DANILO PICAGLI SHIBATA

TRADUÇÃO GRAFEMA-FONEMA PARA A LÍNGUA PORTUGUESA BASEADA EM AUTÔMATOS ADAPTATIVOS

São Paulo
2008

DANILO PICAGLI SHIBATA

TRADUÇÃO GRAFEMA-FONEMA PARA A LÍNGUA PORTUGUESA BASEADA EM AUTÔMATOS ADAPTATIVOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia

Área de Concentração: Engenharia de Computação

Orientador: Prof. Dr. Ricardo Luis de Azevedo da Rocha

São Paulo
2008

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com anuência de seu orientador.

São Paulo __ de Abril de 2008.

Assinatura do Autor _____

Assinatura do Orientador _____

FICHA CATALOGRÁFICA

Shibata, Danilo Picagli
Tradução grafema-fonema para a língua portuguesa baseada em autômatos adaptativos / D.P. Shibata. -- São Paulo, 2008.
104 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Tradução automática 2. Linguagem natural 3. Alfabeto fonético 4. Teoria dos autômatos 5. Síntese da fala I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

AGRADECIMENTOS

À minha família.

Aos amigos do Coop, do Clube, do Grupo de Análise de Segurança e outros.

Ao meu orientador, Prof. Dr. Ricardo Rocha, pela orientação, a confiança (às vezes demasiada) demonstrada, pelas oportunidades de apresentar o trabalho para seus alunos e especialmente pela amizade.

Ao Fábio Koike que implementou o sintetizador de fala sem o qual o objetivo de criar um tradutor texto-voz não seria concretizado.

Ao Prof. Dr. Marcelo Finger do IME, por fornecer o etiquetador morfológico utilizado durante a fase de testes.

Ao Prof. Dr. João José Neto pela amizade, pela ajuda na resolução de dúvidas sobre os autômatos e pelo incentivo.

Ao Prof. Dr. João Batista Camargo Jr., ao Prof. Dr. Jorge Rady de Almeida Jr. e ao Prof. Dr. Paulo Sérgio Cugnasca do GAS, pela oportunidade de trabalhar na área de análise de sistemas críticos em segurança ampliando meus horizontes como engenheiro, pela orientação durante esse período, pela compreensão e pelo incentivo.

RESUMO

Este trabalho apresenta um estudo sobre a utilização de dispositivos adaptativos para realizar tradução texto-voz. O foco do trabalho é a criação de um método para a tradução grafema-fonema para a língua portuguesa baseado em autômatos adaptativos e seu uso em um software de tradução texto-voz. O método apresentado busca mimetizar o comportamento humano no tratamento de regras de tonicidade, separação de sílabas e as influências que as sílabas exercem sobre suas vizinhas. Essa característica torna o método facilmente utilizável para outras variações da língua portuguesa, considerando que essas características são invariantes em relação à localidade e a época da variedade escolhida. A variação contemporânea da língua falada na cidade de São Paulo foi escolhida como alvo de análise e testes neste trabalho. Para essa variação, o modelo apresenta resultados satisfatórios superando 95% de acerto na tradução grafema-fonema de palavras, chegando a 90% de acerto levando em consideração a resolução de dúvidas geradas por palavras que podem possuir duas representações sonoras e gerando uma saída sonora inteligível aos nativos da língua por meio da síntese por concatenação baseada em sílabas. Como resultado do trabalho, além do modelo para tradução grafema-fonema de palavras baseado em autômatos adaptativos, foi criado um método para escolha da representação fonética correta em caso de ambigüidade e foram criados dois softwares, um para simulação de autômatos adaptativos e outro para a tradução grafema-fonema de palavras utilizando o modelo de tradução criado e o método de escolha da representação correta. Esse último software foi unificado ao sintetizador desenvolvido por Koike et al. (2007) para a criação de um tradutor texto-voz para a língua portuguesa. O trabalho mostra a viabilidade da utilização de autômatos adaptativos como base ou como um elemento auxiliar para o processo de tradução texto-voz na língua portuguesa.

Palavras-chave: Autômatos Adaptativos. Processamento de Linguagens Naturais. Lingüística Computacional. Tradução Texto-Voz. Alfabeto Fonético Internacional.

ABSTRACT

This work presents a study on the use of adaptive devices for text-to-speech translation. The work focuses on the development of a grapheme-phoneme translation method for Portuguese based on Adaptive Automata and the use of this method in a text-to-speech translation software. The presented method resembles human behavior when handling syllable separation rules, syllable stress definition and influences syllables have on each other. This feature makes the method easy to use with different variations of Portuguese, since these characteristics are invariants of the language. Portuguese spoken nowadays in São Paulo, Brazil has been chosen as the target for analysis and tests in this work. The method has good results for such variation of Portuguese, reaching 95% accuracy rate for grapheme-phoneme translation, clearing the 90% mark after resolution of ambiguous cases in which different representations are accepted for a grapheme and generating phonetic output intelligible for native speakers based on concatenation synthesis using syllables as concatenation units. As final results of this work, a model is presented for grapheme-phoneme translation for Portuguese words based on Adaptive Automata, a methodology to choose the correct phonetic representation for the grapheme in ambiguous cases, a software for Adaptive Automata simulation and a software for grapheme-phoneme translation of texts using both the model of translation and methodology for disambiguation. The latter software was unified with the speech synthesizer developed by Koike et al. (2007) to create a text-to-speech translator for Portuguese. This work evidences the feasibility of text-to-speech translation for Portuguese using Adaptive Automata as the main instrument for such task.

Keywords: Natural Language Processing. Computational Linguistics. Adaptive Automata. Text-to-Speech Translation. International Phonetic Alphabet.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

[...]: representação fonética utilizando o Alfabeto Fonético Internacional.

$|c|$: tamanho de uma coleção c.

$e \leftarrow f$: atribui a e o primeiro elemento da fila f.

$e \rightarrow c$: adiciona o elemento e à coleção c. No caso de filas, o elemento é adicionado no final.

$\coloneqq$: atribui a uma variável (à esquerda do símbolo) um valor ou o resultado de uma expressão (à direta do símbolo).

α: símbolos do autômato adaptativo utilizados para representação de influência anterior.

π: símbolos do autômato adaptativo utilizados para representação de influência posterior.

τ: símbolos do autômato adaptativo utilizados para representação de tonicidade.

μ: símbolos do autômato adaptativo utilizados para marcar transições.

σ: símbolos utilizados para representar Unidades de Analise Fonética.

SUMÁRIO

1 Introdução

A partir da criação de sistemas computacionais, um dos maiores desafios relacionados à utilização desses sistemas por seres humanos se encontra na criação de interfaces adequadas de comunicação entre as máquinas e seus controladores. Por isso, as interfaces de comunicação entre os usuários e as máquinas passam por um processo de melhora contínuo desde o advento desses sistemas. Um exemplo da evolução das interfaces pode ser encontrado nos sistemas operacionais, cujas interfaces de controle evoluíram das linhas de comando para interfaces gráficas em que o mouse e comandos de atalho do teclado podem realizar diversos comandos como copiar, salvar ou apagar arquivos. Hoje em dia, as interfaces dos sistemas operacionais são de simples entendimento, tanto que em poucas horas de uso de um sistema operacional novo é possível se sentir à vontade no controle de uma máquina.

