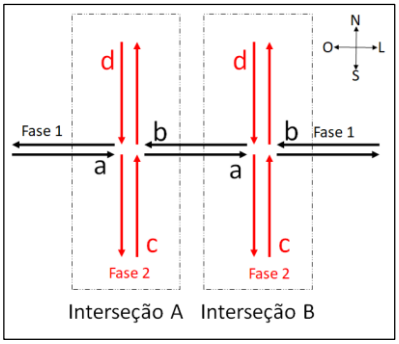


Figura 2. Duas interseções com uma via comum

A segunda função deste modelo é a inibição da fase ativa atual de um agente nos casos em que o nível de ocupação dos veículos na via comum da próxima interseção esteja alta, é concretizada da seguinte forma. A inibição da fase 1 da interseção A, nos casos em que a ocupação na via comum a na interseção B ($q_{a,B}$) é alta, é realizada pelos interneurônios $b_{A\text{-}B}$ que possuem a informação do estado de ocupação da via a da interseção B, Figura 2. Os interneurônios $b_{A\text{-}B}$ inibem os neurônios p da fase comum 1 da interseção A ($p_{1,A}$), em dependência do nível de ocupação da via comum a da interseção

B ($q_{a,B}$), nos casos em que o nível de ocupação da via comum da próxima interseção esteja alta, o que significa que essa via comum não está pronta para receber um fluxo de veículo pois está lotada, evitando engarrafamentos.

Aliás, o mecanismo para concretizar a primeira função do modelo de coordenação de interseções; armazenar a informação quando uma fase de um agente vizinho for ativada até que a fase comum do seguinte agente correspondente esteja pronta para se tornar ativa, provocando ondas de luzes verdes em fases comuns de interseções vizinhas; é apresentado a seguir. O momento ideal para ativar a fase semafórica comum da próxima interseção depende da sua ocupação. Se a próxima interseção tiver uma ocupação alta, sua fase comum deverá se tornar ativa assim que a fase comum da interseção anterior for ativada, para aliviar a ocupação antes que um novo fluxo de veículos chegar. Pelo contrário, se a próxima interseção estiver com uma baixa ocupação, sua fase de semáforo deve esperar que o fluxo de veículos se aproxime da interseção antes de ser ativada.

III. COEFICIENTE DE RELAÇÃO ENTRE INTERSEÇÕES

A relação existente entre cada agente, lembrando que cada agente controla uma interseção, pode variar devido às características físicas como: distância entre as interseções (comprimento da via comum), configuração das interseções, características da via comum entre essas interseções (número de faixas da via e velocidade máxima permitida) e velocidade média atingida pelos veículos nessa via que pode influir no tempo de demora dos veículos para chegar na interseção vizinha.

Para diferençar a relação existente na coordenação entre quaisquer duas interseções pode-se utilizar as sinapses w_{bp} e w_a, (Figura 1). A sinapse w_{bp} descreve o grau de influência do estado de ocupação da via comum da próxima interseção (nível de ocupação da fase a da interseção B, Figura 2) na interseção em questão (interseção *A*, Figura 2), no caso de inibição da fase comum na interseção em questão pois a próxima interseção não está pronta para receber um fluxo de veículos, evitando o engarrafamento.

Um valor alto do parâmetro w_{bp} significa que a interseção A possui uma alta influência na interseção *B*, provocando que os veículos chegarem com maior rapidez. Portanto, a interseção B diminui rapidamente a ativação da fase comum da interseção A pois não está pronta para receber um fluxo de veículos. A alta influência da interseção *A* na interseção *B*, pode ser devido a fatores como: pequena distância entre as interseções, alta velocidade média atingida pelos veículos ou um aumento do número de faixas com respeito à via anterior.

Pelo contrário, um valor baixo do parâmetro significa que o grau de influência da interseção *A* na interseção *B* é baixo, e não afeta em grande medida na inibição da interseção *A*, pois os veículos vão demorar para chegarem na próxima *B*.

A sinapse w_a também descreve o grau de influência da interseção *A* na interseção *B*, mas é utilizada, como explicado anteriormente, para concretizar a segunda função do modelo para a coordenação de interseções. Portanto pode se concluir que os coeficientes de relação para duas interseções vizinhas com uma fase comum são:

$$|w_a| = 1 - |w_{bp}|$$
$$w_{bp} = f(d, v_{max}, \#_f)$$

Onde d é a distância desde uma intersecção até a próxima intersecção vizinha, v_{max} é a velocidade máxima que podem atingir os veículos na via comum entre as duas intersecções vizinhas e $\#_f$ é o número de faixas da via comum.

IV. CÁLCULO DO COEFICIENTE DE RELAÇÃO

Posteriormente, para obter o melhor coeficiente de relação (w_{bp}) possível para cada via comum com diferentes valores nos parâmetros reais das vias, foram realizadas várias simulações variando os coeficientes de relação (entre 0.01 e 0.99) e escolhendo a melhor resposta para cada grupo de parâmetros. O cenário usado para realizar esses testes consiste em duas ruas paralelas e uma rua perpendicular (via comum entre as duas intersecções). A faixa de distância entre as duas interseções analisada foi entre 50 e 500 metros e as velocidades máximas das vias analisadas foram 40, 50 e 60 *km/h*. Por último, o número de faixas na via comum foi variado entre 1 e 4.

Depois dessa análise obtiveram-se 60 pontos (Figura 3) para as diferentes configurações de distância, velocidade máxima e número de faixa e o respectivo coeficiente de relação, que significa a melhor resposta possível para cada configuração analisando desde o ponto de vista do Nível de Ocupação das Vias e o Tempo Meio de Viagem.

