WAGNER JOSÉ DIZERÓ

FORMALISMOS ADAPTATIVOS APLICADOS NA MODELAGEM DE SOFTWARES EDUCACIONAIS

São Paulo
2010

WAGNER JOSÉ DIZERÓ

FORMALISMOS ADAPTATIVOS APLICADOS NA MODELAGEM DE SOFTWARES EDUCACIONAIS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de doutor em Engenharia.

Área de Concentração:
Engenharia Elétrica e Sistemas Digitais

Orientador:
Prof. Livre-Docente João José Neto

São Paulo
2010

DEDICATÓRIA

À minha esposa Joselle
e ao nosso filho Enzo.

AGRADECIMENTOS

Agradeço ao Grande Arquiteto do Universo, que é Deus, por me iluminar com sua Luz e me permitir realizar este trabalho.

À minha esposa, Joselle, por sua presença e seu apoio durante todos os momentos nesses anos de trabalho. Aos meus pais, José e Vicenta, pelo carinho e preocupação comigo. À minha irmã Ana, meu cunhado Claudio e meus queridos sobrinhos Gustavo e Hiago.

Ao meu orientador, João José Neto, que, muito além de um profissional sério, competente e comprometido, é uma pessoa de enorme coração, de quem eu sinto uma imensa satisfação por fazer parte da trajetória de minha vida.

Aos meus amigos e aos companheiros de churrascos Cesar Fassa e Igor Delzan, com quem pude desfrutar momentos muito especiais de diversão e descontração, com muita alegria, risadas e cervejas.

Aos irmãos de coração: Milton Léo e Djalma Cardoso. Pessoas especiais, homens livres e de bons costumes, que admiro e me espelho para um dia ser uma pessoa melhor.

Aos professores Ítalo Santiago Vega e Ricardo Luis de Azevedo Rocha, que participaram da minha banca de qualificação, pelas revisões e importantes comentários.

À Escola Politécnica da USP e seus Professores, que certamente contribuíram imensamente em minha formação. Expresso meu orgulho e gratidão.

Ao Centro Universitário UNILINS por proporcionar as condições que possibilitam a minha capacitação científica.

RESUMO

Esta tese apresenta uma proposta de aplicação de formalismos adaptativos na modelagem e elaboração de cursos para softwares educacionais. A princípio, soluções adaptativas podem ser incorporadas a qualquer tipo de dispositivo guiado por regras. Neste projeto, o dispositivo subjacente utilizado será a Máquina de Moore, que é um autômato finito com saída associada aos seus estados. Assim, para cada estado pode-se associar o material didático a ser apresentado pela função de saída. Aplicando-se os conceitos de adaptatividade nesse transdutor, é possível elaborar cursos dinâmicos, que se auto-modifiquem com base em regras definidas pelo professor e pelas experiências e nível de conhecimento individualizado dos alunos. Para complementar o trabalho, é apresentado um protótipo de curso baseado em Máquina de Moore Adaptativa.

***Palavras-Chave**: tecnologia adaptativa; autômato adaptativo com saída; Máquina de Moore Adaptativa; ensino por computador.*

ABSTRACT

This thesis presents a proposal for application of adaptive formalisms for modeling and development of courses for educational software. In principle, adaptive solutions can be incorporated into any type of device guided by rules. In this project, the device will be used behind the Moore Machine, which is a finite automaton with outputs associated with their states. Thus, for each state we can associate the material to be presented by the function output. Applying the concepts of adaptive technology this transducer, you can develop dynamic courses, which are self-modifying based on rules defined by the teacher, and the level of experience and knowledge of individual students. To complement the work is presented a prototype of course based on Adaptive Moore Machine.

***Keywords**: adaptive technology, adaptive automaton with outputs, Adaptive Machine Moore, teaching by computer.*

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
EAD	Educação a Distância
HA	Hipermídia Adaptativa
HTML	*HyperText Markup Language*
IA	Inteligência Artificial
IEEE	*Institute of Electrical and Electronics Engineers*
ISDL	*Integrated Services Digital Network*
ISO	*International Organization for Standardization*
MM	Máquina de Moore
MMA	Máquina de Moore Adaptativa
PHP	*Personal Home Page*
SE	Software Educacional
SH	Sistemas Hipermídia
SHA	Sistema de Hipermídia Adaptativa
STI	Sistemas Tutores Inteligentes
TA	Tecnologia Adaptativa
UML	*Unified Modeling Language*

SUMÁRIO

1. INTRODUÇÃO

A Tecnologia Adaptativa (TA) pode ser identificada em qualquer modelo que possua a propriedade de modificar seu próprio comportamento, sem a interferência de qualquer agente externo. Dessa forma, se um sistema for definido por um conjunto de regras, é possível incorporar-lhe adaptatividade através da inserção de algum mecanismo que o permita se auto-modificar durante sua operação, alterando o próprio conjunto de regras que define seu comportamento.

A principal característica oferecida pela TA [NETO 1993] reside no fato dela se apoiar em um formalismo com regras dinamicamente variáveis, em contraste com a maioria dos formalismos tradicionais, cujas regras, uma vez estabelecidas, permanecem imutáveis durante toda a sua operação.

Técnicas adaptativas podem ser incorporadas a qualquer tipo de dispositivo guiado por regras. Em [LTA 2009] é possível encontrar diversos tipos de dispositivos adaptativos já propostos, sendo alguns citados a seguir: Autômatos Adaptativos, Statecharts Adaptativos, Redes de Markov Adaptativas, Tabelas de Decisão Adaptativas, Árvores de Decisão Adaptativas, Redes de Petri Adaptativas, ISDL Adaptativos, entre outros.

As aplicações da TA são várias, envolvendo diferentes áreas como: arte por computador, jogos eletrônicos, processamento de linguagem natural, síntese de voz, reconhecimento de padrões, tomada de decisão, compiladores, otimização de código, metamodelagem, computação evolutiva, engenharia de software e robótica. Diversos trabalhos relacionados a TA podem ser encontrados em [LTA 2009].

O ensino por computador é outra área em que a TA demonstra importante potencial [NETO 1993]. Suas características dinâmicas conferem-lhe uma propriedade importante, que pode ser usada como um elemento controlador de atividades ligadas ao ensino auxiliado por computador, facilitando a identificação e o tratamento de muitas nuances do aprendizado assistido.

Aplicando-se os conceitos de adaptatividade, é possível elaborar cursos dinâmicos, que se auto-modifiquem com base nas regras definidas pelo professor, pelas experiências e pelo nível de conhecimento individualizado dos alunos.

Busca-se, nesse contexto, aplicar formalismos adaptativos na representação de cursos dinâmicos para softwares educacionais. O dispositivo subjacente utilizado será a Máquina de Moore. Esse tipo de transdutor é um autômato finito com saída associada aos seus estados. Assim, no caso do modelo de um curso, para cada estado da Máquina de Moore pode-se associar um material didático correspondente, representado pela função de saída.

De tal modo, enquanto o mecanismo formal do transdutor permite que se especifique o sistema através de uma notação matemática rigorosa, o potencial dos dispositivos adaptativos permite ao aluno que utilize cursos baseados nesses formalismos uma experiência de aprendizagem individualizada, apresentando o material didático e seu roteiro de estudo adaptado ao nível de conhecimento e preferências do aprendiz.

1.1. JUSTIFICATIVAS

O uso de tecnologias computacionais tem sido importante no auxílio do aprendizado para inúmeras modalidades de ensino [PAPERT 1980, VALENTE 1993, GARRISON 1997, BRANSFORD et al. 1999], seja quando aplicada à educação infantil ou de nível superior, em programas regulares ou de educação continuada, no ensino presencial ou a distância.

Concomitantemente, o paradigma adaptativo vem apresentando significativos avanços nas pesquisas científicas e tecnológicas [LTA 2009]. São várias as aplicações potenciais que existem para os formalismos adaptativos, entre elas, a aplicação na educação, em especial, como ferramenta de auxilio na construção de soluções para o ensino por computador. Alguns trabalhos como [NETO 1993], [NETO 2001] e [RAMOS et al. 2009] citam esse potencial. Contudo, nenhum trabalho minucioso ainda havia sido realizado sobre este tema.

Sendo a educação das novas gerações uma das principais atividades de qualquer sociedade civilizada, e também uma das mais custosas e difíceis, é de extrema importância qualquer impacto positivo que novas tecnologias possam ter sobre os processos e métodos educacionais.

Assim, a importância de pesquisar e investir continuamente na melhoria do processo de ensino e aprendizagem, buscando-se novas alternativas pedagógicas, aliado ao potencial oferecido pela TA, como suporte ao desenvolvimento de softwares educacionais, são os fatores principais de motivação e justificativa para o desenvolvimento da presente pesquisa.

1.2. OBJETIVOS

A pesquisa desenvolvida nesta tese tem por objetivos gerais:

- ✓ Apresentar uma contribuição à área da adaptatividade com a formalização de um dispositivo adaptativo baseado em Máquina de Moore;
- ✓ Aplicar a adaptatividade, especialmente seus potenciais nos aspectos de aprendizagem e de tomada de decisão, contribuindo com o desenvolvimento de sistemas de ensino por computador, através de um modelo dinâmico capaz de tratar cursos e alunos tanto de forma individualizada como conjunta.

Como objetivos específicos, o presente trabalho pretende:

- ✓ Utilizar a formalização da Máquina de Moore Adaptativa no desenvolvimento de aplicações ao ensino auxiliado por computador.
- ✓ Elaborar de um modelo que permita a modelagem de cursos baseados em diferentes teorias de aprendizagem e para diferentes níveis de ensino.
- ✓ Possibilitar que o material didático elaborado para um determinado curso possa ser reaproveitado em outros cursos.
- ✓ Explorar o uso de adaptatividade no desenvolvimento de um ambiente baseado na plataforma *web* para apoio ao ensino.
- ✓ Implementar um sistema educacional adaptativo genérico, baseado no modelo proposto.
- ✓ Validar o modelo proposto através da realização de simulações e testes feitos a partir de um protótipo desenvolvido.

1.3. ORGANIZAÇÃO DO TRABALHO

No Capítulo 1 é feita a introdução deste trabalho, sendo apresentado um breve descritivo sobre formalismos adaptativos e suas aplicações, bem como são expostos os objetivos e as justificativas para a realização desta tese.

O Capítulo 2 conceitua a TA. Apresenta um levantamento bibliográfico sobre adaptatividade e sua evolução histórica. Expõe uma definição para dispositivos adaptativos e mostra diversas aplicações em que diferentes tipos de dispositivos adaptativos vem sendo desenvolvidos em outras pesquisas.

No Capítulo 3 são apresentados alguns conceitos e definições sobre educação e o uso de computadores como ferramenta pedagógica. Apresenta alguns modelos já desenvolvidos que serviram de base para a elaboração da proposta desta pesquisa.

No Capítulo 4 é apresenta a formulação para Máquina de Moore Adaptativa.

O Capítulo 5 apresenta como aplicar a adaptatividade na modelagem de cursos e exemplos elementares da proposta para demonstrar seu funcionamento.

Posteriormente, no Capítulo 6, são apresentadas as conclusões deste trabalho, juntamente com as contribuições obtidas e trabalhos futuros.

Por fim, na bibliografia é relacionado todo o material consultado durante a elaboração do trabalho e referenciado ao longo do texto.

2. TECNOLOGIA ADAPTATIVA

O termo tecnologia envolve o conhecimento técnico e científico, além de ferramentas, processos e materiais criados e/ou utilizados a partir de tal conhecimento. Para a palavra adaptatividade e outras similares, adotou-se, como definição, a propriedade que um modelo tem de modificar seu próprio comportamento, sem auxílio externo. Logo, TA refere-se às técnicas, métodos e disciplinas que estudam as aplicações da adaptatividade. Assim, TA compreende qualquer modelo de representação formal que tenha capacidade de mudar dinamicamente seu próprio comportamento, em resposta direta a um estímulo de entrada, sem qualquer intervenção externa. Num sistema definido por um conjunto de regras, a incorporação de algum mecanismo, através do qual possa se auto-modificar durante sua execução, o torna adaptativo.

De acordo com Neto [NETO 1993], a principal propriedade apresentada pela adaptatividade consiste no fato dela constituir um formalismo com regras dinamicamente variáveis, em contraste com a maioria dos modelos formais tradicionais, cujas regras, uma vez estabelecidas, permanecem imutáveis durante toda a sua operação.

A adaptatividade pode ser incorporada a qualquer tipo de dispositivo guiado por regras. O termo dispositivo é empregado aqui como alguma abstração formal, na qual o comportamento do dispositivo é descrito por um conjunto finito e explícito de regras. Tais regras especificam, partindo da situação em que se encontre o dispositivo, sua nova situação. Quando se cria uma camada adaptativa num dispositivo subjacente qualquer, tem-se, então, um dispositivo adaptativo guiado por regras, ou simplesmente, um dispositivo adaptativo [NETO 2001].

Conforme citado por Pistori [PISTORI 2003], o formalismo geral que caracteriza os dispositivos adaptativos foi apresentado pela primeira vez à comunidade científica em [NETO 2001], embora em formulações mais restritas venha sendo utilizado a mais tempo. Tal formulação fundamenta-se em um núcleo constituído por um dispositivo não-adaptativo dirigido por regras, acrescido de uma camada adaptativa que provê os recursos de um mecanismo adaptativo ao mesmo. Uma característica fundamental dos dispositivos adaptativos é a possibilidade de reaproveitar integralmente os formalismos consolidados, com aumento de seu poder de representação[1], ao custo de um pequeno acréscimo na sua complexidade formal [PISTORI 2003]. A adaptatividade surgiu da busca por um formalismo simples de usar, mas capaz de representar problemas complexos, envolvendo linguagens não-regulares e até mesmo dependentes de contexto [NETO 1993].

Mas, é necessário diferenciar um sistema **adaptativo** de um sistema **adaptável**. Enquanto a **adaptatividade** refere-se à propriedade de um sistema mudar suas próprias características espontaneamente, a **adaptabilidade** consiste na propriedade de um sistema permitir que um agente externo, como um usuário, personalize o sistema, alterando explicitamente certas características do mesmo, para adequá-lo às suas vontades e necessidades, normalmente dentro de um número limitado de possibilidades. Em um sistema adaptável, por exemplo, o usuário pode: configurar o layout das telas; escolher o padrão de cores; e, ocultar ou exibir funcionalidades.

Nas seções a seguir, são apresentadas: uma revisão da literatura; algumas aplicações de TA; uma formulação geral para dispositivos adaptativos; e, a formulação para Máquina de Moore Adaptativa.

[1] Não se aplica para Máquina de Turing.

2.1. EVOLUÇÃO HISTÓRICA DA ADAPTATIVIDADE

A presente seção traz uma compilação de importantes publicações baseada no levantamento realizado por Neto [NETO 2007], das principais contribuições encontradas na literatura nacional e internacional ao desenvolvimento dos conceitos e das aplicações ligadas à adaptatividade. Essa coleta de referências se apóia em outros resultados anteriores, publicados por autores como [IWAI 2000], [PISTORI 2003], [PARIENTE 2004], [SHUTT 1993] e [JACKSON 2006].

Embora vários trabalhos apresentem expressiva importância, seja pela relevância teórica ou pela aplicação prática, quando se fala em adaptatividade, dois trabalhos apresentados por Neto podem ser citados como marcos históricos:

- ✓ Em [NETO 1993], é apresentada a formalização de um dispositivo com conjunto dinâmico de regras e que opera como os autômatos de pilha estruturados, porém, com conjunto de transições que se modifica durante a operação do dispositivo. A esse formalismo deu-se o nome simplificado de autômato adaptativo. Nesse trabalho, são enfatizadas as aplicações dos autômatos de pilha estruturados adaptativos, utilizados na construção de reconhecedores sintáticos completos para linguagens de programação e de seus compiladores.

- ✓ Em [NETO 2001], foi apresentada uma formalização geral dos dispositivos adaptativos dirigidos por regras. Nesse artigo, foi proposta uma representação unificada e generalista para qualquer dispositivo adaptativo guiado por regras. É ilustrado no artigo um exemplo de tabela de decisão adaptativa. O modelo baseia-se na separação do mecanismo formal subjacente de seu mecanismo adaptativo. Até então, as notações utilizadas ainda não estavam devidamente adequadas para representar o formalismo adaptativo de uma forma simples e que fosse a mais

próxima possível da utilizada para a representação dos mecanismos formais não-adaptativos subjacentes originais.

Diversos outros trabalhos serviram de base e de referencial na concepção dos trabalhos supracitados. Por isso, alguns tópicos não diretamente relacionados com a adaptatividade, que têm ou tiveram papel marcante na evolução desse conceito, foram incluídos por sua importância histórica nesse processo, como é o caso dos autômatos de pilha estruturados e da extensibilidade. Também são apresentados alguns dos diversos trabalhos que surgiram após as publicações mencionadas acima.

2.1.1. AUTÔMATO DE PILHA ESTRUTURADO

Em 1981, no Brasil, Neto e Magalhães publicavam o primeiro artigo [NETO & MAGALHÃES 1981] sobre o formalismo dos autômatos de pilha estruturados, que viria a se tornar a base dos autômatos adaptativos, cuja idéia foi esboçada mais tarde em [NETO 1988] e consolidada em [NETO 1993] e [NETO 1994].

Os autômatos de pilha estruturados são dispositivos reconhecedores constituídos de uma família de submáquinas, cujas transições podem ser internas, como as de um autômato finito, ou então responsáveis pela movimentação entre submáquinas (chamadas e retornos). Nesse caso, uma pilha é utilizada, estritamente para memorizar estados de retorno, os quais são empilhados sempre que uma submáquina é chamada, para memorizar o estado para o qual ela deve retornar ao final da operação da submáquina assim acionada. Ao final da operação da submáquina, tal estado de retorno é desempilhado e utilizado para promover o prosseguimento da operação interrompida da submáquina chamadora.

A concepção dos autômatos de pilha estruturados surgiu dos diagramas separáveis de transições [CONWAY 1963], que consistem essencialmente de um conjunto de autômatos finitos mutuamente recursivos, que assim interligados têm poder computacional suficiente para o tratamento de linguagens livres de contexto. O trabalho de Conway [CONWAY 1963], de grande importância prática, carecia, no entanto, de uma fundamentação teórica explícita, o que motivou Lomet a publicar mais tarde seu artigo formalizando os diagramas separáveis de transições [LOMET 1973].

O estudo dos autômatos de pilha estruturados motivou Neto a publicar, em 1987, um livro introdutório sobre compiladores [NETO 1987], no qual descreve um método prático e eficiente de obtenção, a partir de gramáticas livres de contexto denotadas na Notação de Wirth, de autômatos de pilha estruturados, para uso como núcleos de compiladores dirigidos por sintaxe.

Em 1999, Neto, Pariente e Leonardi apresentam um artigo [NETO et al. 1999] que descreve cuidadosamente um processo de automatização desse método, indicando o algoritmo para a construção de um gerador automático de reconhecedores sintáticos para linguagens livres de contexto.

2.1.2. DEPENDÊNCIA DE CONTEXTO

A mais antiga menção, na literatura, a processos adaptativos é provavelmente a encontrada no trabalho que Di Fiorino publicou em 1963, no qual relata a idéia da utilização de um conjunto dinâmico de regras para a definição sintática de linguagens [DI FIORINO 1963]. Sua gramática, baseada em notações livres de contexto, é composta de produções ditas gerais, que formam um conjunto fixo, e de produções locais, que são construídas conforme a necessidade particular da sentença analisada.

O tema da dependência de contexto, encarada como sintaxe, levou a muitas discussões na literatura. Em 1980, McGettrick já apresentava em seu livro [McGETTRICK 1980], sobre a definição de linguagens de programação, uma distinção entre semântica estática e semântica dinâmica, esclarecendo que a dita semântica estática, assim como a sintaxe, corresponderia a uma atividade desenvolvida pelo compilador, enquanto a semântica dinâmica seria responsável pelas ocorrências durante a execução dos programas.

Pagan [PAGAN 1981], em seu livro sobre especificações formais de linguagens de programação, faz uma comparação de diversos formalismos conhecidos à época e, em particular, levanta novamente a discussão sobre a inadequação do termo semântica estática para designar dependências de contexto.

Em seu artigo de 1990, Meek [MEEK 1990] argumenta fortemente contra o uso desse termo e o aponta como evidência de uma forma incorreta de encarar um fenômeno genuinamente sintático, que é a dependência de contexto. Slonneger e Kurtz, em seu livro de 1995 sobre sintaxe e semântica das linguagens de programação [SLONNEGER & KURTZ 1995], tecem também considerações que reforçam essa posição. Em seu artigo sobre o uso de autômatos em engenharia de computação [NETO 2003], Neto também defende esse mesmo princípio, citando argumentos encontrados em [PAGAN 1981] e [SLONNEGER & KURTZ 1995].

Está cada vez mais disseminada a idéia de incluir nos formalismos que definem as linguagens dependentes de contexto informações mais completas que aquelas usualmente representadas apenas por seu componente livre de contexto. Assim sendo, a tendência é de que o tratamento das dependências de contexto seja cada vez menos atribuído a atividades semânticas, e que esse aspecto lingüístico seja reconhecido cada vez mais como sintaxe autêntica, reservando-se para a semântica aquilo que lhe é mais próprio, ou seja, os

aspectos que se referem estritamente à interpretação da linguagem e à dinâmica propriamente dita da execução dos programas.

2.1.3. EXTENSIBILIDADE

Extensibilidade é um importante conceito que se refere à capacidade de um sistema incorporar, de forma incremental, novas funcionalidades ou novos recursos, através de solicitação do usuário. Isso é realizado mediante o fornecimento de definições para as extensões desejadas, indicando-se para tanto a forma como tais extensões operam, em função da combinação dos recursos já disponíveis. Não se trata propriamente de uma manifestação de adaptatividade, visto que esta é um fenômeno que ocorre estritamente em tempo de execução, enquanto a extensibilidade é tratada tipicamente no momento da compilação.

Pode-se considerar que a extensibilidade de linguagens de programação seja uma importante manifestação na pesquisa da área, sendo tais linguagens anteriores ao aparecimento de trabalhos explicitamente relacionados com o fenômeno da adaptatividade.

Bastante popular entre 1960 e 1975, o conceito de extensibilidade para linguagens de programação prenunciou aquilo que mais tarde viria a se materializar na forma de adaptatividade. Com a extensibilidade, o programador pode, durante a codificação, modificar as capacidades da sua linguagem de programação, adequando-a a seus propósitos particulares, tanto em matéria de aspecto externo quanto de funcionalidade.

Essa tendência sofreu um impacto negativo com o advento das novas idéias acerca da maneira de programar, resultantes das pesquisas iniciadas nos anos 1970, tendo como conseqüência uma abrupta redução dos esforços da

comunidade nas pesquisas na área. Em [STANDISH 1975] é feita uma documentação histórica dos trabalhos mais importantes previamente publicados sobre o assunto.

Em 1970, em Harvard, Wegbreit publica um relatório técnico sobre linguagens de programação extensíveis [WEIGBREIT 1970]. Outros trabalhos significativos nessa área foram o de Hanford e Jones, sobre sintaxe dinâmica [HANFORD & JONES 1974], e o de Cabasino, sobre linguagens evolutivas e *parsers* dinâmicos [CABASINO et al. 1992].

Posteriormente, ocorreu o aparecimento do conceito de programação extensível, no qual a idéia da extensibilidade é retomada, porém, agora com algumas regras mais restritivas para sua utilização, que exigem, entre outros, a existência não apenas de uma sintaxe flexível, mas também que todos os ambientes de compilação, de desenvolvimento, de depuração e de execução também respeitem essa exigência. Diversas publicações surgiram nesse sentido. Entre outras, podem-se citar duas, apenas a título de ilustração dessas tendências: em 2001, Bacharach e Playford propõem um extensor sintático para a linguagem Java, de certa maneira resgatando valores das linguagens extensíveis [BACHARACH & PLAYFORD 2001], e em 2002, Carmi publica um artigo sobre a biblioteca Adapser [CARMI 2002], projetada para operar como ambiente básico para o desenvolvimento de analisadores sintáticos adaptativos LALR(1), que permitam o tratamento de linguagens extensíveis, ao contrário do que ocorre com os geradores automáticos usuais de analisadores sintáticos.

2.1.4. AUTÔMATOS DE TOPOLOGIA VARIÁVEL

Em 1967, na antiga União Soviética, Agasandjan publicou uma nota de duas páginas, que apresenta a idéia de um tipo de autômatos com estrutura variável, que se poderia comparar, em funcionamento e estrutura, a uma forma

embrionária de dispositivo adaptativo da classe dos autômatos [AGASANDJAN 1967]. Na mesma linha, Salomaa apresenta, em 1968, em trabalho teórico extenso, uma classe de autômatos finitos com estrutura variável no tempo [SALOMAA 1968].

Em 1981, no Brasil, Neto e Magalhães publicavam o primeiro artigo sobre o formalismo dos autômatos de pilha estruturados [NETO & MAGALHÃES 1981], os quais foram usados mais tarde como base para os autômatos adaptativos. Esse trabalho constituiu o alicerce de uma prática de ensino da construção de compiladores dirigidos por sintaxe, cujos reconhecedores sintáticos subjacentes são os autômatos de pilha estruturados.

Em 1986, Krithvasan publica seu primeiro trabalho formalizando os autômatos finitos variantes no tempo [KRITHVASAN 1986], em continuidade às idéias precursoras publicadas por Agasandjan [AGASANDJAN 1967] e por Salomaa [SALOMAA 1968]. Em 1988, Krithvasan estende esses estudos, formalizando de maneira similar os autômatos de pilha variantes no tempo [KRITHVASAN 1988].

Em 1988, Neto apresentou uma publicação [NETO 1988] das primeiras idéias do princípio de funcionamento do formalismo reconhecedor, que viria a ser chamado mais tarde de autômato adaptativo. Motivado por problemas de reconhecimento sintático de linguagens de programação, esse artigo apresenta as primeiras idéias do uso de automodificação do autômato como forma de trabalhar sintaticamente elementos de dependências de contexto de uma linguagem de programação, tais como a verificação de tipos e a garantia da utilização exclusiva de variáveis que tenham sido previamente declaradas.

Posteriormente, em 1993, tal idéia viria a permear todo o conteúdo da tese de Neto [NETO 1993], na qual, em extensivo estudo, consolida a formalização de um dispositivo com conjunto dinâmico de regras e que opera como os

autômatos de pilha estruturados, porém, com conjunto de transições que se modifica durante a operação do dispositivo. A esse formalismo deu-se o nome simplificado de autômato adaptativo. Além da apresentação do formalismo adaptativo, apresenta-se também uma técnica para a implementação automática de reconhecedores sintáticos livres de contexto, usando-se autômatos de pilha estruturados.

Mostra-se ainda como construir autômatos adaptativos que unificam o tratamento dos aspectos léxicos, sintáticos livres de contexto e sintáticos dependentes de contexto.

Para isso, diversos aspectos da compilação de linguagens de programação imperativas são extensivamente tratados com o emprego do formalismo proposto: análise léxica, declarações e utilização de nomes, escopos aninhados, declaração e verificação de tipos, definição e uso de macros paramétricas etc.