Devido à evolução apresentada pelos mesmos sistemas computacionais que se tornam cada vez menores e mais eficientes na realização de tarefas mecanizadas, cada vez mais, essas máquinas vêm substituindo pessoas em tarefas que exigem interação com seres humanos. Dessa forma, é preferível ter um sistema com interface intuitiva (que não exija um período de treinamento) e que não frustre as expectativas dos clientes que interagem com a máquina. Um dos exemplos desse tipo de substituição pode ser visto em sistemas de reconhecimento de fala (KOTELLY, 2003) utilizado para atendimento telefônico ao cliente de diversas empresas. Inicialmente essa substituição foi feita utilizando o teclado como base para os comandos, mas com o passar do tempo ele vem sendo substituído por unidades de resposta audível – URAs.

Atrelada à capacidade de reconhecimento de fala, seria interessante que um sistema que interage com seres humanos tivesse a capacidade de sintetizar voz. Métodos de síntese de voz permitem ao sistema transmitir informação ao usuário de forma semelhante à que seria utilizada por outro interlocutor humano, dessa forma a experiência se assemelha à interação entre dois seres humanos. Este trabalho apresenta um experimento relacionado à síntese de voz na língua portuguesa baseado em autômatos adaptativos.

1.1 Objetivos

O trabalho apresentado neste documento tem como objetivo principal o estudo de técnicas de tradução texto-voz para textos escritos na língua portuguesa utilizando a tecnologia adaptativa

como base e regras estatísticas auxiliares para o processo de tradução. Outros trabalhos com o uso de autômatos adaptativos para tratamento de linguagens naturais são apresentados em José Neto e Moraes (2002), Menezes e José Neto (2002), Zuffo e Pistori (2004) e Alfenas et al. (2004), sendo os dois últimos relacionados à síntese de voz.

Devido às variações que a língua falada pode sofrer dependendo de localização e época analisada, as regras para tradução texto-voz foram criadas para se assemelharem às regras que definem a linguagem culta contemporânea da cidade de São Paulo.

O resultado principal desse trabalho é um modelo de autômato adaptativo, que associado a um etiquetador morfológico estatístico, é capaz de transformar o texto de entrada (escrito na língua portuguesa) na seqüência de fonemas que representa a fala de uma pessoa com a entonação típica de São Paulo lendo esse texto em voz alta de forma cadenciada.

Além do modelo de autômato, esse trabalho gerou dois softwares relacionados aos assuntos tratados, desenvolvidos na linguagem JAVA™. O primeiro software, de caráter mais genérico, é um simulador de autômatos adaptativos para demonstração do funcionamento dessas estruturas e teste dos modelos desenvolvidos tendo esse formalismo como base.

O segundo software foi desenvolvido sobre a plataforma do simulador de autômatos adaptativos. Esse segundo software é específico para a simulação do modelo de autômato adaptativo para tradução grafema-fonema desenvolvido nesse trabalho e é utilizado como base para realizar a tradução de textos da língua portuguesa para sua representação fonética.

1.2 Motivações

Existem diversas motivações para o estudo da tradução grafema-fonema a partir de sistemas computacionais. O primeiro deles decorre da evolução natural das interfaces entre homem e computador, buscando uma forma mais ágil e agradável de comunicação, que permita a utilização de computadores no lugar de seres humanos para tarefas mecânicas.

Utilizando ferramentas de síntese de voz é possivel ler documentos de diversos tipos. Essa característica pode ser utilizada para realizar a leitura de e-mails ou páginas da internet enquanto

uma pessoa realiza outra tarefa no computador. Essa mesma característica pode ser utilizada por deficientes visuais para tarefas que exigem leitura.

Com a mesma ferramenta de síntese de voz, é possível ajudar pessoas portadoras de deficiência auditiva e de fala. As pessoas portadoras de deficiência auditiva parcial ou completa no nascimento têm dificuldade para aprender a falar de forma correta. Uma aplicação de síntese de voz permite que as pessoas que apresentam dificuldade na fala possam se comunicar com pessoas que não entendem linguagens de sinais, como aqueles definidos pela língua brasileira de sinais (LIBRAS).

A motivação principal para escolha dos autômatos adaptativos foi determinada pela sua capacidade de reconhecer linguagens sensíveis ao contexto através de modificações em sua estrutura. As linguagens naturais apresentam essa característica devido à relação existente entre as palavras que se encontram em seus textos, o que torna a capacidade de diferenciar partículas do texto analilsado em função do contexto uma característica necessária ao modelo de tradução.

1.3 Metodologia

O trabalho desenvolvido busca obter a melhor representação sonora para o maior número de palavras da língua portuguesa utilizando um processo de tradução grafema-fonema baseado em autômatos adaptativos. O processo de desenvolvimento do sistema de síntese foi dividido em partes, criando inicialmente um núcleo de tradução de palavras em seqüências de fonemas e depois associando novas funcionalidades que buscam corrigir falhas e resolver não determinismos do modelo central para traduzir textos. A Figura 1 apresenta o esquema de funcionamento do Tradutor.

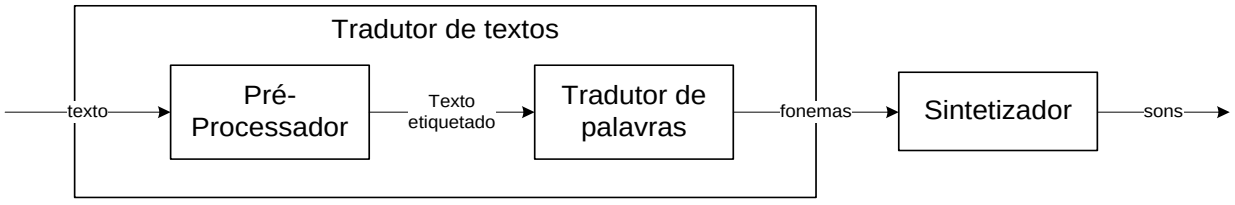

Figura 1 – Esquema do tradutor

O *Tradutor de Palavras*, baseado em autômatos adaptativos, é o núcleo do sistema. Esse tradutor recebe palavras como entrada e gera na saída seqüências de fonemas que devem ser utilizados

para sintetizar o som que representa a forma falada da entrada. Uma dessas seqüências deve ser escolhida para ser usada como tradução da palavra.

O núcleo é precedido por um módulo *Pré-Processador* que recebe o texto de entrada e trata esse texto para que ele possa ser utilizado como entrada para o módulo de tradução grafema-fonema. Esse módulo, transforma entradas impróprias como números, datas e siglas em seqüências de palavra que são aceitas pelo módulo *Tradutor de Palavras*.

Após a tradução do texto de entrada para uma seqüência de fonemas, o módulo *Sintetizador* é utilizado para transformar essa seqüência de fonemas em uma saída sonora.

1.4 Estrutura

A partir do capítulo inicial o trabalho se encontra dividido em três seções. A primeira seção, representada pelos capítulos 2 e 3, apresenta conceitos utilizados e a teoria formulada neste trabalho. A segunda seção, que contém os capítulos 4 e 5, apresenta os softwares desenvolvidos como parte do projeto. A terceira seção contém os capítulos 6 e 7, com os testes do método e conclusões.

