UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARLON DE MATOS DE OLIVEIRA

AUTÔMATOS FINITOS COM SAÍDA: UM AMBIENTE DE CRIAÇÃO E MANIPULAÇÃO DAS MÁQUINAS DE MOORE E MEALY NO AFLAB

CRICIÚMA, JULHO DE 2008

MARLON DE MATOS DE OLIVEIRA

AUTÔMATOS FINITOS COM SAÍDA: UM AMBIENTE DE CRIAÇÃO E MANIPULAÇÃO DAS MÁQUINAS DE MOORE E MEALY NO AFLAB

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientadora: Profª. MSc. Christine Vieira Scarpato

CRICIÚMA, JULHO DE 2008

MARLON DE MATOS DE OLIVEIRA

AUTÔMATOS FINITOS COM SAÍDA: UM AMBIENTE DE CRIAÇÃO E MANIPULAÇÃO DAS MÁQUINAS DE MOORE E MEALY NO AFLAB

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

_____

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

_____

Profª. MSc. Christine Vieira Scarpato (UNESC)
Orientadora

_____

Profª. MSc. Silvana Campos de Azevedo (UNESC)

_____

Prof. Esp. Arildo Sônego (UNESC)

A Deus por mais esta conquista.

AGRADECIMENTOS

A Deus pela minha existência e a Jesus Cristo por ter mudado o rumo da história.

Aos meus pais, Leodovina e Donírio, que além da vida, me deram educação e me incentivaram a chegar à Universidade.

A minha orientadora Christine que me incentivou a pesquisar na área de Linguagens Formais e a desenvolver esse trabalho.

A minha namorada Paula, por compartilhar comigo momentos felizes e momentos difíceis e também por ter me auxiliado na correção deste trabalho.

Aos professores Priscila, Manoel, Alair, Merisandra, Ana, Arildo, Elvis e Daniel, verdadeiros mestres, que me ensinaram mais do que conteúdos e aos demais professores de uma forma ou de outra me ajudaram durante o curso.

Aos meus amigos e colegas Alex, Samuel, Aline, Edroaldo, Dirceu, Luiz Juventino, Fernando, Leonardo, Felipe, José Marcio, Lucélio, Rodrigo, Diego e Geovan que me ajudaram ao longo do curso, no TCC e nos momentos difíceis ou ainda foram parceiros para tomar uma cerveja quando a pressão aumentava e aos outros colegas, que não há necessidade de citar o nome, mas também foram importantes.

A minha ex-colega de trabalho Patrícia, que me ajudou na linguagem e no ambiente de programação.

Aos meus irmãos, principalmente ao Manoel, por ter me dado o exemplo de que é possível concluir um curso superior, mesmo longe de casa.

Quando morrermos, nada pode ser levado conosco, com a exceção das sementes lançadas por nosso trabalho e de nosso conhecimento espiritual (Dalai-Lama).

RESUMO

O presente trabalho apresenta uma pesquisa sobre autômatos finitos com saída, máquinas de Mealy e Moore. Essa se iniciou por um estudo detalhado nas máquinas de estados finitos, autômatos finitos (determinísticos e não determinísticos), máquinas de Mealy e Moore, e também uma análise sobre a ferramenta AFLAB, que já possuía os módulos de autômatos finitos para reconhecimento de sentenças, tanto na forma gráfica quanto na forma tabular. Pelo fato do AFLAB não possuir os módulos de Mealy e Moore foram desenvolvidos, neste trabalho, os presentes módulos, que permitem utilizar autômatos finitos determinísticos e transformá-los em uma máquina de Mealy ou de Moore, para simulação de saídas em Textos, Imagens ou Sons, por meio de uma sentença válida. Por fim, são apresentadas as etapas de desenvolvimento desses módulos, mostrando como foram projetados e qual foi a metodologia utilizada para implementação dos mesmos. Como resultado, obteve-se uma versão do AFLAB com as máquinas de Mealy e Moore implementadas, onde a partir de um autômato finito determinístico consegue-se simular a saída na forma escolhida.

Palavras-chaves: Autômatos Finitos com Saída, Máquina de Moore, Máquina de Mealy e Ferramenta de Ensino.

ABSTRACT

This work presents a research about finite automata with output, Moore and Mealy machines. It begins with a detailed study on the finite state machines, finite automata (deterministic and non-deterministic), Moore and Mealy machines, and the tool AFLAB analysis, that has already have finite automata modules for sentences acknowledgment, as graphic as tabular mode. The AFLAB don't have the Moore and Mealy modules. For this reason, this machines have been developed, accepting the use of deterministic finite automata and change them in a Moore or Mealy machines to simulate Text, Images or Sounds output through a valid sentence. Finally, the develop steps of this machines are introduced, showing how it was projected and what the programming methodology was applied. The result is a new version of the AFLAB with the Moore and Mealy machines, where it´s possible to simulate the chosen output from a deterministic finite automata.

Key words: Finite Automata with Output, Moore Machine, Mealy Machine, Teaching Tool.

LISTA DE FIGURAS

Figura 1. Exemplo de fita de entrada e unidade de controle 20
Figura 2. Diagrama de transições de um AF 22
Figura 3. Exemplo de um AFD na forma de diagrama 24
Figura 4. Diagrama de transições de um AFND 26
Figura 5. AFND na forma de diagrama de transições 28
Figura 6. Representação de um AFND convertido em AFD 28
Figura 7. Diagrama de transições de uma Máquina de Mealy 31
Figura 8. Diagrama de uma Máquina de Mealy: diálogo 33
Figura 9. Exemplo de Máquina de Moore na forma de diagrama 35
Figura 10. Interface do AFLAB, forma gráfica 39
Figura 11. Interface do AFLAB, forma tabular 40
Figura 12. Janela de transições do AFLAB 41
Figura 13. Autômato construído na forma gráfica 42
Figura 14. Diagrama de Caso de Uso (Usuário) 48
Figura 15. Diagrama de Caso de Uso (Aplicação) 49
Figura 16. Diagrama de atividades de Mealy 49
Figura 17. Diagrama de atividades de Moore 50
Figura 18. Diagrama de Estados 51
Figura 19. Parte do código da função que monta a grade de Mealy 53
Figura 20. Parte do código da função do botão “Simular Saída” de Mealy 54
Figura 21. Parte do código da função do botão “Simular Saída” de Moore 55
Figura 22. Parte do código da função que compara as transições em Mealy 56
Figura 23. Parte do código da função de saída com Imagem em Mealy 57

Figura 24. Parte do código da função da saída com Som em Mealy 58

Figura 25. Parte do código da função de saída com Texto em Moore..... 59

Figura 26. AF criado na forma tabular..... 60

Figura 27. Máquina de Mealy 61

Figura 28. Máquina de Mealy simulando saída com Imagem 64

Figura 29. Máquina de Moore simulando saída com Som..... 66

LISTA DE TABELAS

Tabela 1. Tabela de transições de um AF 23

Tabela 2. Representação de uma Máquina de Mealy na forma tabular 31

Tabela 3. Representação de uma Máquina de Moore na forma tabular 35

LISTA DE SIGLAS

AF	Autômato Finito
AFD	Autômato Finito Determinístico
AFND	Autômato Finito Não-Determinístico
AFS	Autômatos Finitos com Saídas
LR	Linguagens Regulares

SUMÁRIO

1 INTRODUÇÃO 14

1.1 OBJETIVO GERAL 15

1.2 OBJETIVOS ESPECÍFICOS 15

1.3 JUSTIFICATIVA 15

1.4 ESTRUTURA DO TRABALHO 17

2 MÁQUINAS DE ESTADOS FINITOS 18

2.1 AUTÔMATOS FINITOS 19

2.2 FORMAS DE REPRESENTAÇÕES DE UM AF 21

2.2.1 Diagrama de Transições 21

2.2.2 Tabelas de Transições 22

2.3 AUTÔMATOS FINITOS DETERMINÍSTICOS 23

2.4 AUTÔMATOS FINITOS NÃO-DETERMINÍSTICOS 25

2.5 AUTÔMATOS FINITOS DETERMINÍSTICOS X AUTÔMATOS FINITOS NÃO-DETERMINÍSTICOS 27

2.6 AUTÔMATO FINITO COM SAÍDA 28

2.6.1 Máquina de Mealy 29

2.6.1.1 Exemplo de Aplicação de uma Máquina de Mealy 31

2.6.2 Máquina de Moore 33

2.6.2.1 Exemplo de Aplicação de uma Máquina de Moore 36

2.6.3 Máquinas de Mealy x Máquinas de Moore 37

3 A FERRAMENTA AFLAB 38

3.1 FUNCIONAMENTO DO AFLAB 39

3.2 DESENVOLVIMENTO DA FERRAMENTA AFLAB 42

4 ESTADO DA ARTE 44

5 AUTÔMATOS FINITOS COM SAÍDA NO AFLAB 47

5.1 METODOLOGIA 47

5.2 DESENVOLVIMENTO 51

5.2.1 A Criação de um AFS a partir de um AFD 52

5.2.1.1 Reprodução das Saídas em Mealy..... 55

5.2.1.2 Reprodução das Saídas em Moore 59

5.3 SIMULAÇÃO DE UMA SAÍDA 60

5.3.1 Simulação das Máquinas de Mealy e Moore 61

5.3.1.1 Saída em Forma de Texto 62

5.3.1.2 Saída em Forma de Imagem..... 63

5.3.1.3 Saída em Forma de Som 64

5.4 RESULTADOS OBTIDOS 66

CONCLUSÃO.... 68

REFERÊNCIAS 70

BIBLIOGRAFIA COMPLEMENTAR 72

APÊNDICE A - ARTIGO 73

1 INTRODUÇÃO

As linguagens formais são conjuntos de palavras sobre alfabetos, essas podem ser representadas de maneira finita e precisa, através de sistemas com sustentação matemática (MENEZES, 2005). Na ciência da computação o uso desse formalismo é indispensável, pois, os profissionais dessa área necessitam entender várias linguagens.

Autômatos finitos são máquinas abstratas de estados e são utilizadas como reconhecedores de linguagens em Linguagens Formais. Esse reconhecimento é limitado à aceitação ou rejeição. Existe ainda uma extensão desses autômatos, que são os autômatos finitos com saída, no qual a saída pode ser associada a uma transição ou a um estado.

Esses conteúdos são abordados na disciplina de Linguagens Formais, que é normalmente estudada da forma tradicional, ou seja, papel e caneta, restringindo assim a dimensão do ensino dos autômatos. Pensando nisso, criou-se o Grupo de Pesquisa em Linguagens Formais do Curso de Ciência da Computação da UNESC, que visa o desenvolvimento de uma ferramenta para simulação de autômatos finitos. A *shell* é denominada AFLAB e tem como objetivo auxiliar os alunos de Linguagens Formais no aprendizado de autômatos finitos e suas derivações.

Baseado na proposta do AFLAB e tendo em vista que o mesmo não possui os módulos de autômatos finitos com saída, verificou-se a necessidade de implementar esses tipos de autômatos.

Esta pesquisa complementará dois módulos do AFLAB, um será a construção de um autômato baseado na máquina de Mealy e o outro na máquina de Moore, em que o usuário poderá escolher a saída para simular o funcionamento de cada

um deles. Este conhecimento é ministrado aos alunos da disciplina de Linguagens Formais, utilizando-se de pouco ou nenhum recurso de software, fazendo com que seus conhecimentos fiquem limitados aos exercícios propostos no papel. Dessa forma, muitas vezes, o aluno sente dificuldade de visualizar como será a saída dessas máquinas.

O objetivo deste trabalho é construir os módulos para o AFLAB, em que os acadêmicos possam interagir utilizando as máquinas e complementando assim seus estudos.

1.1 OBJETIVO GERAL

Desenvolver módulos para as máquinas de Mealy e Moore no AFLAB.

1.2 OBJETIVOS ESPECÍFICOS

São objetivos específicos desse trabalho:

a) compreender autômatos finitos com saídas;

b) representar o funcionamento da máquina de Moore;

c) representar o funcionamento da máquina de Mealy;

d) definir as saídas possíveis para as máquinas;

e) possibilitar ao usuário interação de autômatos com saídas no AFLAB.

1.3 JUSTIFICATIVA

As linguagens formais exigem, daqueles que as estudam e pesquisam, um raciocínio lógico e matemático mais apurado. Desta forma, necessita-se buscar novas

formas de ensinar e aprender estas linguagens e de realizar uma eventual conferência dos exercícios propostos em sala de aula, além de se ter uma outra forma de interagir com os autômatos sem ser da maneira usual, papel e caneta.

O uso de uma ferramenta de software, que possibilite a manipulação e prática dos conhecimentos obtidos na disciplina de Linguagens Formais, pode ajudar o acadêmico a visualizar a dimensão dos estudos de autômatos finitos, pois, estes dispõem de muitas aplicações, muitas vezes não descobertas pela falta de interação com um programa de computador, que facilite a visualização dos mesmos.

Pensando nisso, foi decidido criar a *shell* AFLAB, que é uma ferramenta para simulações de autômatos finitos. Essa *shell* foi dividida em vários módulos, na qual já foi implementada a parte de autônomos finitos, que permitem a sua inserção na forma tabular ou gráfica, para o reconhecimento de sentença. O proposto nesta pesquisa é a implementação de simuladores de autômatos finitos com saída no AFLAB, devido ao mesmo não possuir essas funcionalidades.

Os autômatos finitos com saída são uma extensão da definição de Autômato Finito, incluindo a geração de uma palavra de saída, tornando assim os autômatos mais potentes. As saídas podem ser associadas às transições (Máquina de Mealy) ou aos estados (Máquina de Moore) (MENEZES, 2005). Essas saídas podem ser de vários tipos, tais como imagens, músicas, textos, entre outros.

Autômatos finitos com saída são estudados na disciplina de Linguagens Formais e possuem outras aplicações práticas, como por exemplo, processamento de textos, hipertexto, hipermídia e animação quadro-a-quadro (MENEZES, 2005).

1.4 ESTRUTURA DO TRABALHO

A presente pesquisa foi dividida em seis capítulos. O texto introdutório ao assunto, objetivos e justificativa, estão contidos no Capítulo 1. O Capítulo 2 faz um apanhado sobre máquinas de estados finitos, explicando os autômatos finitos determinísticos e não determinísticos com um paralelo entre um e outro, nessa etapa é explicada a representação e utilização dos mesmos. Os autômatos finitos com saídas, máquinas de Mealy e Moore, também são explicados no Capítulo 2, onde se tem uma abordagem detalhada do funcionamento de cada máquina, com exemplos e demonstrações de como elas funcionam e qual a relação entre as duas.

O funcionamento do AFLAB é explicado no Capítulo 3, que mostra como a ferramenta funciona e como foi desenvolvida. No Capítulo 4 são mostrados alguns trabalhos correlatos, nos quais se vê um pouco sobre o que já foi desenvolvido na área de autômatos finitos.

Por fim, o Capítulo 5 trata especificamente do trabalho desenvolvido, onde se explica o funcionamento do AFLAB para simulação de autômatos finitos com saída, nesse capítulo também é mostrado como os autômatos finitos com saída foram implementados no AFLAB, inclusive sua modelagem. Na conclusão são apresentadas as considerações finais e sugestões para futuros trabalhos.

2 MÁQUINAS DE ESTADOS FINITOS

Máquinas de estados finitos podem ser associadas a diversos tipos de sistemas naturais e construídos, por exemplo, elevadores, processadores de texto, analisadores léxicos (MENEZES, 2005), máquinas de vender refrigerantes, relógios, quebra-cabeças, chegando aos programas de computadores (VIEIRA, 2006).

As máquinas de estados finitos são máquinas teóricas que, basicamente, podem atuar como um computador digital moderno, pois, ambos possuem memória limitada, pré-estabelecida (LEWIS; PAPADIMITRIOU, 2004), e totalmente organizada em torno do conceito de estados (VIEIRA, 2006).

Nessa analogia se tem o computador como uma máquina de estados finitos, onde a informação contida nele, num determinado instante, pode ser considerada o estado da máquina. Assim, o número de possíveis estados é finito (embora grande). As ações do computador são controladas por um relógio interno, a cada pulso do relógio, os dados de entrada podem ser lidos e as alocações podem ser mudadas, modificando também o estado da máquina para um novo estado. Esse novo estado, obviamente, vai depender do estado anterior e dos dados de entrada.

Uma vez que os dois fatores são explícitos, o próximo estado é totalmente previsível. Os dados de saídas são armazenados em cada estado, então quem determina as saídas são os próprios estados. Dessa forma, depois de uma seqüência de pulsos do relógio, a máquina produz uma sucessão de saídas em resposta a uma série de entradas (GERSTING, 1995).

Além das máquinas de estados finitos, existem máquinas mais próximas do computador atual, por exemplo, a máquina de Turing. Porém, isso não torna as máquinas de estados pouco eficientes, pois, sabe-se que a base dos computadores

modernos são operações binárias com bits de 0 e 1, significando aceita/rejeita. Os computadores também possuem memória finita, da mesma forma que as máquinas de estados possuem um número finito de estados (MENEZES, 2005).

2.1 AUTÔMATOS FINITOS

As máquinas de estados finitos podem atuar como reconhecedoras de linguagens ou como transdutoras. As reconhecedoras, na maioria das vezes, são conhecidas como Autômatos Finitos[1] (AF) e possuem a saída limitada à aceitação ou rejeição, já as transdutoras podem ter diversos tipos de saída e são, normalmente, conhecidas como Autômatos Finitos com Saídas (AFS) (VIEIRA, 2006).

As linguagens que podem ou não ser reconhecidas por estes autômatos são denominadas regulares. Os Autômatos Finitos podem ser classificados em Autômatos Finitos Determinísticos (AFD) ou simplesmente AF, e os Autômatos Finitos Não Determinísticos (AFND).

Um Autômato Finito consiste em uma máquina abstrata, composta por um conjunto de estados e um controle que se desloca de estado para estado. O controle pode ser determinístico, quando não puder estar em mais de um estado ao mesmo tempo e não determinístico, quando essa restrição é eliminada (HOPCROFT; ULLMAN; MOTWANI, 2002).

Um autômato finito é composto, basicamente, por três partes:

a) fita de entrada;

b) unidade de controle;

---

[1] A palavra autômato, historicamente, tem sua origem por ser usada para referenciar robôs programados para cumprirem tarefas pré-determinadas (LEWIS; PAPADIMITRIOU, 2004).

c) programa ou função de transição.

A fita de entrada é sempre finita e contém as informações a serem processadas, que devem, de modo geral, ocupá-la totalmente. Sendo também dividida em células, na qual cada uma armazena um símbolo, este deve pertencer ao alfabeto de entrada. Não se pode escrever sobre a fita e não existe memória auxiliar (MENEZES, 2005). Na Figura 1 pode-se visualizar essa idéia.

A unidade de controle pode ser considerada a principal parte do autômato. Ela é um bloco com mecanismos internos, que pode se apresentar em algumas das diferentes células da fita, em um determinado momento.