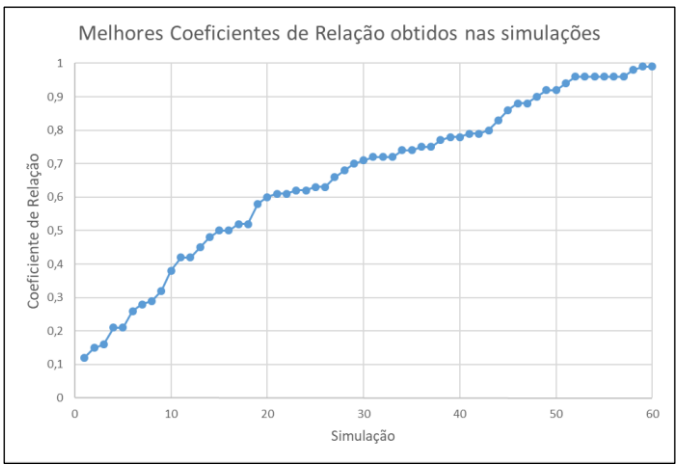


Figura 3. Melhores coeficientes de relação obtidos nas simulações

Para obter a função (1) a partir desses pontos usaram-se três métodos: Rede Neural Artificial (três entradas: $d, v_{max}, \#_f$ y uma saída), Regressão Polinomial com duas variáveis de entrada e grau 4 (tempo para os veículos chegarem na próxima intersecção - $t_{min} = \frac{v_{max}}{d}$ e $\#_f$) e Regressão Polinomial com uma variável de entrada e grau 13 ($t_{min} = \frac{v_{max}}{d}$). Nas figuras 4, 5 e 6 apresentamos tanto a resposta de cada função $f(d, v_{max}, \#_f)$ quanto o erro de aproximação das respostas.

Para analisar o desempenho desses algoritmos no cálculo de um coeficiente de relação, a seguir se analisam as respostas em um cenário com características e demanda de veículos reais. Como foi explicado anteriormente, os parâmetros que são adicionados ao modelo BiNN são invariantes no tempo, portanto, é possível realizar um cálculo inicial off-line para obter esse o valor do coeficiente de relação para quaisquer duas intersecções vizinhas com vias comuns. Esse cálculo é realizado pelos três algoritmos e posteriormente é aplicado o modelo BiNN em um cenário com características e demandas reais.

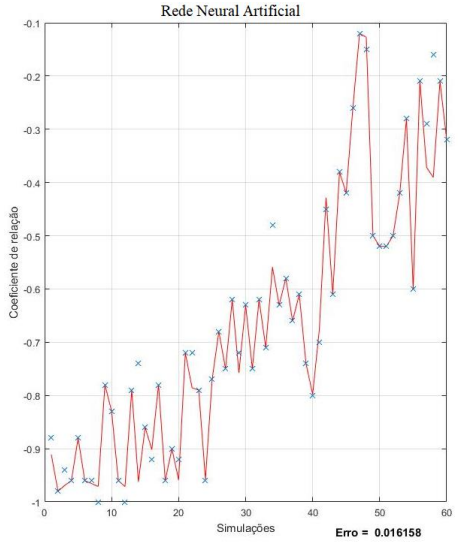


Figura 4. Resposta do método de Redes Neurais Artificiais.

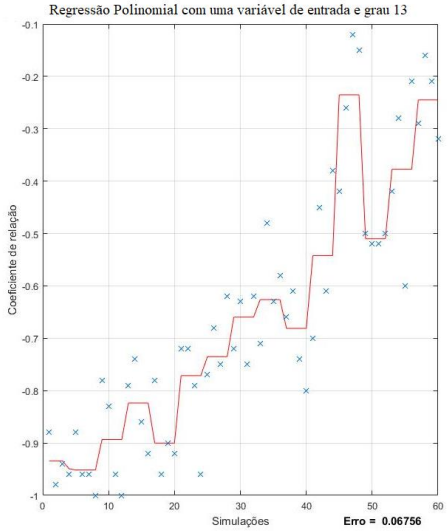


Figura 5. Método de Regressão Polinomial com uma variável.

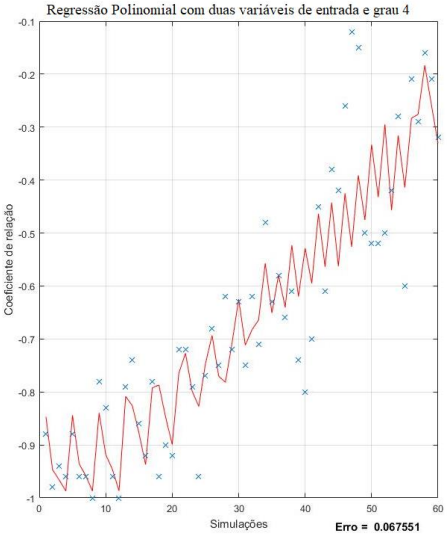


Figura 6. Método de Regressão Polinomial com duas variáveis.