O capítulo 2 contém uma breve introdução aos conceitos relacionados ao tema da dissertação, que são processamento de linguagens naturais enfatizando os etiquetadores morfológicos e os tradutores texto-fala, o estudo de autômatos adaptativos a partir da descrição funcional e de exemplos e a linguagem utilizada para representação fonética das palavras no decorrer do trabalho com enfoque para os símbolos utilizados no português falado no Brasil.

O capítulo 3 apresenta o resultado principal desse trabalho, que é o modelo de autômato utilizado para tradução grafema-fonema de palavras da língua portuguesa. O capítulo se inicia com a descrição funcional, que explica como cada uma das sub-máquinas e funções adaptativas trabalha. Segue a descrição formal desse autômato e por fim são encontrados exemplos da utilização desse autômato para a tradução de algumas palavras da língua portuguesa.

O capítulo 4 descreve o simulador de autômatos adaptativos desenvolvido como base para o tradutor grafema-fonema. As classes desenvolvidas como parte do software são descritas informalmente, mostrando a relação que mantém com as estruturas descritas pelo formalismo. A

seguir, o algoritmo utilizado para simulação é descrito em alto nível com uma breve discussão sobre seu funcionamento.

O capítulo 5 descreve o software utilizado para testar o modelo de autômato apresentado no capítulo 3. Assim como foi feito no capítulo 4 para o simulador de autômatos, neste capítulo são apresentadas características específicas do software tradutor texto-voz. Nesse capítulo também são discutidas características do autômato que determinam particularidades na implementação desse simulador.

O capitulo 6 descreve os resultados obtidos a partir de testes do método criado para tradução grafema-fonema e apresenta uma discussão sobre o conjunto de resultados obtido.

A conclusão do trabalho se encontra no capítulo 7. Nesse capítulo os resultados obtidos são analisados, são sugeridos caminhos de pesquisa relacionados ao mesmo tema e evoluções necessárias ao produto desenvolvido nesse trabalho.

2 Conceitos

Este capítulo contém uma breve introdução aos tópicos teóricos que fundamentam a pesquisa descrita nesta dissertação. A pesquisa se baseia principalmente em dois campos de estudo que são o Processamento de Linguagens Naturais e o estudo de dispositivos reconhecedores, enfocado em Autômatos Adaptativos. Trabalhos desses dois campos são mesclados para criar o produto final, que é um modelo para tradução grafema-fonema de palavras baseado em autômatos adaptativos. Este capítulo apresenta também o Alfabeto Fonético Internacional (HIPA), utilizado no decorrer do trabalho para representar as palavras foneticamente.

2.1 Processamento de Linguagens Naturais

Linguagens são conjuntos de cadeias definidos sobre um conjunto finito de símbolos (LEWIS; PAPADIMITRIOU, 2000). Linguagens naturais são aquelas utilizadas por seres humanos para comunicação de propósito geral, através da fala, escrita ou por sinais (táteis ou visuais).

As regras para formação de sentenças aceitas pelas gramáticas são flexíveis (MANNING; SCHÜTZE, 1999). Essa flexibilidade gera alguns problemas para sistemas de processamento baseados em regras principalmente quanto à existência de ambigüidades e não-determinismos (ROCHA, 2007), o que explica a razão para o sucesso de sistemas de processamento de linguagens naturais baseados em métodos estatísticos de inferência como os trabalhos de Kepler (2005) e Ratnaparkhi (1996). Algumas das dificuldades encontradas são as seguintes:

1. Os diferentes significados que uma palavra pode assumir conforme sua localização na sentença estudada.

2. As ambigüidades existentes nas sentenças geradas a partir das gramáticas dessas linguagens.

3. As regras podem ser diferentes conforme o local ou a época escolhidos para análise das sentenças da linguagem estudada.

4. A necessidade de entender expressões que aparentam não ter um significado segundo as regras da gramática (metáforas e gírias).

Apesar das dificuldades enfrentadas para gerar sistemas computacionais capazes de realizar processamento de linguagens naturais superando esses problemas, existem diversos trabalhos na área que se baseiam no processamento de voz e processamento de texto. Manning e Schütze (1999) apresentam técnicas para processamento de texto, enquanto processamento de voz pode ser encontrado em Tetschner (1993) A maior parte dos trabalhos encontrados no decorrer da pesquisa estuda o processamento de linguagens naturais nas duas formas citadas (voz e texto), mas há trabalhos como o de Pistori e José Neto (2004) que processa gestos.

As áreas de processamento de voz e texto são as mais importantes para este trabalho, pois se encontram diretamente relacionadas a ele. O produto final obtido nesta pesquisa é um software para tradução grafema-fonema, utilizado para criar um software de tradução texto-fala, uma sub-divisão da área de processamento de voz (TETSCHNER; 1993). A área de processamento de texto está relacionada ao trabalho, pois além de processar o texto de entrada para transformá-lo em uma seqüência de sons, o software utiliza a ajuda de um etiquetador morfológico durante o processo de tradução para melhorar seus resultados.

2.1.1 Tradução Texto-Fala

O objetivo dos tradutores texto-fala é sintetizar a fala de um ser humano a partir de um texto de entrada em linguagem natural. Os primeiros testes para síntese de fala utilizavam aparatos mecânicos até o advento de sistemas computacionais capazes de processar e sintetizar voz (SCHROEDER, 1993). Na forma eletrônica, a síntese de voz pode ser feita através dos métodos de síntese articulatória, da síntese fonética ou da síntese por concatenação (LEMMETTY; 1999), sendo este último o método utilizado no sintetizador associado a este trabalho.

Os métodos de síntese por concatenação utilizam um conjunto de sons derivados de fala, armazenados em uma base de dados, para sintetizar a fala de um ser humano. Os modelos que utilizam esse método costumam ser mais limitados que outros modelos, mas necessitam de menor capacidade de processamento em troca de maior capacidade de memória.

Uma das questões importantes relacionadas à síntese por concatenação é definir adequadamente o tamanho da unidade de concatenação utilizada. Segundo Koike et al. (2007), quanto maior a unidade de concatenação maior a inteligibilidade, porém o número de unidades de concatenação aumenta. Um sistema de concatenação de palavras utiliza menos processamento e tem melhor

inteligibilidade que um de concatenação de sílabas, mas necessita de uma base de dados significativamente maior.

Os métodos de síntese articulatória e fonética se baseiam no desenvolvimento de modelos. No primeiro caso o modelo reproduz os órgãos vocais humanos e no segundo ele gera sons a partir de unidades fundamentais. Esses modelos não são utilizados no processo de síntese de voz utilizado neste trabalho.

2.1.2 Etiquetador Morfológico

Etiquetadores são utilizados para gerar um conjunto de etiquetas associado a palavras de um texto. Neste trabalho são utilizados dois etiquetadores, um para dizer o tipo de grafema lido (se é palavra, número, sigla) para definir qual o padrão de leitura do grafema e um para classificar morfologicamente as palavras (se é artigo, verbo, substantivo) para diferenciar palavras homógrafas com significados diferentes.

Os etiquetadores que trabalham com as classes morfológicas das palavras são chamados etiquetadores morfológicos. Os etiquetadores morfológicos fazem parte do grupo de trabalhos relacionados ao processamento de texto na área de processamento de linguagens naturais. Um etiquetador morfológico é utilizado para encontrar parte da informação que se encontra no texto de forma implícita. A partir de um texto de entrada em uma linguagem natural ele gera uma seqüência de etiquetas que classifique as palavras do texto.