O controle finito pode ler o símbolo gravado em qualquer posição da fita, um de cada vez, com o auxílio de um cabeçote móvel de leitura, este cabeçote inicia a leitura sempre na célula mais à esquerda da fita e se move, uma posição à direita, a cada símbolo lido (LEWIS; PAPADIMITRIOU, 2004). Na Figura 1 pode-se observar o controle na segunda posição à direita, assim se supõe que o primeiro símbolo de entrada *a* já foi lido e então o controle se deslocou.

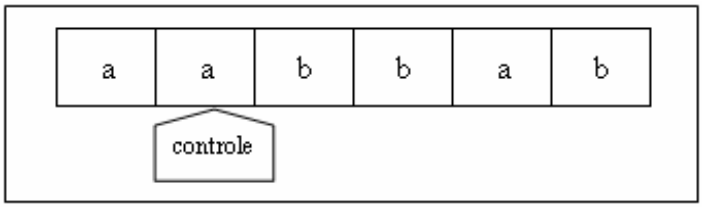


Figura 1. Exemplo de fita de entrada e unidade de controle

O programa ou função de transição é uma função parcial, que determina o novo estado do autômato, lembrando que Autômatos Finitos não possuem memória de trabalho, portanto, para armazenar as informações necessárias deve-se usar o conceito de estados (MENEZES, 2005).

Autômatos Finitos podem ter várias funcionalidades, dentre elas pode-se destacar um corretor ortográfico, que utiliza a idéia de AF para identificar palavras que

não estejam em seu vocabulário. Caso o usuário digite alguma palavra desconhecida, ele exibe como sugestão, uma palavra similar, se o usuário não concordar com a sugestão, ele pode inserir uma nova palavra no vocabulário ou ignorar a correção (STACHOVOSKI, 2005).

2.2 FORMAS DE REPRESENTAÇÕES DE UM AF

Representar um AF em uma 5-tupla, com descrição detalhada das funções de transição é muito tedioso e difícil de ler, mas, para diminuir este problema existem duas notações mais simples de especificar os autômatos (HOPCROFT; ULLMAN; MOTWANI, 2002). São elas:

a) diagrama de transições, onde o autômato é representado por um grafo;
b) tabela de transições, que consiste em uma listagem tabular de uma determinada função, que por implicação informa o conjunto de estados e o alfabeto de entrada.

2.2.1 Diagrama de Transições

Um diagrama de transição para um AFD é um grafo direcionado e rotulado, cada estado é representado por um vértice também chamado de nó, o estado inicial possui uma seta que não se origina de nenhum nó, os estados finais são representados por círculos duplos (ou mais espessos) e os demais estados por um círculo simples (ou menos espessos).

Os arcos ou arestas representam as transições, sendo que, entre dois estados *p* e *q*, existirá uma aresta direcionada de *p* para *q*, com rótulo *a* (HOPCROFT; ULLMAN; MOTWANI, 2002).

Ao analisar a Figura 2 é possível identificar os estados q_0, q_1 e q_2, sendo q_0 um estado inicial e q_2 um estado final. As arestas representam as transições, entre q_0 e q_1 existe uma aresta rotulada de *0* (símbolo lido), isso significa que se o controle do autômato está no estado q_0 e o símbolo *0* é lido, o próximo estado será o q_1.

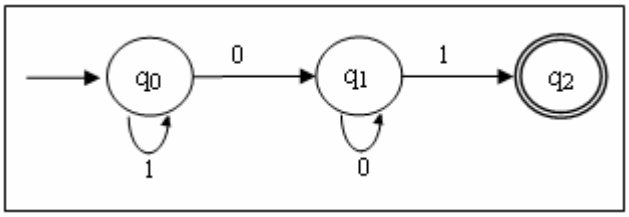


Figura 2. Diagrama de transições de um AF

2.2.2 Tabelas de Transições

Tabelas de transições são representações convencionais e tabulares de uma função, que recebe dois argumentos (o símbolo lido e o estado) e retorna um valor (o próximo estado).

Nessa tabela, as linhas representam os estados, onde o inicial é indicado por uma seta e os finais por um asterisco. As colunas representam os símbolos de entrada e o conteúdo da posição. Na tabela de transições são encontrados todos os dados necessários que se precisa para especificar um AF.

No exemplo da Tabela 1 pode-se verificar os estados q_0, q_1 e q_2, sendo q_0 um estado inicial e q_2 um estado final. A seta indica o estado inicial do autômato, onde no estado q_0 com o símbolo *0*, ter-se-á q_1 como próximo estado.

Tabela 1. Tabela de transições de um AF

δ	*0*	*1*
→q_0	q_1	q_0
q_1	q_1	q_2
*q_2	–	–

Pode-se perceber na Tabela 1 a existência de algumas células representadas com o caractere “-”, esse símbolo indica que não existe nenhuma transição de um determinado estado sobre um dado símbolo de entrada (HOPCROFT; ULLMAN; MOTWANI, 2002).

2.3 AUTÔMATOS FINITOS DETERMINÍSTICOS

Segundo Menezes (2005) e Vieira (2006) um autômato finito determinístico é formalmente definido por uma quíntupla $M= (\Sigma, Q, \delta, q_0, F)$, assim sendo:

a) Σ refere-se ao conjunto finito de um ou mais estados possíveis no autômato;

b) Q refere-se ao alfabeto dos símbolos de entrada;

c) δ refere-se à função de transição, formalmente δ: Σ x Q → Σ;

d) q_0 refere-se ao estado inicial do Autômato que pertence ao conjunto Σ;

f) F refere-se ao conjunto dos estados finais, pertencente ao conjunto Σ.

Observe que as funções de transição são as regras que os autômatos utilizam para escolher o próximo estado (LEWIS, 2004). Essas funções podem ser representadas por grafos finitos diretos, onde o estado inicial é representado por um nó apontado por uma seta, e os estados finais são representados por uma linha dupla, ou mais espessa, em torno do nó (MENEZES, 2005). Já os arcos do grafo servem para refletir a transição

especificada pela função. Observe na Figura 3, na qual e_0 e e_1 são os estados, *a* e *b* são os símbolos de entrada, e_1 é um estado final e as flechas representam as transições.

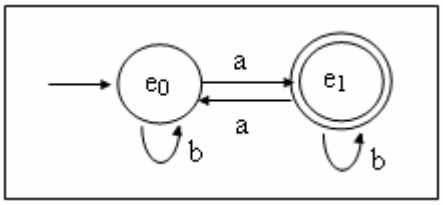


Figura 3. Exemplo de um AFD na forma de diagrama

A característica de cada estado lido, com a respectiva função de transição, levar o autômato para apenas um próximo estado, explica o termo determinístico (HOPCROFT; ULLMAN; MOTWANI, 2002).

O processamento de um AFD consiste, basicamente, na sucessiva aplicação da função programa para cada símbolo da fita de entrada lido (da esquerda para direita), até que ocorra uma condição de parada.

Um AFD sempre pára, pois, como o conjunto de estados é finito e a cada símbolo lido o controle se desloca para o próximo estado ou para o mesmo estado, dependendo da função de transição, não existe a possibilidade de ciclo infinito.

O motivo de parada do processamento pode ser de duas maneiras, aceitando ou rejeitando uma determinada entrada. As condições de parada são as seguintes (MENEZES, 2005):

a) após percorrer toda a fita lendo todos os símbolos, o autômato pára em um estado final, nesse caso a sentença é considerada aceita;
b) após percorrer toda a fita lendo todos os símbolos de entrada, o autômato pára em um estado não final, esta sentença será considerada rejeitada;
c) uma determinada função é indefinida para o argumento, ou seja, um determinado estado com o símbolo lido não o leva a nenhum outro

estado, assim a máquina pára e a sentença é rejeitada;

d) lê na fita de entrada um símbolo que não faz parte dos símbolos aceitos pelo AF.

Em outras palavras, um Autômato Finito lê um símbolo de entrada de cada vez, iniciando pela célula mais à esquerda da fita, depois de ler o símbolo, ele executa a função de transição, que determina qual o novo estado do AF e posiciona o cabeçote de leitura na célula da fita imediata à direita, esse processo é repetido continuamente até o final da fita ou quando ocorre alguma condição de parada.

Se o autômato percorreu toda a fita e chegou a um estado pertencente ao conjunto de estados finais, a sentença é considerada aceita. Por outro lado, se em algum momento o autômato não conseguir prosseguir a leitura ou o AF chegar ao final da fita, e este for um estado que não pertença ao conjunto de estados finais, a sentença é considerada rejeitada.

2.4 AUTÔMATOS FINITOS NÃO-DETERMINÍSTICOS

De acordo com Menezes (2005) e Vieira (2006) um AFND também é formalmente definido por uma 5-tupla, sendo $M = (\Sigma, Q, \delta, q_0, F)$, no qual:

a) os elementos Σ, Q, q_0, F possuem as mesmas definições dos AFD;

b) δ consiste em uma função programa ou de transição, definido em $Q \times \Sigma = \rho(\Sigma)$; sendo que $\rho(\Sigma)$ é um subconjunto de Q; isto equivale a dizer que $\delta(q, a) = p_1, p_2, ..., p_n$. A interpretação de δ é que um determinado AF no estado q, com o símbolo de entrada a pode ir tanto para o estado p_1 como para o estado p_2, ..., como para o estado p_n.

O não-determinismo de um autômato não quer dizer que ele é mais eficiente no reconhecimento de linguagens, embora, seja de extrema importância no estudo da teoria da computação e das linguagens formais, por ser uma generalização das máquinas (MENEZES, 2005).

Nos AFND podem-se ter várias combinações possíveis para a mesma palavra, assim o não determinismo se identifica por causa de uma *indecisão* associada a um estado, neste existem duas ou mais transições prováveis sob o mesmo símbolo de entrada (VIEIRA, 2006). Na Figura 4 a *indecisão* ocorre em dois pontos, na transição do estado q_0 com o símbolo de entrada, *a* e do estado q_1 com o símbolo de entrada *b*.

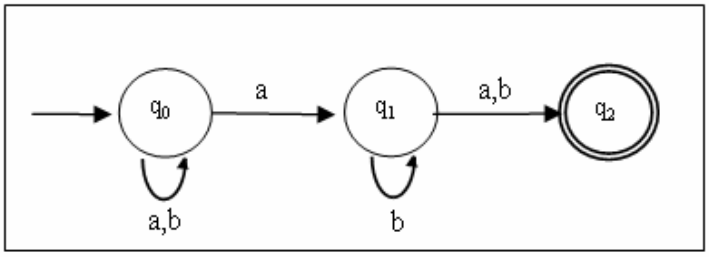


Figura 4. Diagrama de transições de um AFND

Ao analisar um AFND, tem-se o questionamento de quando uma sentença é reconhecida ou não, a resposta é simples, pois dado um único caminho de uma AF no qual todos os símbolos de entrada sejam lidos e o último estado é um final, a palavra é reconhecida. Vieira (2006, p. 90) define que “uma palavra é reconhecida se, e somente se, existe uma computação que consome e termina em estado final”.

Existem duas formas de um AFND reconhecer uma sentença, uma delas é o entendimento de que em cada ponto de indecisão ocorre uma multiplicação da unidade de controle, a ponto de explorar todas as possibilidades de reconhecimento de uma determinada *palavra* (MENEZES, 2005). O outro entendimento é que o autômato escolhe um caminho a percorrer. Caso esse não reconheça a sentença, serão percorridos

outros caminhos até esgotar as possibilidades ou a sentença for reconhecida, no contrário a sentença será rejeitada (GAIDZINSKI, 2007).

2.5 AUTÔMATOS FINITOS DETERMINÍSTICOS X AUTÔMATOS FINITOS NÃO-DETERMINÍSTICOS

Um AFD nada mais é do que um tipo de AFND, assim para qualquer AFND pode-se construir um AFD equivalente, nesse caso, ambos reconhecerão a mesma linguagem (VIEIRA, 2006).

Menezes (2005) costuma dizer que é mais fácil construir um AFND e a partir dele criar um AFD, pois, em um AFD ter-se-á um número igual ou relativamente maior de estados comparado ao mesmo autômato construído na solução não-determinística. Pode-se ressaltar também que um AFD tem implementação relativamente simples, porém não é clara na representação de algumas Linguagens Regulares (LR). Por outro lado, tem-se nos AFND uma implementação complexa, porém uma representação mais clara de determinadas LR.

As diferenças entre AFD e AFND são claramente explicadas por Aho, Sethi e Ullman (1995, p. 51):

> Tanto os autômatos finitos determinísticos quanto os não-determinísticos são capazes de reconhecer precisamente os conjuntos regulares. Consequentemente, ambos podem, reconhecer exatamente o que as expressões regulares podem denotar. Entretanto, existe uma barganha tempo-espaço: enquanto os autômatos finitos determinísticos podem levar a reconhecedores mais rápidos do que os não determinísticos, o autômato finito-determinístico pode ser muito maior do que um autômato finito não determinístico equivalente.

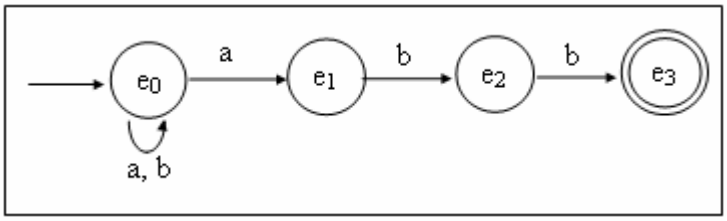


Figura 5. AFND na forma de diagrama de transições

Na Figura 5 fica claro o não determinismo na transição do estado e_0 com o símbolo de entrada *a*. A Linguagem Regular *(a+b)*abb* pode ser claramente visualizada na Figura 5, no entanto, na Figura 6 essa LR já não é tão explicita, embora os dois autômatos reconheçam as mesmas linguagens, ou seja, são equivalentes. Sendo que o AFD representado na Figura 6 é o AFND da Figura 5 transformado em um AFD.

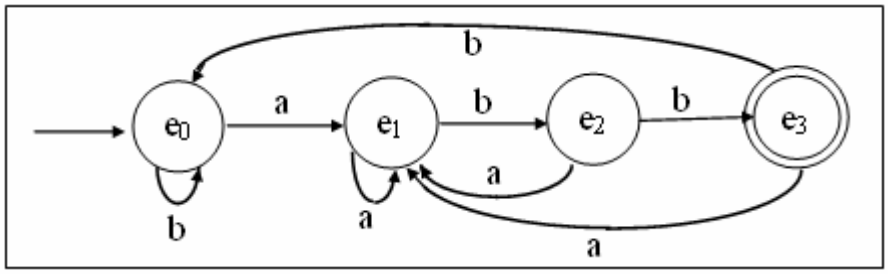


Figura 6. Representação de um AFND convertido em AFD
Fonte: Adaptado de AHO, A.; SETHI, R.; ULLMAN, J. (1995)

2.6 AUTÔMATO FINITO COM SAÍDA

Sem alterar a classe dos autômatos finitos como reconhecedores de linguagens regulares, pode-se acrescentar-lhes algumas saídas, então esses autômatos passam a se chamar de Autômatos Finitos com Saída (AFS). Esses autômatos não têm a saída limitada à lógica binária aceita/rejeita, podendo ter essa saída de vários tipos, por exemplo a geração de palavras (MENEZES, 2005).

Segundo Crespo (2001) essas extensões dos autômatos foram criadas para torná-los mais potentes, uma vez que *frases* podem ser associadas às saídas. A

incorporação dessa frase de saída pode ser dada de duas formas, dependendo de onde estão incorporadas as ações de escrita das *palavras*.

Quando as saídas são associadas às transições têm-se as máquinas de Mealy[2], já quando essas saídas estão ligadas aos estados, têm-se as máquinas de Moore. Porém, em ambas as máquinas, as saídas não podem ser lidas, sendo assim, não podem ser utilizadas como memória auxiliar (MENEZES, 2005).

De acordo com Menezes (2005) essas máquinas são como segue:

a) definidas sobre um alfabeto especial, chamado alfabeto de símbolos de saída, que pode ser igual ao alfabeto de entrada;
b) a saída é armazenada em uma fita de saída, independente da fita de entrada;
c) a cabeça da fita de saída move uma célula para direita a cada símbolo gravado;
d) o resultado do processamento do autômato é o seu estado final (aceita/rejeita) e as informações gravadas na fita de saída.

2.6.1 Máquina de Mealy

A Máquina de Mealy é uma modificação de um autômato finito, de forma a gerar uma saída para cada transição da máquina (MENEZES, 2005).

---

[2] O nome "máquina de Mealy" tem origem no nome do criador do conceito: G. H. Mealy, um pioneiro das máquinas de estado.

Uma Máquina de Mealy é formalmente definida por uma sêxtupla, na qual as saídas são associadas às transições. Conforme Menezes (2005) o modelo matemático para uma máquina de Mealy é M = (Σ, Q, δ , q_0, F, Δ) na qual:

a) Σ é um conjunto de estados não vazio e finito;

b) Q é o alfabeto de símbolos de entrada (conjunto finitos e não vazio);

c) δ é a função de transição de estados: δ: Σ x Q $\rightarrow$ Σ x Δ;

d) q_0 é o estado inicial, um elemento distinguido de Σ;

e) F é um subconjunto de Σ, chamado de conjunto de estados finais;

f) Δ é um conjunto de símbolos de saída, ou simplesmente alfabeto de saída.

Note que em uma Máquina de Mealy, Σ, Q, δ e q_0 são como nos Autômatos Finitos Determinísticos, portanto Mealy e AFD são semelhantes, com a diferença que em vez de aceitação/rejeição existirá uma palavra de saída (VIEIRA, 2006).

Na representação de Mealy por tabela de transições, tem-se as linhas contendo os estados e as colunas formadas por símbolos do alfabeto de entrada. Os elementos das células da tabela possuem o formato *estado seguinte* / *x*, no qual x é uma palavra do símbolo de saída (CRESPO, 2001).

Na Tabela 2 tem-se o exemplo de uma tabela de transição, onde q_0 é um estado inicial, q_1 é um estado final, a transição de q_0 com o símbolo de entrada *0* mantém o controle do autômato em q_0 e escreve uma palavra na fita de saída *a*, a transição de q_0 com o símbolo de entrada igual a *1* leva o a autômato para q_1 e escreve na fita de saída *b*, já a transição de q_1 com o símbolo de entrada *0* escreve *a* e mantém o controle do autômato no estado q_1.

Tabela 2. Representação de uma Máquina de Mealy na forma tabular

Estado Atual (Σ)	*Alfabeto de entrada (Q)*	
	0	*1*
→q_0	q_0/a	q_1/b
* q_1	q_1/a	-

Em um grafo rotulado, as etiquetas das setas possuem o formato y/x, no qual y e x são palavras do alfabeto de entrada e do alfabeto de saída, respectivamente (CRESPO, 2001). Veja na Figura 7 o mesmo exemplo da tabela de transição na forma gráfica.