V. SCENARIO

A cidade de São Paulo tem grandes problemas com o nível engarrafamento no trânsito. Sendo um de os maiores aglomerados urbanos no mundo, segundo [15], com uma população de 12.038.175 só em sua região metropolitana, a qual tem uma extensão de 1521,11 km^2 tendo uma densidade de população de 7.914,07 habitantes/km^2, em toda sua área. A cidade conta com uma frota veicular de aproximadamente 7 milhões de veículos e uma extensão de vias de aproximadamente 17.000 km [16]. Por este fato, São Paulo tem uns dos maiores índices de engarrafamento no trânsito, chegando a ter segundo [16], engarrafamentos de mais de 150 km na região do centro expandido com muita frequência nos horários de picos do trânsito.

Figura 7. Cenário escolhido

O cenário escolhido encontra-se na região de Butantã perto da estação do mesmo nome, é bem do tem um total de 7,4 km de ruas e avenidas (Figura 7), com 9 intersecções com semáforos. A Rodovia Raposo Tavares tem um fluxo bem alto de veículos procedente da região Sudoeste da Grande São Paulo, mesma característica apresentada pela Av. Professor Francisco Morato. Além disso esta região tem uma característica muito importante na hora de avaliar nossa proposta, as distâncias das vias comuns entre as intersecções vizinhas, essas distancias variam entre 70 e 400 metros, as velocidades máximas permitidas nessas vias também variam entre 40 e 60 *km/h*. As rotas dos veículos foram programadas com as mesmas características das rotas de trânsito descritas e observadas em [16], [20].

VI. SIMULAÇÃO

As simulações correspondentes foram realizadas com o auxílio das ferramentas MATLAB e SUMO (“Simulation of Urban Mobility”) [17], [18]. O modelo BiNN foi programado na linguagem M do MATLAB, enquanto o modelo do sistema de trânsito urbano foi programado em XML (“eXtensible Markup Language”), linguagem utilizada pelo SUMO. Para executar as simulações e analisar os resultados, foi utilizado o protocolo TraCI4Matlab [19], que adota o paradigma cliente-servidor e permite a interação entre o SUMO (servidor) e o MATLAB (cliente). Todas as simulações tem um tempo de duração de 3600 s (uma hora).

Para o análises dos resultados foram realizadas 8 tipos de simulações diferentes, 6 simulações com o modelo BiNN e os três algoritmos apresentados para o cálculo do coeficiente de relação, além de um algoritmo de Controle de Semáforos com Tempos Fixos, atualmente usado para o controle dessas intersecções. As demandas de veículos usadas para avaliar o desempenho da nossa proposta são: demanda baixa, com 2.2 veículos por segundo começando o percorrido (entrando ao cenário) e demanda alta com 2.9 veículos começando o percorrido.

Por último, todos os veículos utilizados têm as mesmas características, 5 metros de comprimento, com aceleração de 0.8 m/s^2 e velocidade máxima de 16.67 *m/s*. Além de um comportamento de direção estocástico: sigma (parâmetro do simulador SUMO) igual a 0,5.

VII. RESULTADOS

Os indicadores de desempenhos adotados são o Tempo Médio de Viagem dos veículos (TMV), que foi utilizado no analises realizado por [7], [21], e o Nível de Ocupação das Vias do cenário (NOV) que foi utilizado por [8], [9]. Para o cálculo do tempo médio de viagem dos veículos calculou-se a média do tempo de viagem de todos os veículos que iniciaram e a acabaram o percorrido no cenário. Na Figura 8 mostram-se os valores de Tempo Médio de Viagem, obtidos nas 4 simulações realizadas para a análise dos resultados.

Para a demanda de trânsito baixa, o melhor resultado em termos de TMV foi obtido com o modelo BiNN e o método de Redes Neurais para o cálculo do coeficiente de relação entre as intersecções (166 segundos). Enquanto o pior resultado foi obtido pelo Método de Controle de Intersecções com Tempos Fixos. A diferença entre estes resultados é de 31 segundo, e representa uma melhor de um 15.7%.

Para uma demanda alta de trânsito as respostas apresentam um comportamento semelhante com a demanda baixa. Mas a diferença entre o melhor e o pior resultado aumenta em 40 segundos, que significa uma melhora de 16.5%, ligeiramente superior do que a anterior.

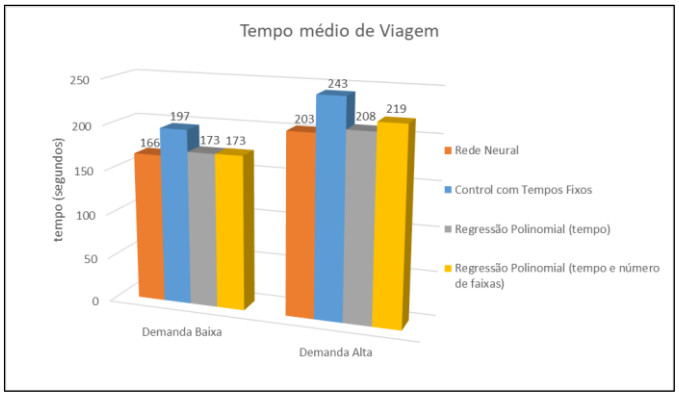


Figura 8. Tempo Médio de Viagem dos veículos

Em relação ao NOV, as simulações realizadas forneceram os resultados mostrados nas Figuras 9 e 10 e os valores médios deste parâmetro apresentam-se na Figura 11. No caso das respostas para a demanda baixa de trânsito, o melhor resultado também foi obtido pelo modelo BiNN com o método de Redes Neurais para o cálculo do coeficiente de relação e a pior resposta foi obtido pelo Método de Controle de Intersecções com Tempos Fixos (Figura 9). Mesmo comportamento que o analisado para o parâmetro anterior. Neste caso na Figura 9 podemos observar que a curva da resposta do parâmetro NOV com Redes Neurais estabiliza-se num valor próximo aos 450 veículos, enquanto a resposta de Tempos Fixos no consegue se estabilizar. Analisando o valor médio deste parâmetro (Figura 11), o primeiro método apresenta uma melhoria de um 15.6% com respeito ao controle com Tempos Fixos.