Tabela 1 – Etiquetagem de "A menina caminha sobre sua caminha"

	A	menina	caminha	sobre	sua	caminha	.
Classe	artigo definido	nome	verbo	preposição	pronome	nome	ponto
QTAG	ARTD	N	V	PRP	PRN	V	PT
Gamallo	DET	NOM	V	PRP	ADJ	NOM	SENT
VISL	ART	N	V	PRP	PRON	V	
VLMC	P	N	VB-P	P	PRO$-F	VB-P	.

Alguns exemplos de etiquetadores podem ser obtidos nos trabalhos de Kepler(2005), Menezes (2000), Brill (1995) e Ratnaparkhi (1996). A Tabela 1 apresenta um exemplo, em que a frase "A

menina caminha sobre sua caminha" é etiquetada utilizando os etiquetadores QTAG (ETIQUETADOR), Gamallo (GAMALLO) e VLMC tagger (KEPLER; 2005) e o parser VISL (VISL).

Os quatro etiquetadores testados utilizam conjuntos de etiquetas diferentes. A Tabela 2 apresenta o significado de algumas etiquetas e a equivalencias entre etiquetas dos três etiquetadores. Os conjuntos completos de etiquetas dos etiquetadores podem ser obtidos nos endereços eletrônicos de cada um deles. A partir da classificação correta das palavras, é possivel perceber que nenhum dos quatro etiquetadores obteve 100% de acerto na análise dessa sentença.

Tabela 2 – Etiquetas utilizadas, significado e equivalência.

Significado	QTAG	Gamallo	VISL	VLMC
Artigo Definido	ARTD	DET	ART	D
Artigo Indefinido	ARTI	DET	ART	D-UM
Substantivo	N	NOM	N	N
Verbo	V	V	V	VB
Preposição	PRP	PRP	PRP	P
Pronome	PRN	P	PRN	PRO
Adjetivo	ADJ	ADJ	ADJ	ADJ

2.2 Dispositivos Adaptativos

Dispositivos são considerados adaptativos se permitirem a alteração de sua estrutura em tempo de execução, admitindo assim a representação de estruturas mais complexas que a dos dispositivos que não dispõem dessa característica. Segundo a definição fornecida por José Neto (2001), um dispositivo é dito adaptativo se o seu comportamento se altera dinamicamente, como resposta direta ao estímulo de entrada, sem interferência de agentes externos.

Existem diversos tipos de dispositivos adaptativos, entre os quais estão os Autômatos Adaptativos de José Neto (1994), os State-Charts Adaptativos apresentados por Almeida Jr. (1995), Autômatos Finitos auto-modificáveis de Rubinstein e Shutt (1995), as Gramáticas

Adaptativas Recursivas descritas em Shutt (1993) e as Gramáticas Adaptativas para linguagens dependentes de contexto de Iwai (2000).

2.2.1 Autômatos Adaptativos

Os autômatos adaptativos apresentados por José Neto (1994), que servem como base para realizar o processo de tradução grafema-fonema proposto por este trabalho, podem ser entendidos como uma extensão adaptativa dos autômatos de pilha estruturados (JOSÉ NETO, 1987).

A camada adaptativa transforma os autômatos de pilha estruturados, que por serem equivalentes aos autômatos de pilha tradicionais (JOSÉ NETO, 2003) são capazes de reconhecer cadeias pertencentes a linguagens livres de contexto, em dispositivos com o poder computacional de Máquinas de Turing (PISTORI, 2003), capazes de reconhecer cadeias pertencentes a linguagens sensíveis ao contexto.

É importante ressaltar que a proposta inicial e grande parte dos trabalhos subseqüentes sobre autômatos adaptativos dão enfase ao seu uso como um acessório no estudo de compiladores. Por esta razão, os termos *funções adaptativas*, *chamadas de funções adaptativas* e *chamadas de sub-máquina* devem ser citados por completo para que não haja confusão com os termos *função* e *chamada de função*, comumente relacionados ao software.

Nesse trabalho não há necessidade de diferenciar essas entidades com tal rigor, pois não há possibilidade de confusão. Desta forma, no decorrer do trabalho, *função adaptativa* vai ser denominada *função*, as *chamadas de função adaptativa* serão denominadas *chamadas de função* ou *chamadas* e as *chamadas de sub-máquina* serão sempre denominadas dessa forma para diferenciá-las das *chamadas de função adaptativa*.

Além disso, toda vez que for necessário referenciar as *alterações geradas por uma ação adaptativa executada por meio de uma chamada de função adaptativa*, isso será referenciado como as *alterações realizadas por uma função* quando se referir às alterações realizadas por qualquer ação adaptativa executada por meio de chamadas dessa função ou *alterações realizadas por uma chamada* quando se referir a alterações realizadas pela ação específica referente a uma chamada de função.

2.2.1.1 Funcionamento dos Autômatos Adaptativos

O processo de reconhecimento consiste em alterar a configuração do autômato em função de suas transições até que ele chegue a uma configuração de aceitação ou até que nenhuma de suas transições possa ser executada. Uma configuração é dita de aceitação se a cadeia de entrada e a pilha se encontram vazias e o estado atual do autômato pertence ao conjunto de estados de aceitação.

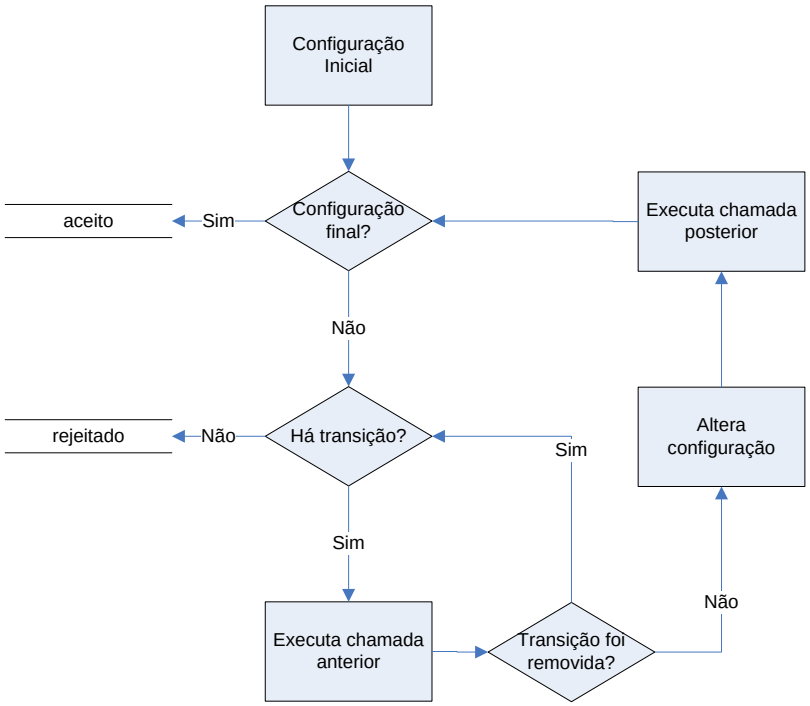

Figura 2 – Fluxograma de funcionamento dos autômatos adaptativos.