Tabelas de símbolos e tabelas de palavras reservadas são eliminadas e substituídas por mecanismos adaptativos equivalentes. Esse trabalho constitui uma referência bastante completa acerca do formalismo dos autômatos adaptativos e da sua aplicação à implementação de linguagens de programação.

Em 1994, Neto publicou na ACM SIGPLAN Notices uma pequena síntese do formalismo dos autômatos adaptativos apresentado em sua tese, síntese essa que deu a conhecer à comunidade internacional o autômato adaptativo [NETO 1994]. Ainda em 1994, no Worcester Polytechnic Institute, Shutt complementa o trabalho iniciado para gramáticas em sua tese [SHUTT 1993], passando então a abranger formalismos de autômatos.

Rubinstein e Shutt publicam então dois artigos, nos quais apresentam os seus autômatos finitos automodificáveis [RUBINSTEIN & SHUTT 1994]. Logo em seguida, em 1995, publicam outro artigo, introdutório, sobre os mesmos autômatos finitos automodificáveis [RUBINSTEIN & SHUTT 1995].

Em 1996, Saitou e Jakiela escrevem um artigo em que estudam os autômatos automontáveis unidimensionais [SAITOU & JAKIELA 1996], dispositivos que também exibem adaptatividade e que são criados dinamicamente através da montagem incremental do autômato desejado a partir de um conjunto finito de partes acopláveis disponíveis.

Em continuidade à tese de Neto, sobre métodos de construção de compiladores para linguagens de programação, Pereira apresenta em 1999, em sua dissertação de mestrado [PEREIRA 1999], uma ferramenta que oferece a seus usuários recursos para a criação e o ensaio de autômatos finitos, de pilha estruturados e adaptativos. Essa ferramenta proporcionou recursos para a definição formal da sintaxe de linguagens livres de contexto e a produção automática de autômatos de pilha estruturados a partir dessa formalização da linguagem, ou seja, com ela tem-se um gerador automático de reconhecedores sintáticos para linguagens livres de contexto.

Outra face da ferramenta é seu aspecto adaptativo, omitido no tratamento de linguagens regulares e livres de contexto, mas presente na ferramenta, de forma que, havendo necessidade de recursos para o tratamento de dependências de contexto, toda a potência da ferramenta e do formalismo dos autômatos adaptativos pode ser posta em ação.

Em 2000, Rocha e Neto apresentaram um estudo conceitual dos autômatos adaptativos, explorando sua aplicabilidade teórica [ROCHA & NETO 2000]. Em particular, esse artigo contém uma demonstração da abrangência dos autômatos adaptativos, provando sua equivalência com as máquinas de Turing.

Em 2001, Neto publicou um texto que teve um papel muito significativo para a área, já que apresenta uma forma generalizada para o fenômeno da adaptatividade, através de uma proposta de formulação geral para os dispositivos formais adaptativos [NETO 2001].

Segundo essa proposta, todos os dispositivos adaptativos cuja operação possa ser considerada uma alteração dinâmica do conjunto de regras que a definem podem ser expressos de maneira uniforme usando-se o formalismo dos dispositivos adaptativos dirigidos por regras, aí apresentados.

Essa arquitetura geral consiste em acrescentar, a um dispositivo subjacente não-adaptativo de qualquer natureza uma camada adaptativa, responsável pela parte dinâmica do conjunto de regras que definem o dispositivo.

Em 2003, Pistori, em sua tese de doutorado, faz uma resenha bastante cuidadosa dos progressos até então atingidos na área, além de dar diversas outras contribuições técnicas ao campo da adaptatividade e suas aplicações. Como algumas das contribuições apresentadas em seu trabalho [PISTORI 2003], o autor define formalmente o conceito de autômato finito adaptativo, propõe diversas simplificações na notação dos autômatos adaptativos e elimina algumas redundâncias de propostas anteriores.

Em 2003, Pistori e Neto publicam uma documentação sobre a ferramenta AdapTools [PISTORI & NETO 2003], desenvolvida como parte da tese de Pistori [PISTORI 2003], que tem sido usada como uma das principais referências para a utilização da ferramenta.

Em 2006, Keung e Tyagi publicam um artigo [KEUNG & TYAGI 2006] contendo a descrição do conceito de reconfigurabilidade do espaço de estados e usam esse conceito em uma proposta de implementação física eficiente dos

autômatos finitos automodificáveis, de Rubinstein e Shutt [RUBINSTEIN & SHUTT 1994].

2.1.5. GRAMÁTICAS DEPENDENTES DE CONTEXTO

Em 1965, Van Wijngaarden dava início à sua importante contribuição apresentando pela primeira vez as chamadas gramáticas de dois níveis, ou gramáticas W, nome dado em sua homenagem, com a publicação do seu artigo sobre o projeto ortogonal de linguagens de programação [WIJNGAARDEN 1965]. O autor propõe, nessa publicação, um novo tipo de gramática, através da qual podem ser aplicadas algumas diretrizes inovadoras para o projeto e descrição formal de linguagens de programação, centradas no conceito da ortogonalidade[2] dos componentes das linguagens de programação.

A ortogonalidade foi acompanhada de outra proposta, relativa à maneira de descrever gramaticalmente a linguagem em questão: em lugar das clássicas gramáticas que definem as linguagens através de conjuntos fixos de regras, usa-se uma metagramática, que permite gerar novas regras “sob medida” para cada situação, as quais vão sendo incorporadas ao conjunto de regras já existentes, completando-as de acordo com a necessidade de cada texto em análise.

Mais tarde, tais gramáticas de dois níveis seriam utilizadas para a especificação formal da linguagem de programação Algol 68, consagrada entre os mais significativos marcos da evolução das linguagens de programação [WIJNGAARDEN 1976].

---

[2] Em computação, a ortogonalidade implica em um conjunto de primitivas que podem se combinar para construir as estruturas de uma linguagem.

A contribuição de Van Wijngaarden foi importantíssima como pioneira na representação estritamente sintática de fenômenos lingüísticos dependentes de contexto, através de métodos gramaticais generativos.

Alguns anos após o trabalho de Wegbreit sobre linguagens de programação extensíveis [WEGBREIT 1970], foi publicado em 1974 o trabalho de Hanford e Jones sobre sintaxe dinâmica [HANFORD & JONES 1974], formalizado como uma aplicação das idéias ao contexto da definição sintática de linguagens de programação com características extensíveis, reforçando assim a tendência ao aparecimento de uma forma gramatical diferente daquela representada pelas gramáticas de dois níveis, para a formalização de dependências de contexto em linguagens de programação.

Pouco depois, em 1976, foi publicado o relatório revisado da linguagem Algol 68 [WIJNGAARDEN 1976], no qual os autores exploram ao limite os recursos oferecidos pelas gramáticas W, de dois níveis, e aplicam ao projeto dessa linguagem o inovador e importante conceito de ortogonalidade, ambos previamente apresentados pelo autor em [WIJNGAARDEN 1965].

Em 1990, diversos artigos foram publicados em torno da adaptatividade em âmbito gramatical. Em abril desse ano, Meek publicou seu trabalho [MEEK 1990] centrado na representação das dependências de contexto encontradas nas linguagens de programação.

No artigo de maio de 1990, Burshteyn discute o conceito e a prática de alterar a gramática da linguagem de programação durante a análise dos programas nela codificados [BURSHTEYN 1990 a]. Em um segundo trabalho, de novembro do mesmo ano, Burshteyn detém-se em considerações sobre a geração de linguagens formais por gramáticas modificáveis e no reconhecimento sintático de programas denotados nas linguagens por elas formalizadas [BURSHTEYN 1990 b].

Esses dois trabalhos muito influenciaram os desenvolvimentos realizados na época, propiciando a compreensão da utilidade de formalismos automodificáveis em aplicações voltadas à especificação das dependências de contexto encontradas nas linguagens de programação.

Ainda em 1990, Christiansen publicou um excelente trabalho [CHRISTIANSEN 1990], motivado pelos artigos de Burshteyn [BURSHTEYN 1990 b] e Meek [MEEK 1990], publicados anteriormente no mesmo ano. Já na introdução de seu artigo, comenta a impropriedade terminológica do trabalho de Meek e propõe o uso das técnicas de Burshteyn como forma prática para representar e implementar linguagens contendo dependências de contexto. Em adição, coleta diversas outras publicações então conhecidas sobre gramáticas adaptáveis e tece comentários sobre as mesmas.

Em 1992, Cabasino, Paolucci e Todesco publicam um artigo sobre gramáticas evolutivas e analisadores sintáticos dinâmicos, contribuindo mais uma vez para a evolução da idéia da adaptatividade em formalismos gramaticais e tecendo considerações acerca da sua realização prática [CABASINO et al, 1992].

Em 1993, Shutt publica sua dissertação de mestrado, sobre gramáticas adaptáveis recursivas [SHUTT 1993], outra contribuição para esses formalismos gramaticais dependentes de contexto, trabalho muito influenciado pelos conceitos de automodificação, contidos nos formalismos similares publicados até então.

No ano de 1994, Boullier publica mais um trabalho sobre formalismos gramaticais dinâmicos [BOULLIER 1994], com os quais o autor propõe uma forma de efetuar não apenas a descrição sintática de uma linguagem de programação, como também a verificação de tipos e outros aspectos das dependências de contexto usualmente encontradas, chegando a ilustrar o uso

de sua proposta para a resolução de aspectos bastante complexos, tais como os polimorfismos, tipos derivados, overloading, e outras formas não-triviais de dependências de contexto, freqüentes nas linguagens modernas de programação.

Em 2000, Iwai publica sua tese [IWAI 2000], na área dos formalismos gramaticais adaptativos, até então inexplorados no Brasil. Em sua contribuição, propõe a formalização de linguagens dependentes de contexto através das gramáticas adaptativas, que utilizam como base gramáticas livres de contexto, às quais se adicionam, como forma principal de incremento do poder de expressão, mecanismos adaptativos de alteração do seu próprio conjunto de produções.

Adicionalmente, para simplificar o tratamento das dependências de contexto, as gramáticas adaptativas também lançam mão de símbolos de contexto, que podem ser associados dinamicamente ao texto reconhecido, em função da identificação de situações particulares de dependência de contexto na sintaxe das cadeias analisadas.

Aycock publica, em 2003, seu artigo de apoio ao uso de um formalismo que denominou gramáticas dinâmicas generativas, e que propõe como modelo de computação [AYCOCK 2003]. Com essa proposta, o autor unifica diversas áreas diferentes de aplicação por meio do emprego desse modelo computacional comum, de caráter gramatical.

Em 2004, Pariente publica sua tese sobre as gramáticas adaptativas com verificação de aparência [PARIENTE 2004]. Nesse trabalho, é feita uma compilação extensiva de diversos tipos de gramáticas com conjuntos variáveis de regras e são estabelecidos entre elas diversos vínculos e comparações.

Em 2006, Jackson publica seu livro sobre adaptatividade e dependências de contexto em análise sintática [JACKSON 2006], que inclui informações sobre uma ferramenta de auxílio ao uso das gramáticas adaptativas denominadas gramáticas-§ para a representação de linguagens com dependências de contexto. Trata-se de um livro publicado que aborda de forma mais extensiva e abrangente esse assunto da representação das dependências de contexto e seu tratamento computacional através de técnicas adaptativas.

2.1.6. OUTROS FORMALISMOS ADAPTATIVOS

Logo após a publicação da tese em que foram apresentados os autômatos adaptativos [NETO 1993], iniciaram-se diversas pesquisas em torno do tema da adaptatividade, nas quais se procurou transpor as idéias já utilizadas com autômatos para outros tipos de dispositivos formais.

Como primeiro resultado completo nesse sentido, Almeida publicou, em 1995, sua tese sobre *statecharts* adaptativos [ALMEIDA 1995], incorporando adaptatividade ao formalismo clássico dos *statecharts*, de Harel [HAREL 1987]. Apresentou ainda, como complemento, uma ferramenta visual para a criação e simulação de *statecharts* adaptativos.

Em 1997, Santos acrescentou extensões, dotando com recursos mais expressivos a teoria e a ferramenta apresentadas por Almeida, para permitir que fossem ensaiados aspectos de sincronização explícita entre *statecharts*. Isso se fez através da adição de redes de Petri aos *statecharts*, para a explicitação do sincronismo entre eventos.

Do mesmo modo que em Almeida [ALMEIDA 1995], da pesquisa de Santos [SANTOS 1997] também resultou uma ferramenta visual para a definição e ensaio dos seus statecharts adaptativos sincronizados.

Outra publicação, que teve um papel muito importante como forma de demonstração prática da aplicabilidade da adaptatividade, foi materializada em 1999, no artigo de Neto e Basseto sobre composição musical automática, baseada em algoritmos adaptativos [BASSETO & NETO 1999]. O formalismo matemático em que tais algoritmos se baseiam é a rede de Markov adaptativa, uma espécie de autômato finito estocástico de topologia completa, a cujas transições são associadas probabilidades, sendo estas modificadas dinamicamente de acordo com as transições realizadas. Famílias de tais dispositivos podem ser usadas, permitindo-se que cada um deles possa modificar, segundo regras bem estabelecidas, as probabilidades associadas às próprias transições e também às dos outros dispositivos com que operam.

Um programa muito simples, denominado Lassus [BASSETO 1999], elaborado com base nesse princípio, ilustra a eficácia dos dispositivos adaptativos como mecanismos inteligentes a serem usados na elaboração computacional de peças musicais, segundo regras determinadas.

Em 2000, Basseto publica sua dissertação de mestrado [BASSETO 2000], na qual são formalizadas as redes de Markov adaptativas. Nesse trabalho, o autor explora diversos aspectos da automatização da composição musical, utilizando como linguagem de especificação uma gramática sensível ao contexto, e implementando-a com redes de Markov adaptativas, abrindo caminho para muitos desdobramentos na área da utilização do computador como coadjuvante nesse tipo de processo da criação artística.

Em 2001, ao lado da apresentação da teoria geral dos dispositivos adaptativos dirigidos por regras [NETO 2001], Neto ilustra os conceitos aí definidos com a apresentação das tabelas de decisão adaptativas como dispositivo adaptativo obtido pela incorporação de mecanismos adaptativos a um formalismo subjacente representado pelas tabelas de decisão tradicionais, o que viria mais

tarde a mostrar-se útil na formulação de sistemas adaptativos de tomada de decisões [PEDRAZZI et al. 2005], [PISTORI 2003].

Em 2006, Pistori, Neto e Pereira publicam um artigo sobre árvores de decisão adaptativas não-determinísticas [PISTORI et al. 2006], muito adequadas para a formulação de tomadas de decisão do tipo diagnóstico inteligente, inferido a partir de sintomas.

Em 2007, Camolesi [CAMOLESI 2007] publica sua tese sobre um meta-ambiente para modelagem de aplicações usando TA. Nela são apresentados alguns dispositivos adaptativos, entre eles o *Interaction System Design Language* (ISDL) adaptativo, capaz de especificar sistemas distribuídos.

2.2. DISPOSITIVO ADAPTATIVO DIRIGIDO POR REGRAS

Historicamente, dispositivos adaptativos dirigidos por regras, ou simplesmente dispositivos adaptativos, emergem do campo de linguagens formais e autômatos. Os métodos formais permitem que se especifique, desenvolva e verifique um sistema computacional através da aplicação de uma notação matemática rigorosa.

Um dispositivo não-adaptativo dirigido por regras, pode vir a ser qualquer máquina formal cujo comportamento dependa exclusivamente de um conjunto finito de regras, que determinem, para cada possível configuração corrente do dispositivo, a sua próxima configuração. Já, um dispositivo formal é dito adaptativo sempre que seu comportamento puder modificar-se dinamicamente, como uma resposta espontânea aos estímulos de entrada que o alimentam. Essa alteração deve se dar sem que haja qualquer interferência de agentes externos, inclusive de usuário. Para que isso possa acontecer, os dispositivos adaptativos devem ser, portanto, automodificáveis.

O mecanismo adaptativo tem dois elementos: as funções adaptativas e as ações adaptativas. Conceitualmente, as funções adaptativas são relacionadas a especificação, enquanto as ações adaptativas são relacionadas a execução. Em outras palavras, funções adaptativas consistem de um conjunto de declarações de funções paramétricas, as quais especificam, de uma forma semelhante ao que ocorre com as declarações de sub-rotinas e funções em uma linguagem de programação usual. Já as ações adaptativas dizem respeito as modificações a serem impostas ao conjunto de regras.

Programam-se as modificações a serem impostas ao conjunto de regras através de um conjunto de ações adaptativas elementares. Uma ação adaptativa elementar indica uma dentre três tipos de primitivas: consultas, exclusões e inclusões.

- ✓ As ações adaptativas elementares de consulta permitem inspecionar o conjunto de regras que definem o dispositivo, em busca de regras que sejam aderentes a algum um padrão fornecido.
- ✓ Ações elementares de exclusão permitem remover do conjunto de regras qualquer regra aderente ao padrão fornecido.
- ✓ Ações de inclusão permitem especificar a adição de uma ou mais novas regras, de acordo com um padrão fornecido.

Quando uma função adaptativa é executada, todas as ações elementares de consulta são executadas em primeiro lugar, seguidas das ações de exclusão e, por último, são efetuadas todas as ações de inclusões.

A segunda parte refere-se à associação de, no máximo, duas ações adaptativas a regras selecionadas do dispositivo subjacente. Uma dessas ações adaptativas é especificada para ser executada **antes** que a regra à qual está associada seja aplicada. A segunda é executada **após** a aplicação da regra. Ambas são opcionais. Portanto, regras que não estejam associadas a ações adaptativas comportam-se como simples regras não-adaptativas.

A figura 1 apresentada o esquema geral para Dispositivos Adaptativos. Na figura, é possível ver uma cadeia de entrada e o comportamento inicial do dispositivo. Para se consumir um símbolo de entrada da cadeia, a camada adaptativa do dispositivo é acionada, realizando, opcionalmente, ações adaptativas antes de executar as regras do dispositivo subjacente. Em seguida, são executadas as regras do próprio dispositivo subjacente. Após a execução das regras do dispositivo subjacente, ações adptativas posteriores podem ser executadas. E, finalmente, um novo comportamento do dispositivo pode ser observado.

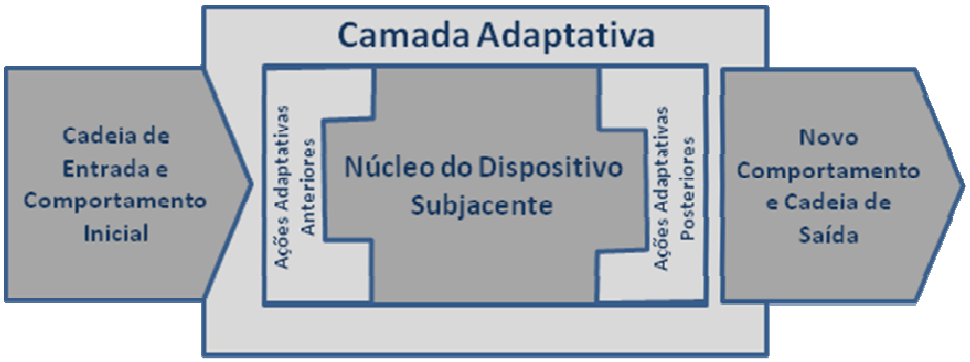

Figura 1: Esquema de um Dispositivo Adaptativo.

Em outras palavras, um dispositivo adaptativo pode ser obtido pela incorporação de ações adaptativas às regras da formulação subjacente, de tal modo que, sempre que alguma delas for aplicada, a ação adaptativa associada é acionada, causando as devidas alterações correspondentes no conjunto de regras do dispositivo não-adaptativo subjacente. Porém, quaisquer possíveis alterações no comportamento do dispositivo devem ser plenamente conhecidas *a priori*, em quaisquer etapas de sua operação em que tais alterações de comportamento devam se efetivar.

A principal característica dessa formulação é que ela apresenta a desejável propriedade de preservar a natureza do formalismo não-adaptativo, de tal forma que o dispositivo adaptativo resultante possa ser facilmente compreendido por todos os que tiverem familiaridade com o formalismo não-adaptativo subjacente original.

A seguir, é apresentada uma classificação de dispositivos adaptativos, e, posteriormente, uma série de aplicações de tais dispositivos.

2.2.1. CLASSIFICAÇÃO DE DISPOSITIVOS ADAPTATIVOS

Os formalismos subjacentes dos dispositivos adaptativos podem ser classificados em variadas categorias, conforme sua forma de operação. Entre outras, têm-se as seguintes, definidas em [NETO 2007]:

- ✓ **dispositivos de reconhecimento**, da classe dos autômatos, baseados na sucessão de mudanças de estados;

- ✓ **dispositivos de processamento**, como as linguagens de programação adaptativas, que permitem descrever a lógica de programas com código automodificável;

- ✓ **dispositivos de geração**, da classe das gramáticas, baseados na aplicação sucessiva de regras de substituição;

- ✓ **dispositivos para a representação de sistemas assíncronos**, tais como os statecharts, que incorporam mecanismos responsáveis pela representação de fenômenos de sincronização;

- ✓ **dispositivos estocásticos**, como as redes de Markov, capazes de representar fenômenos de caráter aleatório;

- ✓ **dispositivos de auxílio à tomada de decisões**, representados principalmente pelas tabelas de decisão e pelas árvores de decisão;

Dentre as categorias citadas acima, há diversos trabalhos já desenvolvidos e outros em curso. A seção a seguir apresenta diversas aplicações de dispositivos adaptativos agrupadas por tipo de aplicação.

2.2.2. APLICAÇÕES DE DISPOSITIVOS ADAPTATIVOS

Diversas aplicações potenciais existem para a tecnologia derivada da adaptatividade, incluindo: inferência, arte usando computador, processamento de linguagem natural, síntese de voz, reconhecimento de padrões, tomada de decisão, linguagens de programação adaptativas, otimização de código, metamodelagem, computação evolutiva, engenharia de software, robótica e segurança. A seguir, são expostas diversas aplicações dos formalismos adaptativos associados a diversas classes de formalismos subjacentes:

2.2.2.1. INFERÊNCIA

Em 1988, Neto e Iwai publicaram um primeiro artigo sobre inferência [NETO 1998], no qual é mostrado um método, empregando autômatos adaptativos, que, partindo de amostras positivas e negativas de uma linguagem regular, é capaz de inferir um autômato finito que representa uma boa aproximação do autômato que aceita a linguagem em questão. Inspirada nesse trabalho, e dando continuidade ao mesmo, em 2006, Matsuno publica, em sua dissertação de mestrado [MATSUNO 2006], os resultados obtidos aplicando a TA à inferência de linguagens regulares e livres de contexto.

2.2.2.2. ARTE USANDO O COMPUTADOR

Nos anos de 1999 e 2000, Basseto cria, usando TA baseada em redes de Markov, uma aplicação muito interessante, na área da geração automática de música por computador, criada com base em regras bem estabelecidas [BASSETO & NETO 1999], [BASSETO 2000], [BASSETO 1999].

2.2.2.3. PROCESSAMENTO DE TEXTOS EM LINGUAGEM NATURAL

Em 2000, a área do processamento de linguagens naturais ganha as contribuições de [MENEZES 2000], com a criação de um método adaptativo de etiquetação automática de textos em língua natural. Essa pesquisa apresentou, como um de seus produtos, um método, baseado em autômatos adaptativos, de análise morfológica para a língua portuguesa.

Em um protótipo, o autor desenvolveu um coletor de palavras usando um autômato adaptativo enriquecido com mecanismos de avaliação da freqüência de utilização dos caminhos, de forma que, dinamicamente, o dispositivo efetua suas buscas percorrendo o caminho até então mais utilizado, ou seja, o de maior freqüência constatada.

O dispositivo explora a idéia de implementar o classificador para operar em duas etapas: a primeira, de aprendizagem, feita sobre um corpus etiquetado, no qual as palavras e as respectivas etiquetas são coletadas e aprendidas; e a segunda, de utilização dos fatos coletados, na qual um texto não etiquetado é analisado e suas palavras, classificadas de acordo com o que tenha sido aprendido na fase anterior.

Em 2001, Taniwaki apresenta sua dissertação [TANIWAKI 2001], na qual mostra a viabilidade da representação de fenômenos lingüísticos da linguagem natural, usando para isso formalismos adaptativos. Mostra ainda a equivalência entre tais formalismos e outros tradicionalmente empregados na análise e processamento de linguagem natural, tais como ATN e DCG.

Após a experiência de Menezes [MENEZES 2000], optou-se por investir na pesquisa da parte sintática das linguagens naturais, em particular, nos seus aspectos dependentes de contexto, e como primeiro resultado nessa direção, Neto e Moraes publicaram, em 2003, um material [NETO & MORAES 2003]

apresentando uma técnica de descrição de dependências de contexto usando como base o mecanismo formal oferecido pelas gramáticas adaptativas, propostas por Iwai em [IWAI 2000].

2.2.2.4. SÍNTESE DE VOZ

Em 2004, é publicado também um artigo sobre a aplicação da TA à síntese de voz [ZUFFO & PISTORI 2004]. Esse foi um dos primeiros desdobramentos de um dos experimentos conduzidos na tese de Pistori [PISTORI 2003], e utiliza autômatos adaptativos para escolher a composição apropriada de fonemas prégravados, com base em regras de articulação vocal aplicadas a textos escritos em português. Em [CAYA & SAPATA 2009] é apresentada uma proposta de um sintetizado de voz para idioma espanhol.

2.2.2.5. RECONHECIMENTO DE PADRÕES

Em 2002, Costa, Hirakawa e Neto publicam um artigo sobre o uso de mecanismos baseados em autômatos adaptativos no reconhecimento automático de padrões geométricos, usando pela primeira vez essa tecnologia em robótica [COSTA et al. 2002].

Segue-se a publicação [PISTORI 2003], explorando em profundidade essa aplicação, orientando-a para padrões mais complexos e variáveis e apresentando diversas aplicações da TA: reconhecimento de formas, síntese de voz e elaboração de interfaces humano-computador.

O projeto SIGUS [SIGUS 2007], originou-se do tema do artigo publicado em 2004 por Pistori e Neto [PISTORI 2004] sobre os primeiros resultados do uso

de TA para o reconhecimento de padrão, em particular, correspondente à identificação dos símbolos da linguagem de sinais.

2.2.2.6. TOMADA DE DECISÃO

Após a publicação, em 2001, do artigo de Neto, apresentando as tabelas de decisão adaptativas, outros trabalhos surgiram explorando a adaptatividade na tomada de decisões. Assim, em 2003, Pistori desenvolveria, a partir das tabelas de decisão adaptativas, os conceitos de árvores de decisão adaptativas, as quais, pela aderência conceitual a certos tipos de tomadas de decisão, tornam-se um formalismo muito interessante para uso nessas aplicações [PISTORI 2003].

Em 2005, Pedrazzi, Tchemra e Rocha contribuíram com um artigo sobre o uso de tabelas de decisão adaptativas em aplicações à resolução de problemas de tomada automática de decisão [PEDRAZZI et al. 2005].

No artigo de 2006 de Pistori, Neto e Pereira [PISTORI et al. 2006], são conceituadas as árvores de decisão adaptativas e ilustra-se sua aplicação à tomada de decisão em diagnósticos automáticos com base em sintomas.