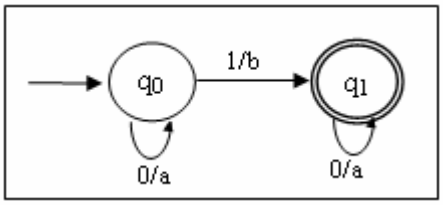


Figura 7. Diagrama de transições de uma Máquina de Mealy

Em outras palavras, a Máquina de Mealy recebe um símbolo de entrada e produz uma saída. No exemplo da Figura 7, considerando uma fita de entrada igual a *0001000* ter-se-á, por exemplo, uma saída igual a **aaabaaa**, a máquina irá parar e a sentença será considerada aceita, pois, a máquina aceita sentenças iniciadas por um número indeterminado de zeros, seguida de um único símbolo um e por último mais uma sequência indeterminada de zeros.

2.6.1.1 Exemplo de Aplicação de uma Máquina de Mealy

Nessa parte do trabalho, será demonstrado um exemplo prático para ajudar no esclarecimento da máquina de Mealy.

Uma aplicação bem comum de uma Máquina de Mealy é o diálogo entre uma pessoa e um programa de computador, esse diálogo pode ser comandado pelo usuário ou pelo programa. Supõe-se então que a referida máquina tem a função de criar e atualizar um arquivo (MENEZES, 2005). Na Figura 8 pode-se visualizar o autômato na forma gráfica, a simbologia adotada foi a seguinte:

a) <...> entrada fornececida pelo usuário;
b) "..." saída gerada pelo programa;
c) [...] ação interna do programa;
d) (...) resultado da ação interna do programa, no diagrama é utilizado como entrada.

No exemplo da Figura 8 tem-se o estado inicial q_0, por exemplo, a abertura de um software para desenhar. A entrada do usuário, estado q_1 seria a criação de um novo desenho. Quando esse arquivo for iniciado (estado q_2), o autômato irá verificar se o nome do arquivo já existe (estado q_3), se não existir o autômato irá para o estado q_6 para gravar as informações. No estado q_7 poderão ser acrescidas informações até que o usuário queira salvar o arquivo, se o usuário desejar salvar o arquivo (estado q_8), a máquina irá salvá-lo e retornar para o estado q_1, podendo finalizar a operação (estado q_f) ou reiniciar as transições a partir do estado q_1.

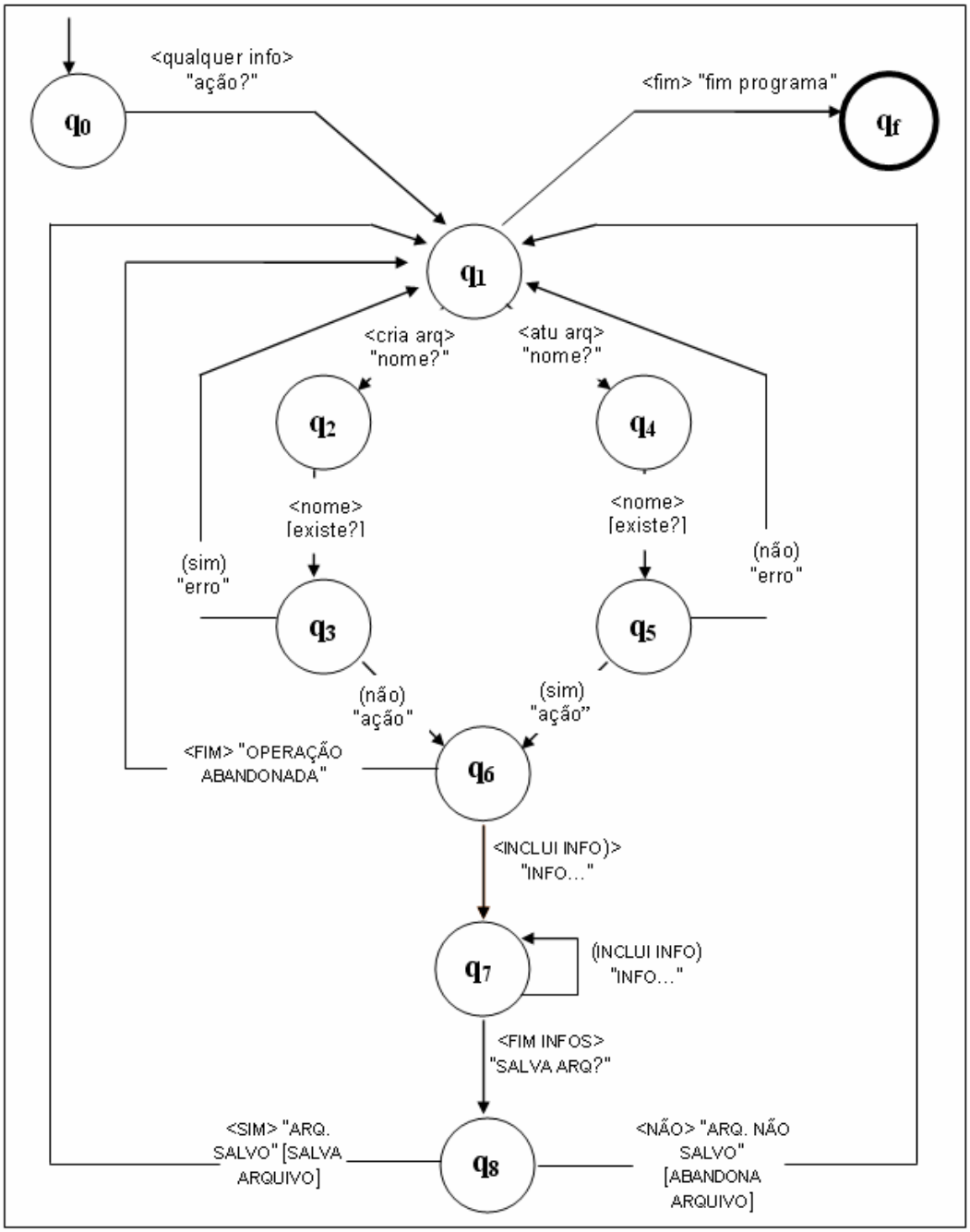


Figura 8. Diagrama de uma Máquina de Mealy: diálogo
Fonte: MENEZES, P. (2005)

2.6.2 Máquina de Moore

Uma Máquina de Moore nada mais é do que uma modificação de um AF, no qual existe uma saída associada a cada estado (VIEIRA, 2006).

Formalmente e de acordo com Menezes (2005) uma Máquina de Moore é definida por uma séptupla M = (Σ, Q, δ , q_0, F, Δ, δs) na qual:

a) Σ é um conjunto finito de estados possíveis;

b) Q é o alfabeto finito de símbolos de entrada;

c) δ é uma função programa ou função de transição, $\delta: \Sigma \times Q \rightarrow \Sigma$;

d) q_0 é o estado inicial, um elemento oriundo de Σ;

e) F é um ou mais elemento de Σ, definido como conjunto de estados finais;

f) Δ é um conjunto de símbolos de saída;

g) δs é uma função de saída, na qual $\delta s: \Sigma \rightarrow \Delta^*$ (função total)

Como visto no exposto Σ, Q, δ , q_0 e F são como nos Autômatos Finítos e Δ é como na Máquina de Mealy. A função de transição pode ser escrita como um diagrama nos AF, somando por sua vez na etiqueta de cada estado, a saída associada, quando diferente de vazio (MENEZES, 2005).

O funcionamento da Máquina de Moore para uma determinada entrada, consiste em uma sucessiva aplicação da função de transição para cada símbolo de entrada lido, até que ocorra uma condição de parada, juntamente com a sucessiva aplicação da função de saída a cada estado alcançado. A saída vazia, resultante da função de saída, corresponde a nenhuma gravação feita na fita e, portanto, a cabeça da fita de saída não se move. Observe que se todas as saídas forem vazias, a Máquina de Moore irá se comportar como um AF (MENEZES, 2005).

A representaçao gráfica de uma Máquina de Moore é semelhante ao de um AF, no entanto, em cada vértice, ao invés de representar um estado, representa um estado e a saída correspondente (VIEIRA, 2006).

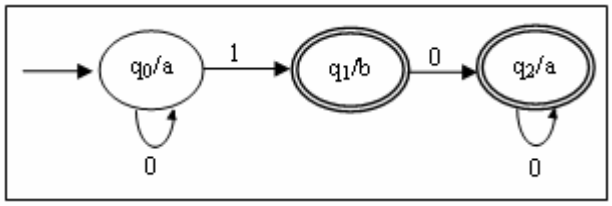


Figura 9. Exemplo de Máquina de Moore na forma de diagrama

O exemplo da Figura 9 é uma máquina de Moore, na qual a função de saída se dá no momento em que o cabeçote de leitura atinge um determinado estado. Na relação y/x, y é o estado lido e x é a saída proporcionada.

Na Figura 9, no estado inicial q_0, a máquina irá produzir uma saída *a*, quando lido um símbolo igual a *0*, a máquina irá produzir mais uma saída *a* e continuará no estado q_0 até que o símbolo lido seja *1* ou a fita de entrada tenha acabado. Se, nesse caso, o símbolo lido for *1*, a máquina aceitará a sentença e produzirá uma saída igual a **aab,** já se a máquina parar ou se houver um símbolo de entrada depois do *1* diferente de *0*, a sentença será considerada rejeitada.

A Máquina de Moore mantém as mesmas características de um AF, na qual uma sentença analisada tem a saída padrão igual a aceita/rejeita e mais uma saída proporcionada pela função de saída (VIEIRA, 2006).

Tabela 3. Representação de uma Máquina de Moore na forma tabular

Estado (Σ)	***Saída (Δ)***	***Símbolo de entrada (Q)***	
		0	***1***
→q_0	A	q_0	q_1
* q_1	B	q_1	-

Na Tabela 3 pode-se observar a tabela de transições de uma máquina de Moore, na qual juntamente com o estado existe a saída, assim no estado q_0 tem-se a saída *a* e no estado q_1 há a saída *b*, por exemplo, o estado q_0 produz a saída *a* e com o

símbolo de entrada 0, continua em q_0 e com o símbolo 1 vai para q_1, já no estado q_1 pode ser lido somente o símbolo 0, pois, para o símbolo 1 não existe transição.

2.6.2.1 Exemplo de Aplicação de uma Máquina de Moore

Máquinas de Moore podem ser utilizadas em várias ocasiões, inclusive na área de compiladores, podendo atuar como analisador léxico (MENEZES, 2005).

Análise léxica é o processo de analisar a entrada de linhas de caracteres (tal como o código-fonte de um programa de computador) e produzir uma seqüência de símbolos chamados *símbolos léxicos* (*lexical tokens*), ou somente *símbolos* (tokens), que podem ser manipulados mais facilmente por um parser (leitor de saída).

Uma Máquina de Moore aplicada a analisadores léxicos age como um autômato finito (geralmente determinístico), que identifica os componentes básicos de uma linguagem, por exemplo, números, identificadores, separadores, entre outros (MENEZES, 2005). As partes da máquina podem ser entendidas como:

a) os estados finais são associados a cada bloco léxico;
b) cada estado final possuí uma saída, que identifica o bloco léxico;
c) para os outros estados, geralmente, as saídas são vazias. Eventualmente, elas podem não ser vazias, no caso de uma informação adicional (mensagem de erro, por exemplo).

Resumindo, uma máquina de Moore é uma forma de verificar determinado alfabeto. Quando se analisa uma palavra, pode-se definir, por meio da análise léxica, se algum símbolo faz ou não parte do alfabeto.

2.6.3 Máquinas de Mealy x Máquinas de Moore

Segundo Crespo (2001) é possível, a partir de uma Máquina de Moore, criar uma Máquina de Mealy e vice-versa. Essas duas máquinas podem ser ditas equivalentes se, e somente se:

a) os símbolos de entradas forem equivalentes;

b) as palavras de saídas forem equivalentes;

c) para todas as entradas, tem-se uma saída equivalente.

As Máquinas de Mealy, geralmente, possuem menos estados que uma Máquina de Moore correspondentes. Por esse motivo, sempre se aconselha usar Máquinas de Mealy em aplicações práticas. Porém, segundo estudos, observou-se que as pessoas preferem associar as saídas aos estados e não às transições (MENEZES, 2005).

O uso de um software pode ser muito útil para o ensinamento dos Autômatos Finitos, pois, a interação do acadêmico com essas máquinas abstratas, faz com que a absorção do conteúdo se torne mais interessante e eficiente.

3 A FERRAMENTA AFLAB

A ferramenta AFLAB foi desenvolvida em um Trabalho de Conclusão de Curso de Ciência da Computação, da Universidade do Extremo Sul Catarinense (UNESC), pelo acadêmico Marco Aurélio Gaidzinski. Esse software possui dois módulos implementados, ambos tratam dos AFD e dos AFND, o primeiro módulo é a representação de um AF em sua forma tabular e o segundo é a representação desse mesmo autômato na forma gráfica.

A *shell* AFLAB foi desenvolvida na linguagem Object Pascal em ambiente de programação Borland® Delphi™ Enterprise versão 7.0, essa linguagem foi escolhida por ser estudada em meio acadêmico, podendo assim ter seu código fonte estudado ou até mesmo modificado. Outro motivo da escolha desse ambiente de programação foi a capacidade de gerar programas leves, que não consumam muita memória do computador (GAIDZINSKI, 2007).

O AFLAB tem como objetivo ajudar os acadêmicos do curso de Ciência da Computação a entenderem os Autômatos Finitos determinísticos e não determinísticos, para o reconhecimento de linguagens, podendo ser usado nas disciplinas de compiladores, linguagens formais e teoria da computação (GAIDZINSKI, 2007).

A *shell* AFLAB possui interface, que permite criar e manipular autômatos na forma gráfica ou tabular, o sistema ainda permite carregar e salvar os autômatos em arquivos com a extensão *Initialization Files* (INI).

3.1 FUNCIONAMENTO DO AFLAB

Ao criar um Autômato no AFLAB é necessário primeiro escolher se o mesmo será criado na forma tabular ou gráfica, isso deve ser feito selecionando a aba correspondente, como pode ser visto na Figura 10 (GAIDZINSKI, 2007).

Figura 10. Interface do AFLAB, forma gráfica
Fonte: GAIDZINSKI, M. (2007)

Depois de escolher a forma que o autômato será criado, deve-se informar os estados, os símbolos de entrada e as transições. Caso a escolha de criação do AF tenha sido pela tabela de transições (guia AF – Forma Tabular), os nomes dos estados, o alfabeto, o estado inicial e os estados finais devem ser informados nos campos

apropriados, logo em seguida o botão **transições** deve ser clicado para informar as transições (GAIDZINSKI, 2007).

Por último, deve ser informada a sentença a ser reconhecida, no campo destinado a ela. O clique do **OK** irá fazer o processamento e retornar se a sentença é verdadeira ou falsa.

Na Figura 11 pode-se observar uma sentença sendo reconhecida por um autômato, na sua forma tabular.

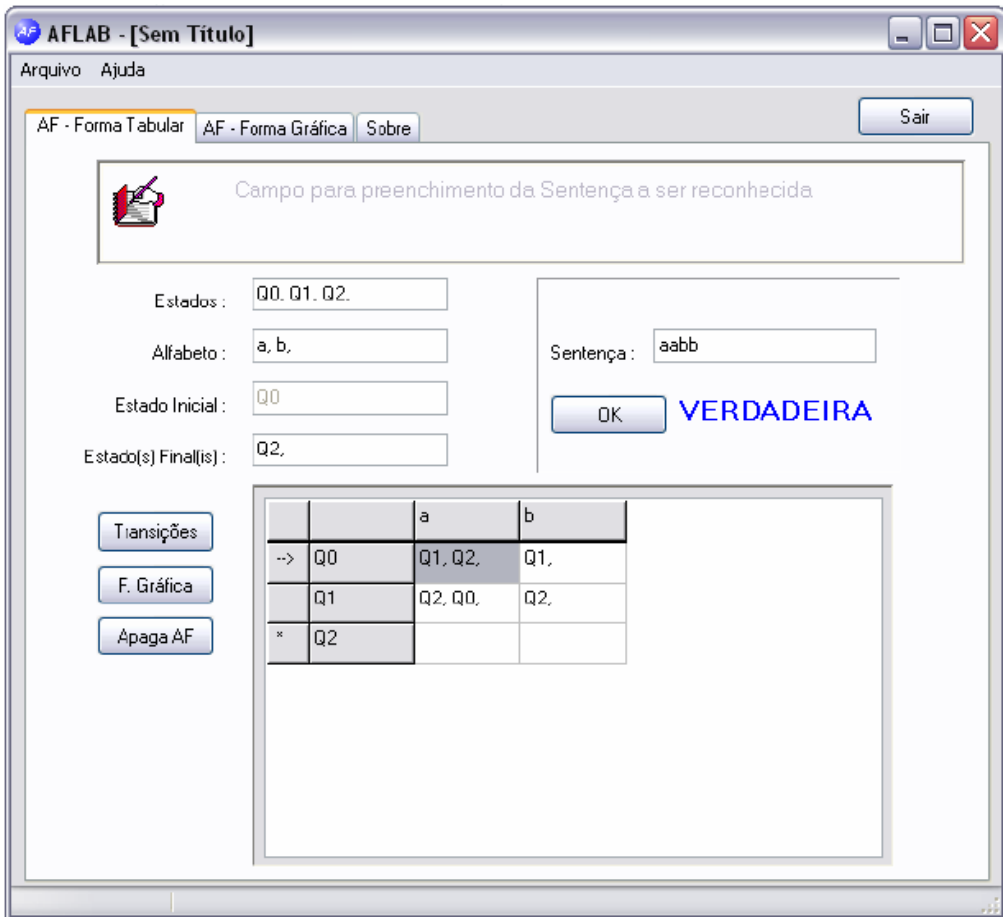


Figura 11. Interface do AFLAB, forma tabular
Fonte: GAIDZINSKI, M. (2007)

Outra forma de se criar um autômato no AFLAB é por meio de diagramas de transição, para isso é necessário selecionar a guia AF – Forma Gráfica (Figura 10). Nesse ambiente serão usados os botões da parte superior esquerda, esses botões irão

determinar se o estado é final ou não, além de permitir a movimentação dos estados. O primeiro estado inserido será sempre o estado Inicial.

Um clique no botão *Transições* irá abrir uma nova janela, onde serão informadas as transições, como pode ser visto na Figura 12. No primeiro *combo box* deverá ser escolhido o estado de origem. No campo *Símbolo(s)* deverão ser informados os símbolos do alfabeto de entrada para essa transição, se tiver mais de um símbolo, esses deverão ser separados por vírgula. No *combo box* Estado Destino deve ser informado o estado de destino.

Figura 12. Janela de transições do AFLAB
Fonte: GAIDZINSKI, M. (2007)

Na Figura 13 pode-se visualizar um autômato construído na forma gráfica, onde as cores ajudam a facilitar a identificação dos estados, o estado inicial está em azul, o estado final está contornado com uma linha dupla em vermelho e o cinza indica os demais estados. As transições estão nas arestas na forma “a, (Q1 -> Q2)” onde “a” representa o símbolo lido e (Q1 -> Q2) indicam os estados de origem e destino, respectivamente.