A partir de uma configuração podem ser utilizadas as transições do autômato que satisfaçam ao mesmo tempo três condições. (1) O estado de origem da transição deve ser igual ao estado atual do autômato, (2) a transição deve consumir o símbolo que se encontra no início da cadeia de entrada ou não deve consumir nenhum símbolo e (3) deve desempilhar a referência para o estado que se encontra no topo da pilha ou não deve empilhar nenhuma referência para estado. O autômato é dito determinístico se há garantia de que em cada passo há apenas uma transição que possa ser utilizada, caso contrário é dito não determinístico.

A alteração de configuração se dá a partir da alteração do estado atual, consumo do símbolo inicial da cadeia e desempilhamento da referência para o estado que se encontra no topo da pilha (se a transição escolhida prevê o consumo de símbolos e desempilhamento). O estado atual deve

ser alterado para o estado destino da transição quando não há desempilhamento ou para o estado cuja referência foi desempilhada caso contrário. Por fim um símbolo pode ser escrito na cadeia[1] de entrada e uma referência para um estado pode ser empilhada.

Duas ações adaptativas podem estar associadas a uma transição. Uma é executada anteriormente e outra posteriormente à alteração da configuração. Essas ações adaptativas alteram a topologia do autômato. Se a ação anterior remove a transição do conjunto de transições, as operações de alteração de configuração e ação adaptativa posterior são ignoradas e uma nova transição deve ser encontrada para continuar a execução.

2.2.1.2 Exemplos

Nesta seção são apresentados dois exemplos do uso de autômatos adaptativos para o reconhecimento de linguagens. Foi escolhida uma linguagem livre de contexto, $L = a^n b^n, n \geq 0$, e uma sensível ao contexto, $L = ww, w = (a \cup b)^*$. Para a linguagem livre de contexto apresenta-se também um autômato de pilha estruturado capaz de reconhecê-la.

1) $L = a^n b^n, n \geq 0$

Essa linguagem apresenta a principal característica das linguagens reconhecidas por autômatos de pilha, que é a possibilidade de trabalhar com estruturas aninhadas. Essa é a característica que diferencia linguagens livres de contexto (aquelas aceitas por autômatos de pilha) de linguagens regulares (que são aceitas também por autômatos finitos). Esse exemplo é capaz de mostrar como os autômatos adaptativos podem facilitar o reconhecimento das linguagens livres de contexto.

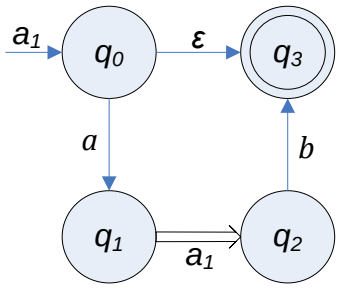

Figura 3 – Autômato de Pilha Estruturado ($a^n b^n$)

[1] A possibilidade de escrever um símbolo na cadeia de entrada apresentada na proposta inicial (JOSÉ NETO, 1994) não é encontrada em definições posteriores pois é possível provar que sua retirada não afeta o poder computacional dos Autômatos Adaptativos. Esse artifício é utilizado extensivamente durante esse trabalho como forma de gerar informação sobre o contexto de forma simples, logo, recomenda-se seu uso sempre que possível.

O autômato de pilha estruturado capaz de reconhecer as cadeias dessa linguagem é representado na Figura 3. As transições internas são representadas por setas simples acompanhadas do símbolo que deve ser lido para acionar a transição. A transição de chamada é representada por uma seta dupla, acompanhada do nome da sub-máquina que deve ser chamada. A seta aponta para o estado de retorno cuja referência deve ser empilhada. As transições de retorno não aparecem representadas por setas. Essas transições são chamadas a partir do estado de saída, que é simbolizado por um círculo com perímetro duplicado.

Esse autômato de pilha estruturado lê símbolos *a* e empilha uma referência para o estado q_2, voltando para o estado q_0. Isso significa que ele faz uma chamada da sub-máquina a_1, que é a única desse autômato, toda vez que lê um símbolo *a* na cadeia de entrada. Após a leitura dos símbolos *a* o autômato passa para o estado q_3, que passa a desempilhar as referências para estados do topo da pilha, retornando para as chamadas anteriores ao fim da execução da sub-máquina. Esse processo se repete até que todos os símbolos *b* tenham sido lidos.

Se a cadeia possuir o mesmo número de símbolos *a* e *b*, o autômato chega ao final da execução tendo lido toda a cadeia e com a pilha vazia. Se a cadeia possuir mais símbolos *a* do que *b*, a pilha não termina vazia. Se o número de símbolos *b* dessa cadeia for maior que o número de símbolos *a*, a cadeia de entrada não é lida por inteiro e o autômato pára no estado q_3. Se houver símbolos *a* e *b* intercalados o autômato não termina a execução, pois chegará a uma configuração em que nenhuma transição pode ser utilizada. Desta forma o processo é terminado indicando que a cadeia de entrada não foi aceita.

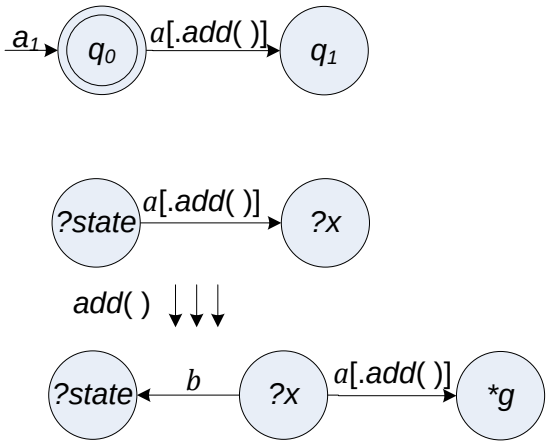

Figura 4 – Autômato Adaptativo ($a^n b^n$)

A Figura 4 apresenta um exemplo de autômato adaptativo utilizado para reconhecer linguagens da cadeia $L = a^n b^n, n \geq 0$. A notação utilizada para a representação da função adaptativa nessa figura é semelhante à definida por Pistori e José Neto (2003). Nessa notação, há a configuração de uma parte do autômato antes da chamada da função e depois da chamada e a alteração é indicada por três setas paralelas acompanhadas do padrão de chamada da função. Neste trabalho, variáveis aparecem prefixadas por ?, geradores por * e parâmetros da função adaptativa por #.

O funcionamento do autômato consiste em criar uma transição de volta (que é chamada a partir da leitura do símbolo *b*), apagar a última transição usada, criar um novo estado (*$*g_i$* em que *i* é um contador dos estados criados) e uma transição que faça o processo equivalente direcionada para esse novo estado. Esse processo se repete toda vez que um símbolo *a* é lido. Existe apenas um caminho de volta que pode ser percorrido apenas pela leitura de um número de símbolos *b* igual ao número de símbolos *a* lidos, até chegar ao estado inicial, que é o único estado final desse autômato.