No caso da demanda alta de trânsito podemos observar que a resposta não tem ponto de estabilização para nenhum dos métodos utilizados, portanto podemos inferir que esta demanda cumpre com o objetivo proposto de saturar as vias. Neste caso, o comportamento das respostas dos diferentes métodos também apresenta a mesma característica, a melhor resposta continua sendo o método de Redes Neurais para o cálculo do coeficiente de relação no modelo BiNN e a melhora do primeiro método com respeito ao segundo é de 17.7%.

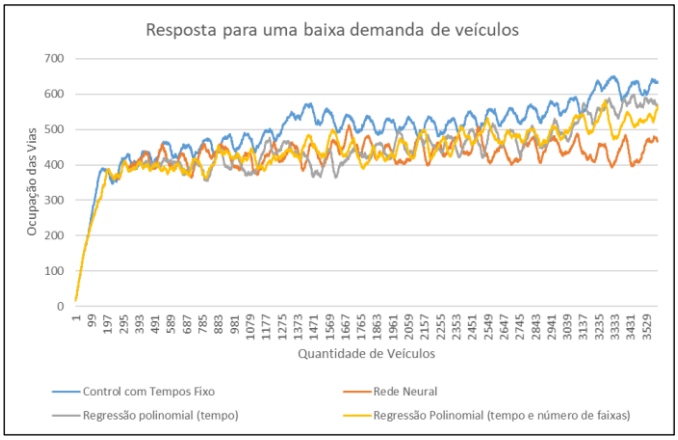


Figura 9. Nível de Ocupação das Vias para demanda baixa de trânsito

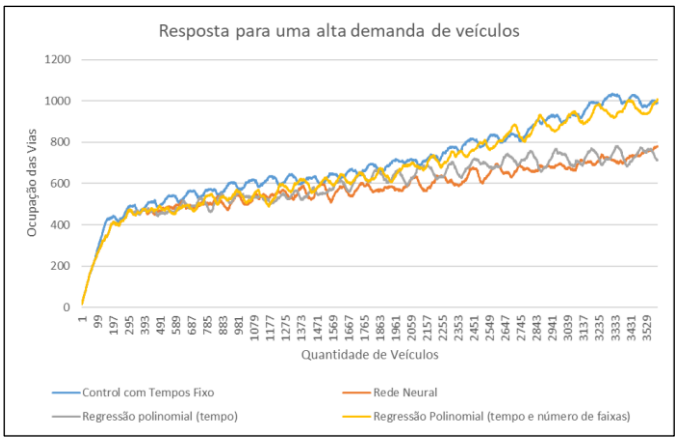


Figura 10. Nível de Ocupação das Vias para demanda alta de trânsito

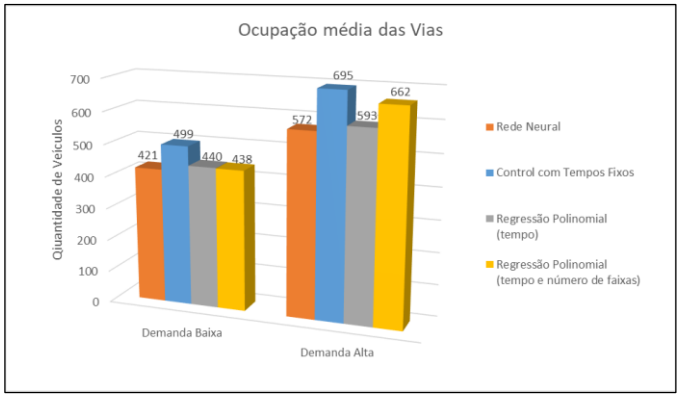


Figura 11. Valor médio do Nível de Ocupação das Vias

VIII. CONCLUSIONS

Este trabalho descreveu a aplicação de um Algoritmo de Controle Adaptativo para o controle multiagente e a sincronização das intersecções vizinhas, diferenciando a relação entre cada intersecção dependendo de parâmetros reais das vias como: distância entre intersecções, velocidade máxima possível da via comum entre as intersecções e número de faixas da via comum. O algoritmo utilizado foi baseado em Redes Neurais Bio-Inspiradas, mas para o cálculo dos coeficientes de relação entre as intersecções foram analisados três métodos: Redes Neurais Artificiais, Regressão Polinomial com duas variáveis de entrada e Regressão Polinomial com uma variável de entrada. Os resultados mostraram que o método de Redes Neurais fornece a melhor solução, obtivendo uma melhoria em média e em comparação com um método de Controle de Tempos Fixos de 16.38% em termos de Tempo Médio de Viagem e Nível de Ocupação das Vias. Portanto podemos concluir que com um modelo adaptativo de controle de trânsito pode se conseguir uma diminuição dos engarrafamentos de trânsito.

ACKNOWLEDGMENT

Este trabalho foi apoiado pela Coordenação de aperfeiçoamento de pessoal de nível superior (CAPES). Agradecemos o apoio do Instituto de Sistemas de Transporte (DLR) por permitir o uso do software de simulação SUMO.

REFERENCES

[1] United Nations, "World population prospects. Key finding and advance tables," New York, 2017.