Em 2009, apresenta sua tese de doutorado [TCHEMRA 2009] sobre tabela de decisão adaptativa na tomada de decisão multicritério.

2.2.2.7. LINGUAGENS PARA PROGRAMAÇÃO ADAPTATIVAS

Em 2003, Rocha e Neto publicam um primeiro trabalho sobre a inclusão de características de adaptatividade em linguagens de programação [ROCHA &

NETO 2003], em particular, linguagens funcionais, mostrando a viabilidade de tal empreendimento através da apresentação de uma possível arquitetura.

Em 2006, Freitas e Neto contribuem com um artigo sobre linguagens de programação adaptativas [FREITAS & NETO 2006] e nesse trabalho discutem o novo estilo de programação daí derivado.

Em 2009, Pelegrini apresenta sua disssertação de mestrado [PELEGRINI 2009] sobre códigos adaptativos e linguagens de programação adaptativas.

2.2.2.8. OTIMIZAÇÃO DE CÓDIGO

Em 2004, Luz publica em sua dissertação de mestrado [LUZ 2004] uma aplicação da TA à otimização de código em compiladores, empregando uma forma de otimização do tipo *peephole*, aperfeiçoada pela inclusão da adaptatividade nos algoritmos de otimização utilizados.

2.2.2.9. META-MODELAGEM

Em 2004, Camolesi e Neto publicam as primeiras idéias da modelagem adaptativa [CAMOLESI 2004] e, com base nelas, Camolesi apresenta sua tese de doutorado [CAMOLESI 2007], na qual a tônica é a proposta de um metambiente, materializada por meio de uma ferramenta com interface visual, destinada a oferecer ao usuário meios para a construção automática de ambientes completos para a definição, formalização, simulação e ensaios de dispositivos formais, adaptativos ou não, especificados pelo próprio usuário.

2.2.2.10. COMPUTAÇÃO EVOLUTIVA

Em 2005, Pistori, Martins e Castro publicaram um artigo confrontando algoritmos genéticos com autômatos finitos adaptativos, em uma aplicação na qual foram comparadas e correlacionadas a adaptação dos indivíduos e a evolução da população [PISTORI et al. 2005].

Em 2007, é publicado um artigo em que se aplica a adaptatividade para simular algoritmos genéticos, como parte de um projeto envolvendo a aplicação da TA ao estudo da biodiversidade [BRAVO et al. 2007].

2.2.2.11. ENGENHARIA DE SOFTWARE

Também em 2005, Silva e Neto apresentaram um artigo [SILVA & NETO 2005] propondo um esquema para o projeto de linguagens para a especificação de software, que corresponde a uma primeira aproximação entre a TA e a engenharia de software.

2.2.2.12. ROBÓTICA

Em continuidade ao trabalho de Costa, Hirakawa e Neto em 2002, sobre reconhecimento de padrões geométricos [COSTA et al. 2002], no mesmo evento de 2005, Souza e Hirakawa publicam [SOUZA & HIRAKAWA 2005] outra contribuição da adaptatividade à robótica, envolvendo o mapeamento automático e a navegação em ambientes desconhecidos usando como formalismo de representação os autômatos adaptativos.

2.2.2.13. SEGURANÇA (SECURITY)

Em [PELEGRINI & NETO 2007] é proposto o uso de dispositivos adaptativos para segurança de dados, tendo um transdutor como dispositivo subjacente. Em [CEREDA & ZORZO 2008] propõe-se um controle de acesso através de autômato adaptativo. Ainda em 2008, Cereda apresenta, em sua dissertação de mestrado [CEREDA 2008], um modelo de controle de acesso adaptativo.

2.2.2.14. ENSINO POR COMPUTADOR

Em diversos trabalhos, como [NETO 1993], [NETO 2001] e [RAMOS et al. 2009], é citado o potencial da TA como ferramenta de auxilio na construção de soluções para o ensino por computador. Em janeiro de 2009, é apresentado por Dizeró e Neto [DIZERO & NETO 2009] um primeiro esboço do uso de Máquina de Moore Adaptativa na modelagem de cursos. Sendo uma Máquina de Moore um autômato finito com saída associada aos seus estados, é possível associar, para cada estado, o material didático a ser apresentado pela função de saída. Aplicando-se os conceitos de adaptatividade nesse transdutor, pode-se elaborar cursos dinâmicos, que se auto-modifiquem com base nas experiências e nível de conhecimento individualizado dos alunos.

3. COMPUTADORES NA EDUCAÇÃO

De forma geral, os computadores têm tido um papel de destaque no processo educativo [PAPERT 1980, KEEGAN 1986, PERRATON 1988, VALENTE 1993, GARRISON 1997, BRANSFORD et al. 1999]. Ao contrário de outras mídias utilizadas na educação, o computador tem a característica de processar e manipular a informação que recebe. Ele pode calcular, traduzir, ordenar, transformar, arrumar e mesmo fazer inferências a partir de informações fornecidas.

O primeiro sistema de ensino automatizado que se tem notícia data de 1926 quando Sidney L. Pressey, professor da universidade estadual de Ohio, inventou uma máquina que apresentava questões de múltipla escolha através de um tambor cilíndrico que podia ser rotacionado. Nos últimos anos, diversas soluções educacionais vêm sendo propostas e desenvolvidas para as mais variadas áreas do conhecimento [FAGUNDES 1999].

O uso de tecnologias computacionais tem sido importante no auxilio do aprendizado para inúmeras modalidades de ensino, seja quando aplicada à educação infantil ou de nível superior, em programas regulares ou de educação continuada, no ensino presencial ou a distância.

Contudo, muitos ambientes tradicionais de educação não passam de um repositório estático, com os mesmos conteúdos, estruturas e apresentação para todos os alunos. Prover um ambiente de ensino com funcionalidades que permitam a adaptação do ambiente à situação específica vivida pelo usuário a cada intervalo de tempo é uma tarefa inovadora e investigativa.

O uso de tecnologias computacionais no ensino implica na utilização do computador como ferramenta pedagógica que auxilia no processo de construção do conhecimento. Ou seja, implica que o aluno, através da máquina, possa adquirir conceitos sobre praticamente qualquer domínio. Nesse sentido, o computador transforma-se em um poderoso recurso de suporte à aprendizagem, com inúmeras possibilidades pedagógicas.

> *"Entender o binômio "Computador e Educação" é ter em vista o fato de que o computador se tornou um instrumento, uma ferramenta para aprendizagem, desenvolvendo habilidades intelectuais e cognitivas, levando o indivíduo a explorar suas potencialidades, sua criatividade e sua inventividade. O produto final desse processo é a formação de indivíduos autônomos, que aprendem por si mesmo, porque aprenderam a aprender, através da busca, da investigação, da descoberta e da invenção."*
>
> [VEIGA 2001]

No processo de ensino-aprendizagem, o computador tem sido utilizado tanto para ensinar sobre computação, como para ensinar praticamente qualquer assunto. No ensino de computação, o computador é usado como objeto de estudo. Ou seja, o aluno usa o computador para adquirir conceitos computacionais, como arquitetura do computador, noções de programação, implicações sociais do computador na sociedade, etc. Já no ensino mediado por computador, ele pode ser usado como uma máquina de ensinar, que essencialmente realiza as atividades didáticas tradicionais de forma automatizada, ou ainda como uma ferramenta pedagógica.

Valente [VALENTE 1993] afirma que para a implantação do computador na educação são necessários basicamente quatro ingredientes: o computador, o software educativo, o professor capacitado para usar o computador como meio educacional e o aluno. Todos eles com igual importância. De forma complementar, o importante trabalho de Bransford [BRANSFORD 1999] propõe

quatro perspectivas para avaliar um ambiente de aprendizagem: o conhecimento, o aprendiz, a comunidade e a avaliação.

Uma questão a ser enfatizada é que ao considerar a aplicação do computador ou qualquer produto tecnológico na educação, é preciso ter claro, e em destaque, que a aprendizagem - a aquisição de um conhecimento novo - só ocorre com o engajamento pessoal do aprendiz. Nenhuma máquina é capaz de colocar conhecimento em uma pessoa. Ela pode ser usada para ampliar as condições do aprendiz de desenvolver suas próprias potencialidades.

Outro item relevante é destacar que atrás de qualquer tecnologia utilizada para sistemas de ensino existem pessoas, que preparam os materiais e os disponibilizam. Assim, as tecnologias não mudam necessariamente a relação pedagógica. As tecnologias tanto servem para reforçar uma visão conservadora, como uma visão progressista. Aqui serão apresentados algumas das principais teorias da aprendizagem, porém não serão defendidos pontos de vista ou discutidos assuntos relativos a questões didático-pedagógicas, sendo apenas apresentados conceitos e aplicações de soluções relacionadas ao uso dos computadores e da informática na educação.

A seguir, são apresentados alguns conceitos sobre as principais teorias da aprendizagem, softwares educacionais, além de alguns trabalhos relacionados. Não é pretensão esgotar os temas existentes sobre ensino por computador, mas, sim, fazer uma breve explanação sobre alguns conceitos e soluções sobre os temas citados que serviram de suporte na elaboração da proposta apresentada no capítulo 4 desta tese.

3.1. TEORIAS DA APRENDIZAGEM

Denominam-se de teorias da aprendizagem os diversos modelos que visam explicar o processo de aprendizagem pelos indivíduos. O processo de aprendizagem, por sua vez, pode ser definido como o modo como os seres adquirem novos conhecimentos, seja pela experiência, pela observação ou pela prática motivada. Sob o ponto de vista filosófico e psicológico, essa definição tem sofrido alterações significativas ao longo dos séculos. A partir do final do século 19, quando a psicologia passou a ser reconhecida como ciência, as principais teorias da aprendizagem que se manifestam dominantes são [CFAECO 2010, SCHULTZ 2005]: behaviorismo, cognitivismo e construtivismo. A primeira está associada à psicologia do comportamento e as restantes à psicologia cognitiva. O construtivismo é uma evolução do cognitivismo.

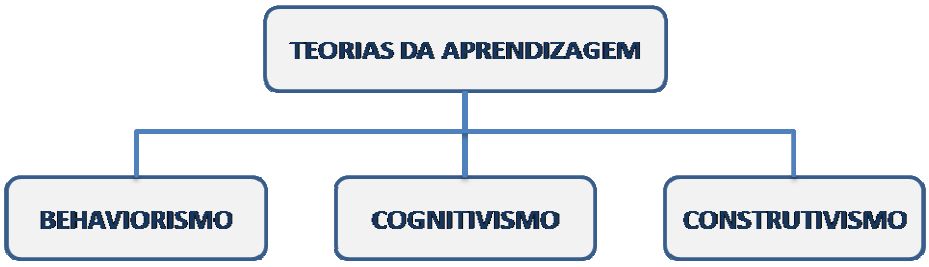

Figura 2: Principais teorias da aprendizagem.

3.1.1. BEHAVIORISMO

O behaviorismo [WATSON 2005] é uma teoria da aprendizagem que se centra em comportamentos observáveis e ignora as atividades mentais. Tem origem em 1913, após a publicação do artigo “Psicologia: como os behavioristas a vêem”, de John B. Watson. O behaviorismo baseia-se nas mudanças de comportamento observáveis. Um dado modelo de comportamento é repetido até que o mesmo se torne automático. Os behavioristas definem aprendizagem

como a aquisição de novos comportamentos. No modelo behaviorista, os indivíduos reagem reflexivamente ao ambiente. Os processos cognitivos são ignorados. Certas respostas podem ser condicionadas reforçando o comportamento desejado com um estímulo.

A teoria behaviorista concentra-se no estudo dos comportamentos que podem ser observados a partir das reações do indivíduo a estímulos do meio ambiente. A mente é vista como uma caixa negra porque responde a estímulos que podem ser observáveis, ignorando totalmente a possibilidade de ocorrência de processos mentais. Consoante a resposta dada pelo aprendiz esteja certa ou errada, é lhe fornecido um reforço positivo ou negativo. O objetivo é aumentar a probabilidade de ocorrência da resposta desejada no futuro. Assim, para o behaviorismo o conhecimento é visto como dado e absoluto, isto é, existe na realidade exterior e universalmente aceite. A aprendizagem é um processo passivo, sem interesse pelos processos mentais que ocorrem no aprendiz.

3.1.2. COGNITIVISMO

O cognitivismo baseia-se nos processos mentais subjacentes ao comportamento. As mudanças no comportamento são observadas e utilizadas como indicadores do que está a acontecer na mente do aprendiz. Contrariamente ao behaviorismo, que considera que o comportamento é uma resposta mecânica à sujeição de estímulos, ou seja, a aprendizagem é determinada pelo meio ambiente e o organismo humano adapta-se às circunstâncias do meio, os cognitivistas interessaram-se por descobrir o que se passa dentro do cérebro humano e modelar os processos mentais que ocorrem durante a aprendizagem. Segundo eles, à semelhança do computador, a mente humana é um processador de informação, isto é, recebe, interpreta, armazena e recupera ou utiliza informação quando necessita dela.

Embora para a teoria cognitivista os processos mentais que ocorrem no aprendiz sejam o objeto de estudo principal, o conhecimento continua a ser visto como dado e absoluto, tal como acontece com o behaviorismo; a aprendizagem é o processo que cria na memória representações simbólicas da realidade exterior. Outro aspecto importante associado ao cognitivismo, derivado da teoria cognitiva de Piaget, é o respeito pelo estágio de desenvolvimento intelectual dos alunos, garantindo que as estruturas cognitivas destes estejam preparadas para a aquisição de novos conhecimentos.

3.1.3. CONSTRUTIVISMO

O construtivismo [PAPERT 1986] baseia-se na premissa de que todos constroem a própria perspectiva do mundo, através da experiência individual e de acordo com determinado esquema (estrutura mental do conhecimento que um indivíduo possui; o esquema encontra-se em reajustamento constante: um indivíduo ajusta o seu esquema mental para incorporar novas experiências e atribuir significado à nova informação).

O construtivismo é uma filosofia de aprendizagem que se baseia na premissa de que os indivíduos constroem o próprio conhecimento do mundo em que vivem, refletindo nas próprias experiências pessoais. Cada indivíduo gera regras e modelos mentais, e utiliza-os para interpretar e atribuir significado às suas próprias experiências. A aprendizagem consiste, por isso, no processo de ajustamento dos modelos mentais à acomodação de novas experiências.

A aprendizagem construtivista baseia-se numa participação ativa dos alunos na resolução de problemas e na exercitação do pensamento crítico, relativamente às atividades que acham relevantes e atraentes. Eles estão a construir o próprio conhecimento, testando idéias e aproximações baseadas no

conhecimento que possuem e na experiência, aplicando-as a situações novas e integrando o novo conhecimento no pré-existente.

A premissa fundamental do construtivismo é que o aluno constrói ativamente o próprio conhecimento. Em vez de absorver simplesmente as idéias transmitidas pelo professor ou resultantes da prática repetitiva, o aluno é colocado a criar e a desenvolver as próprias idéias.

Os construtivistas vêem a aprendizagem como o resultado de uma construção mental. Os alunos aprendem combinando a nova informação com a que já conhecem. A aprendizagem é também influenciada pelo contexto, atitudes e valores do aluno. O conceito chave do construtivismo é que a aprendizagem é um processo ativo de criação, não de aquisição, do conhecimento.

3.1.4. ANÁLISE PEDAGÓGICA DAS TEORIAS DA APRENDIZAGEM

O behaviorismo e o cognitivismo vêem o conhecimento como absoluto (imposto socialmente, universalmente aceito e existente na realidade exterior) e transmissível. Esta visão objetivista do conhecimento orienta a aprendizagem para um processo passivo, em que a realidade exterior é interpretada de forma convergente por todos os alunos: os behavioristas dizem que a aprendizagem consiste nas respostas do aluno a fatores externos e existentes no meio ambiente; os cognitivistas afirmam que a aprendizagem consiste na representação simbólica da realidade exterior que o aluno projeta na sua mente. Por isso, o foco pedagógico incide, para os behavioristas, na aplicação de estímulos e reforços adequados enquanto, para os cognitivistas incide na manipulação do processo mental que o aluno deve seguir.

O construtivismo apresenta uma visão do conhecimento diferente da visão exposta pelo behaviorismo e pelo cognitivismo. Para o construtivismo o

conhecimento é uma construção pessoal que se realiza através do processo de aprendizagem. O conhecimento não pode ser transmitido de uma pessoa para outra, ele é (re)construído em cada pessoa. Cada aluno interpreta a realidade exterior baseando-se na sua experiência pessoal. Refletindo na experiência individual, o aluno ajusta os seus modelos mentais para inter-relacionar a nova informação com o seu conhecimento prévio. Dessa forma, a realidade exterior é internamente controlada pelo aluno. Ele cria a sua própria interpretação da realidade com base na estrutura cognitiva que possui. Conseqüentemente, o objetivo principal da pedagogia é fomentar e orientar o processo mental que o aluno segue na interpretação da realidade.

Tabela 1: Comparativo das teorias da aprendizagem.

	Aluno	**Professor**	**Aprendizagem**	**Método**
Behaviorismo	Passivo	Repassador do conhecimento (verdades absolutas)	Influenciada por fatores biológicos de conduta (estímulo-resposta)	Entrega de contéudos; Instrução programada.
Cognitivismo	Autônomo; Filtra os materiais que têm significado para si.	Determina a estrutura conceitual e proposicional do conteúdo	"Ampliação" da estrutura cognitiva através da incorporação de novas idéias a ela.	Aulas expositivas; Desenvolvimento de mapas conceituais.
Construtivismo	Ativo	Mentor; Favorecedor dos processos de descobrimento autônomo de conceitos.	Interação sujeito/objeto; Construtivismo seqüencial (níveis); Relação direta com o desenvolvimento	Experiências, pesquisas e solução de problemas.

Do ponto de vista da criação de software educacional, é desejável que o sistema permita a elaboração de cursos baseados nos diferentes modelos apresentados sobre as teorias de aprendizagem. Ficando a critério do professor ou responsável a definição do modelo mais apropriado para suas necessidades e perfil de seus alunos. Não descartando, ainda, o desenvolvimento de cursos híbridos com módulos baseados em diferentes modelos de aprendizagem.

3.2. SOFTWARES EDUCACIONAIS

Como definição geral, um software educacional (SE) é um produto dirigido a diversas finalidades pedagógicas, sendo planejado e desenvolvido de modo a poder ser aplicado para diferentes estratégias de aprendizagem. Em outras palavras, todo programa que utiliza uma metodologia que o contextualize no processo ensino e aprendizagem pode ser considerado educacional. Os softwares educacionais também são conhecidos por outros termos, tais como: softwares educativos, *coursewares*, ou, ainda, programas educativos por computador. Segundo Campos [CAMPOS 1989], todos estes termos possuem o mesmo significado: material educacional para microcomputadores.

Desde a década de 70, o software educacional vem entrando no mercado mundial de forma progressiva. Diversos países como Inglaterra, França e EUA, entre outros, desenvolveram projetos de uso do computador na educação e, conseqüentemente, necessitaram desenvolver produtos de software específicos para suas necessidades. O mesmo ocorreu no Brasil, aonde diversos projetos de pesquisa vêm sendo desenvolvidos não só relacionados ao uso do computador em sala de aula como, também, ao desenvolvimento de software para os mais diversos conteúdos programáticos.

Projetos de desenvolvimento de software educacional, além de envolver em seu desenvolvimento uma equipe multidisciplinar, devem refletir os objetivos educacionais propostos e o ambiente de aprendizagem almejado, criando situações que estimulem o desenvolvimento das habilidades desejadas.

Ao escolher um software para apoiar alguma atividade curricular, o professor conta com vários tipos de software que podem ser usados para atingir resultados eficientes para a aprendizagem e para o desenvolvimento da habilidade de investigação e pensamento crítico.

Sabe-se da importância do computador na educação como agente transformador e a importância do software educacional como co-responsável dessa transformação auxiliando no processo de ensino e aprendizagem. Porém, um problema em relação à questão do software educacional é que ainda não há uma delimitação clara e precisa sobre o que pode ser considerado um software educacional: Um jogo pode ser considerado um software educacional? E se for um jogo pedagógico? E programas que permitem a construção e a manipulação de estatísticas educacionais para uso por supervisores e orientadores pedagógicos? E aqueles voltados para a administração do ensino e da escola? Quais são os critérios para que um determinado software seja considerado educacional? Que ele tenha sido feito sob a ótica da educação para desenvolver algum objetivo didático? Mas, quais as características do software didático? O que significa dizer que um software didático é de boa ou má qualidade? Quais os parâmetros que determinam a qualidade do software didático? O que torna um software didático adequado ou não a determinadas situações de ensino-aprendizagem?

A fim de buscar responder algumas das perguntas supracitadas, as seções a seguir trazem, respectivamente, uma classificação de diversos tipos de softwares educacionais, uma referência sobre qualidade de softwares educacionais e, alguns métodos de avaliação de softwares educacionais.

3.2.1. CLASSIFICAÇÃO DE SOFTWARES EDUCACIONAIS

Existem diferentes maneiras de classificar os programas computacionais que podem ser utilizados na educação. Uma das maneiras, por exemplo, consiste em categorizar de acordo com a natureza do software e suas propriedades. Uma outra forma seria classificar pela finalidade para a qual o programa computacional é utilizado no processo educacional.

Quanto ao enfoque dado à aprendizagem, um software educacional pode direcionar para uma aprendizagem algorítmica ou heurística. A aprendizagem algorítmica está relacionada à teoria de aprendizagem behaviorista, enquanto a aprendizagem heurística trabalha no contexto construtivista.

Em um software de aprendizagem algorítmica a ênfase está na transmissão de conhecimentos, na direção que vai do sujeito que domina o saber para aquele que quer aprender. No modelo algorítmico o desenvolvedor de software tem o papel de programar uma seqüência de instruções planejadas para levar o educando ao conhecimento. Essa modalidade pode ser caracterizada como uma versão computadorizada dos métodos tradicionais de ensino. Segundo [VALENTE 1993], as categorias mais comuns desta modalidade são: os tutoriais, *drill & practice* (repetição e prática), jogos e simulação.

Já em um software orientado pelo modelo de aprendizagem heurística predominam as atividades experimentais em que o programa produz um ambiente com situações variadas para que o aluno as explore e construa conhecimentos por si mesmo. Como exemplo, é possível citar a estratégia de "aprender por projetos" [FAGUNDES 1999], na qual o aluno aprende através da experimentação ativa, buscando soluções para os problemas.

A seguir, são sucintamente apresentadas algumas das principais classificações e tipos de softwares utilizados em educação.

3.2.1.1. SISTEMAS TUTORIAIS

É um tipo de software no qual a informação é organizada de acordo com uma seqüência pedagógica particular. Procuram ensinar controlando o processo de aprendizagem e de acordo com o tempo que o aluno leva para aprender. Um tutorial é usado para introduzir novos tópicos e conceitos para os alunos, proporcionando uma instrução direta. Geralmente os tutoriais são ricos em inovações tecnológicas como hipertexto, interfaces bem elaboradas (com sons, imagens, animações, etc.), versões para Web, etc. que tornam a apresentação do conteúdo atraente, retendo a atenção do aluno. Alguns ainda verificam o aprendizado do aluno, através de perguntas, permitindo interação. Mais recentemente foram desenvolvidos os chamados Sistemas Tutores Inteligentes (STI) que utilizam-se de técnicas de Inteligência Artificial (IA) e teorias pedagógicas para conduzir o aluno dentro de um determinado domínio de conhecimento.

3.2.1.2. REPETIÇÃO E PRÁTICA

Tipo de software que utiliza perguntas e respostas, normalmente utilizadas para revisar material já estudado. Portanto, a atividade computacional proporcionada por um software desse tipo revê um conteúdo que já foi apresentado ao aluno. Seu principal objetivo é aquisição, desenvolvimento e aplicação de um conhecimento específico. Como citado em [LUCENA 1990], Campos e Rocha sugerem que o tipo de software "repetição e prática" deve ser usado após a apresentação de um "tutorial", depois, portanto, do aluno ter adquirido o novo conhecimento para uma avaliação da aprendizagem. Entretanto, ele não é uma boa ferramenta se a estratégia de ensino enfatizar a análise e a síntese do conteúdo pedagógico.

3.2.1.3. SIMULAÇÃO

Um software que apresente "simulação" permite ao aluno realizar atividades das quais normalmente não poderia participar, dando-lhe a oportunidade de testar, tomar decisões, analisar, sintetizar e aplicar o conhecimento adquirido em situações reais. A simulação permite realizações de experiências que métodos convencionais de ensino, usualmente, não proporcionam, fazendo com que o aluno observe e tire conclusões sobre as conseqüências de suas ações e decisões. A simulação deve ser utilizada após a aprendizagem de conceitos e princípios básicos do tema em questão. Exemplos típicos de simulação podem ser observados em sistemas de realidade virtual (RV).

3.2.1.4. APLICATIVOS

São programas que *a priori* não foram desenvolvidos para fins educacionais, mas que são essencialmente interativos, permitindo a organização e o tratamento rápido e produtivo dos dados introduzidos no computador, apresentando grande potencialidade para o uso na prática educacional, podendo, portanto, com criatividade, ser utilizados em diversas atividades curriculares. O uso desses aplicativos dá grande liberdade ao professor para adaptá-los dentro das necessidades de suas disciplinas curriculares. Incluem nessa categoria: processadores de texto, planilhas eletrônicas, editores gráficos, etc.

3.2.1.5. JOGOS EDUCACIONAIS

O jogo é uma atividade lúdica em que crianças e/ou adultos participam de uma situação de engajamento social num tempo e espaços determinados, como

características próprias delimitadas pelas próprias regras de participação na situação “imaginária”. Originalmente criados para o entretenimento, os jogos educativos facilitam e estimulam a aprendizagem através da interação. Incitam a resolução dos problemas propostos permitindo ao utilizador raciocinar e estimular as suas capacidades cognitivas, assim como desenvolver a sua coordenação motora e reflexiva. Existem dois grupos principais de jogos educativos: os de enredo e os de regras.

3.2.1.6. FERRAMENTAS PARA RESOLUÇÃO DE PROBLEMAS

Um software direcionado para a "solução de problemas" apresenta situações que estimulam o aluno a encontrar estratégias próprias para resolver o problema proposto. Neste processo, o aluno avalia e utiliza os conhecimentos já adquiridos que são específicos e necessários para finalizar com sucesso a tarefa proposta. Em geral, o aprendiz deve produzir qual problema quer solucionar. Esse tipo de ferramenta pode atender a quase todas as disciplinas, tanto no conhecimento como no interesse e a capacidade do aluno, são softwares abertos que permitem ao professor constantemente descobrir novas formas de planejar atividades que atendam seus objetivos.

3.4. PROJETOS RELACIONADOS

Atualmente, diversos ramos da computação têm desenvolvido soluções ou sido utilizadas como suporte em sistemas educacionais. Entre elas, é possível citar: RV, IA, videoconferência, aplicações via Internet, multimídia, etc. A seguir, são apresentados alguns conceitos, entre muitos existentes, que serviram de referência no desenvolvimento do modelo proposto no capítulo 4 desta tese. Juntamente como a definição desses conceitos e projetos, são apresentadas, também, algumas das diferenças com o projeto aqui proposto.