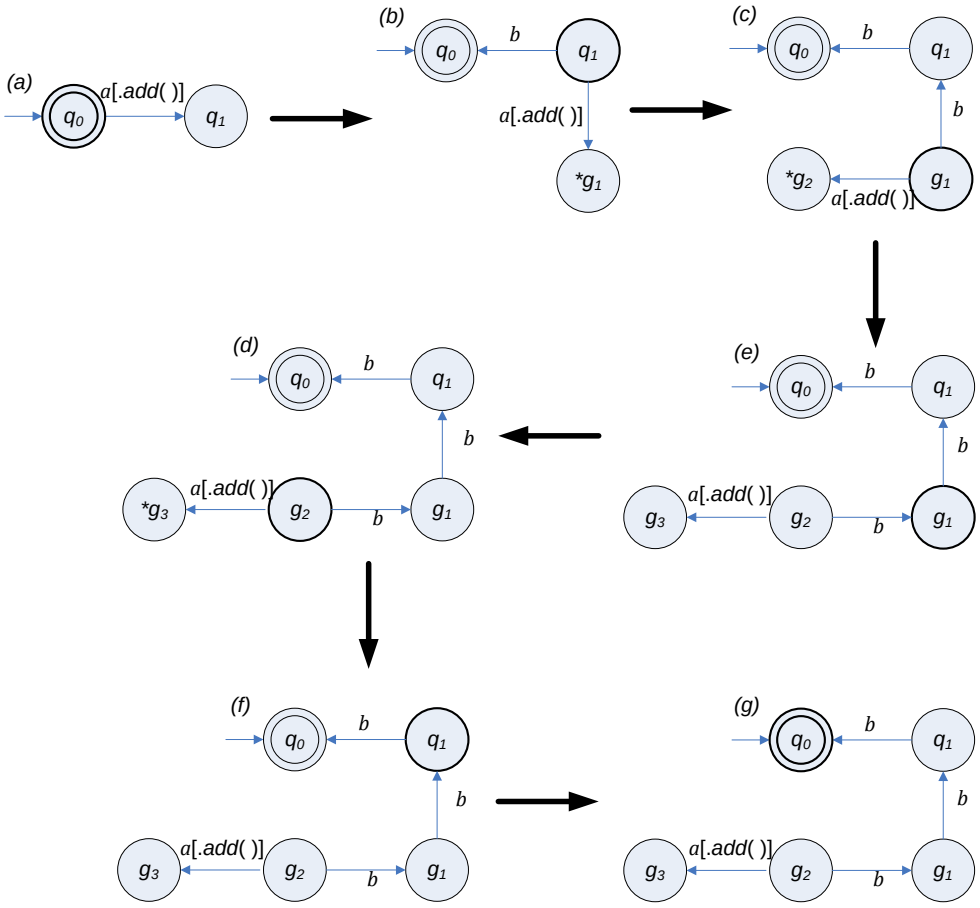

Figura 5 – Autômato Adaptativo – reconhecimento de *aaabbb*

A Figura 5 ilustra o processo de execução do autômato adaptativo descrito, para reconhecer a cadeia de entrada $aaabbb$. Nessa figura o estado corrente é marcado com uma linha mais espessa que a dos outros estados. O autômato definido em *(a)* representa a configuração inicial do autômato. *(b)*, *(c)* e *(d)* representam os passos em que ocorrem leituras de símbolos a. Em cada um desses passos ocorre uma chamada da função adaptativa *add*, que insere um novo estado, apaga a transição utilizada e cria duas novas transições no lugar. *(e)*, *(f)* e *(g)* representam passos em que ocorrem leituras do símbolo b. Dessa forma o autômato volta ao estado inicial (que é um estado de aceitação) e aceita a cadeia de entrada.

2) $L = ww, w = (a \cup b)^*$

O processo de reconhecimento das cadeias dessa linguagem implica em determinar se a primeira metade é igual à segunda. Isso pode ser feito dividindo a cadeia em duas partes iguais (se possível) e verificando a igualdade entre as duas metades (deterministico) ou analisando uma parte e, pressupondo que chegou ao meio da cadeia, comparar o resto com a parte analisada (não determinístico).

A capacidade de contar o número de símbolos (necessária para definir o meio da cadeia) e a capacidade de armazenar a primeira metade (para comparar com a segunda) são características de linguagens sensíveis ao contexto, portanto não há autômatos de pilha ou dispositivos equivalentes capazes de reconhecer todas as cadeias dessa linguagem.

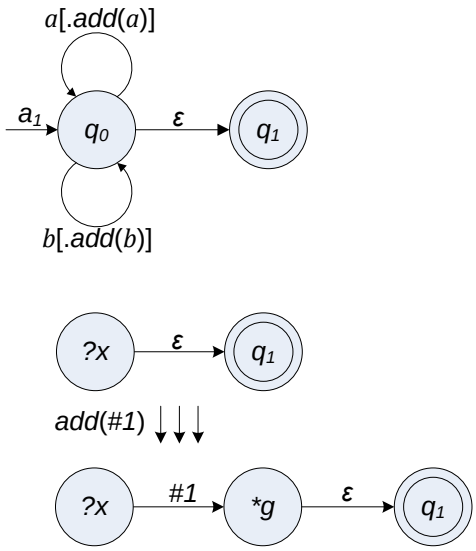

Figura 6 – Autômato Adaptativo - reconhecimento de L = ww, w = $(a \cup b)^*$

A Figura 6 apresenta um autômato adaptativo não-determinístico capaz de reconhecer as cadeias dessa linguagem, baseado na idéia de reconhecer uma parte da cadeia e, supondo que chegou ao meio, verificar se a outra parte é igual. São duas transições, que permitem o processamento da linguagem $(a \cup b)^*$ e uma transição em cadeia vazia que permite que o autômato chegue ao estado final e utilizada como referência para ação adaptativa *add* no momento de criar novas transições.

As transições acionadas por leitura dos símbolos *a* e *b* chamam funções adaptativas que alteram o autômato, criando um novo estado antes do estado q_1, que só pode ser alcançado pela leitura de um símbolo igual ao último lido. Dessa forma, o caminho até o estado q_1 que é o único estado final do autômato só pode ser percorrido se a cadeia lida enquanto o autômato estava em q_0 estiver repetida na cadeia de entrada.

A Figura 7 ilustra o funcionamento do autômato adaptativo descrito acima durante o reconhecimento da cadeia *abaaba*. O caminho que leva ao reconhecimento da cadeia vai da configuração *(a)* até a configuração *(k)*. Nessa execução, as três primeiras transições executadas consomem os três primeiros símbolos e alteram a estrutura do autômato por meio de chamadas da função *add*. Os três outros símbolos são consumidos pelas transições criadas pela função adaptativa *add* levando o autômato ao estado final com a cadeia vazia, situação em que ocorre o reconhecimento da cadeia de entrada.

As situações de não determinismo ocorrem na configuração *(a)* em que o autômato pode utilizar a transição vazia chegando à configuração *(b)* e nas configurações *(d)* e *(f)* em que o autômato pode utilizar a primeira transição para consumir o símbolo *a* chegando às configurações *(e)* e *(g)* respectivamente. Em *(b)* não ocorre aceitação, pois a cadeia de entrada não se encontra vazia e não há possibilidade de executar novas transições. Em *(e)* não há possibilidades de executar novas transições e o estado não é final. Em *(g)* a execução continua, mas nenhum caminho leva à configuração final, pois a leitura já passou da metade da cadeia e não há como chegar ao estado q_1 onde ocorre o reconhecimento.