[2] A. Arroub, B. Zahi, E. Sabir, and M. Sadik, "A literature review on Smart Cities: Paradigms, opportunities and open problems," *2016 Int. Conf. Wirel. Networks Mob. Commun.*, pp. 180–186, 2016.

[3] Y. Hernafi, M. Ben Ahmed, and M. Bouhorma, "An Approaches ' based on Intelligent Transportation Systems to Dissect Driver Behavior and Smart Mobility in Smart City," pp. 886–895, 2016.

[4] F. Vit, N. Mirko, S. Miroslav, and V. Zdenek, "System alliances as a tool for solving Smart Cities problems," *2015 Smart Cities Symp. Prague, SCSP 2015*, 2015.

[5] I. S. and S. SIEMENS, "Solutions for Urban Traffic, Intelligent Traffic Systems."

[6] J. Von Stritzky and C. Cabrerizo, "Ideas paras las ciudades inteligentes del futuro," p. 64, 2011.

[7] G. B. Castro, J. S. C. Martini, and A. R. Hirakawa, "Biologically-Inspired Neural Network for Traffic Signal Control," in *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 2144–2149.

[8] G. B. Castro, D. S. Miguel, B. P. Machado, and A. R. Hirakawa, "Biologically-Inspired Neural Network for Coordinated Urban Traffic Control: Parameter Determination and Stability Analysis," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2015, pp. 209–214.

[9] A. Diveev, E. Sofronova, and V. Mikhalev, "Model Predictive Control for Urban Traffic Flows," *IEEE Int. Conf. Syst. Man, Cybern.*, pp. 3051–3056, 2016.

[10] S. Araghi, A. Khosravi, and D. Creighton, "Optimal Design of Traffic Signal Controller Using Neural Networks and Fuzzy Logic Systems," *Int. Jt. Conf. Neural Networks*, pp. 42–47, 2014.

[11] M. B. W. de Oliveira and A. de A. Neto, "Optimization of Traffic Lights Timing based on Artificial Neural," *IEEE 17th Int. Conf. Intell. Transp. Syst.*, pp. 1921–1922, 2014.

[12] M. Elgarej, M. Khalifa, and M. Youssfi, "Traffic Lights Optimization with Distributed Ant Colony Optimization Based on Multi-agent System," *Springer Int. Publ. AG*, pp. 266–279, 2016.

[13] T. Kaur and S. Agrawal, "Adaptive Traffic Lights Based On Hybrid of Neural Network and Genetic Algorithm for Reduced Traffic Congestion," *Eng. Comput. Sci. (RAECS), 2014 Recent Adv.*, pp. 266–279, 2014.

[14] J. R. Peláez and D. Andina, "Do biological synapses perform probabilistic computations?," *Neurocomputing*, vol. 114, pp. 24–31, 2013.

[15] "Demographia World Urban Areas: 12th Annual Edition," *Demographia*. Frontier Centre for Public Policy, p. 112, 2016.

[16] Prefeitura de São Paulo, "Companhia de Engenharia de Tráfego de São Paulo," 2017. [Online]. Available: http://www.cetsp.com.br/. [Accessed: 02-Mar-2018].

[17] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO – Simulation of Urban MObility An Overview," *Inst. Transp. Syst.*, pp. 1–6, 2011.

[18] R. Hilbrich, "SUMO – Simulation of Urban MObility," *Institute of Transportation Systems*. [Online]. Available: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/. [Accessed: 07-Apr-2017].

[19] A. F. Acosta Gil, "TraCI4Matlab: User's Manual," 1, 2014.

[20] Globo Comunicação e Participações S.A, "Radar do trânsito em tempo real de São Paulo," 2017. [Online]. Available: http://g1.globo.com/sao-paulo/transito/radar-tempo-transito-agora.html.

[21] B. L. Ye, W. Wu, L. Li, and W. Mao, "A hierarchical model predictive control approach for signal splits optimization in large-scale urban road networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2182–2192, 2016.

Parte II

Tutoriais

Reescrita e adaptatividade

P. R. M. Cereda

***Abstract*—Este documento constitui um material complementar ao tutorial homônimo apresentado na décima terceira edição do Workshop de Tecnologia Adaptativa. O objetivo principal é apresentar ao leitor um conjunto de referências recomendadas para a melhor compreensão dos conceitos introduzidos na apresentação oral.**

***Palavras-chave:*—reescrita, macros, expansão, adaptatividade, linguagens de programação, desenvolvimento, implementação**

I. Conceito de tempo

O autor inicia o tutorial definindo o conceito de tempo de acordo com os grandes filósofos da Antiguidade, como Agostinho de Hipona, e contextualiza a importância da revisita àquilo que foi aprendido no passado (referenciado como *memória*) como subsídio para a formulação de novas contribuições (ou *esperança* na definição de Agostinho). As leituras recomendadas para esta seção incluem:

1) A. de Hipona, *Confissões*, 1st ed., L. Mammì, Ed. Penguin books, 2017
2) C. G. Jung, *O homem e seus símbolos*, 1st ed. Nova Fronteira, 2008

II. Sistemas de reescrita

Considerado uma das partes centrais do tutorial, o conceito de sistema de reescrita resume-se na transformação entre termos de acordo com um conjunto de regras de substituição. As leituras recomendadas para esta seção incluem:

1) M. Abadi, L. Cardelli, P. L. Curien, and J. J. Levy, "Explicit substitutions," *Journal of Functional Programming*, vol. 1, no. 4, pp. 375–416, 1991
2) ——, "Explicit substitutions," in *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. 31–46, New York, NY, USA, 1990
3) F. Baader and T. Nipkow, *Term rewriting and all that*. New York: Cambridge University, 1998
4) M. Balaban, E. Barzilay, and M. Elhadad, "Abstraction as a means for end-user computing in creative applications," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 32, no. 6, pp. 640–653, nov 2002
5) H. P. Barendregt, *The Lambda Calculus, Its Syntax and Semantics*, ser. Studies in Logic and the Foundations of Mathematics. North Holland, 1985, vol. 103
6) H. P. Barendregt and E. Barendsen, "Introduction to lambda calculus," Department of Computer Science, Catholic University of Nijmegen, Nijmegen, The Nederlands, Tech. Rep., 1991
7) B. Benninghofen, S. Kemmerich, and M. M. Richter, *Systems of Reductions*, ser. Lecture Notes in Computer Science, G. Goss and J. Hartmanis, Eds. Springer-Verlag, 1987, no. 277
8) D. Bert and R. Echahed, "Abstraction of conditional term rewriting systems," in *International Symposium on Logic Programming*. MIT, 1995, pp. 162–176
9) A. Bove and L. Arbilla, "A confluent calculus of macro expansion and evaluation," *SIGPLAN Lisp Pointers*, vol. 5, no. 1, pp. 278–287, jan 1992
10) ——, "A confluent calculus of macro expansion and evaluation," Instituto de Computación, Universidad de la República, Montevideo, Uruguay, Technical Report INCO-91-01, 1991
11) C. Brabrand and M. I. Schwartzbach, "Growing languages with metamorphic syntax macros," *SIGPLAN Notices*, vol. 37, no. 3, pp. 31–40, jan 2002
12) R. A. Brooker and D. Morris, "A general translation program for phrase structure languages," *Journal of the ACM*, vol. 9, no. 1, pp. 1–10, jan 1962
13) H. Christiansen, "A survey of adaptable grammars," *SIGPLAN Notices*, vol. 25, no. 11, pp. 35–44, November 1990
14) N. Dershowitz, "Termination of rewriting," *Journal of Symbolic Computation*, vol. 3, no. 1–2, pp. 69–116, 1987
15) ——, "Orderings for term-rewriting systems," *Theoretical Computer Science*, no. 17, pp. 279–301, 1982
16) N. Dershowitz and J.-P. Jouannaud, "Rewrite systems," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. North Holland, 1990, pp. 243–320
17) H. B. Enderton, *Computability Theory: an introduction to Recursion Theory*. Academic Press, 2010

III. Aspectos de implementação

Nesta parte do tutorial, o autor discorre acerca dos aspectos de implementação de sistemas de reescrita de propósito geral e específico. As leituras recomendadas para esta seção incluem:

1) P. R. M. Cereda, "Macros como mecanismos de abstração em transformações textuais," Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, 2018
2) R. Ierusalimschy, *Programming in Lua*, 4th ed. Lua.org, 2016
3) R. Ierusalimschy, L. H. Figueiredo, and W. Celes Filho, "Lua: an extensible extension language," *Software: Practice and Experience*, vol. 26, no. 6, pp. 635–652, jun 1996
4) K. Jung and A. Brown, *Beginning Lua Programming*. Wrox, 2007
5) R. M. Keller, *Computer Science: Abstraction to Implementation*. Harvey Mudd College, 2001

O autor pode ser contatado através do seguinte endereço de correio eletrônico: paulo.cereda@alumni.usp.br.

6) M. V. M. Ramos, J. José Neto, and Í. S. Vega, *Linguagens formais: teoria, modelagem e implementação*. Porto Alegre: Bookman, 2009
7) D. Batory, B. Lofaso, and Y. Smaragdakis, "JTS: Tools for implementing domain-specific languages," in *Proceedings of the 5th International Conference on Software Reuse*, 1998, pp. 143–153
8) P. R. M. Cereda and J. José Neto, "Towards performance-focused implementations of adaptive devices," *Procedia Computer Science*, vol. 109, pp. 1164–1169, 2017

IV. Adaptatividade

A adaptatividade é contextualizada, no escopo do tutorial, como fenômeno de automodificação estrutural espontânea. As leituras recomendadas para este seção incluem:

1) P. R. M. Cereda, "Aplicação de técnicas adaptativas na manipulação de textos," Escola Politécnica, Universidade de São Paulo, Tutorial, 2016, tutorial ministrado no X Workshop de Tecnologia Adaptativa – WTA 2016
2) 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*, São Paulo, jan 2016, pp. 16–26
3) ——, "A recommendation engine based on adaptive automata," in *Proceedings of the 17th International Conference on Enterprise Information Systems*, vol. 2, Barcelona, Spain, 2015, pp. 594–599
4) ——, "Personalização de recursos com suporte à privacidade através de técnicas adaptativas," in *Libro de Actas CACIC 2015 – XXI Congreso Argentino de Ciencias de la Computación*, Junín, Buenos Aires, Oct 2015, pp. 62–71
5) ——, "Persistência em dispositivos adaptativos," in *Memórias do VIII Workshop de Tecnologia Adaptativa – WTA 2014*, 2014, pp. 120–125
6) P. R. M. Cereda, N. K. Miura, and J. José Neto, "Syntactic analysis of natural language sentences based on rewriting systems and adaptivity," *Procedia Computer Science*, vol. 130, pp. 1102–1107, 2018
7) P. R. M. Cereda, "Macros como mecanismos de abstração em transformações textuais," Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, 2018
8) J. José Neto, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa," *IEEE Latin America Transactions*, vol. 5, pp. 496–505, 2007
9) ——, "Adaptive rule-driven devices: general formulation and case study," in *International Conference on Implementation and Application of Automata*, Pretoria, 2001
10) ——, "Adaptive automata for context-sensitive languages," *SIGPLAN Notices*, vol. 29, no. 9, pp. 115–124, set 1994