3.4.1. OBJETOS DE APRENDIZAGEM

De acordo com [IEEE 2002], um objeto de aprendizagem (*Learning Object*) é qualquer entidade digital ou não digital que possa ser usada, re-usada ou referenciada durante a aprendizagem baseada em Tecnologias da Informação. Assim, um objeto de aprendizagem é o menor bloco de informação capaz de transmitir conhecimento e deve ser elaborado de forma independente.

Ao dividir a estrutura de um curso em objetos reutilizáveis, é possível aproveitar o mesmo conteúdo para vários objetivos e situações. Esse modelo de curso, baseado em objetos de aprendizagem, é caracterizado pela alternativa de se criar blocos independentes de conteúdo educacional que forneçam uma experiência educacional para algum propósito pedagógico.

Objetos de aprendizagem são estruturados através da combinação do conteúdo didático e seus respectivos metadados educacionais. Para permitir que desenvolvedores e instrutores encontrem com facilidade os objetos que sirvam a seus propósitos, a descrição, objetivos educacionais e outras características de cada objeto existente no repositório são colocadas em metadados. Quando o objeto de aprendizagem é armazenado para

distribuição, um sistema de registro de características efetua essa catalogação, o que permite criadores e usuários em potencial encontrarem o objeto rápida e facilmente. Através do metadado é possível obter informações características de catalogação do objeto de aprendizagem, tais como: aplicações previstas; nível do aprendiz; tipo de interatividade; formato de mídia; autor; data de criação e atualização, etc.

Na produção de componentes didáticos digitais, considera-se os objetos de aprendizagem como blocos de conteúdo educacional, podendo fazer referência a outros blocos, e podendo ser combinados ou seqüenciados para formar interações educacionais. Esses objetos de aprendizagem podem ser usados como recursos simples ou combinados para formar uma unidade de instrução maior. Podem, também, ser usados em um determinado contexto e depois reutilizados em contextos similares.

IEEE (1484.12.1-2002 IEEE Standard for Learning Object Metadata) e ISO (SC 36 WG 2 - Information Technology for Learning, Education and Training) têm grupos trabalhando na elaboração de propostas para sua estruturação e categorização (metadados). Além da reusabilidade dos recursos, que possibilita incorporá-los em múltiplas aplicações, destacam-se também outros benefícios da catalogação de objetos educacionais:

- ✓ acessibilidade: pela possibilidade de acessar recursos educacionais em um local remoto e usá-los em muitos outros locais;
- ✓ interoperabilidade: podendo utilizar componentes desenvolvidos em um local, com algum conjunto de ferramentas ou plataformas, em outros locais com outras ferramentas e plataformas;
- ✓ durabilidade: para continuar usando recursos educacionais quando a base tecnológica muda, sem reprojeto ou recodificação.

Esses modelos de padronização vêm ganhando aceitação mundial, mas ainda é prematuro classificá-los como padrões universalmente aceitos.
Neste projeto, será utilizado apenas o conceito principal de objeto de aprendizagem. Ou seja, cada material didático será representado como um objeto de aprendizagem permitindo sua concatenação com outros objetos de aprendizagem, além da possibilidade de se reutilizar seu conteúdo. Questões como a categorização dos objetos de aprendizagem através de metadados não serão tratadas.

3.4.2. SISTEMAS DE HIPERMÍDIA ADAPTATIVA

Sistemas Hipermídia (SH) podem ser conceituados a partir da relação entre os conceitos de Hipertexto e Multimídia. Hipertexto é um sistema para a visualização de informação cujos documentos contêm referências internas (chamadas de links ou nós de informação) onde o usuário pode iniciar uma leitura e navegar de forma não linear. Multimídia compreende os múltiplos meios que podem ser usados na representação de uma informação, como, por exemplo, texto, imagem, áudio, animação e vídeo.

A Hipermídia Adaptativa (HA) estuda o desenvolvimento de sistemas capazes de promover a adaptação de conteúdos e recursos hipermídia, vindos de qualquer fonte (bancos de dados, Internet, serviços etc.) e apresentados em qualquer formato (texto, áudio, vídeo, etc e suas combinações) ao perfil ou modelo de seus usuários [PALAZZO, 1999].

A idéia por trás da expressão Sistema de Hipermídia Adaptativa (SHA) é a expectativa de oferecer a cada usuário uma interface modelada de acordo com suas características específicas. Em outras palavras, em SHA os usuários acessam interfaces cujo estilo, conteúdo, recursos e links serão dinamicamente

selecionados, entre diversas possibilidades, reunidos e apresentados a eles conforme seus objetivos, necessidades, preferências e desejos.

Um SHA deve então satisfazer a três critérios básicos [PALAZZO, 1999]:

- ✓ ser um sistema hipertexto ou hipermídia;
- ✓ possuir um modelo do usuário; e,
- ✓ ser capaz de adaptar a hipermídia do sistema usando tal modelo.

SHA são especialmente úteis quando há a necessidade de disponibilizar informação seletiva e contextual a usuários com diferentes objetivos e níveis de conhecimento. Entre os principais usos da HA encontram-se hoje os sistemas educacionais baseados em hipermídia, sistemas de informações on-line, sistemas de ajuda on-line, sistemas de informações institucionais e a construção de visões personalizadas.

Segundo [BRUSILOVSKY, 2001], existem duas classes diferentes de HA, caracterizando o primeiro a apresentação adaptativa e o segundo a navegação adaptativa.

Na **apresentação adaptativa**, a idéia é adaptar o conteúdo de um *link* acessado por um particular usuário, ao conhecimento, objetivos e outras características deste usuário. Por exemplo, a um usuário experiente é possível a apresentação de informação mais profunda e detalhada, enquanto que a um iniciante podem ser oferecidas explicações adicionais.

A **navegação adaptativa** busca auxiliar os usuários a achar seus caminhos no hiperespaço através da adaptação da forma de apresentar os *links* na rede hipermídia. Nesta área é possível distinguir seis principais tecnologias para a apresentação dos *links* ao usuário:

- ✓ Orientação direta: Em cada ponto, qual o melhor caminho?
- ✓ Classificação: Em que ordem os links devem ser apresentados?
- ✓ Ocultação: Quais links não devem ser apresentados?
- ✓ Anotação: Como agregar mais informação aos links?
- ✓ Geração de links: Como links interessantes podem ser gerados?
- ✓ Adaptação de mapas e índices: Como apresentar mapas e índices?

Abaixo, a figura 4 ilustra a classificação dos espaços da HA, proposta em [BRUSILOVSKY, 2001].

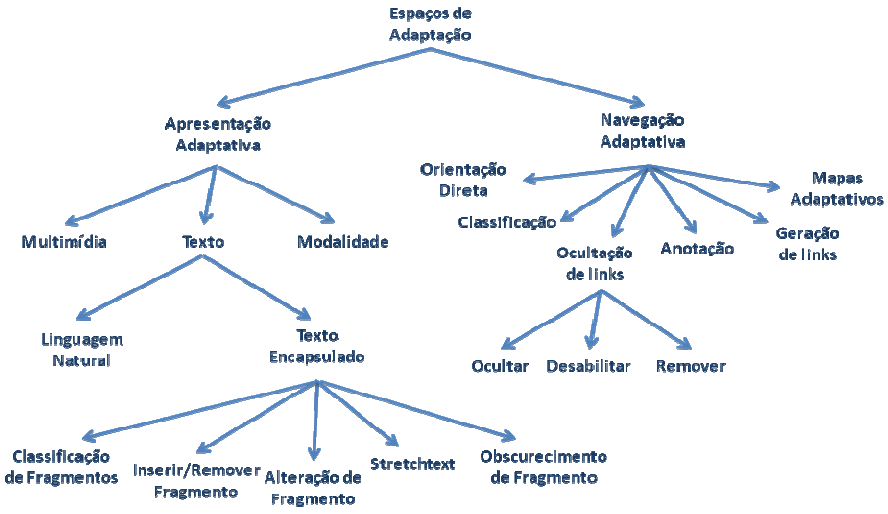

Figura 3: Espaços de adaptação em Hipermídia Adaptativa.

Apesar da palavra "adaptativa" fazer parte da expressão SHA, sua forma de aplicação e utilização pouco tem relação com os formalismos adaptativos. Em SHA, a adaptatividade é obtida através de técnicas heurísticas de IA. Neste trabalho, serão utilizadas algumas das idéias relacionadas aos espaços de adaptação, apresentados na figura 9, porém, representados através de dispositivos adaptativos.

3.4.3. REPRESENTAÇÃO DE CURSOS ATRAVÉS DE AUTOMATOS

No artigo intitulado “Cursos são Autômatos” [MACHADO, 1999], é apresentada uma forma de modelagem de cursos para *Web* utilizando autômatos finitos determinísticos com saída, cuja principal finalidade é a criação de material hipermídia independente do esquema de navegação. Na proposta, um curso pode ser representado tanto por Máquina de Moore ou quanto por Máquina de Mealy, com material didático do curso vinculado, respectivamente, aos estados ou às transições do transdutor. Em ambos os casos, esse autômato com saída fornece uma máquina abstrata para o controle da navegação em hipertextos.

Segundo esse modelo, cada autômato constitui um curso definido sobre um conjunto de hiperdocumentos (unidades de informação) independentes. As *n-uplas* dos autômatos de cursos apresentam uma correspondência às estruturas de hiperdocumentos na *Web*. O alfabeto de símbolos de entrada é um conjunto de nomes que identificam as âncoras de navegação no hipertexto. As saídas estão relacionadas a unidades de informação constituídas por hiperdocumentos e as palavras de saída, presentes na função de saída, são hipertextos apresentados como uma única página *Web* no navegador do usuário. As transições, definidas na função programa, funcionam como ligações lógicas (e não ligações físicas no documento) entre os conteúdos dos cursos e definem possíveis *links* a serem selecionados durante a navegação pelo hipertexto.

Uma das vantagens da solução utilizada é que a estrutura dos autômatos com saída permite a criação do material hipermídia de forma independente do autômato em si, possibilitando a programação de seqüências de estudo com objetivos e enfoques específicos, reuso de parte ou íntegra das páginas *Web* em diversos cursos, eliminando a redundância na criação de páginas, bem como a modularização que facilita a expansibilidade do sistema.

A estrutura do sistema de hipertexto que não utiliza *links* diretos permite a estruturação e edição do conjunto de materiais hipermídia sem a necessidade do autor dos documentos se preocupar com a inserção de *links* diretamente nos textos, facilitando a tarefa de edição dos documentos. Um benefício adicional é que, como páginas de um curso podem ser constituídas por outras sub-páginas concatenadas, o autor pode construir funções de saída/programa diferentes que contenham acesso a páginas de mesmo conteúdo, evitando-se redundância na criação dos documentos *Html*.

É possível perceber que num mesmo estado podem existir mais de uma unidade de informação associada, uma vez que a função de saída pode gerar palavras a partir da concatenação de símbolos do alfabeto de saída. É possível, assim, fazer reuso das unidades de informação, que podem aparecer por mais de uma vez durante o curso. Da mesma maneira, é também verdade que esse material didático possa ser reaproveitado em outros cursos.

Para a modelagem de um curso estático, o funcionamento de uma Máquina de Moore ou de uma Máquina de Mealy é suficiente para se conseguir especificar formalmente todo o modelo. Contudo, um enriquecimento nesse tipo de máquina, através da camada adaptativa, permite maior poder de representação, podendo tornar o acompanhamento do curso mais flexível e dinâmico e podendo tratar questões dependentes de contexto. Neste trabalho, a Máquina de Moore é utilizada como dispositivo subjacente, conforme apresentado a seguir no Capítulo 4.

4. MÁQUINA DE MOORE ADAPTATIVA

A formulação para Máquina de Moore Adaptativa, apresentada nesta seção, é adaptada da formulação geral para dispositivos adaptativos dirigidos por regras proposto originalmente em [NETO 2001] e novamente publicado em [RAMOS et al. 2009]. Complementarmente a um autômato adaptativo, uma Máquina de Moore Adaptativa (MMA) pode adicionar ou remover as saídas associadas a cada estado, além de ter funções adaptativas para adicionar ou remover estados e transições. Para isso, é proposto apenas um ajuste na gramática das funções adaptativas para que as ações adaptativas elementares possam ser aplicadas também na função de saída do dispositivo subjacente original.

4.1. FORMULAÇÃO PARA MÁQUINA DE MOORE (NÃO ADAPTATIVA)

Uma Máquina de Moore (MM) [HARRISON 1978], [HOPCROFT 1979], [LEWIS & PAPADIMITRIOU 2000] é um autômato finito determinístico modificado, que possui saídas associadas aos estados. Essa máquina possui uma função que gera uma palavra de saída para cada estado da máquina, podendo essa ser uma palavra vazia. Outro tipo de transdutor, conhecido como máquina de Mealy, também é um autômato finito modificado, mas que possui as palavras de saída associadas com as transições entre os estados. Neste tipo de máquina, as palavras de saída dependem do estado atual e do valor das entradas.

O processamento de uma MM para uma dada entrada w consiste na sucessiva aplicação da função-programa para cada símbolo de w, até ocorrer uma condição de parada. A palavra vazia como saída da função programa indica

que nenhuma gravação é realizada. Se todos os estados geram saída vazia, então a Máquina de Moore se comporta como se fosse um autômato finito.

A MM é representada por uma héptupla: $MM = (Q, \delta, \Sigma, q_0, \lambda, \Delta, \delta_s)$, onde:

- ✓ MM é uma Máquina de Moore (não-adaptativa), cuja operação é determinada pelo conjunto de regras δ.
- ✓ Q é o conjunto de todas as suas possíveis configurações, e $q_0 \in Q$ é sua configuração inicial.
- ✓ Σ é o conjunto (finito) de todos os símbolos que são estímulos de entrada válidos para MM, com $\varepsilon \in \Sigma$.
- ✓ $\lambda \subseteq Q$, (respectivamente, $F = Q - \lambda$) é o subconjunto de todos os seus estados finais (respectivamente, estados não-finais).
- ✓ ε denota "vazio", e representa o elemento nulo de conjunto ao qual ele pertence.
- ✓ $w = w_1 \; w_2 \; ... \; w_n$ é uma cadeia de estímulos de entrada, onde $w_k \in \Sigma - \{\varepsilon\}, k = 1, ... , n$, com $n \geq 0$.
- ✓ Δ é um conjunto (finito), com $\varepsilon \in \Delta$, de todos os possíveis símbolos a serem emitidos como saídas por MM.
- ✓ δ_s é a função de saída por meio da relação $\delta_s \subseteq Q \times \Delta^*$ a qual é uma função total que determina a geração de uma palavra de saída para cada estado.
- ✓ δ é o conjunto de regras que definem MM por meio da relação $\delta \subseteq Q \times \Sigma \times Q$. As regras $r \in \delta$ apresentam-se na forma $r = (q_i, \; s, \; (q_j, \; \delta_s(q_j)))$, indicando que, em resposta a um estímulo de

entrada qualquer $s \in \Sigma$, r modifica a configuração corrente de q_i para a nova configuração q_j, executa a função de saída $\delta_s(q_j)$, e consome s.

Uma regra $r = (q_i, s, (q_j, \delta_s(q_j)))$, com $r \in \delta$; $q_i, q_j \in Q$; $s \in \Sigma$, é dita compatível com a configuração corrente q se e apenas se $q_i = q$, desde que s seja vazio ou igual ao estímulo de entrada corrente do dispositivo. Neste caso, a aplicação de uma regra compatível move o dispositivo para a configuração q_j (denotada por $q_i \Rightarrow^s q_j$). Note-se que s pode ser vazio.

Uma seqüência opcional de movimentos em vazio, finalizada por um movimento que consuma o símbolo w_k, é denotada por: $q_i \Rightarrow^{\sim} q_m$, $m \geq 0$, e abrevia $q_i \Rightarrow^{\varepsilon} q_1 \Rightarrow^{\varepsilon} q_2 \Rightarrow^{\varepsilon} \ldots \Rightarrow^{\varepsilon} q_m$.

Um seqüência opcional de movimentos em vazio, finalizada por um movimento que consuma o símbolo w_k, é denotada por: $q_i \Rightarrow^{\sim w_k} q_j$ e representa $q_i \Rightarrow^{\sim} q_m \Rightarrow^{w_k} q_j$.

Diz-se que uma cadeia de entrada $w = w_1 w_2 \ldots w_n$ é aceita por MM quando: $q_0 \Rightarrow^{\sim w_1} q_1 \Rightarrow^{\sim w_2} \ldots \Rightarrow^{\sim w_n} q_n \Rightarrow^{\sim} c$ (abreviadamente denotada como $q_0 \Rightarrow^{w} q$, com $q \in \lambda$). De forma complementar, diz-se que w é rejeitada por MM quando $q \in F$.

A linguagem definida por MM é o conjunto: $L(MM) = \{w \in \Sigma^* \mid q_0 \Rightarrow^{w} q, q \in \lambda\}$ de todas as cadeias $w \in \Sigma^*$ que são aceitas por MM.

4.1.1. EXEMPLO DE MÁQUINA DE MOORE (NÃO-ADAPTATIVA)

Para ilustrar o funcionamento de uma Máquina de Moore, é apresentado a seguir um simples exemplo de um gerador de código-fonte para um subconjunto da linguagem Pascal. Em [HOPCROFT, 1979], a representação gráfica da Máquina de Moore apresenta os símbolos de saída acima dos circulos, que por sua vez representam os estados. Em [RAMOS et al, 2009], a mesma representação é feita colocando-se ambos, o estado e a saída, dentro da circunferência, separados por uma barra (/). Para efeito de facilitar a visualização, nesta tese, adotou-se representar as saídas colocando-as numa caixa de texto e ligando-a ao seu respectivo estado através de uma linha.

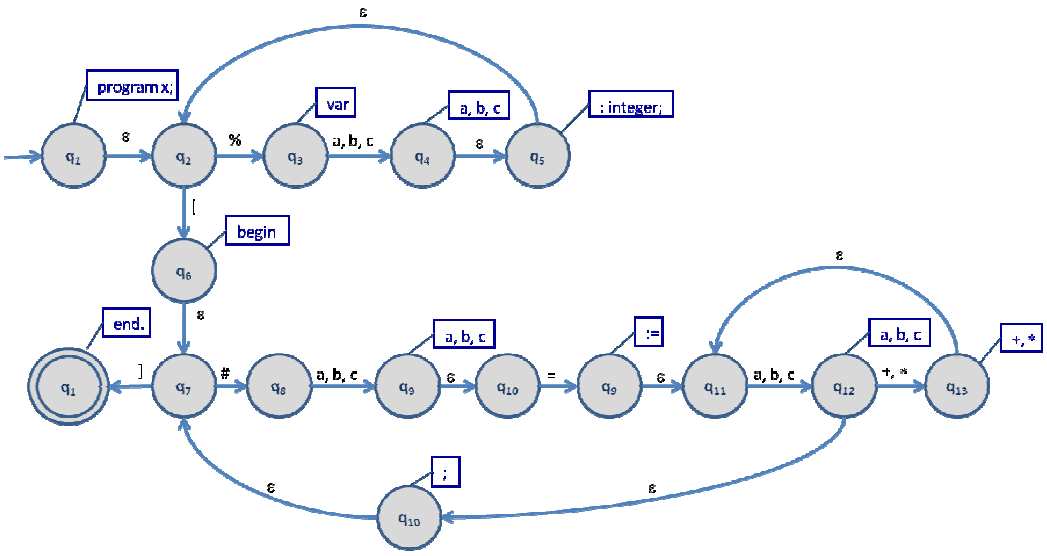

Figura 4: Exemplo de Máquina de Moore.

Um exemplo de cadeia de entrada aceita pelo transdutor descrito na figura 5 é:

```
%a
%b
[
#a=a+b
#b=b*b
]
```

Tal sentença pode ser entendida como um programa contendo a declaração de duas variáveis (“a” e “b”, nas duas linhas iniciais) e o uso de ambas em dois comandos de atribuição (nas duas linhas finais).
Após consumir a cadeia de entrada, a seguinte saída é gerada:

```
program x;
var a: integer;
var b: integer;
begin
    a := a + b;
    b := b * b;
end.
```

Apesar da geração correta da saída, note existir uma dependência de contexto não tratada por esse transdutor: somente os identificadores declarados (após o símbolo “%”) podem ser usados nos comandos (após o símbolo “#”).

Assim como um autômato finito, uma Máquina de Moore só é capaz de representar efetivamente linguagens regulares. Desse modo, a derivação da cadeia de entrada apresentada a seguir, viola a dependência de contexto que vincula referências às variáveis com as respectivas declarações (a variável ”b” é referenciada indevidamente, uma vez que sua declaração foi substituída pela declaração da variável “c”).

```
%a
%c
[
#a=a+b
#b=b*b
]
```

Observe que o transdutor também aceita a nova cadeia de entrada e gera o código-fonte como saída, uma vez que não existem erros de sintaxe na declaração. Daí a necessidade de se adicionar um filtro para excluir tais sentenças do conjunto gerado pelo transdutor.

4.2. FORMULAÇÃO PARA MÁQUINA DE MOORE ADAPTATIVA

Defina-se um contador T, que é iniciado com o valor $zero$, e automaticamente incrementado de uma unidade sempre que uma ação adaptativa não-vazia for executada. Cada valor k assumido por T pode ser usado para indexar os nomes de conjuntos variantes no tempo. Neste caso, tal indexação seleciona o passo k de operação de MMA.

Um dispositivo $MMA = (MM_0, CA)$ é dito adaptativo sempre que, para qualquer passo de operação $k \geq 0$, MMA seguir o comportamento de MM_k, até que a execução de alguma ação adaptativa não-vazia inicie seu $(k+1)$-ésimo passo ao modificar seu próprio conjunto de regras.

Em qualquer passo $k \geq 0$ de operação de MMA, sendo MM_k o correspondente dispositivo subjacente definido por R_k, a execução de qualquer ação adaptativa não-vazia provoca a evolução de R_k para R_{k+1}. Assim, MMA começa a executar seu $(k+1)$-ésimo passo de operação já com o conjunto R_{k+1}, resultante das alterações impostas, no passo anterior, ao conjunto R_k. Daí em diante, MMA

seguirá o comportamento de MM_{k+1} até que alguma outra ação adaptativa não-vazia provoque o início de um novo passo de operação. Esse procedimento se repete até que a cadeia de entrada tenha sido integralmente processada ou que algum erro interrompa a operação de MMA.

O dispositivo adaptativo MMA inicia sua operação em c_0, em sua forma inicial $MMA_0 = (Q_0, \delta_0, \Sigma, q_0, \lambda, \Delta, \delta_{s0}, BA, AA)$. No passo $k \geq 0$, um estímulo de entrada move MMA para sua próxima configuração e então MMA inicia seu $(k+1)$-ésimo passo de operação se e somente se uma nova ação adaptativa não-vazia for executada. Dessa forma, estando MMA em seu passo k, e apresentando-se na forma $MMA_k = (Q_k, \delta_k, \Sigma, q_k, \lambda, \Delta, \delta_{sk}, BA, AA)$, a execução de uma ação adaptativa não-vazia deve conduzi-lo a $MMA_{k+1} = (Q_{k+1}, \delta_{k+1}, \Sigma, q_{k+1}, \lambda, \Delta, \delta_{sk+1}, BA, AA)$.

Nessa formulação, tem-se:

- ✓ $MMA = (MM_0, CA)$ representa algum dispositivo adaptativo, dado por um dispositivo subjacente iniciado por MM_0 e um mecanismo adaptativo CA.
- ✓ MM_k é o dispositivo não-adaptativo subjacente de MMA em algum passo de operação k. MM_0 é o dispositivo não-adaptativo subjacente de MMA, definido pelo conjunto inicial de regras não-adaptativas δ_0. Por definição, quaisquer regras não-adaptativas, em qualquer δ_k, espelham as regras adaptativas correspondentes em δ_k.
- ✓ Q_k é o conjunto de todas as possíveis configurações que MM pode assumir no seu $(k+1)$-ésimo passo de operação, e $q_k \in Q_k$ é a configuração em que inicia tal passo de operação. Para $k = 0$, tem-se, respectivamente, Q_0, o conjunto inicial de configurações válidas, e $q_0 \in Q_0$, a configuração inicial, tanto para MM_0 como para MM.

- ✓ ε denota "vazio", no contexto em que for empregado, a ausência de qualquer outro elemento válido do conjunto correspondente.

- ✓ Σ é o conjunto (finito, invariável) de todos os possíveis eventos que são possíveis estímulos de entrada para MMA ($\varepsilon \in \Sigma$).

- ✓ $\lambda \subseteq Q$, (respectivamente, $F = Q - \lambda$) é o subconjunto de todos os estados finais (respectivamente, estados não-finais).

- ✓ BA e AA são os conjuntos de ações adaptativas, ambos contendo a ação adaptativa vazia ($\varepsilon \in BA \cap AA$).

- ✓ $w = w_1\ w_2\ ...\ w_n$, $k = 1, ..., n$ é uma cadeia de estímulos não-vazios de entrada, ou seja, $w_k \in \Sigma - \{\varepsilon\}$.

- ✓ Δ, com $\varepsilon \in \Delta$, é um conjunto (finito, invariável) de todos os possíveis símoblos passíveis de serem gerados por MMA como efeitos colaterais da aplicação de regras adaptativas.

- ✓ AR_k é um conjunto de regras adaptativas, dadas por uma relação $AR_k \subseteq BA \times Q \times \Sigma \times (Q \times \Delta^*) \times AA$. Em particular, AR_0 define o comportamento inicial de MMA. Ações adaptativas mapeiam o conjunto corrente de regras adaptativas AR_k de MMA em um novo conjunto AR_{k+1}, através da aplicação de um conjunto de operações de inclusão e/ou remoção de regras adaptativas AR_k. Regras adaptativas $ar \in AR_k$ apresentam-se na forma $ar = (ba, q_i, s, (q_j, \partial_1\partial_2\partial_n,), aa)$, significando que, em resposta a algum estímulo de entrada $ar \in AR_k$, ar executa inicialmente a ação adaptativa $ba \in BA$; a execução de ba é descontinuada caso venha a eliminar ar de AR_k; caso contrário, aplica-se a regra não adaptativa subjacente $r = (q_i, s, (q_j, \delta_s(q_j)))$, conforme descrito anteriormente para o dispositivo não-adaptativo; finalmente, executa-se a ação adaptativa $aa \in AA$.

- ✓ Defina-se AR como o conjunto de todas as possíveis regras adaptativas para o dispositivo MMA.

- ✓ Defina-se R como o conjunto de todas as possíveis regras não-adaptativas para MM, o dispositivo subjacente de MMA.

- ✓ $CA \subseteq BA$ x R x AA , definido para um determinado dispositivo adaptativo MMA, é o mecanismo adaptativo de MMA que deve ser aplicado, em qualquer passo k de operação, a cada regra $R_k \subseteq R$, e deve ser tal que opere como função, quando aplicado a qualquer subdomínio $R_k \subseteq \delta$. Isso determina um único para de ações adaptativas a serem associadas a cada regra não-adaptativa.