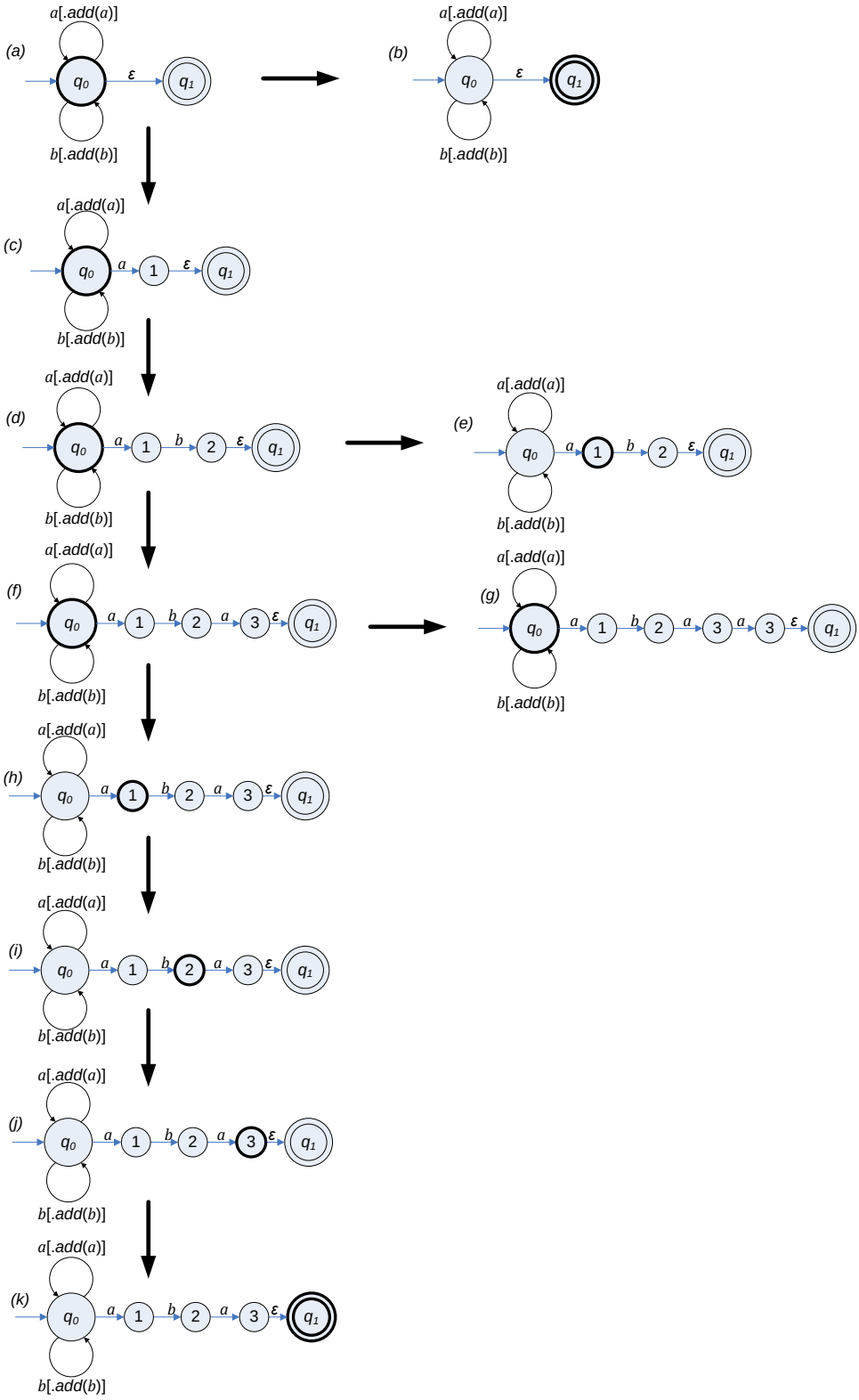

Figura 7 – Execução do autômato para a cadeia *abaaba*

2.3 Alfabeto Fonético Internacional

O Alfabeto Fonético Internacional apresenta uma notação padronizada para a representação de fonemas. Esse alfabeto, desenvolvido pela Associação Internacional de Fonética (IPA), representa características da linguagem como fonemas e entonação. O princípio desse alfabeto é gerar uma representação simbólica unívoca para os sons gerados na fala.

O conjunto de símbolos desse alfabeto é extremamente extenso, apresentando sons de diversos tipos, muitos dos quais não são utilizados na língua portuguesa. Esta seção apresenta uma descrição sucinta do conjunto de símbolos desse alfabeto focando os símbolos utilizados para representar sons da língua portuguesa que foram utilizados nesse trabalho.

O trabalho de Barbosa e Albano (2004) apresenta um vasto estudo sobre a fonética da língua portuguesa, com enfoque na forma contemporânea da região de São Paulo. Por ser justamente essa variação da língua o foco deste trabalho, utilizar-se-á essa fonte como base para a saída fonética gerada pelo tradutor grafema-fonema e também para a descrição apresentada a seguir.

2.3.1 Vogais

Vogais são os fonemas caracterizados pela passagem livre de ar pela boca (HIPA). Segundo Cipro Neto e Infante (1997), na língua portuguesa as vogais representam o núcleo das sílabas, ou seja, não há sílabas sem vogais, e são caracterizadas quanto à zona de articulação, elevação da zona mais alta da língua e timbre. Além disso, podem ser classificadas como orais ou nasais. Todas as vogais da língua portuguesa são sonoras.

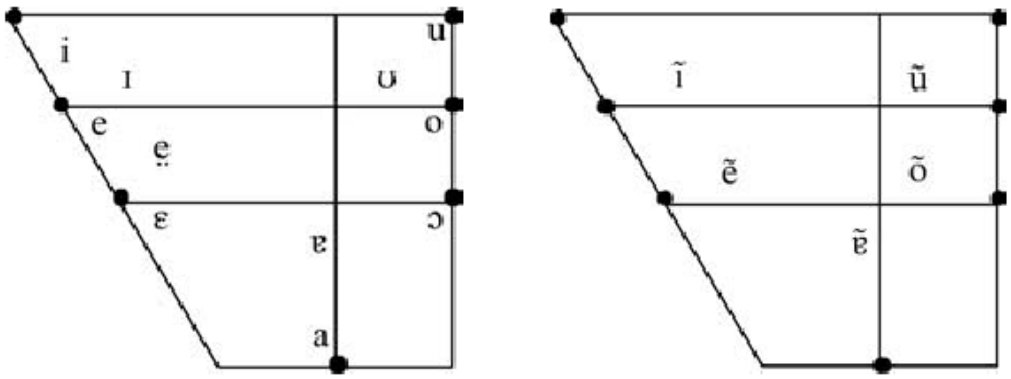

Figura 8 – Vogais da Língua Portuguesa (BARBOSA; ALBANO, 2004)

A Figura 8 apresenta o inventário de vogais utilizadas na língua portuguesa de São Paulo. A tabela à esquerda apresenta as vogais orais enquanto a da direita apresenta variações nasais dessas vogais, que se caracterizam pela expulsão de ar pelo nariz junto do som (CIPRO NETO; INFANTE, 1997). O som [ɐ̟] foi o único não utilizado no processo de tradução.

2.3.2 Consoantes

Consoantes são sons emitidos na fala, caracterizados pelo fechamento completo ou parcial da cavidade bucal (HIPA). Cipro Neto e Infante (1997) caracteriza as vogais da língua portuguesa como pulmônicas (em que o ar é expirado dos pulmões), e as diferencia por modo de articulação (indica o tipo de obstáculo encontrado pelo ar na passagem pela boca), o ponto de articulação (indica o local onde o obstáculo se encontra) e a fonação (indica se há ou não vibração das cordas vocais).