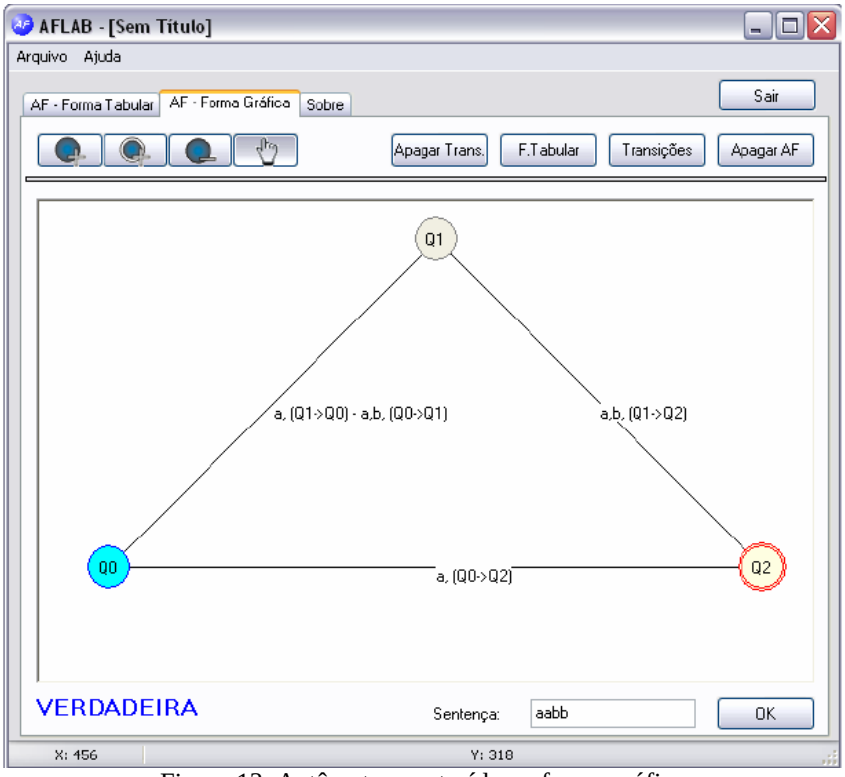


Figura 13. Autômato construído na forma gráfica
Fonte: GAIDZINSKI, M. (2007)

3.2 DESENVOLVIMENTO DA FERRAMENTA AFLAB

O AFLAB utiliza uma única estrutura para o armazenamento dos AF, permitindo que esse seja determinístico ou não determinístico, sem restrições. Outra característica importante é a de que essa mesma estrutura irá armazenar os autômatos, tanto na forma tabular como na forma gráfica, permitindo assim que um autômato criado na forma tabular seja visualizado na forma gráfica e vice-versa (GAIDZINSKI, 2007).

Essa estrutura foi divida, basicamente, em três classes *Testados, Talfabeto* e *Tconexao*, onde, *Testados* tem a função de gravar os atributos dos estados, como rótulo, se é inicial ou final e as coordenadas responsáveis pela localização do mesmo na forma gráfica. A classe *Talfabeto* é responsável pelo armazenamento dos símbolos de entrada.

Por último, a classe *Tconexao* é responsável pelas informações das transições como estado inicial, símbolo de entrada e estado de destino (GAIDZINSKI, 2007).

O armazenamento dos dados foi feito através de variáveis declaradas com tipo de vetores dinâmicos, evitando assim a limitação do número de estados, de símbolos de entrada e até mesmo de transições (GAIDZINSKI, 2007).

No reconhecimento de sentenças, tanto com AFD quanto para AFND, foi utilizada uma técnica disponibilizada por Vieira (2006), onde a estrutura do AF é baseada no conceito de árvores. A base da árvore é criada quando se lê o estado inicial com o primeiro símbolo de entrada, gerando um novo conjunto de estados (novos ramos da árvore). A seguir é lido o próximo símbolo de entrada, que será comparado com os estados desse novo conjunto de estados e gerado um novo conjunto de estados, isso se repete até que o último símbolo de entrada seja lido (GAIDZINSKI, 2007).

A identificação do reconhecimento ou não da linguagem é feito de modo que, se o último conjunto de estado criado possui pelo menos um estado final a linguagem é reconhecida, caso contrário ela será rejeitada (GAIDZINSKI, 2007).

Algumas ferramentas já foram desenvolvidas na área de Teoria da Computação e Linguagens Formais, de certa forma, essas contribuíram para a idealização da ferramenta AFLAB.

4 ESTADO DA ARTE

Além do AFLAB existem algumas ferramentas desenvolvidas por estudantes do curso de Ciência da Computação de diversas Universidades, que também tratam da manipulação de AF, dentre elas, pode-se destacar as seguintes:

a) **Language Emulator:** ferramenta desenvolvida na Universidade Federal de Minas Gerais, como Trabalho do Curso de Pós-Graduação do Departamento de Ciência da Computação. Essa ferramenta tem por objetivo ajudar os alunos do curso a compreender os conceitos da teoria da computação, mais especificament,e a teoria dos autômatos. O Language Emulator foi desenvolvido na linguagem Java e permite a manipulação de expressões regulares, gramáticas regulares, autômatos finitos determinísticos, autômatos finitos não determinísticos, AFND com transições lambda, máquinas de Moore e de Mealy. O inconveniente na ferramenta citada é que as máquinas de Moore e Mealy podem ter como saída somente letras ou palavras (VIEIRA; VIEIRA;VIEIRA, 2002);

b) **AGASTUDIO:** ferramenta desenvolvida na Universidade do Extremo Sul Catarinense, pela acadêmica Lislaine Stachowoski, como Trabalho de Conclusão de Curso. O AGASTUDIO foi baseado em autômatos finitos, mais especificamente na máquina de Mealy, em que as saídas geram animações quadro-a-quadro, sem que o usuário tenha contato direto com o código fonte. O software foi desenvolvido na linguagem Java (STACHOWOSKI, 2005);

c) **AGACATEGORY**: software desenvolvido na Universidade do Extremo

Sul Catarinense, pelo acadêmico Daniel Fernandes, como Trabalho de Conclusão de Curso. Esse software foi implementado para mostrar a aplicação de algumas operações categoriais aplicadas à animações. Estas foram criadas no modelo AGA iniciado por Accorsi (2002), que utiliza os autômatos finitos com saída (máquina de Mealy) para criar animações. A ferramenta foi desenvolvida em linguagem de programação Delphi (FERNANDES, 2005);

d) **SIGA:** ferramenta criada na Universidade Paranaense (UNIPAR), pelos alunos João Giovanetti e Luiz Panick, como trabalho de disciplina de Linguagens Formais, orientado pelo professor Yandre Costa. O software recebe alguns dados como estados, símbolos, transições, estado inicial e estado final, por último é informada a sentença a ser reconhecida e o sistema retorna se a sentença foi aceita ou rejeitada, caso a sentença tenha sido rejeitada o software retorna também o motivo. O SIGA foi desenvolvido em ambiente de programação Visual Basic 6.0 e não trabalha com AFS (GIOVANETTI; PANICK; COSTA, 2000);

e) **ENSINET/NAV:** ferramenta para estruturação de cursos baseados em objetos de aprendizagem, esses cursos são realizados no ambiente ENSINET e estruturados através do conceito de autômatos de navegação e Objetos de Aprendizagem. O trabalho foi baseado em um estudo sobre os projetos *Hyper-Automaton* e *Extensible Hyper-Automaton*, ambos utilizam na sua implementação o formalismo dos Autômatos Finitos com Saída. O site foi desenvolvido em linguagem Pytom e ambiente Zope. Esse projeto foi desenvolvido por pesquisadores da Universidade Federal de Pelotas (SOUZA et al, 2004);

f) **AFCHECKER:** trabalho desenvolvido na disciplina de linguagens formais e autômatos na Universidade Católica de Pelotas (UCPel). O software atua como reconhecedor de sentenças por meio de AFD, AFND e autômatos finitos com movimentos vazios. A ferramenta desenvolvida em linguagem de programação C++ utiliza os conceitos de pilhas e filas encadeadas para realizar o reconhecimento das linguagens (BASTOS; SALVADOR, 2002);

g) **EDITAB: s**oftware desenvolvido na Universidade Federal de Santa Catarina (UFSC). Esse trabalho implementa o reconhecimento de sentenças por meio de um algoritmo que simula um autômato finito na forma tabular. Essa ferramenta foi desenvolvida em ambiente MS-DOS, tendo assim a carência de um ambiente gráfico que facilite o uso. A ferramenta também não possui a opção de exportar a tabela para outras linguagens de programação, enfim, ela permite apenas a visualização ou impressão da mesma (FURTADO; DELUCCA; KREIS, 1992);

h) **EDIGRAF:** ferramenta desenvolvida na UFSC o EDIGRAF é um Editor Gráfico de Autômatos Finitos criado para facilitar o aprendizado dos alunos na disciplina de Introdução a Compiladores. O software atua como um reconhecedor de linguagens e foi desenvolvido para ambiente MS-DOS, por isso possui um ambiente gráfico que dificulta a sua utilização (FURTADO; DELUCCA; KREIS, 1991).

Esses trabalhos serviram de base para que fossem possíveis as melhorias e novas funcionalidades, que estão sendo implementadas no AFLAB.

5 AUTÔMATOS FINITOS COM SAÍDA NO AFLAB

A *shell* AFLAB é uma ferramenta para auxílio ao ensino e aprendizagem de autômatos finitos, inclusive os autômatos finitos com saída, máquina de Mealy e Moore, módulos que foram implementados na aplicação como objetivo dessa pesquisa.

Os módulos de máquinas de Moore e Mealy possuem três tipos de saídas cada um, sendo Textos, Imagens e Sons, esses podem ser associados às transições (Mealy) ou aos estados (Moore).

A ferramenta AFLAB possui uma interface gráfica de fácil entendimento, onde os usuários poderão criar um autômato na forma tabular ou gráfica e depois simular as máquinas de Mealy ou Moore.

Os dados referentes aos autômatos e às máquinas de Mealy e Moore também poderão ser salvos em arquivos textos com extensão .INI.

5.1 METODOLOGIA

Ao longo dessa pesquisa, foram necessárias algumas etapas metodológicas, desenvolvidas ao decorrer da mesma, com a finalidade de atingir os objetivos deste Trabalho de Conclusão de Curso.

Durante o desenvolvimento desse trabalho se fez necessário um levantamento bibliográfico, no qual foram estudadas as linguagens formais, mais especificamente as máquinas de estados, autômatos finitos, autômatos finitos com saídas (máquinas de Moore e Mealy). Nessa etapa também foram realizados estudos sobre modelagem de sistemas em UML e desenvolvimento de software em ambiente de

programação Delphi. O levantamento bibliográfico foi realizado com pesquisas em livros, artigos, trabalhos de conclusão de curso, teses, dissertações, entre outros.

Estudar o código da ferramenta AFLAB também foi indispensável, pois, foi necessário entender como a ferramenta funciona e como foi implementada, assim foi possível criar dois módulos e fazer com que esses funcionassem juntamente com os módulos já existentes.

Antes de iniciar a implementação do software foi necessário definir as saídas mais convenientes, onde se preferiu utilizar, texto (letras, palavras ou frases), imagens (figuras ou fotos bitmaps) e arquivos de áudio (mp3, wav e midi), pois, esses tipos de mídias são comumente utilizadas pelos acadêmicos.

A implementação foi iniciada com a modelagem da ferramenta em UML, nessa etapa o software foi modelado em Linguagem Universal de Modelagem, no qual foram obtidos alguns diagramas.

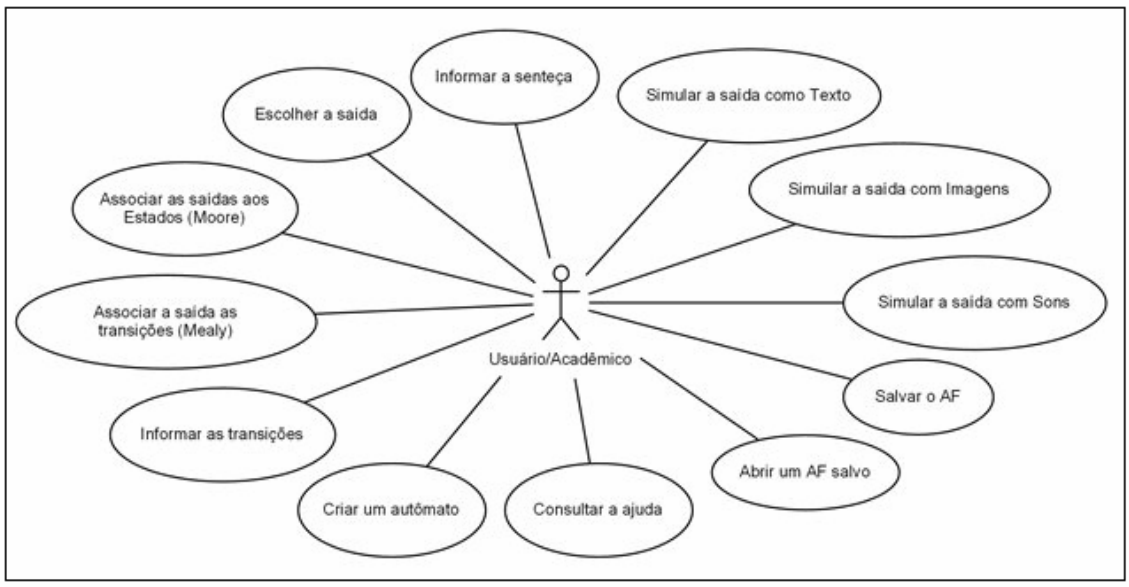


Figura 14. Diagrama de Caso de Uso (Usuário)

Na Figura 14 pode-se observar um diagrama de caso de uso do ponto de vista de um usuário, que normalmente será um acadêmico, esse diagrama é genérico, ou seja, serve tanto para o módulo de Mealy quanto para o módulo de Moore.

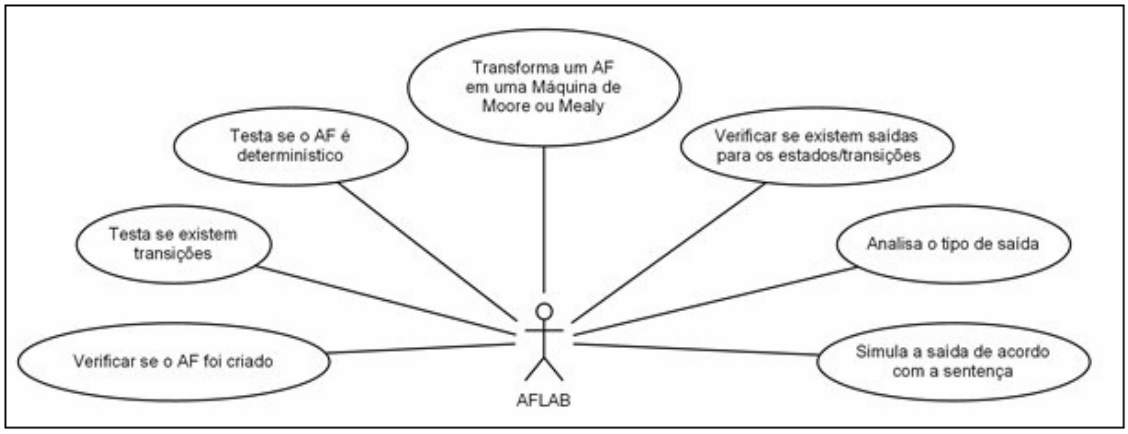


Figura 15. Diagrama de Caso de Uso (Aplicação)

Já na Figura 15 pode-se ver o diagrama de caso de uso do ponto de vista do sistema AFLAB, que por sua vez também serve para Mealy ou Moore, uma vez que as duas máquinas têm o funcionamento básico muito parecido.

Na Figura 16 tem-se um modelo de diagrama de atividades que representa e ajuda a demonstrar o funcionamento de uma máquina de Mealy.

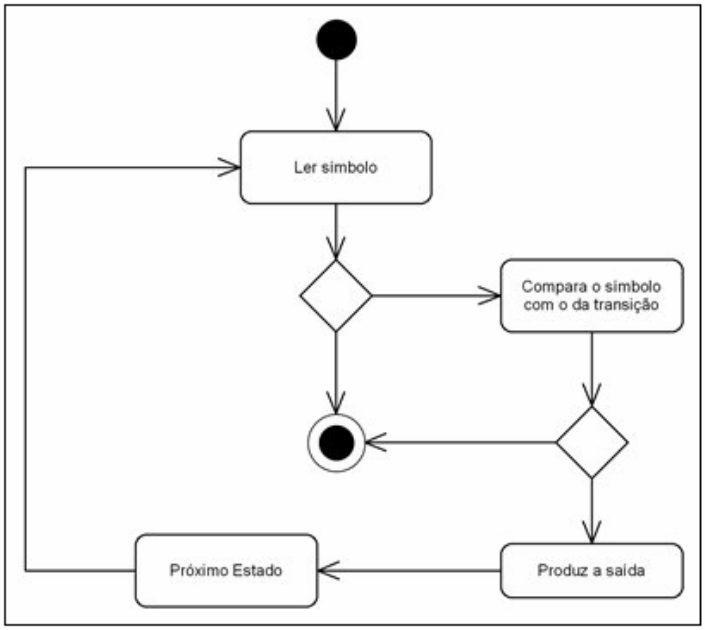


Figura 16. Diagrama de atividades de Mealy

Na Figura 17 é possível visualizar um outro diagrama de atividades, que demonstra o funcionamento da máquina de Moore.

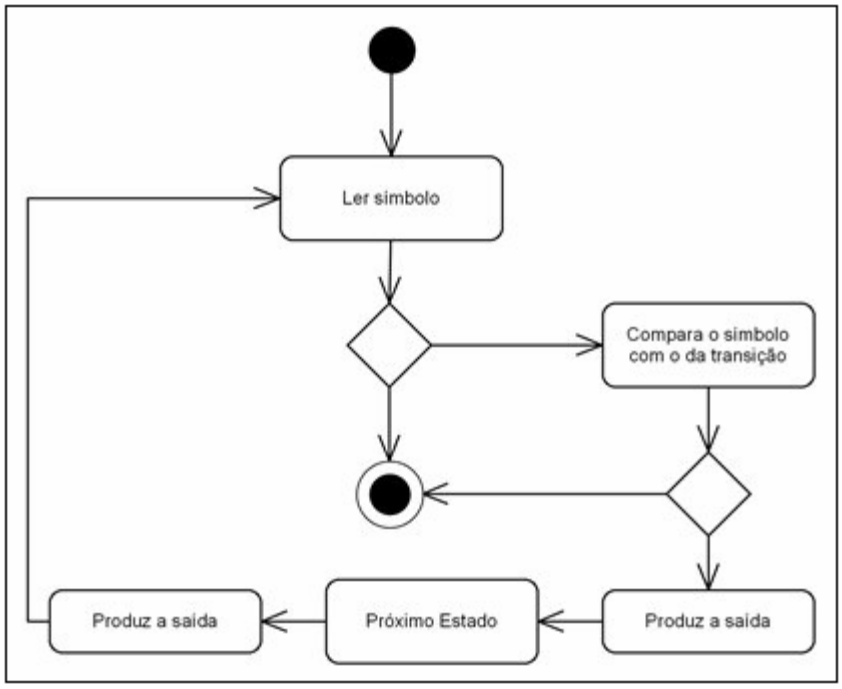


Figura 17. Diagrama de atividades de Moore

Observe que a diferença entre os diagramas de atividades das duas máquinas é a saída, que em Mealy é somente uma para cada transição e em Moore é no mínimo duas, uma para o estado de origem e outra para o estado de destino.