Note-se que a definição de AR_k como subconjunto do produto cartesiano BA x Q x ... x AA pode ser refinada, à luz da definição do mecanismo adaptativo CA. Assim, pode-se reinterpretar AR_k como subconjunto do produto cartesiano BA x AR_k x AA.

O conjunto $AR_k \subseteq AR$ pode ser construído como a coleção de todas as regras adaptativas que possam ser obtidas pela associação de pares de ações adaptativas com as regras não-adaptativas em AR_k. Equivalentemente, como as regras de δ_k espelham as de AR_k em cada momento, é possível construir Δ_k simplesmente removendo-se, das regras de AR_k, todas as referências a ações adaptativas.

O algoritmo sugerido em [NETO 2001] é apresentado a seguir e descreve a operação completa de um dispositivo adaptativo qualquer, dirigido por regras:

4.2.1. ALGORITMO DE OPERAÇÃO

O roteiro abaixo descreve de forma geral a operação, a partir do instante $T = 0$, de qualquer dispositivo adaptativo dirigido por regras:

1. Posiciona-se o dispositivo adaptativo em sua configuração inicial;
2. Dispara a função de saída associada ao estado inicial e apresenta a palavra de saída;
3. Posiciona-se a cadeia de entrada para que seja lido em primeiro lugar seu evento mais à esquerda;
4. Caso não haja mais nenhum evento a ser processado, então desviar para o passo 9 deste roteiro, caso contrário, alimente o dispositivo fornecendo-lhe o próximo evento a ser processado, extraído da cadeia de entrada;
5. Escolhe-se a próxima regra adaptativa a ser aplicada: dada a configuração corrente $c_T \in C_T$ e o estímulo $s \in S$, extraído da parte ainda não processada da cadeia de entrada, localizam-se, no conjunto NR_T, as regras compatíveis, ou seja, todas as regras que possam ser aplicadas sob as circunstâncias correntes, coletando-as em um conjunto $CR_T = \{ ar \in AR_T, \mid ar = (ba, c_T, s, c, rs, aa), s \in \{ s_T , \varepsilon \}; c, c_T \in C_T; ba \in BA; aa \in AA \}$.

   Há três casos a considerar:

   - se CR_T for vazio, nenhum movimento será possível para o dispositivo (por ser AR_T incompletamente especificado), então a cadeia de entrada será rejeitada;
   - se $CR_T = \{ ar^k \}$ para alguma $ar^k = (ba^k, c_T, s, c^k, rs^k, aa^k) \in AR_T$, então existirá uma única regra disponível para aplicação determinística.

Dessa forma, o dispositivo alcançará um estado pré-definido na próxima configuração $c_T + 1 = c^k$;

- se $CR_T = \{ ar^k = (ba^k, c_T, s, c^k, rs^k, aa^k) \mid k = 1, 2, ..., m \}$, então todas essas m regras serão igualmente aplicáveis, de forma não-determinística, à configuração corrente. Assim, todas elas devem ser aplicadas em paralelo, como é usual na operação de dispositivos não-determinísticos. Obviamente, em ambientes seqüenciais, o paralelismo deve ser exaustivamente simulado, por exemplo, aplicando-se algum tipo de estratégia de retrocesso à configuração anterior do dispositivo (*backtracking*).

6. Se uma ação adaptativa ba^k for a ação adaptativa ε (vazia) na regra que estiver sendo aplicada, então passar-se-á à execução do passo 6. Caso contrário, aplique-se a ação adaptativa ba^k ao conjunto de regras atuais, gerando uma nova configuração intermediária para o dispositivo AD. Note-se que, em alguns casos, até mesmo a regra adaptativa ar^k, eventualmente, pode ser removida em decorrência da aplicação de ba^k. Nesse caso extremo, a aplicação de ar^k é abortada, retorne ao passo 4;

7. Aplique a regra $nr^k = (c_T, s, c^k, rs^k)$, conforme definido para dispositivo não-adaptativo (subjacente), gerando uma configuração (intermediária) de AD;

8. Dispara a função de saída associada ao estado inicial e apresenta a palavra de saída;

9. Se a ação adaptativa executada aa^k, indicada na regra adaptativa ar^k, for a ação adaptativa ε (vazia), desvie para o passo 4; caso contrário, aplica-se aa^k ao conjunto corrente de regras adaptativas, gerando, finalmente, uma nova configuração do dispositivo. Da mesma forma que no passo 6, a execução da ação adaptativa também pode apagar ar^k. Nesse caso, porém, não acarreta qualquer ação adicional;

10. Se a configuração corrente for uma configuração de aceitação, então a cadeia de entrada será considerada aceita, caso contrário será rejeitada. Em qualquer caso, a operação do dispositivo será finalizada.

4.2.2. FUNÇÕES ADAPTATIVAS

Uma função adaptativa determina quais modificações (ações adaptativas elementares) serão realizadas na camada subjacente do dispositivo. Formalmente, define-se uma função adaptativa como uma 9-upla:
$FA = (F, P, V, G, C, E, I, A, B)$ na qual:

- ✓ F é o nome da função adaptativa.
- ✓ P é a lista $(r_0, r_1, ..., r_m)$ de parâmetros formais.
- ✓ V é a lista $(v_1, v_2, ..., v_n)$ de identificadores de variáveis.
- ✓ G é a lista $(g*_1, g*_2, ..., g*_p)$ de identificadores de geradores.
- ✓ C é a lista de ações de consulta.
- ✓ R é a lista de ações de remoção.
- ✓ I é a lista de ações de inserção.
- ✓ A é uma ação adaptativa inicial, opcional, executada antes de F.
- ✓ B é uma ação adaptativa final, opcional, executada depois de F.

Todas as ações adaptativas são da forma (F', P'), na qual:

- ✓ F' é o nome de uma função adaptativa.
- ✓ P' é a lista $(p_0, p_1, ..., p_k)$ de argumentos a serem passados para F'.

A especificação de uma função adaptativa é iniciada com um mecanismo de passagem de parâmetro por valor automaticamente atribuindo, a cada parâmetro formal, r_i, o valor atualmente atribuído ao argumento p_i, em P'. Depois disso, ocorre a execução da ação adaptativa inicial (opcionalmente), que pode repassar, se necessário, os valores dos parâmetros recebidos por F,

para uma outra função adaptativa. A ação adaptativa inicial, no entanto, não tem acesso a variáveis e geradores, pois seus valores ainda não estão definidos neste ponto da execução da função adaptativa.

Ações elementares possuem, basicamente, a definição formal de uma regra da camada subjacente, possivelmente contendo nomes de variáveis e de geradores no lugar dos elementos de regra. Na execução das ações elementares de consulta, um mecanismo de busca por padrões é responsável por atribuir valores a estas variáveis, tendo como base o conjunto de regras do mecanismo subjacente. Uma vez atribuídos, os valores de cada variável não podem mais ser alterados durante a execução da função adaptativa. Quando o mecanismo de busca por padrões não encontra qualquer regra na camada subjacente que satisfaça o formato determinado pela ação elementar de consulta, as variáveis permanecem indefinidas.

A execução de ações elementares de remoção segue, inicialmente, o mesmo princípio da execução das ações elementares de consulta, com uma busca de padrões sendo utilizada para determinar valores para variáveis (que podem também estar presentes nas ações elementares de remoção). Após a busca, caso todos os valores de variáveis tenham sido definidos, a regra correspondente deve ser eliminada da camada subjacente (caso contrário – existem variáveis indefinidas -, a ação elementar é simplesmente ignorada).

Para a execução das ações elementares de inserção, que ocorre sempre depois da execução das ações de consulta e remoção, todas as variáveis devem ter sido previamente instanciadas (ou marcadas como indefinidas). Para os casos em que todas as variáveis contidas na ação elementar de inserção já estejam definidas, procede-se a inserção das regras correspondentes na camada subjacente. Ações elementares de inserção podem também conter nomes de geradores, ao invés de variáveis. Nesse caso, antes da inserção da nova regra na camada subjacente, todos os geradores são substituídos por novos símbolos, diferentes de qualquer símbolo utilizado na camada

subjacente. Por fim, executa-se a ação adaptativa final, que pode agora fazer referência à variáveis e geradores.

Em [NETO 1993] e [NETO 1994] é proposta uma notação para funções adaptativas que possui basicamente o formato exibido no quadro seguinte, com "padrão" sendo uma estrutura dependente do formato das regras da camada subjacente:

Os prefixos ?, – e + denotam ações elementares de consulta, remoção e inserção, respectivamente.

$F(r_1, ..., r_m) = \{$
$v_1, v_2, ..., v_n, \quad g^*_1, g^*_2, ..., g^*_p :$
$A(p_1, ..., p_k)$
? [padrão]
? [padrão]
...
- [padrão]
- [padrão]
...
+ [padrão]
+ [padrão]
...
$B(p_1, ..., p_k)$
$\}$

Um exemplo de Máquina de Moore Adaptativa escrito através dessa notação será apresentado a seguir.

4.2.3. EXEMPLO DE MÁQUINA DE MOORE ADAPTATIVA

Para ilustrar o funcionamento de uma Máquina de Moore Adaptativa, é apresentado a seguir um simples exemplo de um gerador de código-fonte para um subconjunto das linguagens Pascal e C. Para isso, o transdutor foi posicionado em uma configuração a partir da qual torna-se acionado uma configuração no início do transdutor que determina, de acordo com a cadeia de entrada, qual será a lipossível selecionar o mecanismo de geração de código-fonte analisando-se o prefixo da cadeia de entrada. Com o uso da camada adaptativa no transdutor, tornou-se possível incorporar uma característica importante das linguagens usuais de programação, em que a utilização das variáveis e de outros elementos da linguagem deve ser precedida de sua declaração. A Máquina de Moore Adaptativa deste exemplo possui a topologia inicial mostrada na figura 6.

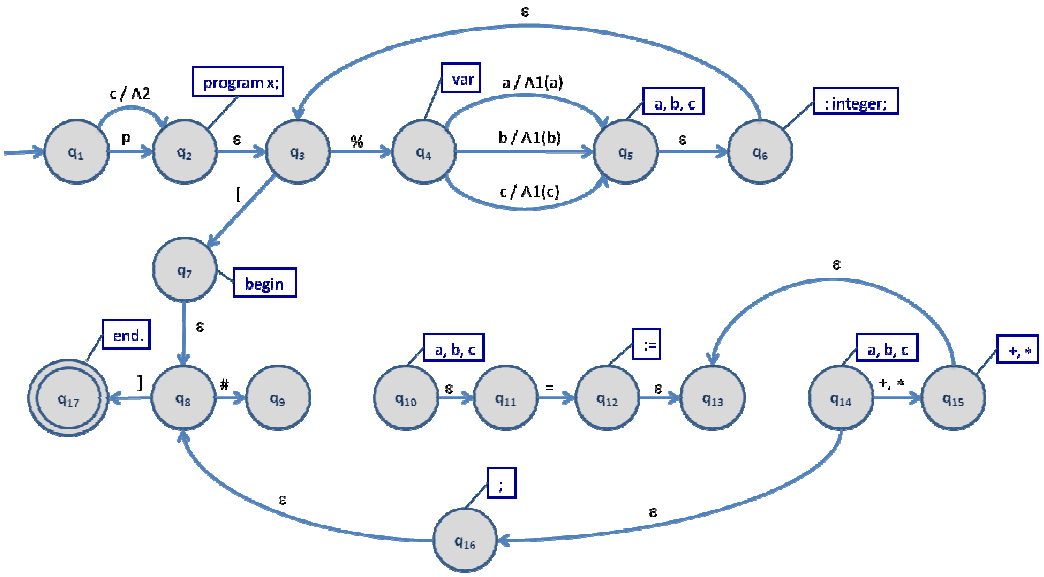

Figura 5: Topologia inicial da Máquina de Moore Adaptativa.

Ao início da execução do transdutor, estando o mesmo no estado q_1, somente dois estímulos são aceitos na cadeia de entrada: p ou c, que representam a escolha da linguagem a qual se deseja gerar o código-fonte como saída,

respectivamente, Pascal ou C. A linguagem Pascal é tida no modelo como padrão. Ao escolher a linguagem C, a função adaptativa A2 é executada e realiza as mudanças necessárias nas saídas do transdutor para se obter o código-fonte esperado.

Esse transdutor possui duas funções adaptativas: A1(σ) e A2, descritas informalmente a seguir:

Função adaptativa A1(σ):

- ✓ Esta função adaptativa é executada após a execução da transição à qual está associada;
- ✓ O parâmetro σ representa o símbolo de entrada associado à transição que invocou a função adaptativa A1;
- ✓ A transição associada ao símbolo de entrada σ, entre os estados q_4 e q_5, é removida do transdutor pela execução da ação adaptativa A (ou seja, a transição adaptativa recém-executada elimina a si própria);
- ✓ Uma nova transição, associada ao símbolo σ, é adicionada por A entre os estados q_9 e q_{10};
- ✓ Uma nova transição, associada ao símbolo σ, é adicionada por A entre os estados q_{13} e q_{14}.

Função adaptativa A2:

- ✓ Esta função adaptativa é executada após a execução da transição à qual está associada;
- ✓ Novas saídas são associadas aos estados q_2, q_4, q_6, q_{12} e q_{17}.
- ✓ Nenhum estado ou transição é modificado.

A operação desta Máquina de Moore Adaptativa, com algumas cadeias de entrada distintas, é ilustrada a seguir:

- ✓ Cadeia de entrada p%a%c[#a=c#c=a+c*c]

Nota-se que, ao consumir o símbolo p, o transdutor se comporta como uma Máquina de Moore típica, simplesmente avançando da configuração q_1, indo até q_2 e apresentando a saída "program x;". Ao consumir %a tem-se a saída "var a: integer;" e a função adaptativa A(a) é executada após consumir o símbolo a na transição do estado q_4 para q_5, causando o seguinte efeito:

- ✓ A transição que vai do estado q_4 para o estado q_5 consumindo o símbolo a é removida;
- ✓ A transição que vai do estado q_9 para o estado q_{10} consumindo o símbolo a é adicionada;
- ✓ A transição que vai do estado q_{13} para o estado q_{14} consumindo o símbolo a é adicionada;

Dessa forma, o autômato passa a aceitar o nome a no trecho de comandos da cadeia de entrada e, adicionalmente, deixa de aceitar novas declarações para esse mesmo nome, conforme a figura 7:

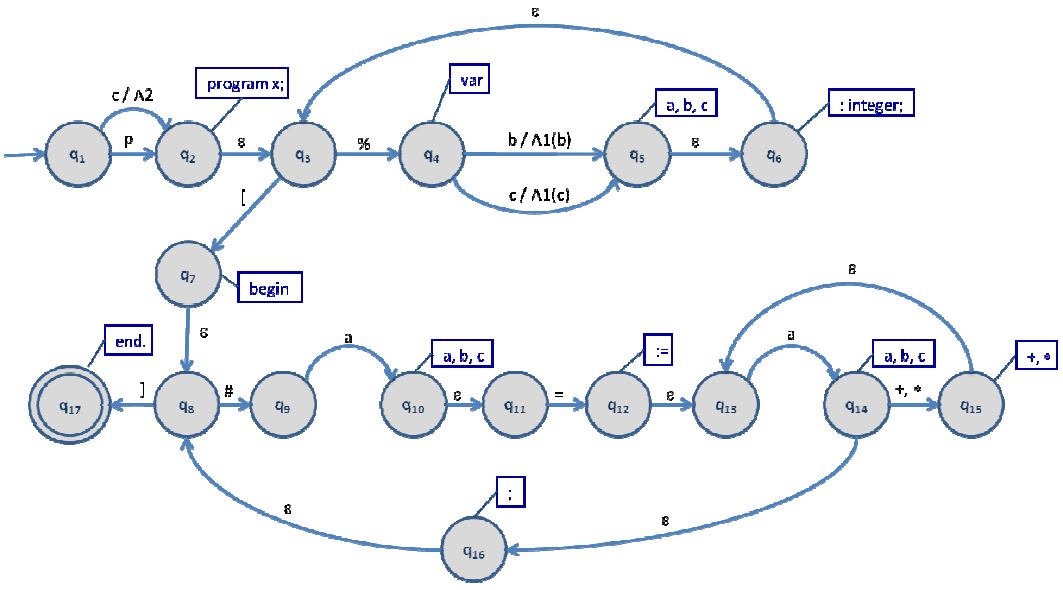

Figura 6: Topologia da Máquina de Moore Adaptativa após consumir *p%a*.

Comportamento semelhante ocorre quando o autômato processa a declaração do segundo nome (entrada %c, na transição que vai do estado q_4 para o estado q_5). A topologia do autômato é igualmente modificada, conforme ilustra a próxima figura. Uma vez efetuadas todas as declarações, o autômato atinge a sua topologia definitiva (vide figura 5) e efetua o reconhecimento do restante da cadeia de entrada ([#a=c#c=a+c*c]) sem novas adaptações. Ele termina a sua seqüência de movimentações no estado q_{17}, que é um estado final, e com a cadeia de entrada esgotada. Logo, a cadeia é aceita.

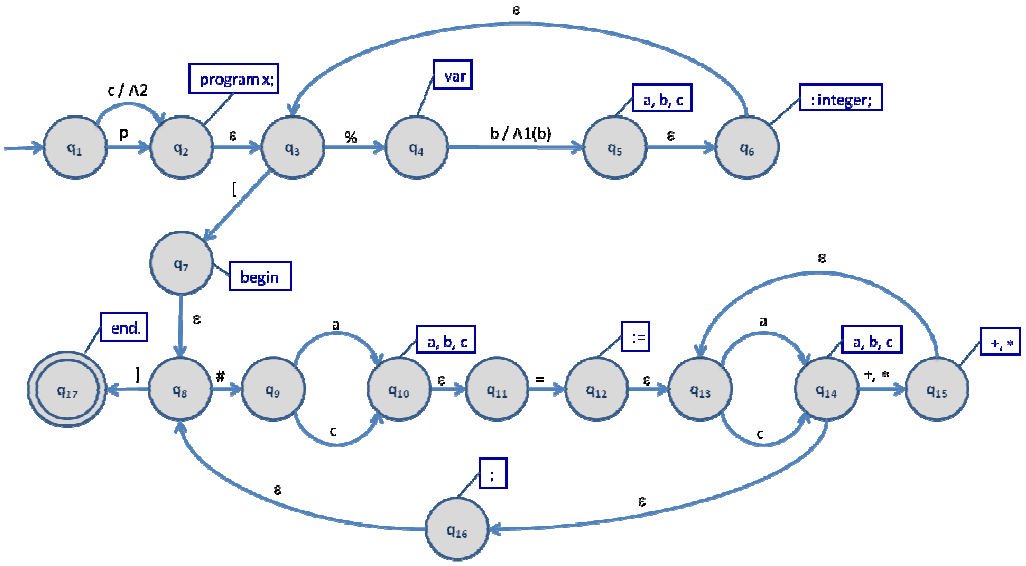

Figura 7: Topologia da Máquina de Moore Adaptativa após consumir *p%a%c*.

✓ Cadeia de entrada p%a%c[#b=c]

Após as declarações dos nomes a e c, a topologia final do autômato é a mesma ilustrada na figura 8. No entanto, a tentativa de se usar o nome b no lado esquerdo de um comando de atribuição faz com que o autômato pare no estado q_9 (após a transição com o símbolo #) sem poder se movimentar, por falta de transição que consuma o nome b a partir do estado q_9. Nessa

configuração, como o estado q_9 não é final e a cadeia de entrada não está esgotada, então a cadeia de entrada é rejeitada.

✓ Cadeia de entrada p%a%c[#a=a*b]

Este caso é similar ao anterior, exceto pelo fato de que um símbolo não declarado previamente (b) é encontrado, no caso, no lado direito do comando de atribuição. Assim, o transdutor adaptativo se movimenta como um autômato finito, no trecho entre o símbolo # e o símbolo *, parando no estado q_{13}, sem novas possibilidades de movimentação, com o símbolo b na estrada. A cadeia de entrada é rejeitada, pois o estado q_{13} não é um estado final. A cadeia de entrada também não está esgotada, o que seria outro motivo suficiente para rejeitá-la.

✓ Cadeia de entrada p%a%c%a[#a=c]

Após as duas primeiras declarações (dos nomes a e c), a tentativa de se declarar novamente o nome a conduz o autômato mais uma vez do estado q_4 para o estado q_5 (na topologia da figura 6), porém desta vez não existe mais a transição que consumiria esse símbolo no estado q_4. Logo, o autômato pára e a cadeia de entrada é rejeitada, uma vez que o estado q_4 não é final e a cadeia de entrada não está esgotada.

✓ Cadeia de entrada c%a%c[#a=c#c=a+c*c]

Nessa cadeia de entrada, o primeiro símbolo indica que a saída gerada pela Máquina de Moore Adaptativa deverá ser um código-fonte em linguagem C. Assim, após consumir o símbolo c na transição do estado q_1 para o estado q_2, a

função adaptativa A2 é executada trocando as saídas do transdutor conforme ilustra a figura 9.

Neste caso, nenhuma transição ou estado foi adicionado ou removido, mas as palavras de saída associadas às configurações dos estados q_2, q_4, q_6, q_{12} e q_{17} foram substituídas.

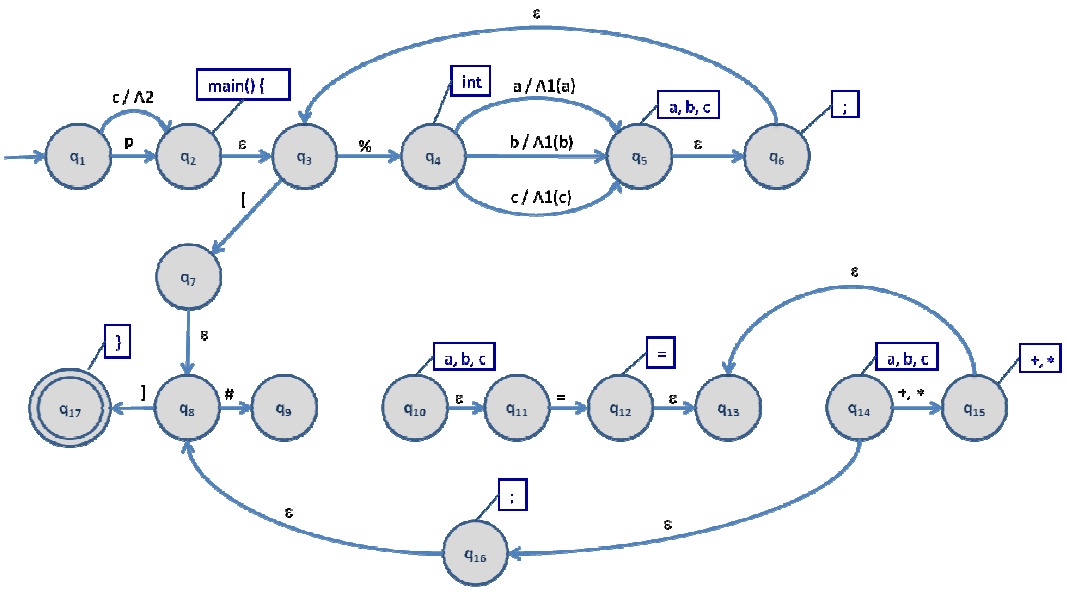

Figura 8: Topologia para uma cadeia de entrada iniciada por *c*.

Após consumir toda a cadeia de entrada, a seguinte saída é gerada:

```
main()
{
    int a;
    int c;
    a = c;
    c = a + c * c;
}
```

Com o uso de Máquina de Moore como dispositivo subjacente, foi possível criar a função adaptativa A2 que transformou o autômato do exemplo num meta-gerador de código-fonte. Sem a necessidade de nenhuma modificação na topologia do autômato, é possível gerar o código-fonte para diferentes linguagens trocando apenas o primeiro *token* da cadeia de entrada. Essa mesma idéia pode ser aplicada, por exemplo, na geração de código binário para diferentes plataformas computacionais.

5. USO DE ADAPTATIVIDADE NA MODELAGEM DE CURSOS

Este capítulo apresenta uma proposta de desenvolvimento de um projeto voltado para modelagem de cursos para softwares educacionais baseada em formalismos adaptativos.

Um dos objetivos da pesquisa é elaborar um ambiente para a modelagem de cursos. A idéia é criar um sistema baseado nos formalismos adaptativos apresentados com Capítulo 2 capaz de representar cursos da maneira mais independente possível, podendo aplicar as diferentes teorias de aprendizagem expostas no Capítulo 3.

Ou seja, o ambiente não deve limitar os métodos pedagógicos adotados pelos professores. Dessa forma, as questões pedagógicas devem ser tratadas e discutidas pelos docentes e não limitadas pela ferramenta. As regras para o acompanhamento do curso são definidas pelo Professor, sendo aplicadas de acordo com a disciplina a ser abordada ou a linha pedagógica do professor.

O ambiente é baseado em hipertexto e desenvolvido para rodar em ambiente *Web*. Assim, a estrutura principal do ambiente é baseada na linguagem *Html*. Sendo a lógica de funcionamento do ambiente controlada pela linguagem *PHP* [PHP 2010]. Para armazenamento das informações persistentes, foi utilizado o gerenciador de banco de dados *MySQL* [MYSQL 2010].

Na representação da proposta, cada curso é definido como uma máquina de Moore e referencia um conjunto de objetos de aprendizagem independentes, tal que o roteiro de estudos do curso permanece separado do material didático. Cada estado representa um tópico de estudo dentro do curso. As transições

compõem o roteiro das aulas. A função de saída funciona como a ligação lógica dos estados com os objetos de aprendizagem.

Dessa forma, cada máquina de Moore constitui um curso definido sobre um conjunto de hiperdocumentos (unidades de informação) independentes. As *n-uplas* dos autômatos de cursos apresentam uma correspondência às estruturas de hiperdocumentos na *Web*. O alfabeto de símbolos de entrada é um conjunto de rótulos que identificam as âncoras de navegação no hipertexto. As saídas estão relacionadas a unidades de informação constituídas por hiperdocumentos e as palavras de saída, presentes na função de saída, são hipertextos apresentados como uma única página *Web* no navegador do usuário. As transições, definidas na função programa, funcionam como ligações lógicas (e não físicas) entre os conteúdos dos cursos e definem possíveis *links* a serem selecionados durante a navegação pelo hipertexto.

Uma das vantagens da solução utilizada é que a estrutura dos autômatos com saída permite a criação do material hipermídia de forma independente do autômato em si, possibilitando a programação de seqüências de estudo com objetivos e enfoques específicos, reuso de parte ou íntegra das páginas Web em diversos cursos, eliminando a redundância na criação de páginas, bem como a modularização que facilita a expansibilidade do sistema.

A estrutura do sistema de hipertexto que não utiliza *links* diretos permite a estruturação e edição do conjunto de materiais hipermídia sem a necessidade do autor dos documentos se preocupar com a inserção de *links* diretamente nos textos, facilitando a tarefa de edição dos documentos. Um benefício adicional é que, como páginas de um curso podem ser constituídas por outras sub-páginas concatenadas, o autor pode construir funções de saída diferentes que contenham acesso as páginas de mesmo conteúdo, evitando-se redundância na criação dos documentos *Html*. Por exemplo, pode existir uma função que gere uma página contendo uma "definição" e um "exemplo",

enquanto outra página pode ter o mesmo "exemplo" como "dica" para um exercício.