V. Álgebras e linguagens

Por fim, o autor apresenta uma discussão sobre álgebras e linguagens de reescrita de propósito geral e específico. As leituras recomendadas para esta seção incluem:

1) M. Abadi, L. Cardelli, P. L. Curien, and J. J. Levy, "Explicit substitutions," *Journal of Functional Programming*, vol. 1, no. 4, pp. 375–416, 1991
2) ——, "Explicit substitutions," in *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. 31–46, New York, NY, USA, 1990
3) J. Andersen and C. Brabrand, "Syntactic language extension via an algebra of languages and transformations," *Electronic Notes in Theoretical Computer Science*, vol. 253, no. 7, pp. 19–35, sep 2009
4) J. Andersen, C. Brabrand, and D. R. Christiansen, "Banana algebra: Compositional syntactic language extension," *Science of Computer Programming*, vol. 78, no. 10, pp. 1845–1870, sep 2013
5) D. Bert and R. Echahed, "Abstraction of conditional term rewriting systems," in *International Symposium on Logic Programming*. MIT, 1995, pp. 162–176
6) A. Bove and L. Arbilla, "A confluent calculus of macro expansion and evaluation," *SIGPLAN Lisp Pointers*, vol. 5, no. 1, pp. 278–287, jan 1992
7) ——, "A confluent calculus of macro expansion and evaluation," Instituto de Computación, Universidad de la República, Montevideo, Uruguay, Technical Report INCO-91-01, 1991
8) C. Brabrand and M. I. Schwartzbach, "Growing languages with metamorphic syntax macros," *SIGPLAN Notices*, vol. 37, no. 3, pp. 31–40, jan 2002
9) A. Church, "A formulation of the simple theory of types," *Journal of Symbolic Logic*, vol. 5, pp. 56–68, 1940
10) H. Friedman and M. Sheard, "Elementary descent recursion and proof theory," *Annals of Pure and Applied Logic*, vol. 71, no. 1, pp. 1–45, January 1995
11) A. L. Galdino, "Uma formalização da teoria de reescrita em linguagem de ordem superior," Doutorado, Departamento de Matemática, Instituto de Ciências Exatas, Universidade de Brasília, Brasília, 2008
12) K. Gmeiner and B. Gramlich, "Transformations of conditional rewrite systems revisited," in *Recent Trends in Algebraic Development Techniques*, ser. Lecture Notes in Computer Science, A. Corradini and U. Montanari, Eds. Heidelberg: Springer-Verlag, 2008, vol. 2486
13) D. M. Harland, *Polymorphic Programming Languages: Design and Implementation*. New York, NY, USA: John Wiley & Sons, Inc., 1984
14) P. R. M. Cereda, "JIP: a modern, type-safe term rewriting language." Escola Politécnica, Universidade de São Paulo, São Paulo, Technical report, Jan. 2019

Referências

[1] A. de Hipona, *Confissões*, 1st ed., L. Mammì, Ed. Penguin books, 2017.
[2] C. G. Jung, *O homem e seus símbolos*, 1st ed. Nova Fronteira, 2008.
[3] M. Abadi, L. Cardelli, P. L. Curien, and J. J. Levy, "Explicit substitutions," *Journal of Functional Programming*, vol. 1, no. 4, pp. 375–416, 1991.
[4] ——, "Explicit substitutions," in *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. 31–46, New York, NY, USA, 1990.

[5] F. Baader and T. Nipkow, *Term rewriting and all that*. New York: Cambridge University, 1998.

[6] M. Balaban, E. Barzilay, and M. Elhadad, "Abstraction as a means for end-user computing in creative applications," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 32, no. 6, pp. 640–653, nov 2002.

[7] H. P. Barendregt, *The Lambda Calculus, Its Syntax and Semantics*, ser. Studies in Logic and the Foundations of Mathematics. North Holland, 1985, vol. 103.

[8] H. P. Barendregt and E. Barendsen, "Introduction to lambda calculus," Department of Computer Science, Catholic University of Nijmegen, Nijmegen, The Nederlands, Tech. Rep., 1991.

[9] B. Benninghofen, S. Kemmerich, and M. M. Richter, *Systems of Reductions*, ser. Lecture Notes in Computer Science, G. Goss and J. Hartmanis, Eds. Springer-Verlag, 1987, no. 277.

[10] D. Bert and R. Echahed, "Abstraction of conditional term rewriting systems," in *International Symposium on Logic Programming*. MIT, 1995, pp. 162–176.

[11] A. Bove and L. Arbilla, "A confluent calculus of macro expansion and evaluation," *SIGPLAN Lisp Pointers*, vol. 5, no. 1, pp. 278–287, jan 1992.

[12] ——, "A confluent calculus of macro expansion and evaluation," Instituto de Computación, Universidad de la República, Montevideo, Uruguay, Technical Report INCO-91-01, 1991.

[13] C. Brabrand and M. I. Schwartzbach, "Growing languages with metamorphic syntax macros," *SIGPLAN Notices*, vol. 37, no. 3, pp. 31–40, jan 2002.