Por fim, foi criado um diagrama de estados, que pode ser visualizado na Figura 18, esse diagrama mostra o estado da máquina para cada ação do usuário.

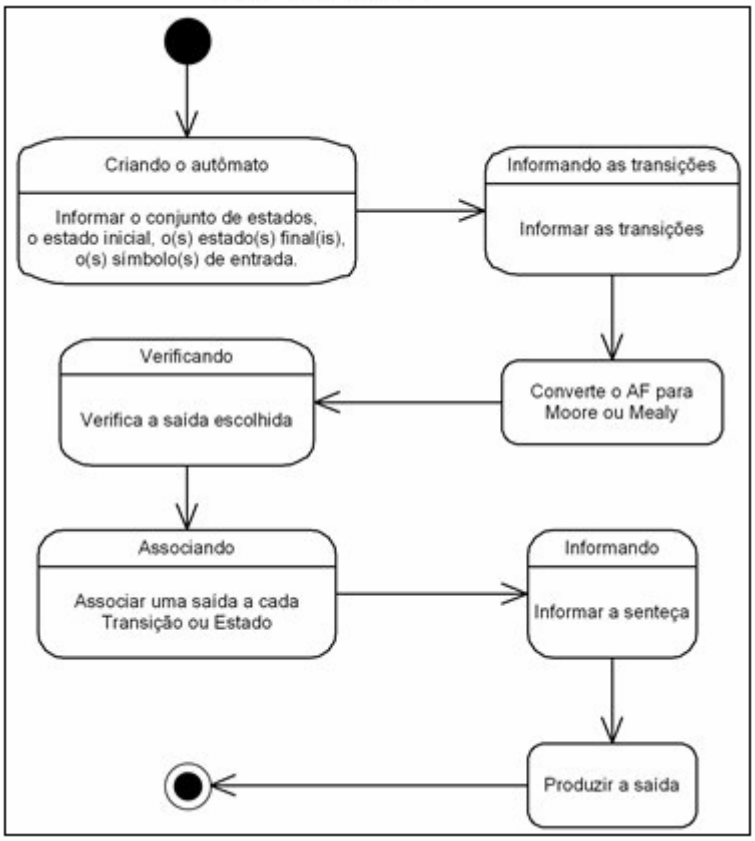


Figura 18. Diagrama de Estados

Terminada a etapa de modelagem iniciou-se a parte de implementação, na qual o ambiente de desenvolvimento Borland Delphi teve que ser estudado, pois o código original do AFLAB estava escrito em *object pascal*, desenvolvido nesse ambiente.

Após a implementação dos módulos de Moore e Mealy no AFLAB, foram realizados alguns testes a fim de deixar a ferramenta mais estável e de fácil utilização pelos acadêmicos, nessa etapa também foram corrigidos eventuais erros encontrados.

5.2 DESENVOLVIMENTO

Os módulos de Mealy e Moore foram desenvolvidos baseados na estrutura dos autômatos criados no AFLAB, ou seja, o autômato é gerado e armazenado

conforme descrito no Capítulo 3, e a partir disso os autômatos finitos com saídas são criados.

No início da implementação pensou-se em criar novos módulos que também armazenassem os autômatos nas classes previamente criadas, porém isso seria somente cópia de código para cada um dos dois módulos propostos, então se chegou à conclusão que o melhor seria utilizar toda a estrutura de criação de autômatos finitos já existentes e somente associar uma saída à transição (Mealy) ou ao estado (Moore). Isso faria com que o código fonte não ficasse muito extenso e também mais compreensível.

Não foi contemplada, nessa pesquisa, a simulação de autômatos finitos não determinísticos com saída. Sendo assim, há uma restrição no sistema, que quando um AFND é gerado e se tenta criar um AFS o sistema emite uma mensagem “AF não determinístico não pode ser simulado por Mealy/Moore no AFLAB”.

Após criar um autômato na forma tabular ou gráfica e acionar o botão Mealy ou Moore, o sistema irá buscar os dados do autômato nas classes armazenadas e irá construir um AFS de acordo com a máquina escolhida. Para que isso seja possível é necessário que o autômato esteja devidamente criado e com as transições informadas, pois AFS somente podem ser simulados por AF que possuam transições, assim foram criadas restrições no sistema, a fim de não permitir a geração de autômatos com saída sem AF, criado com pelo menos uma transição.

5.2.1 A Criação de um AFS a partir de um AFD

Ao clicar no botão *Mealy* ou *Moore* o sistema faz uma série de verificações, tais como, se o autômato está criado, se existem transições e se esse autômato é determinístico, depois do sistema ter feito os devidos testes, os dados do AF são

buscados e montados em uma grade onde aparece uma coluna mostrando todas as transições (Mealy) ou estados (Moore), e outra vazia para que o usuário possa informar a saída desejada.

Na Figura 19 é apresentado uma parte do código fonte que processa esses dados, no qual é utilizada uma estrutura de repetição *for* que passa por todas as conexões e insere em um *client data set* (liga as aplicações aos bancos de dados) chamado *cds_mealy* a transição no formato: "(estado de origem, símbolo do alfabeto de entrada) = estado de destino"

```
for i := 0 to NConexoes - 1  do
begin
  if  (NConexoes >= 0) and (Conexao[i].ContaConexoes <= NConexoes)  then
  begin
    cds_mealy.Append;
    cds_mealy['Campo1']:='('+Conexao[i].estado_origem+','+Conexao[i].Caractete+') = '+Conexao[i].estado_destino;
    cds_mealy.Post;
  end;
end;
```

Figura 19. Parte do código da função que monta a grade de Mealy

O componente *client data set*, foi utilizado, porque esse tipo permite muita flexibilidade, pois, basta conectar uma *dbgrid* (grade para banco de dados) a um desses componentes para se ter acesso às informações inseridas nela. No caso do AFLAB não existe conexão com banco de dados, todo o processamento é feito na memória principal.

Nos *client datas sets* foram criadas duas colunas, uma chamada internamente de *Campo1* para transição, no caso de Mealy ou estados no caso de Moore, e outra chamada *Campo2* para se informar a saída associada a ela. Na grade associada a este componente, a primeira coluna se chama *Transições* ou *Estados*, dependendo da máquina, e a segunda coluna se chama *Saída* em ambos os casos.

Na ativação da guia Mealy ou Moore, depois de escolhido o tipo de saída, deverá ser associada uma saída a uma transição ou a um estado e informada a sentença para que seja simulada a saída, para isso, o botão *Simular saída* deve ser clicado. Ao clicar neste botão o sistema faz algumas validações, por exemplo, qual o tipo da saída

escolhida e se existe uma sentença informada. Na Figura 20 pode-se observar uma parte do código que processa esses testes:

```
procedure Tfrm_principal.bt_saida_mealyClick(Sender: TObject);
var
teste: boolean;
begin
teste := true;
mm_tex_mealy.Clear;

if(Length(ed_sent_mealy.Text) > 0) then //testa se a sentença foi informada
begin
  //se a saida for texto
  if rg_saida_mealy.ItemIndex = 0 then
  begin
    teste := SaidasMealyTexto(ed_ei_mealy.Text, ed_sent_mealy.text, 1000);
  end
```

Figura 20. Parte do código da função do botão “Simular Saída” de Mealy

O funcionamento das funções de saída é basicamente como a de reconhecimento, o sistema passa por todos os estados acessíveis e a cada transição válida, ele consulta na grade qual a saída está associada à transição ou a cada estado, dependendo da máquina. A execução dessa saída pode ser de três formas:

a) texto, onde é exibido um componente do tipo *TMemo* (quadro de texto), que recebe uma linha para cada saída da máquina.

b) imagem, no qual é exibido um componente do tipo *TImage* que recebe as imagens associadas às transições ou estados, dependendo da máquina.

c) som, que por sua vez executa o som em um componente do tipo *TMediaPlayer* (reprodutor de sons), esse componente é invisível, embora seja utilizado em toda função de saída.

Observe na Figura 21 uma parte do código da função associada ao botão *Simular Saída* da máquina de Moore, onde é mostrada a chamada da função de imagem e de som, de acordo com o tipo de saída escolhido.

```
//se a saida for imagem
if rg_saida_moore.ItemIndex = 1 then
begin
  teste := SaidasMooreImagem(ed_ei_moore.Text, ed_sent_moore.text);
end
else
  //se a saida for som
  if rg_saida_moore.ItemIndex = 2 then
  begin
    mp_moore.Enabled := true;
    mm_tex_moore.Visible := true;
    mm_tex_moore.Clear;
    mm_tex_moore.Lines.Add('ORDEM DOS SONS...');
    teste := SaidasMooreTexto(ed_ei_moore.Text, ed_sent_moore.text, 0);
    if (teste = false) then
      exit
    else
    begin
      sb_play_moore.Enabled := true;
      sb_play_moore.Down := true;
      sb_next_moore.Enabled := true;
      mp_moore.Enabled := true;
      teste := SaidasMooreSom(ed_ei_moore.Text, ed_sent_moore.text);
    end;
```

Figura 21. Parte do código da função do botão “Simular Saída” de Moore

5.2.1.1 Reprodução das Saídas em Mealy

A reprodução da saída de Mealy foi divida em três funções: *SaidasMealyTexto(EstadoInicial, Sentenca: string; tempo: integer), SaidasMealyImagem(EstadoInicial, Sentenca: string) e SaidasMealySom(EstadoInicial, Sentenca: string),* ambas retornam um tipo *booleano,* ou seja, *true* se verdadeiro ou *false* se falso. Esse retorno serve para saber se a sentença informada é válida ou não, pois, caso não seja, a saída do autômato será executada até a parte em que for válida, no caso de texto e imagem, já no caso de som a saída não será simulada. Essas funções recebem parâmetros do tipo *string* (conjunto de caracteres alfanuméricos) com o estado inicial e a sentença a ser simulada. A função de Texto possui mais um parâmetro de número inteiro, que serve para controlar o tempo, pois esta função é também utilizada para mostrar a ordem dos sons na função de Som.

A comparação de cada símbolo de entrada com o da transição é feita por meio da função exibida na Figura 22.

```
cds_mealy.First; //marca a primeira posicao da grid
while not cds_mealy.Eof do //enquanto nao chegar ao fim da grid
begin
  for i := 0 to NConexoes -1 do
  begin
    if ((cds_mealy['Campo1']+',') = ('('+EstadoInicial + ','+S+') = '+S2))  then
    begin
      mm_tex_mealy.Lines.Add(cds_mealy['Campo2']); //escreve no memo a saida
      sleep(tempo); //da uma pausa em milisegundos...
      teste := 1;
    end;//fecha o if
    if (teste = 1) then
      break;
end;//fecha o for
cds_mealy.Next; //passa grid para pxoxima celula
break;
```

Figura 22. Parte do código da função que compara as transições em Mealy

O *client data set* é posicionado na primeira posição e enquanto não chegar na posição final, a estrutura de repetição *for* fica rodando até que acabem as conexões, nesse caso da Figura 22 o símbolo lido é o primeiro, pois, ele é comparado com o caractere do alfabeto da transição, onde o estado é o inicial. Nesse caso o sistema irá comparar o *Estado inicial + primeiro símbolo lido da sentença + estado de destino* com a primeira coluna da grade, se for verdadeiro irá executar a saída, neste caso escreverá no *memo*, pois a função é de texto.

Nos demais símbolos de entrada foi utilizada uma função semelhante a da Figura 22, porém, a comparação não é feita com o estado inicial e sim com o estado de origem da transição. O próximo estado é chamado através da função *BuscaProximoEstado(Estado, Alfabeto: string),* que recebe o estado atual e o símbolo de entrada e retorna uma string com o próximo estado, que por sua vez é comparado com o estado de origem, o alfabeto e o estado de destino do autômato, se esse conjunto de dados for igual a transição informada na primeira coluna da grade, o sistema irá produzir a saída. Isso pode ser visualizado melhor na Figura 23.

```
c := Pos(',', S2);
teste := 0;
while c > 0 do //equanto tiver estados
begin
  if ListaConexoes.BuscaProximoEstado(Copy(S2, 1, c - 1), S) <> '' then
  begin
    S3 :=  Trim(S3) + ListaConexoes.BuscaProximoEstado(Copy(S2, 1, c - 1), S);
    cds_mealy.First; //marca a primeira posicao da grid
    while not cds_mealy.Eof do //enquanto nao chegar ao fim da grid
    begin
      for i := 0 to NConexoes -1 do
      begin
        S4 := Trim(copy(S2, 1, c-1));
        if (cds_mealy['Campo1']+',') = ('('+S4 + ','+S+') = '+S3) then
        begin
          image_mealy.Picture.LoadFromFile( cds_mealy['Campo2'] );
          image_mealy.Show;
          application.ProcessMessages;
          sleep(1000);
          teste := 1;
        end;//fecha if
        if (teste = 1) then
          break;
      cds_mealy.Next;//passa grid para pxoxima celula
      end;//fecha o for
    break;
    end;//fecha o while da grid
  break;
  end;//fecha o if da proxima conexao
Delete(S2, 1, c);
c := Pos(',', S2);
end;//fecha o while do c>0
S2 := Trim(S3);
S3 := '';
```

Figura 23. Parte do código da função de saída com Imagem em Mealy

Observe na Figura 23 que a saída produzida é uma imagem e não mais um texto como na Figura 22. A cada transição produzida pela sentença informada igual à transição montada na primeira coluna da grade, o sistema irá carregar o arquivo de imagem correspondente à linha da transição correta. Com o objetivo de melhorar o desempenho do código, a cada transição encontrada o sistema pára e prossegue no próximo bloco de código.

Nas saídas de Mealy em forma de som, foi implementada uma função como as outras, porém, foi adicionada uma outra estrutura de condição, que compara se o componente *MediaPlayer* está ativo, pois este componente é ativado a cada clique no

botão *Simula saída* e desativado cada vez que o botão *Parar* for pressionado ou a função tenha sido completada.

```
cds_mealy.First; //marca a primeira posicao da grid
while not cds_mealy.Eof do //enquanto nao chegar ao fim da grid
begin
  for i := 0 to NConexoes -1 do
  begin
    S4 := Trim(copy(S2, 1, c-1));
    if ((cds_mealy['Campo1']+',') = ('('+S4 + ','+S+') = '+S3)) then
    begin
      if(mp_mealy.Enabled = true) then
      begin
        mp_mealy.FileName := cds_mealy['Campo2'];
        mp_mealy.Open;
        mp_mealy.Play;
        while mp_mealy.Mode = mpPlaying do
        begin
          lb_toc_mealy.Caption:='Tocando música ' + cds_mealy['Campo2'];
          application.ProcessMessages;
        end; //fecha o while
      end//fecha o if do enabled
      else
        exit;
      teste := 1;
    end;//fecha if
    if (teste = 1) then
      break;
  cds_mealy.Next;//passa grid para pxoxima celula
  end;//fecha o for
break;
end;//fecha o while da grid
```

Figura 24. Parte do código da função da saída com Som em Mealy

Na Figura 24 observe uma parte do código da função de saída na forma de som da máquina de Mealy, existe uma estrutura de repetição que fica testando se o som está em execução. Se o botão *Próximo* for clicado, o som pára e o sistema continua a execução da função se ainda existirem sons a serem executados, senão existirem o sistema sai da função. Porém, se o botão *Parar* for clicado, nenhum outro som será reproduzido, pois a função será encerrada.

5.2.1.2 Reprodução das Saídas em Moore

A implementação da máquina de Moore foi um pouco mais complexa do que a de Mealy, pois em Moore, a cada estado atingido, o sistema precisa executar a saída correspondente, sendo assim, nas funções de saída teve-se que testar se era o primeiro símbolo lido, se esse símbolo estava em alguma transição e executar a saída correspondente ao estado inicial dessa transição. Nos demais símbolos, apenas foi preciso consultar a grade e verificar se havia um estado igual ao estado da transição e executar a saída. No último símbolo lido foi preciso executar a saída referente à transição dele e também a do símbolo atingido na referida transição.

```
if ListaConexoes.BuscaProximoEstado(Copy(S2, 1, c - 1), S) <> '' then
begin
  S3 := Trim(S3)+ListaConexoes.BuscaProximoEstado(Copy(S2, 1, c - 1), S);
  cds_moore.First; //marca a primeira posicao da grid
  while not cds_moore.Eof do //enquanto nao chegar ao fim da grid
  begin
    S4 := Trim(copy(S2, 1, c - 1));
    for i := 0 to NConexoes -1 do
    begin
      if ((cds_moore['Campo1']) = (S4))then
      begin
        mm_tex_moore.Lines.Add(cds_moore['Campo2']); //escreve no memo
        sleep(tempo);
        teste := 1;
      end;//fecha if
      if (teste = 1) then
        break;
      cds_moore.Next; //passa grid para pxoxima celula
    end;//fecha o for
  break;
  end;//fecha o while da grid
break;
end;//fecha o if da proxima conexao
```

Figura 25. Parte do código da função de saída com Texto em Moore

Na Figura 25 pode-se observar que é comparada a primeira coluna da grade com o estado que retorna da função *BuscaProximoEstado*, sempre que essa comparação for verdadeira será inserido no *memo* a saída que está na segunda coluna da grade de moore. A mesma lógica utilizada na Figura 25 é aplicada para as saída de Imagem e

Som, obviamente cada uma com suas particularidades, de acordo com os componentes utilizados.

5.3 SIMULAÇÃO DE UMA SAÍDA

A simulação de uma saída em uma máquina de Moore ou Mealy é precedida da criação de um autômato finito determinístico no AFLAB, para isso, devem-se informar os estados, o alfabeto e as transições, que pode ser tanto na forma gráfica quanto na forma tabular. Após criar o autômato, o botão Mealy ou o botão Moore deve ser pressionado, como pode ser visto melhor na Figura 26.