A camada adaptativa do modelo permite maior poder de representação, podendo tornar o acompanhamento do curso mais flexível e dinâmico, podendo tratar questões dependentes de contexto. Desta forma, a camada adaptativa tenta antecipar as necessidades e desejos dos alunos a partir de modelos que representam o seu perfil, nível de conhecimento e preferências, ou ainda, pode direcionar o aluno indicando-lhe rotas de estudo a serem seguidas.

A camada adaptativa pode ser aplicada em três níveis:

- ✓ Adaptatividade na navegação
- ✓ Adaptatividade na apresentação
- ✓ Adaptatividade no conteúdo

A adaptatividade na navegação tem por objetivo apontar um caminho ideal para que um aluno em particular alcance seus objetivos, evitando que o mesmo se disperse, diante de um amplo conjunto de opções e sinta-se desorientado diante da grande quantidade de material oferecido. A adaptação na navegação é obtida acrescentando-se novas transições no modelo do curso ou, ainda, removendo-se algumas das transições existentes. Uma vez que as transições definem o roteiro de aulas do curso, pode-se manter o mesmo conteúdo programático, porém oferecer seqüências alternativas para seguir do curso. De acordo com o perfil de cada aluno, pode-se ofertar um número maior de caminhos a serem seguidos ou restringir essas possibilidades, tornando o curso mais linear e assim limitar o espaço de busca. As regras adaptativas nesse nível incidem exclusivamente sobre a criação e remoção de transições.

A adaptatividade na apresentação diz respeito à forma como as informações serão exibidas para os alunos, dependendo do perfil de cada aluno. Ou seja, um mesmo conteúdo pode ser apresentado utilizando-se diferentes layouts ou

mesmo através de diferentes mídias, conforme as preferências do usuário. Nota-se, também, que o material a ser apresentado ao aluno não está necessariamente pronto e pode ser gerado em tempo real. Essencialmente, a adaptatividade na apresentação implica em se modificar as palavras de saída associadas aos estados.

A adaptatividade no conteúdo refere-se a possibilidade de se substituir um objeto de aprendizagem por outro mais elaborado, ou mesmo mais simplificado, dependendo da situação. Com isso, o curso pode ser atualizado trocando-se as palavras de saída ou até mesmo criando-se novos estados, com tópicos complementares não contidos no curso original. Esse nível de adaptatividade permite, por exemplo, a elaboração de cursos baseados em aprendizagem por projeto, ou ainda, trabalhos desenvolvidos em grupo, explorando o potencial da Inteligência Coletiva [LÉVY 1998].

A fim de demonstrar a aplicabilidade de dispositivos adaptativos na modelagem de cursos, as próximas seções deste capítulo trazem:

- ✓ A representação do modelo estático de dados para Máquina de Moore Adaptativa. A representação é feita através de diagrama de classes da *Unified Modeling Language* (UML) [BOOCH et al, 2000].
- ✓ Alguns exemplos ilustrativos do funcionamento do modelo para representar cursos, tratar questões que envolvem dependência de contexto através de pré-requisitos de conteúdos
- ✓ Um protótipo de um curso de Algoritmos no qual cada aluno possui sua própria instância do curso.

5.1. REPRESENTAÇÃO DO MODELO

A representação interna do modelo usado foi feita utilizando-se diagrama de classes da UML [BOOCH et al, 2000]. O diagrama de classes é usado para descrever os tipos de objetos do sistema e os vários tipos de relacionamentos estáticos existentes entre eles. Os objetos são representados através de abstrações chamadas de classes, representadas por seus atributos e métodos. Os métodos das classes foram omitidos nessa representação.

Ao elaborar a representação do modelo através de diagrama de classes, foi tomado o devido cuidado para não se perder as características pertinentes ao formalismo adaptativo apresentado no Capítulo 2 desta tese.

Primeiramente, é representada apenas a camada subjacente: no caso, a camada com a Máquina de Moore. Posteriormente, é apresentada a camada adaptativa do dispositivo. Somente, então, é apresentada uma representação unificada das duas camadas. A idéia foi evidenciar a separação e a independência das camadas do dispositivo adaptativo, como proposto originalmente em [NETO 2001]. No mesmo sentido, isso apresenta como característica que a troca da camada do dispositivo subjacente em nada afetará a camada que trata das funções e ações adaptativas, o que torna o modelo bastante modular. Por fim, ainda são feitas algumas complementações no modelo para que o mesmo seja capaz de representar diversas instâncias de máquinas simultaneamente.

As subseções a seguir apresentam os passos na modelagem do sistema, mostrando o respectivo diagrama de classes e descrevendo detalhadamente sua classes, seus relacionamentos e seus atributos.

5.1.1. CAMADA SUBJACENTE DO DISPOSITIVO

A figura 10 representa uma Máquina de Moore (não-adaptativa), que será usada na camada do dispositivo subjacente.

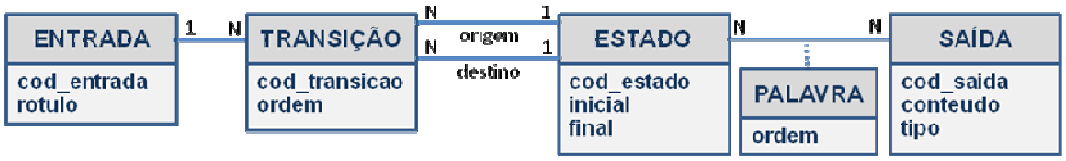

Figura 9: Representação da camada subjacente do dispositivo.

A classe “entrada” representa o conjunto (finito) de todos os possíveis eventos que são estímulos de entrada válidos da máquina. Os símbolos de entrada são representados pelo atributo ‘cod_entrada’, enquanto o atributo ‘rótulo’ foi criado para efeito de implementação. O rótulo possui o conteúdo a ser apresentado na interface para que seja selecionada a correspondente entrada escolhida.

A classe “estado” representa o conjunto de todas as suas possíveis configurações da máquina através do atributo ‘cod_estado’. Nesta mesma classe, os atributos inicial e final representam, respectivamente, a configuração inicial da máquina e os estados que formam o conjunto de estados finais.

A classe “transição” representa o conjunto de regras que definem o funcionamento da máquina. Conforme item 2.3.1, em resposta a um estímulo de entrada qualquer, uma regra modifica a configuração corrente para uma nova configuração após consumir um determinado símbolo de entrada. Para isso, cada transição precisa possuir dois estados associados, sendo um de origem (estado atual) e outro de destino, além de um símbolo de entrada, responsável por disparar a transição. Novamente, para efeito de implementação, foram incorporados dois atributos adicionais: ‘cod_transicao’, como um identificador único de cada transição; e, o atributo ‘ordem’, com a

finalidade de organizar a exibição das entradas válidas para cada configuração da máquina.

Cada transição possui uma única entrada, mas uma mesma entrada pode ser válida para diversas transições. Da mesma forma, um mesmo estado pode servir de origem e/ou destino para uma ou mais transições.

A classe “saída” representa, neste contexto, os materiais didáticos disponíveis. O atributo ‘cod_saida’ é usado como identificador único de cada material. O atributo ‘conteúdo’ representa efetivamente o conteúdo do material didático, que pode ser de diferentes formatos. Alguns tipos de formato possíveis são: Texto, Imagem, Áudio e Vídeo. O atributo ‘tipo’ é usado para o software saber como o material didático deverá ser exibido.

Num mesmo estado podem-se ter diferentes saídas associadas. Uma mesma saída também pode estar associada a diferentes estados. Desse relacionamento, surge a classe “palavra”, que representa a geração das palavras de saída associadas a cada estado, representando a função de saída da Máquina de Moore.

Se fosse desejado trabalhar, por exemplo, com Máquina de Mealy, bastaria retirar o relacionamento entre as classes “estado” e “saída” e criar um novo relacionamento entre as classes “transição” e “saída” para a geração da classe “palavra”. Isso acontece devido ao fato das máquinas de Moore e de Mealy terem comportamentos parecidos, evidentemente, por ambas serem derivadas dos autômatos determinísticos. Para outros formalismos, é necessário um estudo particular. Contudo, o mais importante aqui é notar que nessa fase de modelagem o foco é exclusivamente dado ao dispositivo subjacente escolhido, sem se preocupar com a camada adaptativa.

5.1.2. CAMADA ADAPTATIVA DO DISPOSITIVO

Uma vez definida a estrutura da camada subjacente, o próximo passo é a modelagem da camada adaptativa desse modelo. Abaixo, é possível ver, na figura 11, o diagrama de classes para representação da camada adaptativa.

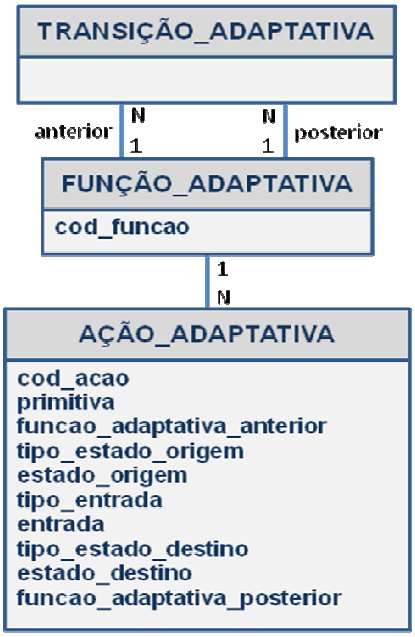

Figura 10: Representação da camada adaptativa do dispositivo.

Na parte superior do diagrama, tem-se a classe “transição_adaptativa”. Essa classe possui dois relacionamentos com a classe função_adaptativa, uma vez que é possível ter-se uma função adaptativa anterior e outra posterior para cada transição adaptativa. Posteriormente, esta classe transição adaptativa será usada como elo da camada adaptativa com a camada subjacente do dispositivo.

Ao centro do diagrama apresentado na figura anterior, tem-se a classe “função_adaptativa”, que possui apenas um atributo chamado cod_func usado como identificador. Uma função adaptativa é formada por uma lista de ações adaptativas.

A classe “ação_adaptativa” representa as ações adaptativas elementares responsáveis pelas modificações a serem impostas ao conjunto de regras da camada subjacente. O atributo ‘primitiva’ representa o tipo de ação adaptativa elementar, podendo ser de consulta, exclusão ou inclusão. Este atributo é, portanto, um campo multivalorado que pode conter os valores ?, - e +, respectivamente, para ações de consulta, exclusão e inclusão. Os atributos funcao_adaptativa_anterior e funcao_adaptativa_posterior são usados quando a ação elementar for de inclusão e essa inclusão envolver outras funções adaptativas. Os atributos tipo_estado_origem, tipo_entrada e tipo_estado_destino servem para dizer se os respectivos valores para estado_origem, entrada e estado_destino são constantes, variáveis, geradores ou parâmetros (c, v, g, p). Finalmente, os atributos estado_origem, entrada e estado_destino representam as novas regras a serem criadas quando a ação adaptativa for do tipo inclusão.

Esse diagrama da camada adaptativa foi elaborado de forma genérica, de maneira a poder ser acoplado a qualquer tipo de dispositivo subjacente. A princípio, seria possível omitir a classe “transição_adaptativa” e fazer um relacionamento da classe “função_adaptativa” diretamente como a classe “transição” da camada subjacente. Isso até simplificaria a representação do modelo, porém, ira ferir a independência da camada subjacente, indo contrário ao formalismo proposto em [NETO 2001]. Por isso, optou por criar uma classe extra responsável por fazer a ligação das duas camadas.

A próxima subseção mostra essa ligação entre as camadas, através do diagrama de classes completo para representar uma Máquina de Moore Adaptativa.

5.1.3. JUNÇÃO DAS DUAS CAMADAS DO DISPOSITIVO ADAPTATIVO

Uma vez modeladas as duas camadas do dispositivo adaptativo, basta agora fazer a junção num único diagrama, conforme apresentado a seguir. Como citado anteriormente, do ponto de vista de implementação, seria possível fundir as classes “transição” e “transição_adaptativa” numa única classe, mas para manter a compatibilidade do modelo original proposto em [NETO 2001], optou-se por essa separação para que as duas camadas do dispositivo adaptativo possam ser tratadas de forma independente. Seguindo o modelo, cada vez que uma nova entrada válida ocorrer, verifica-se se existe uma transição adaptativa a ser executada. Em caso afirmativo, executa-se a função adaptativa anterior, seguida da transição definida na máquina e, por fim, executa-se a função adaptativa posterior. Se não houver uma transição adaptativa associada, simplesmente se executa a transição (não-adaptativa).

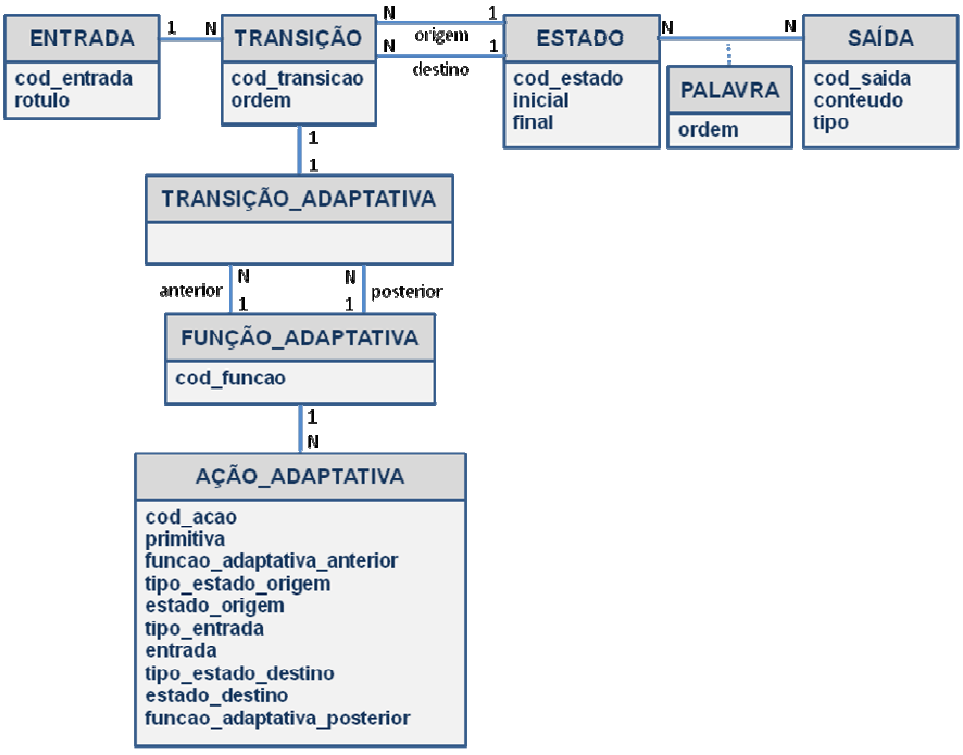

Figura 11: Camada adaptativa acoplada à camada do dispositivo subjacente.

5.1.4. MODELO AJUSTADO PARA TRABALHAR COM DIVERSOS CURSOS

O modelo proposto no item 4.1.3 permite a representação de qualquer MMA e, portanto, para diferentes tipos de curso, independente dos critérios didático-pedagógicos que possam ser aplicados pelo docente. Contudo, ele representa apenas uma única máquina, ou seja, o modelo consegue representar um único curso de cada vez. Para permitir a representação de diversas instâncias de cursos utilizando a mesma representação, algumas classes foram incorporadas, como pode ser visto do diagrama da figura 13.

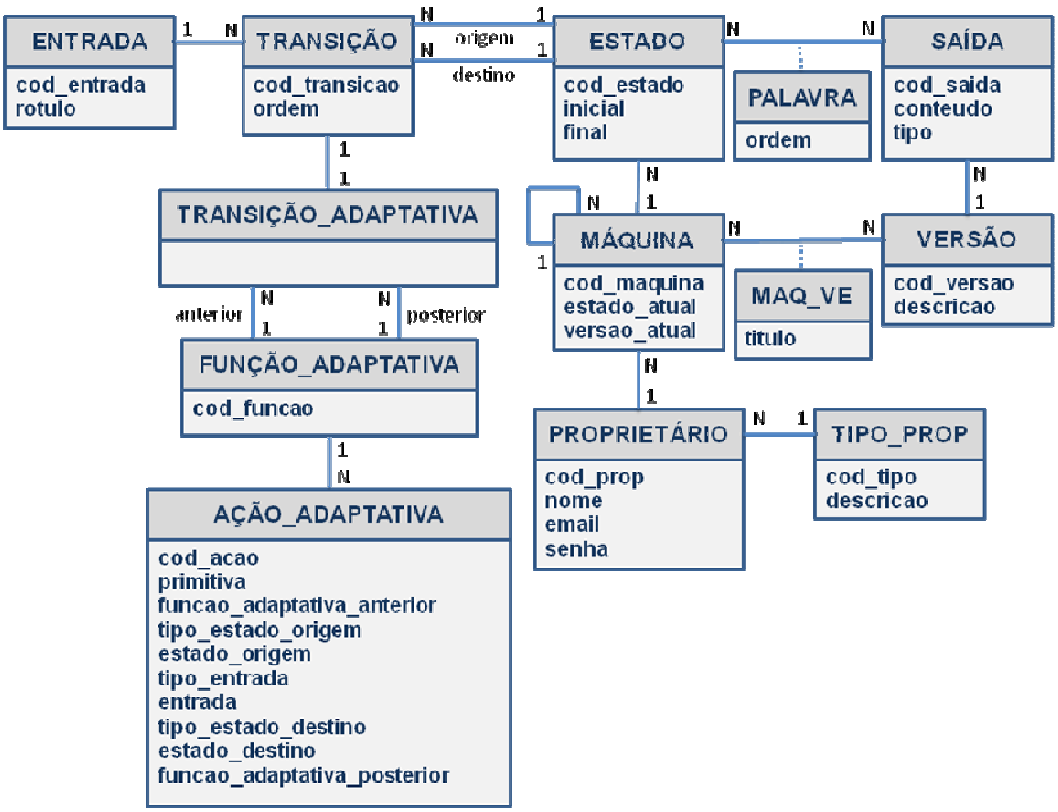

Figura 12: Diagrama ajustado para aceitar múltiplas instâncias de cursos.

A classe “máquina” do diagrama acima permite que esse modelo estenda o diagrama de classes de forma que represente não apenas um único curso, mas sim, quantas máquinas (cursos) forem necessárias. Basicamente, a máquina identificará um curso através de seu título/nome. O atributo ‘estado_atual’ servirá apenas como um marcador persistente do estado atual de cada máquina na ocorrência de interrupção do andamento do curso (seja essa interrupção proposital ou involuntária). Outros atributos como carga horária, nível escolar, etc. também poderiam ser introduzidos nessa classe. Existe um auto-relacionamento na classe “máquina”. O objetivo desse relacionamento recursivo é diferenciar as máquinas originais criadas pelos professores das demais instâncias pertencentes a cada aluno. Assim, cada clone gerado para os alunos irá manter um vínculo com a máquina original, na qual o aluno iniciou o curso. Caberá aos professores cadastrarem novos cursos. Mas, para cada aluno matriculado num curso, será necessário gerar um clone do curso original. Na medida que cada aluno for avançando nos estudos, apenas a sua própria máquina será atualizada, independente do andamento das aulas dos demais alunos e sem afetar a máquina do curso original criada pelo docente.

Na classe “proprietário” foram definidos os atributos: nome, e-mail e senha. Outros atributos como endereço, telefone, etc. também poderiam ser incluídos, não sendo colocados apenas para efeito de simplificar o diagrama. Cada proprietário possui um tipo, que pode ser, por exemplo, ‘professor’, ‘tutor’, ‘aluno’ ou ‘administrador’. Cada tipo de proprietário será usado para o sistema diferenciar os níveis de acesso do usuário no ambiente. Cada proprietário pode ser dono de um ou mais máquinas, ou seja, um professor pode ministrar um ou mais cursos, assim como um aluno pode acompanhar um ou mais cursos.

Existe, também, uma classe “versão”, criada para se reaproveitar um mesmo roteiro de aulas mudando-se apenas o conteúdo apresentado. Isso pode ser útil, por exemplo, para oferecer um mesmo curso em idiomas diferentes.

5.2. EXEMPLOS DE MODELAGEM DE CURSO

Com a finalidade de exemplificar o uso de formalismos adaptativos, são apresentados alguns exemplos de situações típicas encontradas na modelagem de cursos e como podem ser representadas.

5.2.1. REPRESENTAÇÃO DE CURSOS (SEM FUNÇÕES ADAPTATIVAS)

Para ilustrar, é apresentado um simples exemplo de como um curso qualquer pode ser modelado através de uma Máquina de Moore (não-adaptativa). A representação é feita por uma heptupla, conforme proposto no item 4.1.

$ND = (C, NR, S, c_0, A, NA, RS)$

$C = \{ q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8 \}$

$c_0 = q_1$

$S = \{ p, a, e, r, s \}$

- p – próximo
- v – voltar
- e – exercício
- r – resumos
- s – saída

$A = \{ q_8 \}$

$NA = \{ A, B, C, D, E, F, G, H \}$

- A – Introdução do curso
- B – Teoria sobre o assunto X
- C – Exemplo sobre o assunto X
- D – Exercícios sobre o assunto X
- E – Teoria sobre o assunto Y
- F – Exemplo sobre o assunto Y
- G – Exercícios sobre o assunto Y
- H – Conclusões

$NR = \{ (q_1, p, q_2), (q_2, e, q_3), (q_2, p, q_4), (q_3, v, q_2), (q_4, e, q_5), (q_4, p, q_6), (q_5, v, q_4), (q_6, r, q_7), (q_6, s, q_8), (q_7, v, q_6) \}$

$RS = \{ (q_1, A), (q_2, BC), (q_3, D), (q_4, EF), (q_5, G), (q_6, H), (q_7, ABE) \}$

A figura a seguir ilustra a configuração de Máquina de Moore apresentada anteriormente, com suas regras de transição e regras de saída para a representação de um curso qualquer.

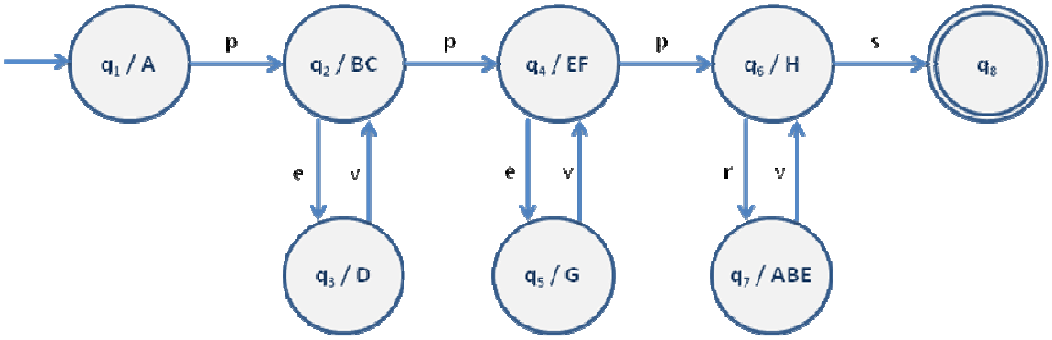

Figura 13: Exemplo de modelagem de um curso com Máquina de Moore

Como é característico em qualquer Máquina de Moore, para cada estado atingido, a função de saída determina a geração de uma palavra de saída (que eventualmente pode ser vazia). Assim, estando no estado inicial q_1, a função de saída irá exibir A, que representa o material didático referente a "Introdução do curso". Recebendo como entrada 'p', que representa o próximo tópico do curso, avança-se para a configuração q_2 e, em seguida, são exibidos os materiais B e C, respectivamente, representando a "Teoria sobre o assunto X" e "Exemplo sobre o assunto X". E, dessa forma, a cada nova entrada válida obtém-se uma nova configuração e são exibidos os materiais didáticos relacionados na função de saída. É interessante notar a capacidade de reuso dos materiais didáticos. Isso pode ser observado, por exemplo, no estado q_7 da máquina, que apresenta ABE, sendo esses conteúdos já utilizados anteriormente pela mesma máquina.

Na implementação desse modelo em ambiente hipertexto, por exemplo, pode-se entender que as entradas são fornecidas pelo aluno através de sua navegação pelo site, clicando-se em *links* que o permitem navegar pelo conteúdo do curso.

A figura 15 mostra como fica a organização do curso dentro do modelo relacional mapeado pelo diagrama apresentado no item 4.1.1. Esses dados podem, por exemplo, estar gravados nesse formato dentro de um banco de dados, ou carregados em memória através de vetores. É possível simular a realização do curso e rastrear sua execução acompanhando seu estado atual e sua palavra de saída, de acordo com o consumo de cada estímulo da cadeia de entrada.

ENTRADA

cod_entrada	rotulo
p	próximo
v	voltar
e	exercício
r	resumos
s	saída

ESTADO

cod_estado	inicial	final
q1	S	N
q2	N	N
q3	N	N
q4	N	N
q5	N	N
q6	N	N
q7	N	N
q8	N	S

SAÍDA

cod_saida	conteúdo	tipo
A	Introdução do curso	texto
B	Teoria sobre o assunto X	texto
C	Exemplo sobre o assunto X	texto
D	Exercícios sobre o assunto X	texto
E	Teoria sobre o assunto Y	texto
F	Exemplo sobre o assunto Y	texto
G	Exercícios sobre o assunto Y	texto
H	Conclusões	texto

TRANSIÇÃO

cod_transicao	cod_origem	cod_entrada	cod_destino	ordem
t01	q1	p	q2	1
t02	q2	e	q3	1
t03	q2	p	q4	2
t04	q3	v	q2	1
t05	q4	e	q5	1
t06	q4	p	q6	2
t07	q5	v	q4	1
t08	q6	r	q7	1
t09	q6	s	q8	2
t10	q7	v	q6	1

PALAVRA

cod_estado	cod_saida	ordem
q1	A	1
q2	B	1
q2	C	2
q3	D	1
q4	E	1
q4	F	2
q5	G	1
q6	H	1
q7	A	1
q7	B	2
q7	E	3

Figura 14: Modelo relacional para representação da Máquina de Moore.

A seguir, são simulados os resultados para a cadeia de entrada: **p p p r v s**. As próximas figuras ilustram as mudanças na configuração da máquina juntamente com os saídas resultantes.

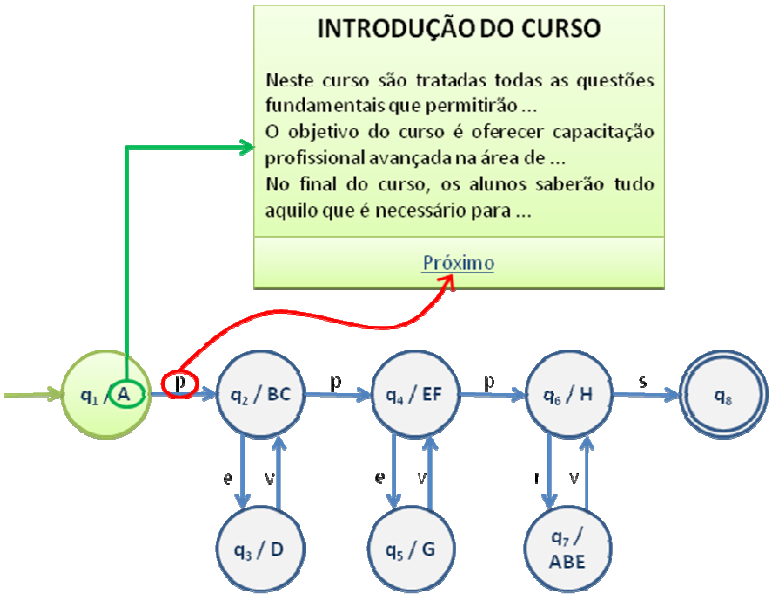

Figura 15: Configuração inicial da máquina no estado q_1, exibindo a saída A.