[14] R. A. Brooker and D. Morris, "A general translation program for phrase structure languages," *Journal of the ACM*, vol. 9, no. 1, pp. 1–10, jan 1962.

[15] H. Christiansen, "A survey of adaptable grammars," *SIGPLAN Notices*, vol. 25, no. 11, pp. 35–44, November 1990.

[16] N. Dershowitz, "Termination of rewriting," *Journal of Symbolic Computation*, vol. 3, no. 1–2, pp. 69–116, 1987.

[17] ——, "Orderings for term-rewriting systems," *Theoretical Computer Science*, no. 17, pp. 279–301, 1982.

[18] N. Dershowitz and J.-P. Jouannaud, "Rewrite systems," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. North Holland, 1990, pp. 243–320.

[19] H. B. Enderton, *Computability Theory: an introduction to Recursion Theory*. Academic Press, 2010.

[20] P. R. M. Cereda, "Macros como mecanismos de abstração em transformações textuais," Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, 2018.

[21] R. Ierusalimschy, *Programming in Lua*, 4th ed. Lua.org, 2016.

[22] R. Ierusalimschy, L. H. Figueiredo, and W. Celes Filho, "Lua: an extensible extension language," *Software: Practice and Experience*, vol. 26, no. 6, pp. 635–652, jun 1996.

[23] K. Jung and A. Brown, *Beginning Lua Programming*. Wrox, 2007.

[24] R. M. Keller, *Computer Science: Abstraction to Implementation*. Harvey Mudd College, 2001.

[25] M. V. M. Ramos, J. José Neto, and Í. S. Vega, *Linguagens formais: teoria, modelagem e implementação*. Porto Alegre: Bookman, 2009.

[26] D. Batory, B. Lofaso, and Y. Smaragdakis, "JTS: Tools for implementing domain-specific languages," in *Proceedings of the 5th International Conference on Software Reuse*, 1998, pp. 143–153.

[27] P. R. M. Cereda and J. José Neto, "Towards performance-focused implementations of adaptive devices," *Procedia Computer Science*, vol. 109, pp. 1164–1169, 2017.

[28] P. R. M. Cereda, "Aplicação de técnicas adaptativas na manipulação de textos," Escola Politécnica, Universidade de São Paulo, Tutorial, 2016, tutorial ministrado no X Workshop de Tecnologia Adaptativa – WTA 2016.

[29] 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*, São Paulo, jan 2016, pp. 16–26.

[30] ——, "A recommendation engine based on adaptive automata," in *Proceedings of the 17th International Conference on Enterprise Information Systems*, vol. 2, Barcelona, Spain, 2015, pp. 594–599.

[31] ——, "Personalização de recursos com suporte à privacidade através de técnicas adaptativas," in *Libro de Actas CACIC 2015 – XXI Congreso Argentino de Ciencias de la Computación*, Junín, Buenos Aires, Oct 2015, pp. 62–71.

[32] ——, "Persistência em dispositivos adaptativos," in *Memórias do VIII Workshop de Tecnologia Adaptativa – WTA 2014*, 2014, pp. 120–125.

[33] P. R. M. Cereda, N. K. Miura, and J. José Neto, "Syntactic analysis of natural language sentences based on rewriting systems and adaptivity," *Procedia Computer Science*, vol. 130, pp. 1102–1107, 2018.

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

[35] ——, "Adaptive rule-driven devices: general formulation and case study," in *International Conference on Implementation and Application of Automata*, Pretoria, 2001.

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

[37] J. Andersen and C. Brabrand, "Syntactic language extension via an algebra of languages and transformations," *Electronic Notes in Theoretical Computer Science*, vol. 253, no. 7, pp. 19–35, sep 2009.

[38] J. Andersen, C. Brabrand, and D. R. Christiansen, "Banana algebra: Compositional syntactic language extension," *Science of Computer Programming*, vol. 78, no. 10, pp. 1845–1870, sep 2013.

[39] A. Church, "A formulation of the simple theory of types," *Journal of Symbolic Logic*, vol. 5, pp. 56–68, 1940.

[40] H. Friedman and M. Sheard, "Elementary descent recursion and proof theory," *Annals of Pure and Applied Logic*, vol. 71, no. 1, pp. 1–45, January 1995.

[41] A. L. Galdino, "Uma formalização da teoria de reescrita em linguagem de ordem superior," Doutorado, Departamento de Matemática, Instituto de Ciências Exatas, Universidade de Brasília, Brasília, 2008.

[42] K. Gmeiner and B. Gramlich, "Transformations of conditional rewrite systems revisited," in *Recent Trends in Algebraic Development Techniques*, ser. Lecture Notes in Computer Science, A. Corradini and U. Montanari, Eds. Heidelberg: Springer-Verlag, 2008, vol. 2486.

[43] D. M. Harland, *Polymorphic Programming Languages: Design and Implementation*. New York, NY, USA: John Wiley & Sons, Inc., 1984.

[44] P. R. M. Cereda, "JIP: a modern, type-safe term rewriting language." Escola Politécnica, Universidade de São Paulo, São Paulo, Technical report, Jan. 2019.

Paulo Roberto Massa Cereda é graduado em Ciência da Computação pelo Centro Universitário Central Paulista (2005), mestre em Ciência da Computação pela Universidade Federal de São Carlos (2008) e doutor em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (2018). 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, sistemas de reescrita, cálculo lambda, complexidade computacional, linguagens de programação e construção de compiladores.

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.