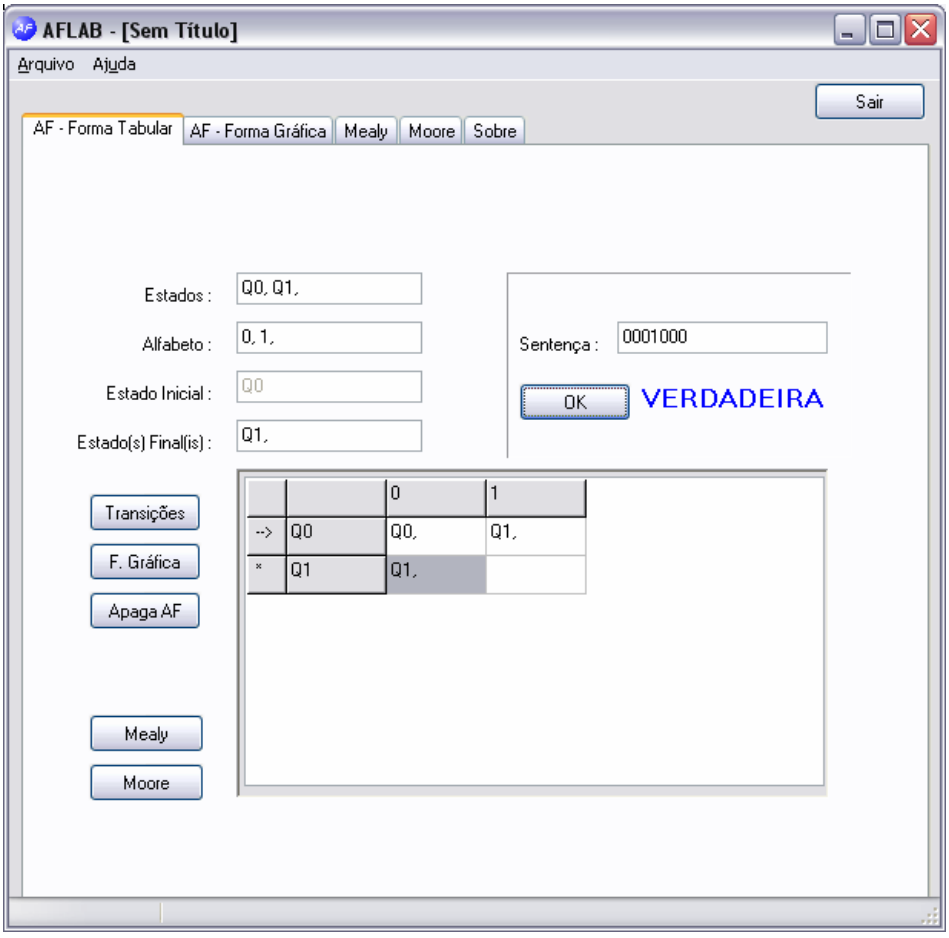


Figura 26. AF criado na forma tabular

5.3.1 Simulação das Máquinas de Mealy e Moore

Antes de simular uma máquina de Mealy ou Moore, deve-se escolher o tipo da saída, que pode ser Texto, Imagem ou Som. No caso de escolhida a opção Texto, se faz necessário informar o conjunto de saídas textos no campo *Saídas*. As outras opções não necessitam dessa informação, uma vez que são informadas diretamente na grade. Isso pode ser visualizado melhor na Figura 27.

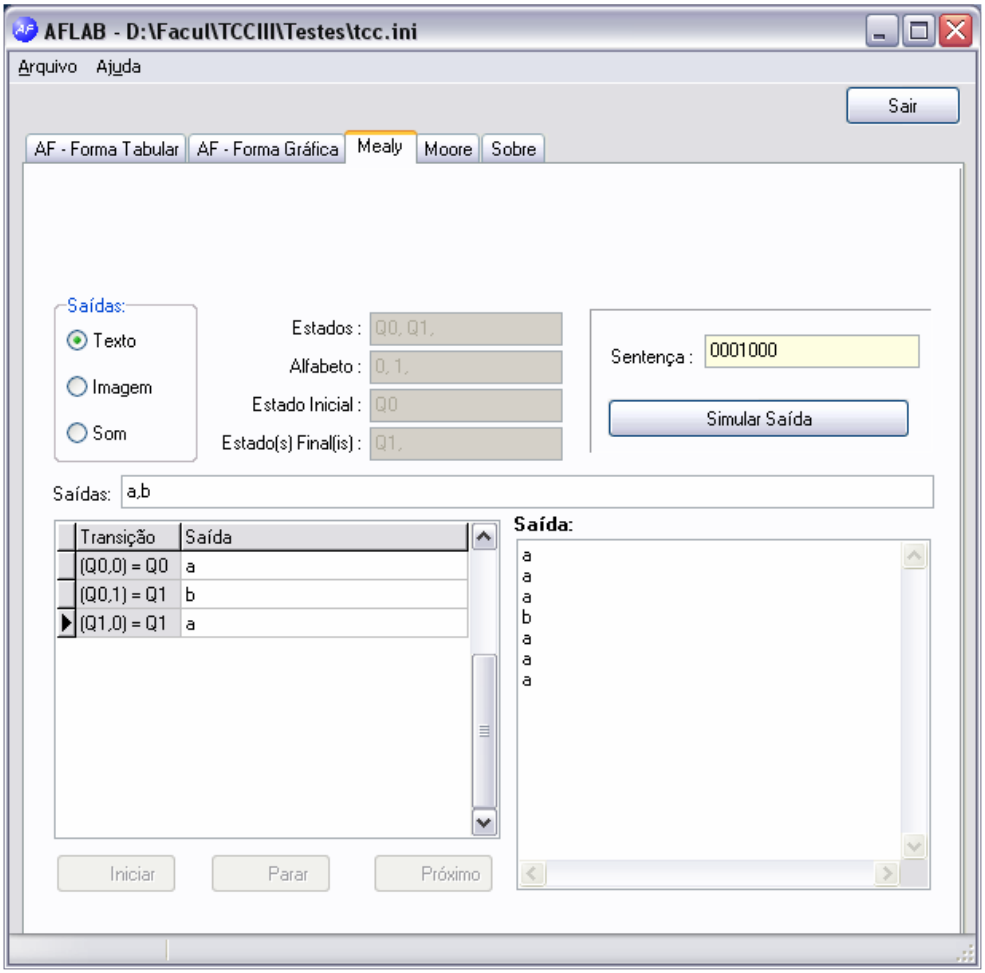


Figura 27. Máquina de Mealy

5.3.1.1 Saída em Forma de Texto

Após ter escolhido o tipo da saída *Texto* e informado o conjunto de saídas no devido campo (separados por vírgula), é necessário associar um texto para cada transição, no caso de Mealy, ou para cada estado, no caso de Moore. Isso se fará possível na grade, onde na primeira coluna se visualiza a transição (Mealy) ou o estado (Moore) e na segunda coluna se informa a saída. Essa será uma das informadas anteriormente. Com a intenção de facilitar ao usuário, quando este tiver que associar a saída de texto a uma transição ou a um estado, está disponível na segunda coluna da grade um campo do tipo *combo box* com as opções possíveis.

A sentença deverá obrigatoriamente ser informada no campo *Sentença*, e o botão *Simular saída* deve ser clicado, para visualização do texto de saída da máquina. Essa sentença deve ser válida, caso seja informada uma sentença inválida, o sistema irá emitir uma mensagem: *A sentença informada é inválida*, já se a sentença não for informada, a aplicação irá mostrar uma mensagem: *Não existe sentença a ser simulada*.

Na ação do botão *Simular saída*, o sistema também exige que todas as transições ou estados tenham uma saída associada, caso isso não ocorra, será exibida uma mensagem: *Existe alguma transição/estado sem saída informada*.

Por fim, a saída de texto é exibida no *memo*, localizado na parte inferior direita da janela, as saídas são separadas por linha com um intervalo de 1 segundo entre uma e outra. Esse tempo é fixado no código fonte e não pode ser alterado pelo usuário.

5.3.1.2 Saída em Forma de Imagem

Após ter escolhido o tipo de saída *Imagem* é necessário associar uma figura para cada transição no caso de Mealy, ou para cada estado, no caso de Moore. Isso se fará possível na grade, pois, na primeira coluna se visualiza a transição/estados e na segunda se informa a saída. Essa saída será, obrigatoriamente, um arquivo de imagem com extensão: Bitmaps (Bmp), Icons (ico), Enhanced Metafiles (emf) ou Metafiles (wmf). Sempre que a saída for de imagens, a segunda coluna da grade libera um botão com o símbolo: "...", que, quando clicado, abre uma nova janela para o usuário selecionar o arquivo da imagem.

A sentença para visualização das saídas da máquina deve ser informada no mesmo campo e com as mesmas restrições da saída *Texto*. Da mesma forma será a ação do botão *Simular saída.* Por fim, a seqüência de imagens é exibida na parte inferior direita da janela, com um intervalo de 1 segundo entre uma e outra. Esse tempo é fixado no código fonte e não pode ser alterado pelo usuário. Na Figura 28 se pode ter uma visão melhor da saída em forma de imagem.

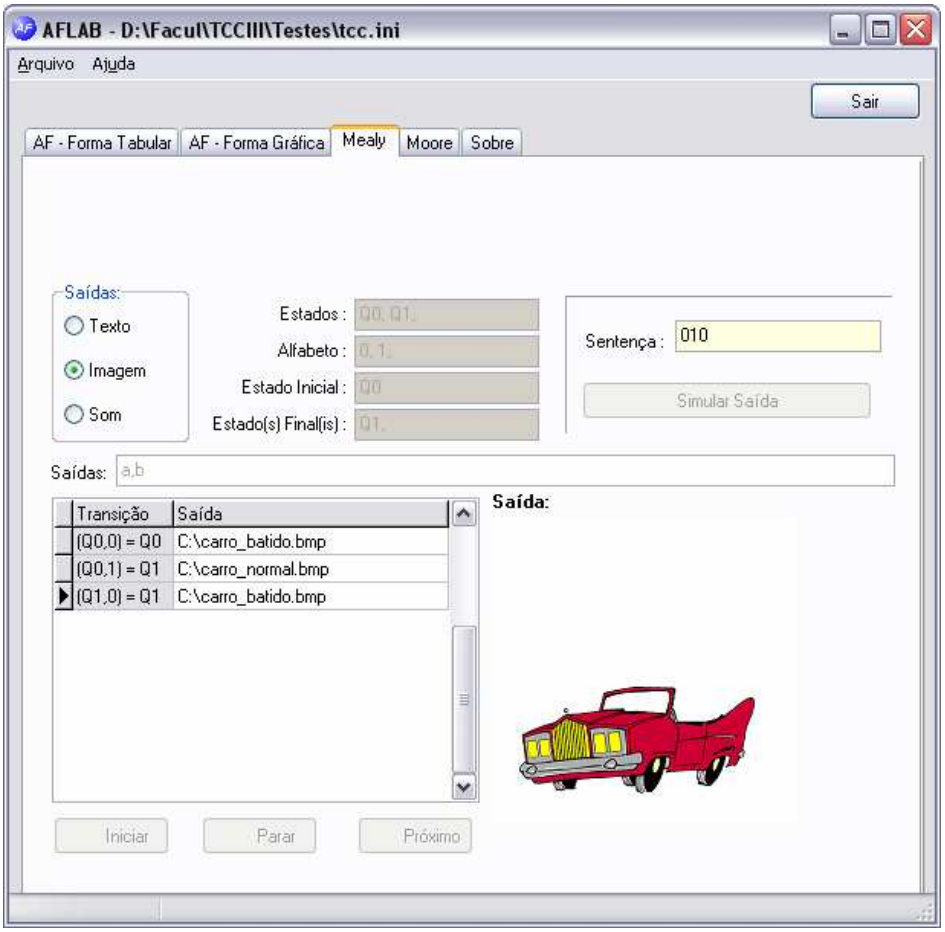


Figura 28. Máquina de Mealy simulando saída com Imagem

Ao terminar a exibição das imagens, o sistema apaga a última figura e libera o botão *Simular saída* para uma próxima simulação.

5.3.1.3 Saída em Forma de Som

Depois de ter escolhido o tipo de saída *Som*, é necessário associar um arquivo de áudio para cada transição, no caso de Mealy, ou para cada estado, no caso de Moore. Isso se fará possível na grade, pois, na primeira coluna se visualiza a transição/estados e na segunda se informa a saída. Essa saída será, obrigatoriamente, um arquivo de som com extensão: Waw, Midi ou Mp3. Sempre que a saída for Som, a

segunda coluna da grade libera um botão com o símbolo: “...”, que, quando clicado, abre uma nova janela para que o usuário selecione o arquivo a ser aberto.

A sentença para visualização das saídas da máquina deve ser informada no mesmo campo e com as mesmas restrições da saída *Texto*. Da mesma forma será a ação do botão *Simular saída*. Por fim, a seqüência de sons é executada e os botões *Iniciar*, *Parar* e *Próximo* são liberados na parte inferior da janela.

Enquanto um arquivo de áudio estiver sendo executado, o botão *Iniciar* ficará pressionado e os botões *Parar* e *Próximo* ficam habilitados. Quando pressionado o botão *Parar,* o sistema pára a execução da música e cancela a simulação da máquina. Já ao pressionar o botão *Próximo,* o sistema pula para o próximo som. Na Figura 29 pode se notar a execução de um arquivo do tipo mp3 na máquina de Moore, onde o nome e o caminho do áudio, que está em execução, aparecem no *label* localizado no final da janela. Na simulação de AFS com som também foi utilizado o *memo* da saída *Texto* para mostrar a ordem da execução dos sons.

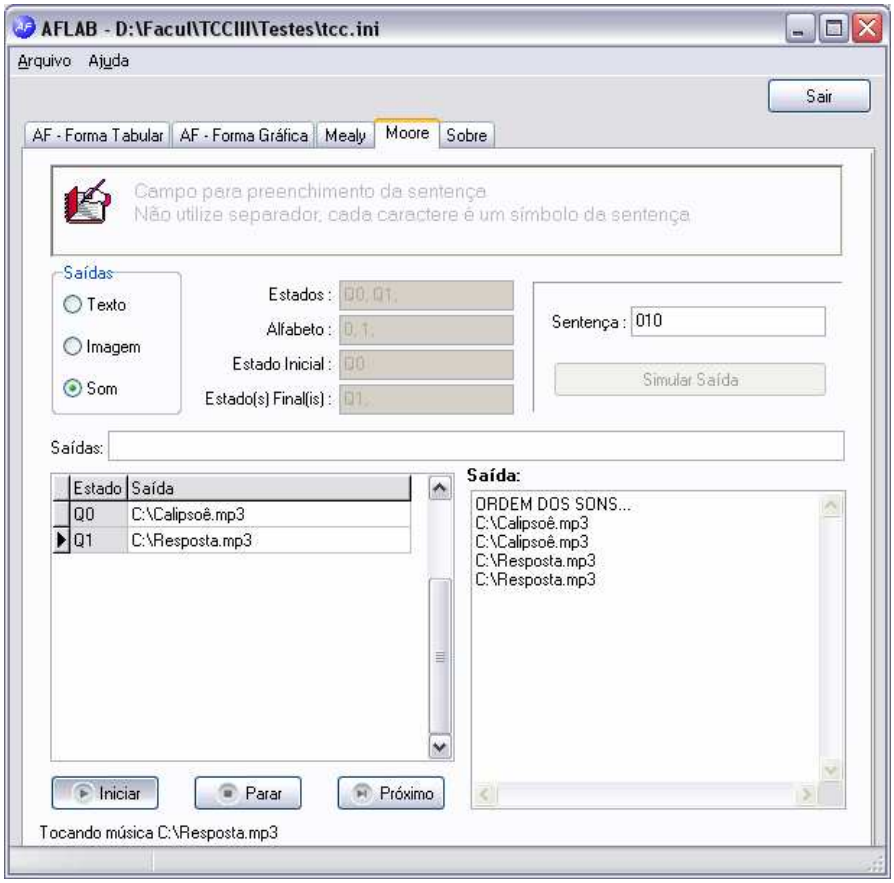


Figura 29. Máquina de Moore simulando saída com Som

Ao terminar a reprodução dos sons, o sistema desabilita os botões de controle do media player e libera o botão *Simular saída* para uma próxima simulação.

5.4 RESULTADOS OBTIDOS

Os módulos de máquina de Mealy e Moore foram implementados totalmente baseados na estrutura de autômatos finitos já existentes no AFLAB, isso reduziu o número de linhas do código fonte e o tornou mais claro.

Durante e após a implementação foram necessários alguns testes. Por exemplo, em uma máquina de Mealy foi testado se para uma determinada sentença o sistema simulava a saída correspondente às transições; da mesma forma foram procedidos testes em Moore, no qual foi possível verificar que se para cada símbolo da

sentença o sistema executava a saída associada ao estado de origem e ao estado de destino. Por fim, foi testado se a ferramenta suportava um número relativamente grande de estados.

Terminados os testes da ferramenta, constatou-se que a mesma atingiu o objetivo geral e os objetivos específicos, na qual os usuários, geralmente acadêmicos, poderão interagir facilmente com os autômatos finitos com saídas, criando autômatos finitos na forma gráfica ou tabular e transformando-os em máquinas de Mealy ou Moore, basta escolher o tipo de saída (texto, imagem ou som), associar uma saída para cada transição ou estado, informar a sentença e simular a saída que será exibida na forma do tipo escolhido.

CONCLUSÃO

O avanço tecnológico vivido atualmente torna de fundamental importância o uso de ferramentas de ensino aprendizagem, pois o aluno que interage com o assunto abordado consegue absorver mais conteúdo do que aquele que estuda da forma tradicional.

Pensando nisso, criou-se a *shell* AFLAB, que teve suas funcionalidades ampliadas nessa pesquisa, pela implementação dos módulos relacionados aos autômatos finitos com saída (Máquina de Mealy e Máquina de Moore). Essa ferramenta pode ser utilizada livremente pelos acadêmicos do Curso de Ciência da Computação, na disciplina de Linguagens Formais, para complementar seus estudos sobre autômatos finitos com saída.

Durante toda a elaboração desse trabalho se fez necessário um estudo bibliográfico sobre máquinas de estados, autômatos finitos (determinísticos e não determinísticos) e autômatos com saída, porém, teve-se a dificuldade de encontrar material publicado sobre essa área da Ciência da Computação. Compreendido os autômatos, precisou-se entender como o reconhecimento de sentença foi implementado no AFLAB, uma vez que não existe transformação entre autômatos não determinístico e determinístico.

Após ter compreendido os autômatos e o funcionamento do AFLAB, iniciou-se a modelagem e implementação do sistema, durante essa etapa houve dificuldades específicas de cada máquina. A implementação propriamente dita, foi iniciada pela máquina de Mealy, na qual houveram alguns testes, por exemplo, para saber se havia apenas um símbolo e se este atingia um estado inicial e final, concomitantemente.

A implementação da Máquina de Moore foi mais complexa, pois se fez necessário desenvolver um algoritmo que lesse o símbolo da sentença, comparasse com o da transição para descobrir o estado de origem e destino, e a partir desses estados, executasse a saída associada. Em Moore foi necessário um tratamento destinado à sentença que possua somente um símbolo, pois este deve executar a saída do estado de origem e de destino. Nas sentenças com mais de um símbolo, somente é executada a saída associada ao estado de origem. Por fim, no último símbolo é executada a saída do estado de origem e de destino.