A figura 16 apresenta a máquina em seu estado inicial q_1, que possui a palavra de saída 'A' associada. Assim que a máquina é colocada nesse estado, a função de saída se encarrega de formar e apresentar a palavra associada ao estado atual. O quadrilátero na parte superior da figura representa um dispositivo físico de saída qualquer, por exemplo, a tela de um monitor. Nele, é exibido o conteúdo do material didático associado à saída 'A', juntamente com os *links* que indicam as entradas válidas para o estado atual, no caso, a entrada *p*, cujo rótulo exibido é *próximo*.

A máquina mantém-se no corrente estado até que o aluno termine de ler o material didático exibido e clique no *link* com a opção *próximo*. Essa ação provoca a transição do estado q_1 para o estado q_2 consumindo o símbolo *p*. Ao atingir o estado q_2, novamente se aciona a função de saída, exibindo-se o correspondente material didático, dessa vez, formado por BC. Estando a máquina na configuração q_2, são apontadas as próximas entradas válidas: *e* e *p*, respectivamente para acessar os exercícios ou avançar ao próximo tópico do curso. A figura 17 ilustra essa configuração.

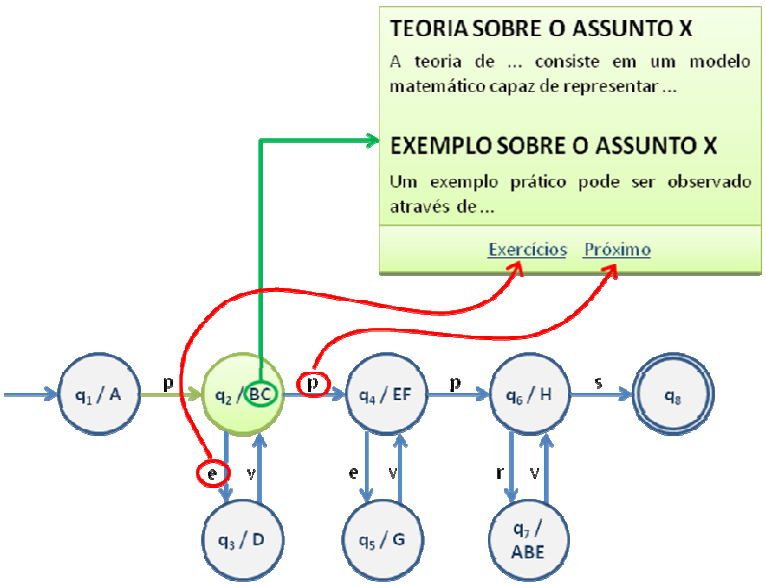

Figura 16: Configuração exibindo a saída BC e as próximas entradas válidas.

Assumindo que o aluno optou por não realizar os exercícios sobre o tópico atual e optou por avançar para o próximo assunto usando a da entrada p. A figura 18 ilustra a nova configuração da máquina.

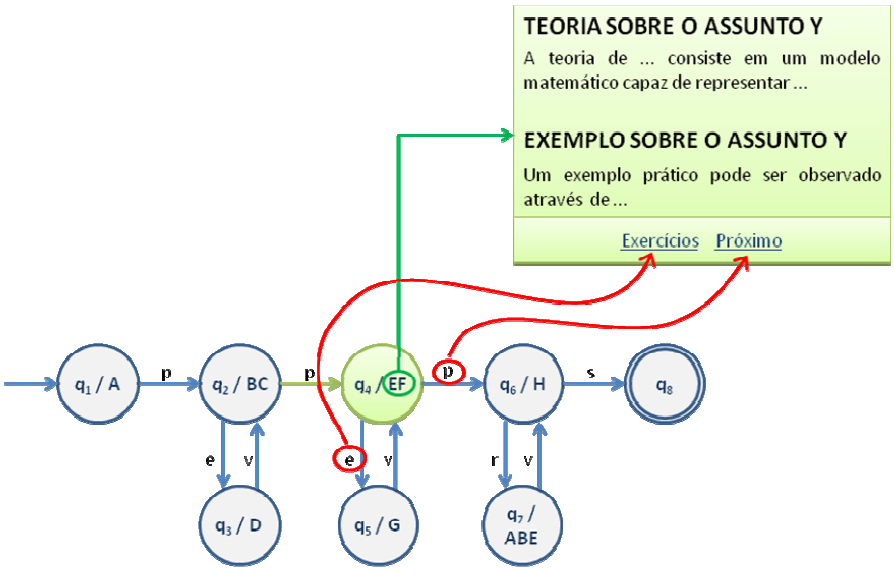

Figura 17: Nova configuração com as respectivas saídas e próximas entradas.

Uma nova entrada p leva o aluno até o estado q_6, que possui as conclusões acerca do curso. A figura 19 ilustra esse novo estado atingido.

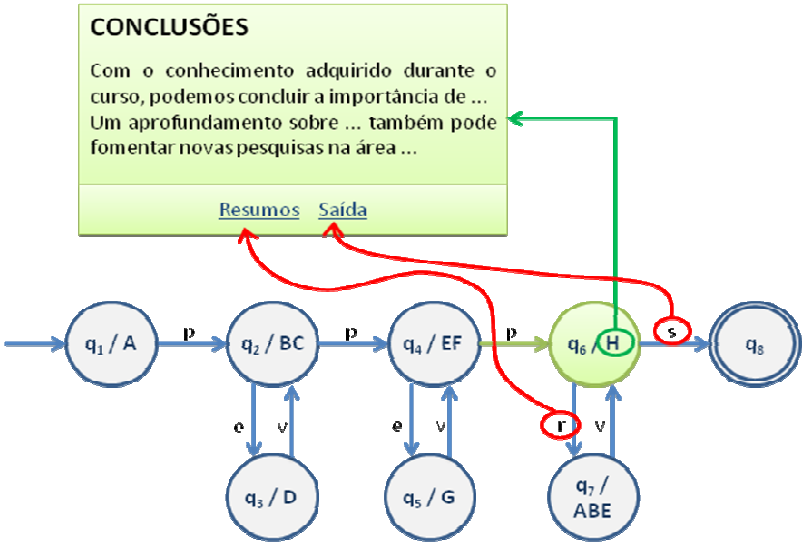

Figura 18: Configuração em q_6, exibindo as conclusões do curso.

Note que, estando no estado q_6, que apresenta as conclusões do curso, é possível acessar o estado q_7 através do estímulo r, representado através do rótulo *resumos*. No estado q_7 é apresentada a palavra de saída ABE. Note que as saída A, B e E já foram usadas para formar outras palavras de saída da máquina. Isso significa que o mesmo material didático está sendo reutilizado no modelo.

Assim, quanto mais modular for a criação do material didático, aproveitando o conceito de objeto de aprendizagem, mais facilmente esse conteúdo poderá ser reaproveitado em diferentes situações. Por exemplo, ao invés de criar uma lista de exercícios e uma avaliação, é possível criar diversas questões separadas e, então, poder fazer uso dessas questões tanto em listas de exercícios quanto em provas, selecionando as questões mais adequadas para cada situação, sem precisar refazer o material.

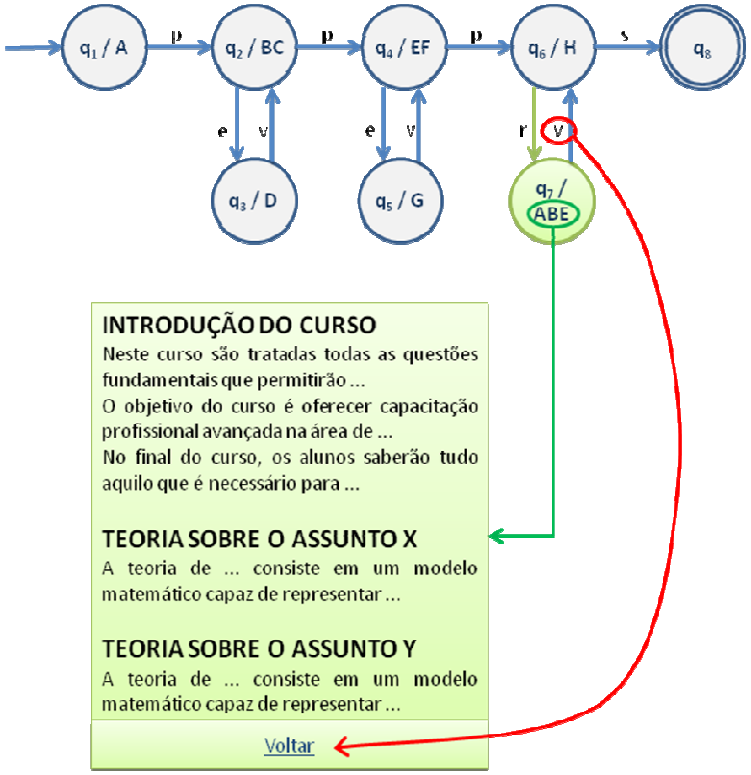

Figura 19: Resumos dos conteúdos abordados fazendo reuso do material didático.

Depois de ler os resumos, a única entrada válida do exemplo é o estímulo *v* para voltar para as conclusões. A figura 21 ilustra a configuração da máquina após a final do curso, finalizando no estado final q_8.

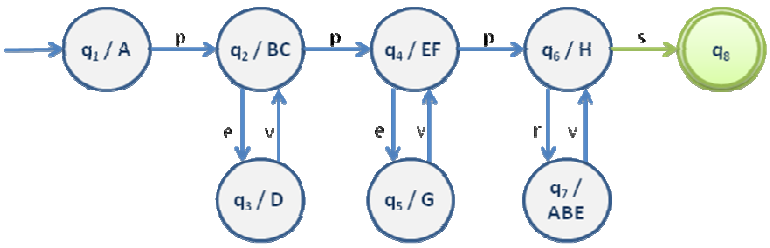

Figura 20: Configuração da máquina no estado q_8, finalizando o curso.

A estruturação do curso fica a critério do professor responsável pelo curso, podendo ser montadas estruturas mais rígidas, forçando o aluno a seguir um

determinado caminho seqüencial de aulas, ou um roteiro de aulas mais flexível, no qual o aluno possa escolher o caminho a ser percorrido durante seus estudos. Para isso, basta que sejam criadas novas transições para a máquina.

5.2.2. TRATAMENTO DE PRÉ-REQUISITO

Em algumas situações, é desejável que o aluno possua conhecimentos prévios sobre um determinado assunto para poder explorar novos conceitos. Ou, ainda, ter conhecimentos básicos sobre um determinado conteúdo para poder estudar assuntos mais avançados dessa mesma matéria. Nessa situação, é desejável que o sistema faça o tratamento de pré-requisitos, ou seja, o sistema deve considerar o conhecimento do aluno dentro do contexto do curso.

Para exemplificar, supondo que o aluno escolha um tópico qualquer para estudar, o mesmo só terá acesso ao conteúdo básico desse assunto. Mas, ao acessar o conteúdo básico, uma função adaptativa pode criar a transição necessária para se acessar o conteúdo avançado desse tópico. Nesse exemplo, a função adaptativa A1 da figura 23 executa três ações adaptativas:

a.) remove a própria transição do estado “tópico” para o estado “básico”

b.) adiciona uma nova transição de “tópico” para “básico” através da entrada ‘b’, mas, desta vez, sem a função adaptativa A1.

c.) adiciona uma nova transição do estado inicial “tópico”, consumindo o símbolo de entrada “a” e atingindo o estado de destino “avançado”.

Antes de acompanhar a execução desse exemplo, é apresentada, a seguir, a configuração do curso através do modelo relacional. Note que, dessa vez, a camada adaptativa também está sendo utilizada.

ESTADO

cod_estado	inicial	final
q1	S	N
q2	N	N
q3	N	N

ENTRADA

cod_entrada	rótulo
b	básico
a	avançado
v	voltar

SAÍDA

cod_saida	conteúdo	tipo
T	Tópicos abordados	texto
B	Conceitos básicos	texto
A	Conceitos avançados	texto

TRANSIÇÃO

cod_transicao	cod_origem	cod_entrada	cod_destino	ordem
t01	q1	b	q2	1
t02	q2	v	q1	1
t03	q3	v	q1	1

PALAVRA

cod_estado	cod_saida	ordem
q1	T	1
q2	B	1
q3	A	1

TRANSIÇÃO_ADAPTATIVA

anterior	transicao	posterior
	t01	A1

FUNÇÃO_ADAPTATIVA

cod_funcao
A1

AÇÃO_ADAPTATIVA

cod_acao	cod_funcao	primitiva	anterior	tipo_origem	origem	tipo_entrada	entrada	tipo_destino	destino	posterior
1	A1	-		C	q1	C	b	C	q2	A1
2	A1	+		C	q1	C	b	C	q2	
3	A1	+		C	q1	C	a	C	q3	

Figura 21: Modelo relacional para o controle de pré-requisito.

Agora, a figura 23 ilustra a situação inicial da máquina quando o estado atual é "tópico" e nenhum estímulo foi consumido.

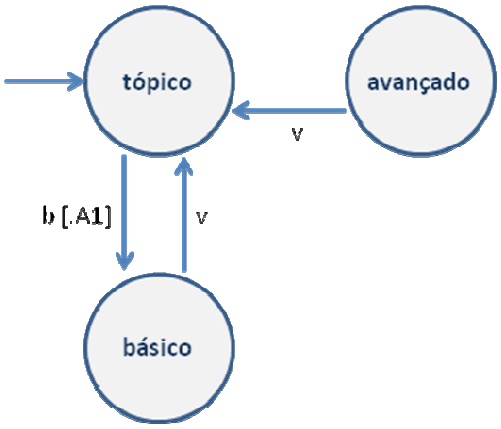

Figura 22: Módulo avançado inacessível devido ao pré-requisito não realizado.

Avançando para o estado "básico" através do estímo "b", são apresentados os conceitos básicos de um tópico qualquer, habilitando o acesso ao conteúdo "avançado" do mesmo tópico, através da execução da função adaptativa A1.

Abaixo, é possível acompanhar a execução das três ações adaptativas elementares associadas a função adaptativa A1. No primeiro passo, a ação adaptativa remove a própria transição, pois essa função adaptativa só deverá ser invocada uma única vez. A seguir, uma nova transição é criada, mas desta vez, sem a existência da função adaptativa A1. E, finalmente, a terceira ação adaptativa cria uma nova transição do estado q_1 para o estado q_3, permitindo que o aluno possa acessar o material avançado.

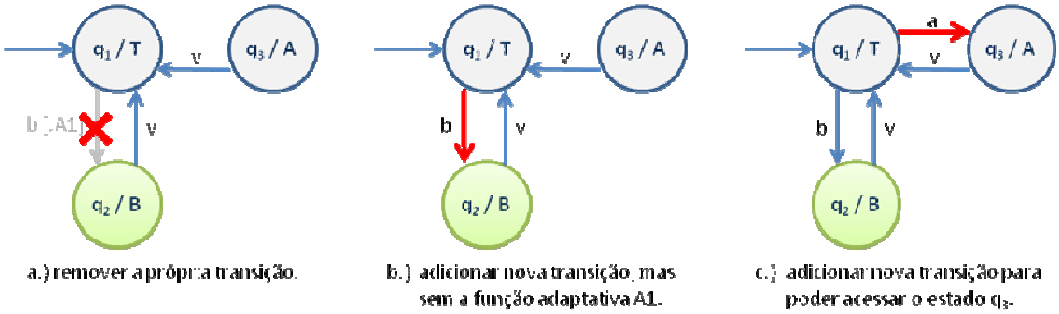

Figura 23: Passos de execução da função adaptativa A1.

A figura 25 ilustra a nova configuração da máquina desse curso após a execução da função adaptativa A1.

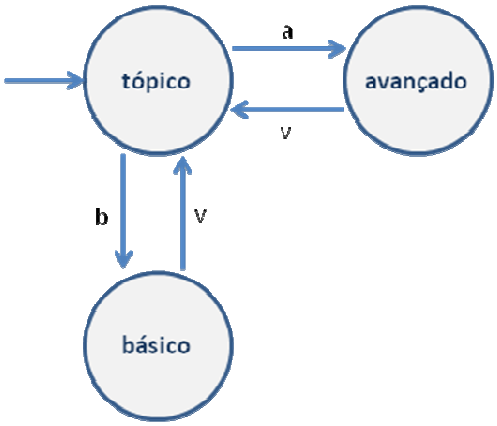

Figura 24: Módulo avançado disponível após cumprimento do módulo básico.

5.2.3. CONTROLE DE REALIZAÇÃO DE PROVA

Uma outra situação que pode ser desejada é exigir que o aluno possa realizar uma prova somente após realizar os exercícios sobre a teoria apresentada. E, ainda, a prova só pode ser realizada uma única vez pelo aluno.

Estando no estado inicial "teoria", na configuração da máquina do exemplo a seguir, no tempo $t = 1$, o único caminho válido é atingir o estado "exercício" através do estímulo "e". Nesse caso, a função adaptativa A1 é disparada, executando três ações adaptativas:

a.) remove a própria transição do estado "teoria" para o estado "exercício"

b.) adiciona uma nova transição de "teoria" para "exercício" através da entrada 'e', mas, desta vez, sem criar a função adaptativa A1.

c.) adiciona uma nova transição do estado "teoria", consumindo o símbolo de entrada "p" e atingindo o estado de destino "prova". Ao criar essa transição, uma nova função adaptativa (A2) é disponibilizada para a configuração da máquina no tempo $t = 2$.

O mapeamento do curso é ilustrado na figura 26.

ESTADO

cod_estado	inicial	final
q1	S	N
q2	N	N
q3	N	N

ENTRADA

cod_entrada	rótulo
e	exercício
p	prova
v	voltar

SAÍDA

cod_saida	conteúdo	tipo
T	Teoria	texto
E	Exercícios	texto
P	Prova	texto

TRANSIÇÃO

cod_transicao	cod_origem	cod_entrada	cod_destino	ordem
t01	q1	e	q2	1
t02	q2	v	q1	1
t03	q3	v	q1	1

PALAVRA

cod_estado	cod_saida	ordem
q1	T	1
q2	E	1
q3	P	1

FUNÇÃO_ADAPTATIVA

cod_funcao
A1
A2

TRANSIÇÃO_ADAPTATIVA

anterior	transicao	posterior
	t01	A1

AÇÃO_ADAPTATIVA

cod_acao	cod_funcao	primitiva	anterior	tipo_origem	origem	tipo_entrada	entrada	tipo_destino	destino	posterior
1	A1	-		C	q1	C	e	C	q2	A1
2	A1	+		C	q1	C	e	C	q2	
3	A1	+		C	q1	C	p	C	q3	A2
4	A2	-		C	q1	C	p	C	q3	

Figura 25: Modelo relacional para o controle de realização de prova,

A seguir, é apresentada a configuração inicial do curso.

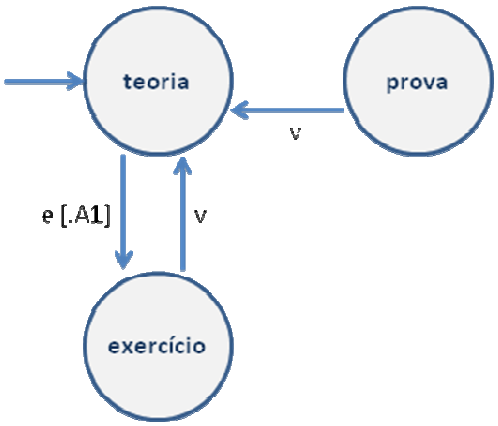

Figura 26: Tempo t=1, com prova não disponível ao aluno.

Nessa configuração inicial, é compulsório que se realize os exercícios antes de fazer a prova. Note que não há uma transição válida do estado “teoria” para o

estado “prova”. Porém ao acessar os exercícios, a função adaptativa A1 é disparada, atingindo a nova configuração com tempo t=2.

Com o contador de tempo t=2, o aluno tem disponibilizada uma transição para que possa acessar e realizar sua prova. Nada impede que o aluno refaça os exercícios antes de realizar a prova. Mas, a realização da prova só pode ser feita uma única vez. Portanto, estando no estado “teoria”, com o contador de tempo t=2 e tendo como estímulo de entrada “p”, o aluno terá acesso ao conteúdo da prova. Mas essa transição implica na execução da função adaptativa A2 (vide figura a seguir), que possui uma única ação adaptativa: remover a própria transição, tornando a prova inacessível novamente após sua realização.

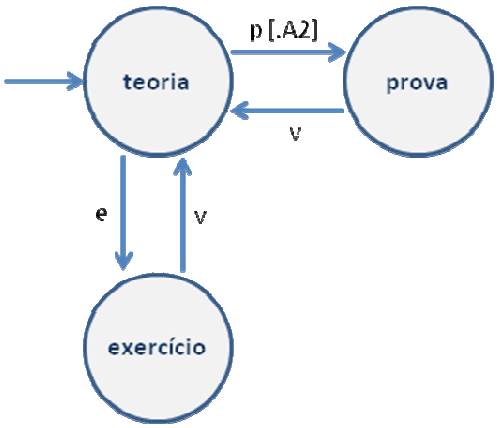

Figura 27: Tempo t=2, com a prova disponível após realização de exercícios.

A configuração da máquina com t=3 é parecida com a configuração inicial t=1. Contudo, note na figura 29 que a função adaptativa A1 não mais existe. Isso quer dizer que o aluno pode fazer os exercícios novamente, mas esta ação não lhe permitirá realizar novamente a prova.

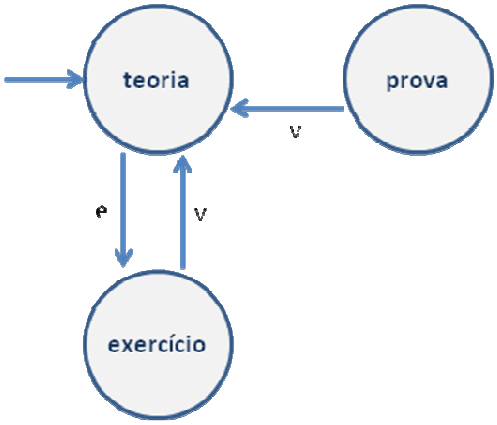

Figura 28: Tempo t=3, com a prova inacessível após sua realização.

5.3. DESENVOLVIMENTO DO PROTÓTIPO

Foi desenvolvido um protótipo no qual ao carregar o sistema, o usuário é direcionado para uma tela de controle de acesso (*login*). Após se autenticar, é apresentado um menu com os cursos cujo aluno esteja previamente matriculado. Após a escolha do curso desejado, é realizada uma chamada à sub-máquina correspondente, passando de "A" e retornando em "B" após a realização do curso. Ao término do curso, retorna-se ao menu de escolha de cursos. As figuras 30 e ilustram o autômato do funcionamento geral do sistema.

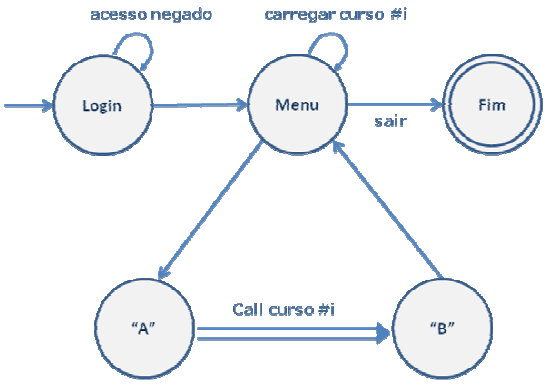

Figura 29: Autômato do esquema das chamadas aos cursos disponíveis.

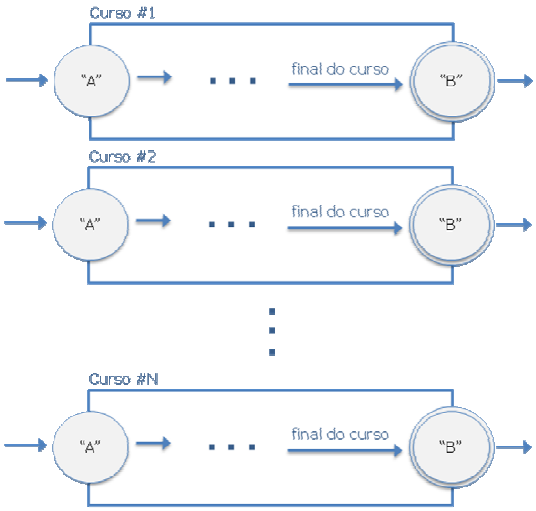

Figura 30: Representação da chamada de sub-máquinas dos cursos.

Cada chamada à sub-máquina irá carregar a instância da Máquina de Moore Adaptativa respectiva ao aluno que esteja utilizando o sistema.

O sistema foi projetado para operar em ambiente *Web* e foi desenvolvido através da linguagem PHP [PHP 2010]. Assim, para se utilizar o sistema é preciso apenas possuir um *browser* (navegador) instalado e estar conectado à Internet (apenas de ser possível o uso do sistema mesmo sem possuir conexão de rede, restringindo-se a execução de algumas atividades).

Para poder efetuar testes, foi modelado um curso de Algoritmos que se encontra disponível a partir do site www.pcs.usp.br/~lta. O protótipo criado possui as funcionalidades necessárias para um aluno matriculado poder acompanhar o curso. Não foi elaborada uma interface visual para o professor elaborar seus materiais didáticos e o roteiro de aulas de seus cursos. Para isso, no momento, a estrutura do curso é carregada diretamente na base de dados do sistema.

6. CONCLUSÕES

Esta tese buscou contribuir para o desenvolvimento de aplicações para softwares educacionais empregando formalismos adaptativos. Procurou-se trazer contribuições tanto para a área teórica quanto para a área prática. Na área teórica, apoiando-se na representação geral para dispositivos adaptativos elaborada por Neto [NETO 2001], foi definido um dispositivo adaptativo: Máquina de Moore Adaptativa, como uma variação dos autômatos adaptativos capaz de representar palavras de saída associadas a cada estado da máquina. Na área prática, foi desenvolvida uma aplicação, baseada no modelo proposto, utilizando-se dos formalismos adaptativos na modelagem de cursos para softwares educacionais. Buscou-se criar um modelo genérico, capaz de permitir a criação de cursos para diferentes níveis de ensino e através de diferentes práticas pedagógicas.