Desta forma, os objetivos desta pesquisa foram atingidos, pois a ferramenta permite a simulação de máquinas de Mealy e Moore e, consequentemente, a interação do usuário com os autômatos finitos com saída, conforme proposto.

Não foi contemplada, nessa pesquisa, a simulação de saídas para autômatos finitos não determinísticos, por esse motivo foi colocada uma verificação que não permite a transformação de AFND em Máquina de Mealy ou Moore. Assim, fica sugerida como trabalhos futuros, a transformação de AFND em AFD para que possam ser simulados autômatos finitos com saída por meio de autômatos finitos não determinísticos.

A criação de uma Máquina de Turing e autômatos com pilha também serão bem-vindas ao AFLAB. Outra sugestão é a conversão do código do sistema para uma linguagem multi-plataforma, permitindo ser executado via Web e em outros sistemas operacionais.

REFERÊNCIAS

ACCORSI, Fernando. **Animação Bidimensional para World Wide Web Baseada em Autômatos Finitos**. Dissertação (Mestrado em Ciência da Computação), Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D. **Compiladores: princípios, técnicas e ferramentas**. Tradução de Daniel de Ariosto Pinto. Rio de Janeiro: Livros Técnicos e Científicos , 1995.

BASTOS, Eduardo; SALVADOR, Otávio. **AFCHECKER**. Trabalho na disciplina de Linguagens Formais. Local – UCPEL, 2002. Disponível em http://afchecker.codigolivre.org.br/licenca.php. Acessado em: 20 ago. 2007.

CRESPO, Rui Gustavo. **Processadores de linguagens:** da concepção à implementação. Lisboa: IST Press, 2001.

FERNANDES, Daniel; **Desenvolvimento e Implementação de Animações Computacionais Baseadas em Autômatos Finitos com Saída utilizando a Teoria das Categorias**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Universidade do Extremo Sul Catarinense, Criciúma, 2005.

FURTADO, Olinto J. Varela; DE LUCCA, José Eduardo; KREISS, Rodrigo. **EDITAB**: editor de autômatos finitos por tabela. Trabalho na disciplina de Introdução a Compiladores. Universidade Federal de Santa Catarina, Florianópolis, 1992. Disponível em http://www.inf.ufsc.br/~olinto/publica.htm. Acessado em: 20 ago. 2007.

FURTADO, Olinto J. Varela; DE LUCCA, José Eduardo; KREISS, Rodrigo. **EDIGRAF**: editor gráfico de autômatos finitos. Trabalho na disciplina de Introdução a Compiladores. Universidade Federal de Santa Catarina, Florianópolis, 1991. Disponível em http://www.inf.ufsc.br/~olinto/publica.htm. Acessado em: 20 ago. 2007.

GAIDZINSKI, Marco Aurélio. **Ambiente de criação e manipulação de autômatos finitos na forma gráfica ou tabular para o reconhecimento de sentenças**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Universidade do Extremo Sul Catarinense, Criciúma. 2007.

GERSTING, Judith L. **Fundamentos matemáticos para a ciência da computação**. 3.ed. Rio de Janeiro: LTC, 1995.

GIOVANETTI, João Leonardo R.; PANICK, Luiz César; COSTA, Yandre M. Gomes. **Sistema de Gerenciamento de Autômatos Finitos Determinísticos.** Trabalho da disciplina de Linguagens Formais e Autômatos. Universidade Paranaense, Paranavaí, 2000.

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. **Introdução à teoria de autômatos, linguagens e computação**. Tradução de Vandenberg D. de Souza. Rio de Janeiro: Elsevier, 2002.

LEWIS, Harry R.; PAPADIMITRIOU, Christos H. **Elementos da teoria da computação**. 2. ed. Porto Alegre: Bookman, 2004.

MENEZES, Paulo Fernando Blauth. **Linguagens formais e autômatos**. 5.ed. Porto Alegre: Sagra Luzzatto, 2005.

SOUZA, Diego et al. **ENSINET/NAV:** Uma Ferramenta para Estruturação de Cursos Baseados em Objetos de Aprendizagem**.** Novas Tecnologias na Educação CINTED-UFRGS. Porto Alegre: 2004.

STACHOVOSKI, Lislaine. **AGASTUDIO**: ambiente de criação e manipulação de animações baseadas em autômatos finitos. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Universidade do Extremo Sul Catarinense, Criciúma, 2005.

VIEIRA, Luiz Filipe M.; VIEIRA, Marcos Augusto M.; VIEIRA, Newton José; **LANGUAGE EMULATOR.** Trabalho no Curso de Pós Graduação. Universidade Federal de Minas Gerais, Belo Horizonte, 2002. Disponível em http://homepages.dcc.ufmg.br/~lfvieira/ftc.html. Acessado em: 01 set. 2007.

VIEIRA, Newton José; **Introdução aos Fundamentos da Computação:** Linguagens e Máquinas. São Paulo: Pioneira Thomson Learning, 2006.

BIBLIOGRAFIA COMPLEMENTAR

CANTÚ, Marco. **Dominando o Delphi 7**: a bíblia. Tradução de João Eduardo Nóbreg Tortello. São Paulo: Makron Books, 2003.

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. **Introduction to automata theory, languages and computation**. 2.ed. New York: Addison-Wesley, 2001.

MACHADO, J. P. **Hyper-Automaton: hipertextos e cursos na web usando autômatos finitos com saída,** Dissertação (Mestrado em Ciência da Computação). Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.

RUMBAUGH, James; JACOBSON, Ivar. **UML**: guia do usuário. Rio de Janeiro: Ed. Campus, 2000.

APÊNDICE A - ARTIGO

Autômatos Finitos com Saída: Um Ambiente de Criação e Manipulação das Máquinas de Moore e Mealy no AFLAB

Marlon de Matos de Oliveira[1], Christine Vieira Scarpato[1].

[1]Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
Caixa Postal 3167 - 88.806-000 – Criciúma – SC – Brasil

oliwer.marlon@gmail.com, cvi@unesc.net.

Abstract. *This paper presents a research about finite automata with output, Moore and Mealy machines with the implementation respective in the AFLAB. The AFLAB don't have the Moore and Mealy modules. For this reason, this machines have been developed, accepting the use of deterministic finite automata and change them in a Moore or Mealy machines to simulate Text, Images or Sounds output through a valid sentence. The result is a new version of the AFLAB with those machines, where it's possible to simulate the chosen output from deterministic finite automata.*

Resumo. *O presente artigo apresenta uma pesquisa sobre autômatos finitos com saída, máquinas de Mealy e Moore com sua respectiva implementação no AFLAB. Pelo fato do AFLAB não possuir os módulos de Mealy e Moore foram desenvolvidos, neste trabalho, os presentes módulos, que permitem utilizar autômatos finitos determinísticos e transformá-los em um autômato finito com saída, para simulação de saídas em Textos, Imagens ou Sons, por meio de uma sentença válida. Como resultado, obteve-se uma versão do AFLAB com essas máquinas, onde a partir de um autômato finito determinístico consegue-se simular a saída na forma escolhida.*

1. Introdução

As linguagens formais são conjuntos de palavras sobre alfabetos Essas podem ser representadas de maneira finita e precisa, através de sistemas com sustentação matemática [Menezes 2005]. Na ciência da computação o uso desse formalismo é indispensável, pois, os profissionais dessa área necessitam entender várias linguagens.

Autômatos finitos são máquinas abstratas de estados e são utilizadas como reconhecedores de linguagens em Linguagens Formais. Esse reconhecimento é limitado à aceitação ou rejeição. Existe ainda uma extensão desses autômatos, que são os autômatos finitos com saída, no qual a saída pode ser associada a uma transição ou a um estado.

Esses conteúdos são abordados na disciplina de Linguagens Formais, que é normalmente estudada da forma tradicional, ou seja, papel e caneta, restringindo assim a dimensão do ensino dos autômatos. Pensando nisso, criou-se o Grupo de Pesquisa em Linguagens Formais do Curso de Ciência da Computação da UNESC, que visa o desenvolvimento de uma ferramenta para simulação de autômatos finitos. A *shell* é denominada AFLAB e tem como objetivo auxiliar os alunos de Linguagens Formais no aprendizado de autômatos finitos e suas derivações.

Baseado na proposta do AFLAB e tendo em vista que o mesmo não possui os módulos de autômatos finitos com saída, verificou-se a necessidade de implementar esses tipos de autômatos. Esta pesquisa complementará dois módulos do AFLAB: um será a construção de um autômato baseado na máquina de Mealy e o outro na máquina de Moore, nos quais o usuário poderá escolher a saída para simular o funcionamento de cada um deles.

2. Autômato Finito com Saída

Sem alterar a classe dos autômatos finitos como reconhecedores de linguagens regulares, pode-se acrescentar-lhes algumas saídas, então esses autômatos passam a se chamar de Autômatos Finitos com Saída (AFS). Esses autômatos não têm a saída limitada à lógica binária aceita/rejeita, podendo ter essa saída de vários tipos, por exemplo, a geração de palavras [Menezes 2005].

Essas extensões dos autômatos foram criadas para torná-los mais potentes, uma vez que “frases” podem ser associadas às saídas. A incorporação dessa frase de saída pode ser dada de duas formas, dependendo de onde estão incorporadas as ações de escrita das “palavras” [Crespo 2001].

Quando as saídas são associadas às transições têm-se as máquinas de Mealy, já quando essas saídas estão ligadas aos estados, têm-se as máquinas de Moore. Porém, em ambas as máquinas, as saídas não podem ser lidas, sendo assim, não podem ser utilizadas como memória auxiliar [Menezes 2005].

Autômatos finitos com saída são como segue [Menezes 2005]:

e) definidas sobre um alfabeto especial, chamado alfabeto de símbolos de saída, que pode ser igual ao alfabeto de entrada;
f) a saída é armazenada em uma fita de saída, independente da fita de entrada;
g) a cabeça da fita de saída move uma célula para direita a cada símbolo gravado;
h) o resultado do processamento do autômato é o seu estado final (aceita/rejeita) e as informações gravadas na fita de saída.

2.1. Máquina de Mealy

A Máquina de Mealy é uma modificação de um autômato finito, de forma a gerar uma saída para cada transição da máquina [Menezes 2005].

Uma Máquina de Mealy é formalmente definida por uma sêxtupla, na qual as saídas são associadas às transições. O modelo matemático para uma máquina de Mealy é M = (Σ, Q, δ , q_0, F, Δ) na qual [Menezes 2005]:

a) Σ é um conjunto de estados não vazio e finito;
b) Q é o alfabeto de símbolos de entrada (conjunto finitos e não vazio);δ é a função de transição de estados: δ: Σ x Q $\rightarrow$ Σ x Δ;
c) q_0 é o estado inicial, um elemento distinguido de Σ;
d) F é um subconjunto de Σ, chamado de conjunto de estados finais;
e) Δ é um conjunto de símbolos de saída, ou simplesmente alfabeto de saída.

Note que em uma Máquina de Mealy, Σ, Q, δ e q_0 são como nos Autômatos Finitos Determinísticos, portanto Mealy e AFD são semelhantes, com a diferença que em vez de aceitação/rejeição existirá uma palavra de saída [Vieira 2006].

Em um grafo rotulado, as etiquetas das setas possuem o formato y/x, no qual y e x são palavras do alfabeto de entrada e do alfabeto de saída, respectivamente [Crespo 2001]. Veja na Figura 1 um exemplo de máquina de Mealy na forma gráfica.

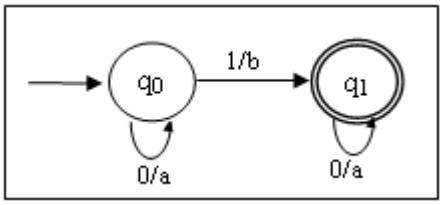


Figura 1. Diagrama de transições de uma Máquina de Mealy

2.2. Máquina de Moore

Uma Máquina de Moore nada mais é do que uma modificação de um autômato finito (AF), no qual existe uma saída associada a cada estado [Vieira 2006].

Formalmente uma Máquina de Moore é definida por uma séptupla M = (Σ, Q, δ , q_0, F, Δ, δs) na qual [Menezes 2005]:

h) Σ é um conjunto finito de estados possíveis;
i) Q é o alfabeto finito de símbolos de entrada;
j) δ é uma função programa ou função de transição, δ: Σ x Q $\rightarrow$ Σ;
k) q_0 é o estado inicial, um elemento oriundo de Σ;
l) F é um ou mais elemento de Σ, definido como conjunto de estados finais;
m)Δ é um conjunto de símbolos de saída;
n) δs é uma função de saída, na qual δs: $\Sigma \rightarrow \Delta^*$ (função total)

Como visto no exposto Σ, Q, δ , q_0 e F são como nos Autômatos Finítos e Δ é como na Máquina de Mealy. A função de transição pode ser escrita como um diagrama nos AF, somando por sua vez, na etiqueta de cada estado, a saída associada, quando diferente de vazio [Menezes 2005].

O funcionamento da Máquina de Moore para uma determinada entrada, consiste em uma sucessiva aplicação da função de transição para cada símbolo de entrada lido, até que ocorra uma condição de parada, juntamente com a sucessiva aplicação da função de saída a cada estado alcançado. A saída vazia, resultante da função de saída, corresponde a nenhuma gravação feita na fita e, portanto, a cabeça da fita de saída não se move. Observe que se todas as saídas forem vazias, a Máquina de Moore irá se comportar como um AF [Menezes 2005].

A representaçao gráfica de uma Máquina de Moore é semelhante ao de um AF, no entanto, em cada vértice, ao invés de representar somente um estado, representa um estado e a saída correspondente [Vieira 2006].

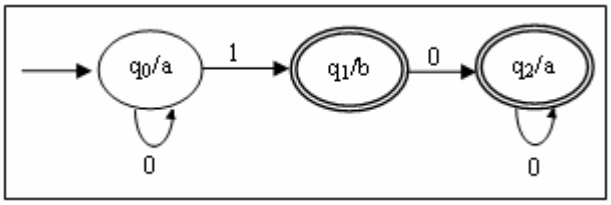


Figura 2. Exemplo de Máquina de Moore na forma de diagrama

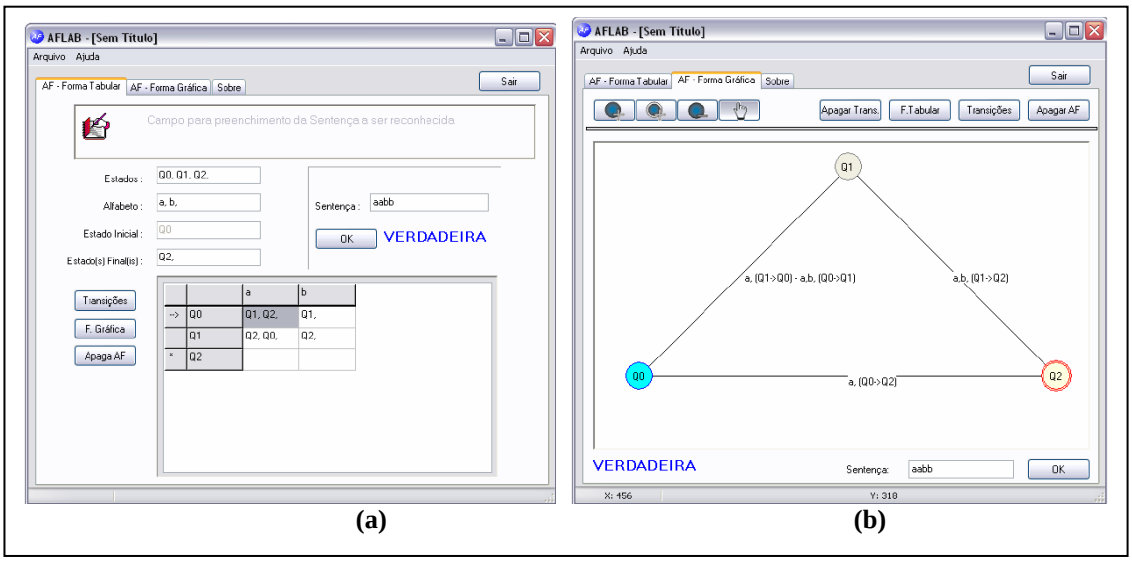
O exemplo da Figura 2 é uma máquina de Moore, na qual a função de saída se dá no momento em que o cabeçote de leitura atinge um determinado estado. Na relação y/x, y é o estado lido e x é a saída proporcionada.

A Máquina de Moore mantém as mesmas características de um AF, na qual uma sentença analisada tem a saída padrão igual a aceita/rejeita e mais uma saída proporcionada pela função de saída [Vieira 2006].

3. A Ferramenta AFLAB

A ferramenta AFLAB foi desenvolvida em um Trabalho de Conclusão de Curso de Ciência da Computação, da Universidade do Extremo Sul Catarinense (UNESC), pelo acadêmico Marco Aurélio Gaidzinski. Esse software possui um módulo para representação de um AF em sua forma tabular e o outro é a representação desse mesmo autômato na forma gráfica.

Ao criar um autômato no AFLAB é necessário primeiro escolher se o mesmo será criado na forma tabular ou gráfica, isso deve ser feito selecionando a aba correspondente, como pode ser visto na figura 3(a) [Gaidzinski 2007].



(a) **(b)**

Figura 3. Interface do AFLAB, forma tabular e gráfica

Depois de escolher a forma que o autômato será criado, deve-se informar os estados, os símbolos de entrada e as transições. Caso a escolha de criação do AF tenha sido pela tabela de transições (guia AF – Forma Tabular), os nomes dos estados, o alfabeto, o estado inicial e os estados finais devem ser informados nos campos apropriados, logo em seguida o botão *transições* deve ser clicado para informar as transições [Gaidzinski 2007].

Na Figura 3(b) pode-se visualizar um autômato construído na forma gráfica, onde as cores ajudam a facilitar a identificação dos estados, o estado inicial será azul, o estado final será contornado com uma linha dupla em vermelho e o cinza indica os demais estados. As transições estão nas arestas na forma “a, (Q1 -> Q2)” onde “a” representa o símbolo lido e (Q1 -> Q2) indicam os estados de origem e destino, respectivamente.

O AFLAB utiliza uma única estrutura para o armazenamento dos AF, permitindo que esse seja determinístico ou não determinístico, sem restrições. Outra característica importante é a de que essa mesma estrutura irá armazenar os autômatos, tanto na forma tabular como na forma gráfica, permitindo assim que um autômato criado na forma tabular seja visualizado na forma gráfica e vice-versa [Gaidzinski 2007].

Essa estrutura foi divida, basicamente, em três classes *Testados, Talfabeto* e *Tconexao*, onde, *Testados* tem a função de gravar os atributos dos estados, como rótulo, se é inicial ou final e as coordenadas responsáveis pela localização do mesmo na forma gráfica. A classe *Talfabeto* é responsável pelo armazenamento dos símbolos de entrada. Por último, a classe *Tconexao* é responsável pelas informações das transições como estado inicial, símbolo de entrada e estado de destino [Gaidzinski 2007].

No reconhecimento de sentenças, tanto com AFD quanto para autômato finito não determinístico (AFND), foi utilizada uma técnica onde a estrutura do AF é baseada no conceito de árvores. A base da árvore é criada quando se lê o estado inicial com o primeiro símbolo de entrada, gerando um novo conjunto de estados (novos ramos da árvore). A seguir é lido o próximo símbolo de entrada, que será comparado com os estados desse novo conjunto de estados e gerado um novo conjunto de estados, isso se repete até que o último símbolo de entrada seja lido [Gaidzinski 2007].

A identificação do reconhecimento ou não da linguagem é feito de modo que, se o último conjunto de estado criado possui pelo menos um estado final a linguagem é reconhecida, caso contrário ela será rejeitada [Gaidzinski 2007].

4. Autômatos Finitos com Saída no AFLAB

Os módulos de máquinas de Moore e Mealy foram implementados no AFLAB com a finalidade de ampliar suas funcionalidades. Essas possuem três tipos de saídas cada uma: Textos, Imagens e Sons, esses podem ser associados às transições ou aos estados, dependendo da máquina.

4.1. Metodologia

Antes de iniciar a implementação do software foi necessário definir as saídas mais convenientes, onde se preferiu utilizar, texto (letras, palavras ou frases), imagens (figuras ou fotos bitmaps) e arquivos de áudio (mp3, wav e midi), pois, esses tipos de mídias são comumente utilizadas pelos acadêmicos.

A implementação foi iniciada com a modelagem da ferramenta em UML. Nessa etapa o software foi modelado em Linguagem Universal de Modelagem, no qual foram obtidos alguns diagramas.

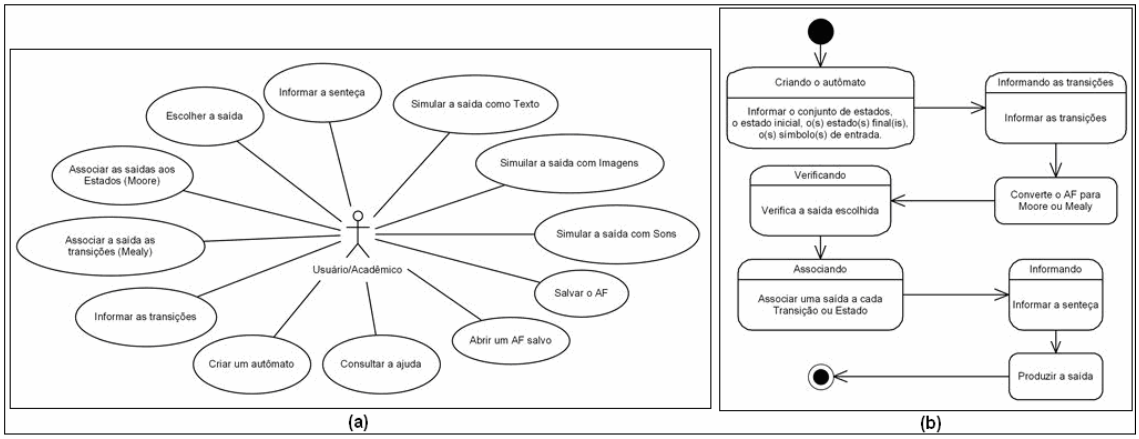


Figura 4. Diagrama de Caso de Uso e de estados

Na Figura 4(a) pode-se observar um diagrama de caso de uso do ponto de vista de um usuário, que normalmente será um acadêmico, esse diagrama é genérico, ou seja, serve tanto para o módulo de Mealy quanto para o módulo de Moore. Por fim, foi criado um diagrama de estados, que pode ser visualizado na Figura 4(b), esse diagrama mostra o estado da máquina para cada ação do usuário.

4.2. Desenvolvimento

Os módulos de Mealy e Moore foram desenvolvidos baseados na estrutura dos autômatos criados no AFLAB, ou seja, o autômato é gerado e armazenado conforme descrito no item 3 desse artigo, e a partir disso os autômatos finitos com saídas são criados.

Após criar um autômato na forma tabular ou gráfica e acionar o botão Mealy ou Moore, o sistema irá buscar os dados do autômato nas classes armazenadas e irá construir um AFS de acordo com a máquina escolhida. Para que isso seja possível é necessário que o autômato esteja devidamente criado e com as transições informadas, pois AFS somente podem ser simulados por AF que possuam transições.

4.3. A Criação de um AFS a partir de um AFD

Ao clicar no botão *Mealy* ou *Moore* o sistema faz uma série de verificações, tais como, se o autômato está criado, se existem transições e se esse autômato é determinístico, depois do sistema ter feito os devidos testes, os dados do AF são buscados e montados em uma grade onde aparece uma coluna mostrando todas as transições (Mealy) ou estados (Moore), e outra vazia para que o usuário possa informar a saída desejada.

O componente *client data set* (liga as aplicações aos bancos de dados)**,** foi utilizado, porque esse tipo permite muita flexibilidade, pois, basta conectar uma dbgrid (grade para banco de dados) a um desses componentes para se ter acesso às informações inseridas nela. No caso do AFLAB não existe conexão com banco de dados, todo o processamento é feito na memória principal.

Nos client datas sets foram criadas duas colunas, uma chamada internamente de Campo1 para transição, no caso de Mealy ou estados no caso de Moore, e outra chamada Campo2 para se informar a saída associada a ela. Na grade associada a este componente, a primeira coluna se chama Transições ou Estados, dependendo da máquina, e a segunda coluna se chama Saída em ambos os casos.

O funcionamento das funções de saída é basicamente como a de reconhecimento. O sistema passa por todos os estados acessíveis e a cada transição

válida, é consultada na grade qual a saída está associada à transição ou a cada estado, dependendo da máquina.

A execução dessa saída pode ser de três formas:

a) texto: onde é exibido um componente do tipo TMemo (quadro de texto), que recebe uma linha para cada saída da máquina.
b) imagem: no qual é exibido um componente do tipo TImage que recebe as imagens associadas às transições ou estados, dependendo da máquina.
c) som: que por sua vez executa o som em um componente do tipo TMediaPlayer (reprodutor de sons), esse componente é invisível, embora seja utilizado em toda função de saída.

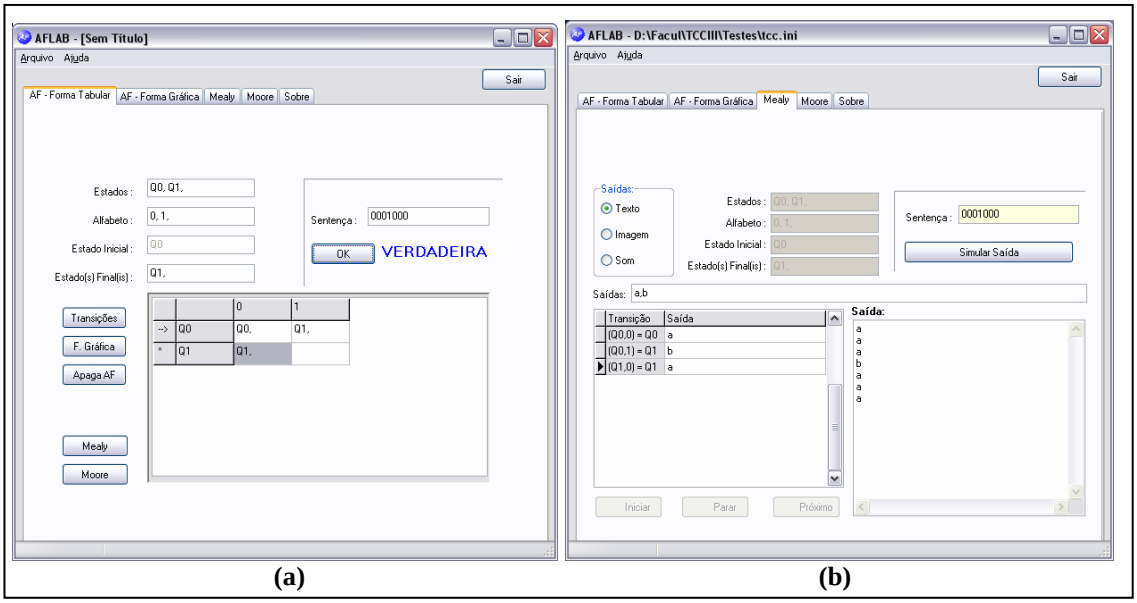
A reprodução da saída de Mealy foi divida em três funções: *SaidasMealyTexto(EstadoInicial, Sentenca: string; tempo: integer), SaidasMealyImagem(EstadoInicial, Sentenca: string) e SaidasMealySom(EstadoInicial, Sentenca: string),* ambas retornam um tipo *booleano*, ou seja, *true* se verdadeiro ou *false* se falso. Esse retorno serve para saber se a sentença informada é válida ou não, pois, caso não seja, a saída do autômato será executada até a parte em que for válida, no caso de texto e imagem, já no caso de som a saída não será simulada. Essas funções recebem parâmetros do tipo *string* (conjunto de caracteres alfanuméricos) com o estado inicial e a sentença a ser simulada.

A implementação da máquina de Moore foi um pouco mais complexa do que a de Mealy, pois nessa, a cada estado atingido, o sistema precisa executar a saída correspondente, sendo assim, nas funções de saída teve-se que testar se era o primeiro símbolo lido, se esse símbolo estava em alguma transição e executar a saída correspondente ao estado inicial dessa transição. Nos demais símbolos, apenas foi preciso consultar a grade, verificar se havia um estado igual ao estado da transição e executar a saída. No último símbolo lido foi preciso executar a saída referente à transição dele e também a do símbolo atingido na referida transição.

4.4. Simulação de uma Saída

A simulação de uma saída em uma máquina de Moore ou Mealy é precedida da criação de um autômato finito determinístico (AFD) no AFLAB, para isso, devem-se informar os estados, o alfabeto e as transições, que pode ser tanto na forma gráfica quanto na forma tabular. Após criar o autômato, o botão Mealy ou o botão Moore deve ser pressionado. Observe na Figura 5(a) um autômato criado na forma tabular com a respectiva máquina de Mealy (Figura 5(b)).

A sentença deverá obrigatoriamente ser informada no campo Sentença, e o botão Simular saída deve ser acionado para visualização do texto de saída da máquina. Essa sentença deve ser válida, caso contrário, o sistema irá emitir uma mensagem: *A sentença informada é inválida*. Já se a sentença não for informada, a aplicação irá mostrar a mensagem: *Não existe sentença a ser simulada*.



(a) **(b)**

Figura 5. Máquina de Mealy criada a partir de um AF

Na ação do botão Simular saída, o sistema também exige que todas as transições ou estados tenham uma saída associada, caso isso não ocorra, será exibida a mensagem: *Existe alguma transição/estado sem saída informada.*

4.4.1. Saída em Forma de Texto

Após ter escolhido o tipo da saída *Texto* e informado o conjunto de saídas no devido campo (separados por vírgula), é necessário associar um texto para cada transição, no caso de Mealy, ou para cada estado, no caso de Moore. Isso se fará possível na grade, onde na primeira coluna se visualiza a transição (Mealy) ou o estado (Moore) e na segunda coluna se informa a saída. Essa será uma das informadas anteriormente. Com a intenção de facilitar ao usuário, quando este tiver que associar a saída de texto a uma transição ou a um estado, está disponível na segunda coluna da grade um campo do tipo *combo box* com as opções possíveis.

Por fim, a saída de texto é exibida no *memo*, localizado na parte inferior direita da janela, as saídas são separadas por linha com um intervalo de 1 segundo entre uma e outra. Esse tempo é fixado no código fonte e não pode ser alterado pelo usuário.

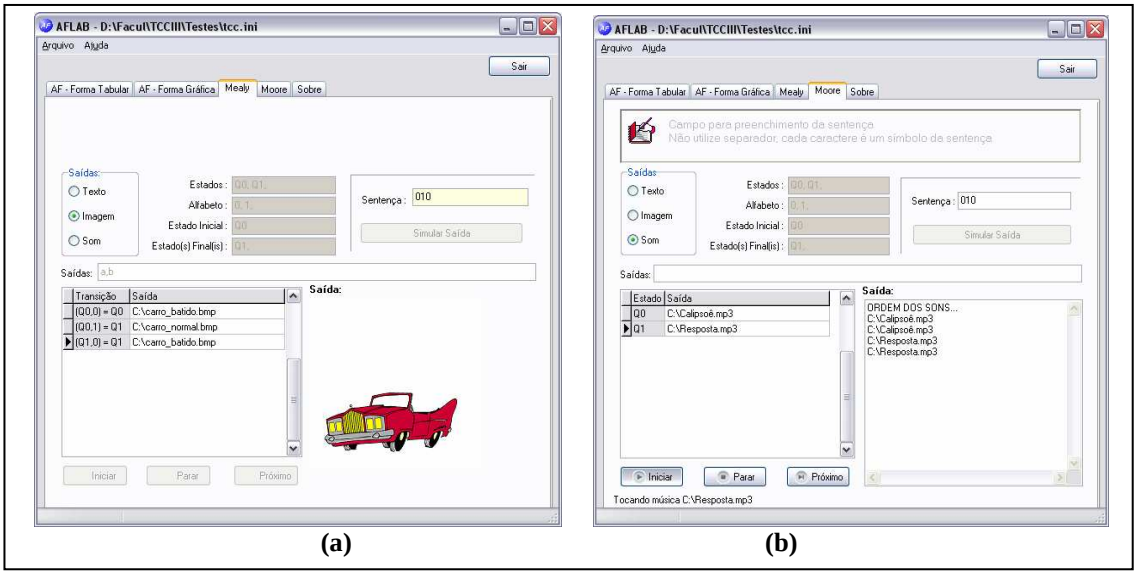
4.4.2. Saída em Forma de Imagem e Som

Após ter escolhido o tipo de saída *Imagem* ou *Som* é necessário associar uma figura/música para cada transição no caso de Mealy, ou para cada estado, no caso de Moore. Isso se fará possível na grade, pois, na primeira coluna se visualiza a transição/estados e na segunda se informa a saída. Essa saída será, obrigatoriamente, um arquivo de imagem com extensão: Bmp, ico, emf, wmf ou um arquivo de som com extensão: Waw, Midi ou Mp3. Sempre que a saída for de imagens ou som, a segunda coluna da grade libera um botão com o símbolo: "...", que, quando clicado, abre uma nova janela para o usuário selecionar o referido arquivo.

Por fim, a seqüência de imagens é exibida na parte inferior direita da janela, com um intervalo de 1 segundo entre uma e outra. Na Figura 6(a) se pode ter uma visão melhor da saída em forma de imagem. Na Figura 6(b) pode-se observar a seqüência de

sons executada e os botões *Iniciar, Parar* e *Próximo* liberados na parte inferior da janela.

Enquanto um arquivo de áudio estiver sendo executado, o botão *Iniciar* ficará pressionado e os botões *Parar* e *Próximo* ficam habilitados. Quando pressionado o botão *Parar,* o sistema pára a execução da música e cancela a simulação da máquina. Já ao pressionar o botão *Próximo,* o sistema pula para o próximo som. Na Figura 6(b) pode se notar a execução de um arquivo do tipo mp3 na máquina de Moore, onde o nome e o caminho do áudio, que está em execução, aparecem no *label* localizado no final da janela. Na simulação de AFS com som também foi utilizado o *memo* da saída *Texto* para mostrar a ordem da execução dos sons.



(a) **(b)**

Figura 6. Máquina de Mealy simulando saída com Imagem e Som

Ao terminar a exibição das imagens, o sistema apaga a última figura no caso de imagem ou desabilita os botões de controle do media player no caso de som, e libera o botão *Simular saída* para uma próxima simulação.

5. Resultados Obtidos

Os módulos de máquina de Mealy e Moore foram implementados totalmente baseados na estrutura de autômatos finitos já existentes no AFLAB, isso reduziu o número de linhas do código fonte e o tornou mais claro.

Durante e após a implementação foram necessários alguns testes. Por exemplo, em uma máquina de Mealy foi testado se para uma determinada sentença o sistema simulava a saída correspondente às transições; da mesma forma foram procedidos testes em Moore, no qual foi possível verificar que se para cada símbolo da sentença o sistema executava a saída associada ao estado de origem e ao estado de destino. Por fim, foi testado se a ferramenta suportava um número relativamente grande de estados.

Terminados os testes da ferramenta, constatou-se que a mesma atingiu seu objetivo, na qual os usuários poderão interagir facilmente com os autômatos finitos com saídas, criando autômatos finitos na forma gráfica ou tabular e transformando-os em máquinas de Mealy ou Moore.

6. Considerações Finais

O avanço tecnológico vivido atualmente torna de fundamental importância o uso de ferramentas de ensino aprendizagem, pois o aluno que interage com o assunto abordado consegue absorver melhor o conteúdo do que aquele que estuda da forma tradicional.

Pensando nisso, criou-se a *shell* AFLAB, que teve suas funcionalidades ampliadas nessa pesquisa, pela implementação dos módulos relacionados aos autômatos finitos com saída (Máquina de Mealy e Máquina de Moore). Essa ferramenta pode ser utilizada livremente pelos acadêmicos do Curso de Ciência da Computação, na disciplina de Linguagens Formais, para complementar seus estudos sobre autômatos finitos com saída.

Após ter compreendido os autômatos e o funcionamento do AFLAB, iniciou-se a modelagem e implementação do sistema. Durante essa etapa houve dificuldades específicas de cada máquina. A implementação propriamente dita, foi iniciada pela máquina de Mealy, na qual foram realizados alguns testes, por exemplo, para saber se havia apenas um símbolo e se este atingia um estado inicial e final, concomitantemente.

A implementação da Máquina de Moore foi mais complexa, pois se fez necessário desenvolver um algoritmo que lesse o símbolo da sentença, comparasse com o da transição para descobrir o estado de origem e destino, e a partir desses estados, executasse a saída associada.

6.1. Trabalhos Futuros

Com a intenção de continuar a ampliação das funcionalidades do AFLAB, fica como sugestão para trabalhos futuros a transformação de AFND em AFD para que possa ser simulado autômatos finitos com saída por meio de autômatos finitos não determinísticos. A criação de uma Máquina de Turing e autômatos com pilha também serão bem-vindas ao AFLAB. Outra sugestão é a conversão do código do sistema em uma linguagem multi-plataforma, permitindo ser executado via Web e em outros sistemas operacionais.

7. Referências

Crespo, Rui Gustavo. (2001) Processadores de linguagens: da concepção à implementação. Lisboa: IST Press.

Gaidzinski, Marco Aurélio. (2007) Ambiente de criação e manipulação de autômatos finitos na forma gráfica ou tabular para o reconhecimento de sentenças. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Universidade do Extremo Sul Catarinense, Criciúma.

Menezes, Paulo Fernando Blauth. (2005) Linguagens formais e autômatos. 5.ed. Porto Alegre: Sagra Luzzatto.

Vieira, Newton José. (2006) Introdução aos Fundamentos da Computação: Linguagens e Máquinas. São Paulo: Pioneira Thomson Learning.