Com base nos estudos realizados, foi possível propor uma estrutura de dados capaz de representar os elementos conceituais do dispositivo adaptativo definido. A elaboração do diagrama de classes para representação interna do modelo mostrou-se de fundamental importância, pois facilitou significativamente a codificação do software. Essencialmente, todo conteúdo dos cursos, incluindo não só o material didático, mas também o roteiro de apresentação desse material e o controle de contexto do aluno no curso são inseridos dentro do modelo. O software do sistema, por sua vez, apenas realiza a leitura das informações relativas às configurações da máquina, executa as funções adaptativas (quando existentes), movimenta a máquina para a próxima configuração válida, gera e apresenta o conteúdo de saída.

O uso de um modelo baseado em Máquina de Moore Adaptativa para projetar sistemas de ensino assistidos por computador apresentou um grande potencial

para se criar ambientes de ensino flexíveis, que se ajustem ao perfil de cada estudante, respeitando ritmo de aprendizagem individual de cada discente.

O protótipo criado é capaz de executar o modelo proposto de Máquina de Moore Adaptativa, representando cursos sobre diferentes assuntos, obedecendo ao roteiro de aulas previstas e controlando, por exemplo, questões como pré-requisitos entre módulos.

Uma das características importantes, que ficou evidenciada na proposta apresentada, é a possibilidade de se criar o material didático de forma independente do curso. A independência do material didático em relação ao roteiro das aulas permite, por exemplo: reutilização do material instrucional em diversos cursos. Outra característica importante é a possibilidade de se criar, para um mesmo curso, diferentes roteiros de aulas, com enfoques diferenciados. Outra característica importante do modelo é a possibilidade de tratamento de dependências de contexto, possibilitando um controle mais efetivo sobre as atividades desenvolvidas pelos alunos.

A possibilidade de criação de novas versões para um mesmo curso também é um recurso interessante. Esse recurso pode ser usado tanto para oferecimento de um mesmo curso em idiomas diferentes, aproveitando-se do mesmo roteiro de aulas, quanto para oferecer o mesmo conteúdo do curso, por exemplo, em versão texto para usuários com conexão de rede limitada e outra versão em formato multimídia para usuários que dispõe de banda larga.

A seguir, são expostas as principais contribuições obtidas por este trabalho assim como sugestões para trabalhos futuros.

6.1. CONTRIBUIÇÕES

As principais contribuições alcançadas por desta tese são elencadas a seguir:

- ✓ Definição do dispositivo adaptativo baseado em Máquina de Moore;
- ✓ Aplicação prática do formalismo adaptativo da Máquina de Moore Adaptativa na elaboração de cursos para softwares educacionais.
- ✓ Definição de um modelo lógico para mapeamento dos conceitos da formulação geral para dispositivos adaptativos;
- ✓ Proposição e criação de um ambiente para modelagem de cursos para softwares educacionais;
- ✓ Elaboração de exemplos de cunho didático para ilustrar o uso de dispositivos adaptativos;
- ✓ Realização de estudo de caso valendo-se do dispositivo definido e da ferramenta desenvolvida;
- ✓ Proposição do uso do modelo para outras aplicações, em especial, voltadas para a Internet;

6.2. TRABALHOS FUTUROS

Os trabalhos desenvolvidos nesta tese abrem uma série de oportunidades para novas pesquisas na área dos dispositivos adaptativos. Entre os avanços tecnológicos que poderão ser perseguidos no futuro se destacam:

- ✓ Criar uma ferramenta de autoria, que ofereça uma interface gráfica amigável para que o professor possa montar o roteiro do curso, sem a necessidade de ser treinado em Tecnologia Adaptativa.
- ✓ Monitorar o comportamento de diversos alunos e permitir que os materiais didáticos do curso e/ou roteiros de estudos do curso possam ser modificados/criados pelo grupo;
- ✓ Aplicar em campo (sala de aula) o ambiente proposto, para que seja possível coletar e analisar os resultados obtidos junto aos alunos;
- ✓ Expandir o conceito usado para curso, para que o mesmo possa ser aplicado, por exemplo, numa grade curricular;
- ✓ Elaborar provas adaptativas;
- ✓ Desenvolver softwares educacionais direcionados ao ensino infantil, fundamental, médio, pós-médio, universitário, treinamento corporativo, educação continuada ou educação a distância;
- ✓ Criar softwares educacionais baseados no modelo behaviorista, cognitivista ou construtivista, ou ainda, cursos híbridos que façam uso de cada uma das teorias nas situações mais apropriadas;

REFERÊNCIAS

AGASANDJAN, G.A.; ***Automata with a Variable Structure***; Dokl. Akad. Nauk SSSR, vol. 174, pp. 529-530, **1967**.

ALMEIDA Jr, J. R.; ***STAD – Uma Ferramenta para Representação e Simulação de Sistemas Através de Statecharts Adaptativos***; Tese de Doutorado; Escola Politécnica; Universidade de São Paulo; São Paulo; 202p;**1995**.

AYCOCK, J.; ***A Program Execution Model Based on Generative Dynamic Grammars;*** IASTED CST 2003 – Cancun, México; **2003**.

BACHARACH, J. & **PLAYFORD**, K. ***The Java Syntactic Extender***; disponível em http://www.ai.mit.edu/~jrb/jse/jse.pdf. 2001.

BASSETO, B. A. & NETO, J. J.; ***A Stochastic Musical Composer Based on Adaptive Algorithms***; Anais do XIX Congresso da Sociedade Brasileira de Computação – SBC´99; PUC-Rio; Vol. 3; pp. 105-113; Rio de Janeiro-RJ; **1999**.

BASSETO, B. A.; ***Lassus*** (Compositor Musical de Corais Tonais a Quatro Vozes). http://www.pcs.usp.br/~lta/download/lassus.zip; **1999**.

BASSETO, B. A.; ***Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos***; Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, **2000**

BOOCH, Grandy & **RUMBAUGH,** James & **JACOBSON**, Ivar;***UML: Guia do Usuário***. Ed. Campus, 1ª edição, Rio de Janeiro, **2000**.

BOULLIER, P.; ***Dynamic Grammars and Semantic Analysis***; INRIA Research Report 2322, August **1994**.

BRANSFORD, John D. et al.;***How People Learn: Brain, Mind, Experience, and School***. National Academy Press, Washington, D.C., **1999**.

BRAVO, C. et al. ***Towards an Adaptive Implementation of Genetic Algorithms*** INBI 2007, XXXIII CLEI – Conferencia Latinoamericana de Informática, San José, Costa Rica, **2007**.

BRUSILOVSKI, P.; ***Methods and Techniques of Adaptive Hypermedia***. User Modeling and User Adapted Interaction. v.6, n.2-3, pp.87-129, **1996**.

BRUSILOVSKY, Peter; ***Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies***; In: Proceedings of Workshop www-based tutoring, 4th International Conference on Intelligent Tutoring Systems, San Antonio, TX, **1998**.

BURSHTEYN, B.; ***On the Modification of the Formal Grammar at Parse Time;*** ACM SIGPLAN Notices Vol. 25, No. 5, pp. 117-123,**1990**

BURSHTEYN, B.; ***Generation and Recognition of Formal Languages by Modifiable Grammars***, ACM SIGPLAN Notices 25 no. 12 pp. 45-53. **1990**

CABASINO, S.; **PAOLUCCI**. P.S.; **TODESCO**, G.M.; ***Dynamic Parsers and Evolving Grammars***, ACM SIGPLAN Notices, 27(11), 39-48, **1992**.

CAMOLESI, A. R. & **NETO**, J. J.; ***Modelagem Adaptativa de Aplicações Complexas***. XXX Conferencia Latinoamericana de Informática – CLEI'04. Arequipa - Peru, Setiembre 27 - Octubre 1, **2004**.

CAMOLESI, A. R., **NETO**, J. J.; ***Modelagem AMBER_Adp de um Ambiente para Gerenciamento de Ensino a Distância***; XIII SBIE – Simpósio Brasileiro de Informática na Educação; Rio Grande do Sul; **2002**.

CAMOLESI, A. R.; ***Proposta de um Gerador de Ambientes para a Modelagem de Aplicações usando Tecnologia Adaptativa***; Tese de Doutorado; Escola Politécnica; Universidade de São Paulo; São Paulo; **2007**.

CAMPOS, Gilda Helena Bemardino de; ***Construção e validação de ficha de avaliação de produtos educacionais para microcomputadores***. Rio de Janeiro, Dissertação (Mestrado) — Faculdade de Educação, UFRJ, **1989**.

CARMI, A.; ***Adapser: An LALR(1) Adaptive Parser,*** The Israeli Workshop on Programming Languages & Development Environments, Haifa, Israel, **2002**.

CAYA, R. & ZAPATA C., A.; ***Estudio de la mejora de la calidad de voz para um sintetizador en idioma castellano usando el método de Autómatas Adaptativos***. III Workshop de Tecnologia Adaptativa, São Paulo-SP, **2009**.

CEFAECO, Centro de Formação de Associação de Escolas do Conselho de Ovar;***Concepção e Desenvolvimento de Projectos de Ensino a Distância***. Disponível em http://www.prof2000.pt/users/ajlopes/af24/plano_sessao_2.htm Site visitado em fevereiro de **2010**.

CEREDA, P.R.M. & **ZORZO**, S.D.; ***Access Control Model Formalism using Adaptive Automaton***. IEEE Latin America Transactions. Volume 6, Issue 5, ISSN: 1548-0992, September **2008**.

CEREDA, P.R.M.; ***Modelo de Controle de Acesso Adaptativo***. Dissertação de Mestrado, UFSCAR, Sâo Carlos/SP, **2008**.

CHRISTIANSEN, H.; ***A survey of adaptable grammars***. SIGPLAN Notices, vol.25, n.11, p.35-44, **1990**.

CONWAY, M.E.; ***Design of a Separable Transition-Diagram Compiler***; Communications of the ACM vol 6 n. 7, pp. 396-408. July, **1963**.

COSTA, E. R., **HIRAKAWA**, A. R., **NETO**, J. J.; ***An Adaptive Alternative for Syntactic Pattern Recognition***. Proceeding of 3rd International Symposium on Robotics and Automation, ISRA 2002, , pp. 409-413. Toluca, Mexico, **2002**.

DI FIORINO, A.C.; ***Some Remarks on the Syntax of Symbolic Programming Languages***, Communications of the ACM, 6(8), 456–460, **1963**.

DIZERÓ, W.J. & NETO, J.J., A.; ***Uso de Máquina de Moore Adaptativa na Modelagem de Cursos***. III Workshop de Tecnologia Adaptativa, São Paulo-SP, **2009**.

FAGUNDES, Léa et al. ***Aprendizes do Futuro: as inovações começaram!*** Coleção Informática para a Mudança na Educação. Ministério da Educação. Secretaria da Educação a Distância. Programa Nacional de Informática na Educação, **1999**.

FREITAS, A.V. & **NETO**, J. J.; ***Adaptive Languages and a New Programming Style*** 6th WSEAS International Conference on Applied Computer Science (ACS'06) – Tenerife, Canary Islands, Spain, December, **2006**.

GARRISON, D. R. & **SHALE**, D.; ***Mapping the Boundaries of Distance Education: Problems in Defining the Field***; The American Journal of Distance Education, 1997.

HANFORD, K.V.; **JONES**, C.B.; ***Dynamic Syntax: A Concept for the Definition of the Syntax of Programming Languages***, Annual Review in Automatic Programming, volume 7, 115–142. **1974**.

HAREL, D. ***Statecharts: A visual formalism for complex systems*** Science of Computer Programming Volume 8, Issue 3. Pages: 231 – 274. June **1987**.

HARRISON, M. A.; ***Introduction to Formal Language Theory***; Ed. Addison-Wesley; 1a edição; Califórnia – USA, 1978.

HOPCROFT, J. E. & **ULLMAN**, J. D.; ***Introduction to Automata Theory, Languages and Computation***, Addison-Wesley; **1979**.

IEEE Institute of Electrical and Electronics Engineers, ***Inc. Draft Standard for Learning Object Metadata – IEEE*** **(1484.12.1-2002)**. The Learning Technology Standards Committee.

ISO9126-1. (1997) International Organization for Standardization. ***"Information technology - Software quality characteristics and metrics - Part 1: Quality characteristics and sub-characteristics"***. ISO/IEC 9126-1 (Commitee Draft).

IWAI, M. K.; ***Um Formalismo Gramatical Adaptativo para Linguagens Dependentes de Contexto*** ; Tese de Doutorado; Escola Politécnica; Universidade de São Paulo; São Paulo; 191p.; 2000.

JACKSON, Q.T.; ***Adapting to Babel: Adaptivity and Context-Sensitivity in Parsing,*** Ibis Publications, Plymouth, Massachusetts, March **2006**.

KEEGAN, D.; ***The Foundations of Distance Education***; London Croom Helm, 282pp, 1986.

KEUNG, K.M. & **TYAGI**, A.; ***State Space Reconfigurability: An Implementation Architecture for Self Modifying Finite Automata;*** Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems 2006, Seoul, Korea October, **2006**.

KLEIN, A. & **KUTRIB**, M.; ***Self-Assembling Finite Automata***. COCOON 2002: 310-319 Proceedings. Lecture Notes in Computer Science 2387 Springer-Verlag, **2002**.

KRITHVASAN, K. ***Time Variant Pushdown Automata***. International Journal of Computer Mathematics, Vol 24, pp. 223-236. **1988**.

KRITHVASAN, K.; ***Time Variant Finite Automata***. International Journal of Computer Mathematics, Vol 19, pp. 103-123. **1986**.

LÉVY, Pierre. ***A inteligência coletiva'***. São Paulo; Ed. Loyola, **1998**.

LEWIS, H. R. & **PAPADIMITRIOU**, C.H.; ***Elements of The Theory of Computation***; Ed. Bookman; 2ª edição; 1998.

LOMET, D. B.; ***A Formalization of Transition Diagram Systems***; Journal of the Association for Computing Machinery, Vol. 20, No. 2, pp. 235-267, April **1973**.

LTA - Laboratório de Tecnologias Adaptativas; Poli-USP; São Paulo; http://www.pcs.usp.br/~lta/; site visitado em Maio/2004.

LUCENA, Marisa; ***Diretrizes para a capacitação do professor na área de tecnologia educacional: critérios para a avaliação de software educacional***. Programa de Engenharia de Sistemas e Computação; COPPE/UFRJ; Rio de Janeiro; **1990**.

LUZ, J.C.; ***Tecnologia Adaptativa Aplicada à Otimização de Código em Compiladores;*** Dissertação de Mestrado, EPUSP, São Paulo, **2004**.

MACHADO, J. P.; ***Autômatos Finitos: um formalismo para cursos na Web***; Simpósio Brasileiro de Engenharia de Software; **1999**.

MACHADO, J. P.; ***Hyper-Automaton: hipertextos e curso na web usando autômatos finitos com saída***; Disertação de Mestrado Universidade Federal do Rio Grande do Sul, Instituto de Informática. Porto Alegre (RS); **2002**.

MATSUNO, I.P.; ***Um Estudo do Processo de Inferência de Gramáticas Regulares e Livres de Contexto Baseados em Modelos Adaptativos***. Dissertação de Mestrado, USP, São Paulo, **2006**.

McGETTRICK, A.D.; ***The Definition of Programming Languages***; Cambridge Computer Science Series Texts – 11, Cambridge University Press, **1980**.

MEEK, B.; ***The Static Semantics File***, ACM SIGPLAN Notices Vol. 25, No. 4, pp. 33-42, **1990**.

MENEZES, C.E.D.; ***Um Método para a Construção de Analisadores Morfológicos, Aplicado à Língua Portuguesa, Baseado em Autômatos Adaptativos***. Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, **2000**.

MYSQL; ***MySql Community***; Disponível em: http://www.mysql.org. Acesso em: fevereiro de **2010**.

NETO, J. J. & **MAGALHÃES**, M.E.S.; ***Reconhecedores Sintáticos – Uma Alternativa Didática para Uso em Cursos de Engenharia***. XIV Congresso Nacional de Informática, pp. 171-181, São Paulo, **1981**.

NETO, J. J.; ***Introdução à Compilação***. Editora LTC, Rio de Janeiro, **1987**.

NETO, J. J.; ***Uma Solução Adaptativa para Reconhecedores Sintáticos***; Anais EPUSP – Engenharia de Eletricidade – série B, vol. 1, pp. 645-657, São Paulo, **1988**.

NETO, João José; ***Contribuição à metodologia de construção de compiladores***; Tese de Livre Docência; Escola Politécnica; Universidade de São Paulo; São Paulo; **1993**.

NETO, J. J.; ***Adaptive Automata for Context-dependent Languages***; ACM SIGPLAN Notices; v.29; n.9; p.115-124; **1994**.

NETO, J. J. & **ALMEIDA** Jr. J. R. & **SANTOS**, J. M. N. ***Synchronized Statecharts for Reactive Systems*****.** Proceedings of the IASTED International Conference on Applied Modelling and Simulation, Honolulu, Hawaii, p.246-251, **1998**.

NETO, J. J. & **IWAI**, M.K.; ***Adaptive Automata for Syntax Learning***. CLEI 98 – XXIV Conferencia Latinoamericana de Informática, MEMORIAS. pp. 135-149, Quito, Equador, **1998**.

NETO, J. J., **PARIENTE**, C.B., **LEONARDI**, F.; ***Compiler Construction – a Pedagogical Approach***. Proceedings of the V International Congress on Informatic Engineering – ICIE 99, Buenos Aires, Argentina, **1999**.

NETO, João José; ***Adaptive Rule-driven Devices – General Formulation And Case Study***; CIAA 2001 - 6ª International Conference on Implementation and Application of Automata; África do Sul; **2001**.

NETO, J. J. & **PARIENTE**, C. A. B.; ***Adaptive Automata - a reduced complexity proposal***; CIAA 2002 - 7ª International Conference on Implementation and Application of Automata; **2002**.

NETO, J. J. & **MORAES**, M. ***Using Adaptive Formalisms to Describe Context-Dependencies in Natural Language***. Lecture Notes in Artificial Intelligence. Computational Processing of the Portuguese Language 6th International Workshop, PROPOR 2003, Volume 2 721, pp 94-97, Faro, Portugal, Springer-Verlag, **2003**.

NETO, J. J.; ***Autômatos em Engenharia de Computação – Uma Visão Unificada*** Primera Semana de Ciencia y Tecnología de la Sociedad Chotana de Ciencias y la Red Mundial de Científicos Peruanos, Junio 22-27, Ciudad de Chota, Perú, **2003**.

NETO, J. J.; ***Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa***; IEEE Latin America Transactions; vol. 5 n° 7; novembro **2007**.

PAGAN, F.G.; ***Formal Specification of Programming Languages: a Panoramic Primer***, Prentice-Hall **1981**.

PALAZZO, L. A. M.; ***Modelos Proativos para Hipermídia Adaptativa***; Tese de Doutorado; Universidade Federal do Rio Grande do Sul, Instituto de Informática. Porto Alegre (RS); **2000**.

PALAZZO, L. A. M.; ***Sistemas de Hipermídia Adaptativa***; XXI Jornada de Atualização em Informática. SBC 2002. Florianópolis-SC, Julho, **2002**.

PAPERT, S.; ***Constructionism: A New Opportunity for Elementary Science Education. A proposal to the National Science Foundation***, Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, Cambridge, Massachusetts, **1986**.

PAPERT, S.; ***Mindstorms – Children, computers and powerful ideas***. New York, Basic Books, **1980**.

PARIENTE, C. A. B.; ***Gramáticas Livres de Contexto Adaptativas com Verificação de Aparência***; Tese de Doutorado; Escola Politécnica; Universidade de São Paulo; São Paulo; **2004.**

PEDRAZZI, T., **TCHEMRA**, A. H., **ROCHA**, R. L. A.; ***Adaptive Decision Tables - a Case Study of their Application to Decision-Taking Problems*****;** Proceedings of International Conference on Adaptive and Natural Computing Algorithms - ICANNGA 2005, Coimbra, March 21-23, **2005**.

PELEGRINI, E. J. & **NETO**, J. J.; ***Applying Adaptive Technology in Data Security***. Proceedings of the 6th Peruvian Computer Week - JPC 2007, Trjillo, Peru, Novembro 5-10 (pp. 31-40), **2007**.

PELEGRINI, E. J.; ***Códigos Adaptativos e Linguagem para Programação Adaptativa: Conceitos e Tecnologia***. Dissertação de Mestrado, EPUSP, São Paulo, **2009**.

PEREIRA, J.C.D.; ***Ambiente Integrado de Desenvolvimento de Reconhecedores Sintáticos, Baseado em Autômatos Adaptativos***; Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, **1999**.

PERRATON, H.; ***A Theory for Distance Education***; in D. Stewart, D. Keegan & B. Holmberg – Distance Education: International Perspectives; New York: Routledge, 1988.

PHP; ***PHP Resources***. Disponível em: http://www.php.org. Acesso em: fevereiro de 2010.

PIAGET, J. Aprendizagem e Conhecimento. In: PIAGET; J; GRÉCO, P., ***Aprendizagem e Conhecimento***, Rio de Janeiro: Freitas Bastos, **1974**.

PISTORI, H. & **NETO**, J. J.; ***AdapTools: Aspectos de Implementação e Utilização;*** Boletim Técnico PCS, Escola Politécnica, São Paulo, **2003**.

PISTORI, H. & **NETO**, J. J.; ***An Experiment on Handshape Sign Recognition using Adaptive Technology: Preliminary Results***. XVII Brazilian Symposium on Artificial Intelligence – SBIA'04. São Luis, September 29 – October 1, **2004**.

PISTORI, H. et al. ***Defect Detection in Raw Hide and Wet Blue Leather CompIMAGE*** – Computational Modelling of Objects Represented in Images: Fundamentals, Methods and Applications, Coimbra, **2006**.

PISTORI, H., **MARTINS**, P.S., **CASTRO** Jr., A.A.; ***Adaptive Finite State Automata and Genetic Algorithms – Merging Individual Adaptation and Population Evolution***. Proceedings of International Conference on Adaptive and Natural Computing Algorithms – ICANNGA 2005, Coimbra, March, **2005**.

PISTORI, H.; **NETO**, J. J.; **PEREIRA**, M. C.; ***Adaptive Non-Deterministic Decision Trees: General Formulation and Case Study***. INFOCOMP Journal of Computer Science, Lavras, MG, **2006**.

PISTORI, H.; ***Tecnologia Adaptativa em Engenharia da Computação: Estado da Arte e Aplicações***; Tese de Doutorado; Escola Politécnica; Universidade de São Paulo; São Paulo; **2003**.

RADY, J.; ***STAD-Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos***; Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, **1995**.

RAMOS, M.V.M., **NETO**, J.J. & **VEGA**, Í.S.; ***Linguagens Formais: Teoria, Modelagem e Implementação***; Ed. Bookman; 1ª edição; **2009**.

RICCHETTI, P.M.; ***Geração de Analisadores Sintáticos Baseados em Transdutores a partir de Especificações Gramaticais;*** Dissertação de Mestrado. Escola Politécnica da USP. São Paulo, **2005**.

ROCHA, R. L. A.& **NETO**, J. J.; ***Autômato adaptativo, limites e complexidade em comparação com máquina de turing***; In: Proceedings of the second Congress of Logic Applied to Technology - LAPTEC'2000. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia: [s.n.], **2000**.

ROCHA, R. L. A.& **NETO**, J. J.; ***Uma proposta de linguagem de programação funcional com características adaptativas***. IX Congreso Argentino de Ciencias de la Computación. La Plata, Argentina, Octubre de **2003**.

RUBINSTEIN, R. S.; SHUTT, J. N.; ***Self-modifying Finite Automata: An Introduction***; Information Processing Letters; v.56; n.4; 24; p.185-190, 1995.

RUBINSTEIN, R.S.; **SHUTT**, J.N.; ***SMFA: Self-Modifying Finite Automata;*** IFIP Congress, Vol. 1 pages 493--498, **1994**.

SAITOU, K. & **JAKIELA**, M.J. ***On Classes of One-Dimensional Self- Assembling Automata Complex Systems***; 391-416. **1996**.

SALOMA, A. ***On Finite Automata with a Time-Variant structure***. Information and Control, 13, pp 85-98, **1968**.

SANTOS, J. M. N.; ***Um Formalismo Adaptativo com Mecanismos de Sincronização para Aplicações Concorrentes***; Dissertação de Mestrado; Escola Politécnica; Universidade de São Paulo; São Paulo; 98p.; **1997**.

SCHULTZ, D. P. & **SCHULTZ**, S. E.;***História da Psicologia Moderna.*** Ed. Thomson, 1ª edição, São Paulo, **2005**.

SHUTT, J. N.; ***Self-modifying finite automata - Power and limitations***; Technical Report WPI-CS-TR-95-4. Worcester Polytechnic Institute, Worcester, Massachusetts, December, **1995**.

SHUTT, J. N.; ***Recursive Adaptable Grammar***. Dissertação de Mestrado; Departamento de Ciência da Computação, Worcester Polytecnic Institute, Worcester Massachusetts, 140p, **1993**.

SIGUS – ***Plataforma de Apoio ao Desenvolvimento de Sistemas para Inclusão Digital de Pessoas com Necessidades Especiais*** (UCDB – Campo Grande – MS), disponível em www.gpec.ucdb.br/sigus; acessado em agosto de 2007.

SILVA, P. S. M. & **NETO**, J. J. ***An Adaptive Framework for the Design of Software Specification Languages***. Proceedings of International Conference on Adaptive and Natural Computing Algorithms - ICANNGA 2005, Coimbra, March, **2005**.

SLONNEGER, K. & **KURTZ**, B.L. ***Formal Syntax and Semantics of Programming Languages – a Laboratory Based Approach*** Addison Wesley, **1995**.

SOUSA, M. A. A. & **HIRAKAWA**, A. H.; ***Robotic Mapping and Navigation in Unknown Environments Using Adaptive Automata***. Proceedings of International Conference on Adaptive and Natural Computing Algorithms – ICANNGA 2005, Coimbra, March, **2005**.

STANDISH T. A.; ***Extensibility in Programming Language Design*** ACM SIGPLAN Notices, p. 18-21, vol 10 n. 7, July, **1975**.

TANIWAKI, C. Y. O.; ***Formalismos Adaptativos na Análise Sintática de Linguagem Natural*** Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, **2001**.

TCHEMRA, A. H.; ***Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério***. Tese de Doutorado, Escola Politécnica da USP, São Paulo, **2009**.

VALENTE, J. A.; ***Computadores e Conhecimento: repensando a educação. Cap. 1: Diferentes usos do computador na educação***; Campinas: Gráfica central da Unicamp, Campinas-SP, Brasil, **1993**.

VEIGA, Marise Schmidt. ***Computador e Educacão? Uma ótima combinação***. Petrópolis, **2001**. Pedagogia em Foco. Disponível em: http://www.pedagogiaemfoco.pro.br/inedu01.htm. Acesso em: fevereiro de 2010.

WEGBREIT, B.; ***Studies in Extensible Programming Languages***; Technical Report ESD-TR-70-297. Harvard University, Cambridge, MA, **1970**.

WIJNGAARDEN, A. et al.; ***Revised Report on the Algorithmic Language Algol 68***, 1973 – Springer-Verlag **1976**.

WIJNGAARDEN, A.; ***Orthogonal Design and Description of Formal Languages***. Mathematisch Centrum Amsterdam, MR 76, **1965**.