A Figura 9 apresenta o conjunto de consoantes utilizado na variação da língua portuguesa falada em São Paulo. Apesar da discussão sobre as variações dos sons róticos (WHITLEY, 2003) e das variações que podem ser dadas para o som da letra *r* na língua portuguesa, foi adotado o [ɣ] em detrimento de [h] ou outras variações aceitas (SILVA; ALBANO, 1999), por questão de coerência com o trabalho citado. Pela mesma razão, os sons de *n* e *m* em final de consoante são representados por [ⁿ] e [ᵐ].

	Bilabial	Labiodental	Alveolar	Postalveolar	Palatal	Velar
Plosive	p b		t d			k g
Affricates			(t͡ʃ) (d͡ʒ)			
Nasal	m		n		ɲ	
Tap			ɾ			
Fricative		f v	s z	ʃ ʒ		ɣ
Lateral approximant			l		ʎ	

Figura 9 – Consoantes da Língua Portuguesa (baseado em BARBOSA; ALBANO, 2004)

2.3.3 Semivogais

No português as semivogais se diferenciam das vogais pelo fato de que não são núcleo das sílabas em que se encontram, logo, as semivogais necessariamente acompanham uma vogal (CIPRO NETO; INFANTE, 1997). Na língua portuguesa são encontradas duas semi-vogais, neste trabalho representadas por [ɪ̯] e [ʊ̯]. Novamente, a representação foi escolhida por questão de coerência com o trabalho de Barbosa e Albano (2004) em detrimento de [j] e [w] que são utilizadas por Weingessel (1994) e Cipro Neto e Infante (1997) para representar os mesmos sons. Nessa representação, o símbolo abaixo dos caracteres que representariam vogais indicam que, na verdade, eles são não-vocálicos.

Normalmente, na língua portuguesa, as semivogais são representadas pelos mesmos símbolos que as vogais ([ɪ̯] *e* e *i* e [ʊ̯] por *o* e *u*), mas em diversas variações da língua portuguesa falada no Brasil, inclusive na de São Paulo, podemos encontrar o som de *l* em final de sílaba pronunciado como [ʊ̯].

Tabela 3 – Vogais e Semi-Vogais da Língua Portuguesa

Fonema	Exemplos	Fonema	Exemplos
[a]	c**a**so,	[ɛ]	b**e**la
[e]	b**e**bo,	[ɔ]	m**o**da
[i]	m**i**nha	[ɐ̃]	c**a**ma, c**â**mara, m**ã**e
[o]	b**o**lo	[ẽ]	s**e**nte, f**ê**mur
[u]	b**u**le	[ĩ]	c**i**nto
[ɐ]	mes**a**	[õ]	c**o**mpra
[ɪ]	sed**e**	[ũ]	c**u**mpre
[ʊ]	carr**o**	[ɪ̯]	pa**i**, sáb**i**a, homogên**e**a
[ẹ]	Núm**e**ro	[ʊ̯]	pa**u**ta, táb**u**a, mág**o**a, a**o**, fa**l**ta, freq**ü**ência

2.3.4 Exemplos

A seguir são apresentados alguns exemplos da utilização do Alfabeto Fonético Internacional. Na Tabela 3 são apresentadas as vogais e semi-vogais da língua portuguesa e exemplos de letras que podem ser utilizadas para representar esses fonemas. Na primeira coluna se encontram os fonemas, representados pelo seu símbolo no Alfabeto Fonético Internacional e na segunda coluna

algumas palavras em que há ocorrência do fonema, com a letra que o representa destacada. A Tabela 4 apresenta as consoantes da língua portuguesa, utilizando as mesmas regras utilizadas para a construção da Tabela 3.

Tabela 4 – Consoantes da Língua Portuguesa

Fonema	Exemplos	Fonema	Exemplos
[p]	**p**ato	[b]	**b**ala
[t]	**t**odo	[d]	**D**ata
[f]	**F**aca	[v]	**v**aca
[k]	**qu**ero, fre**q**üência, **c**aldo	[g]	**g**ado, **gu**eto
[ʃ]	**ch**ato, lu**x**o	[ʒ]	**J**aca, **g**ente
[s]	**s**oda, **c**ela, a**ç**ão, má**x**imo, cas**s**a, e**xc**eção, pi**sc**ina, de**sç**a	[z]	**z**ero, e**x**ato, ca**s**a
[ʧ]	**t**ia, **tch**eco	[ʤ]	**d**ia
[l]	**l**ata	[ʎ]	ca**lh**a, famí**li**a
[ɣ]	**r**ato, ca**rr**o	[ɾ]	ca**r**o
[n]	**n**ata	[ɲ]	pamo**nh**a
[m]	**m**ato, a**m**sterdã	[ᵐ]	ca**m**po
[ⁿ]	ca**n**to	[ks]	flu**x**o

3 Tradução Grafema-Fonema

Este capítulo apresenta o autômato adaptativo criado para realizar a tradução grafema-fonema de palavras da língua portuguesa. Esse autômato é associado a métodos auxiliares para realizar a tradução grafema-fonema de textos escritos na língua portuguesa. A solução apresentada neste documento não é a única possível para resolver problemas da tradução texto-voz na língua portuguesa. Os trabalhos de Barbosa et al. (1999) e Alfenas et al. (2004) são algumas soluções para a mesma tarefa. A solução apresentada por Alfenas et al. (2004) foi a base dos estudos iniciais para a criação deste autômato, que gerou um modelo de solução apresentado em Shibata e Rocha (2007) e culminou no resultado apresentado neste trabalho por isso alguns aspectos do autômato apresentado são semelhantes aos das soluções propostas nesses trabalhos.

Este capítulo é dividido em quatro partes: a primeira parte apresenta a análise inicial do problema de síntese de voz na língua portuguesa com uma solução baseada em autômatos adaptativos. A segunda parte descreve o princípio de funcionamento do autômato resumidamente. A terceira parte contém a descrição do autômato adaptativo criado, com as estruturas iniciais de suas sub-máquinas e das funções adaptativas utilizadas. Por fim, a quarta parte apresenta exemplos da aplicação do autômato a algumas palavras da língua portuguesa.

3.1 Análise do Problema

Antes de iniciar a criação de um tradutor texto-voz baseado em autômatos adaptativos foi necessário definir o método de síntese utilizado pelo tradutor e qual seria exatamente a função dos autômatos adaptativos no processo de tradução. A princípio, foi definido que o tradutor seria baseado na síntese por concatenação e utilizaria sílabas como unidades mínimas de concatenação. O autômato seria responsável pela tradução grafema-fonema das palavras da língua portuguesa para seqüências de cadeias utilizando os símbolos do Alfabeto Fonético Internacional. A separação do texto em palavras, o processo de síntese e questões adicionais como prosódia seriam delegados a outros módulos do tradutor.

Como o processo de síntese se baseia na concatenação de sílabas, o modelo inicial para a solução se baseava na separação das palavras em sílabas. No entanto, essa abordagem encontra como empecilho a necessidade de saber *a priori* se encontros vocálicos formam ou não ditongos ou tritongos. Essa informação depende do contexto em que os encontros vocálicos estão inseridos, o

que torna o processo de separação silábica dependente de contexto. Para superar esse problema as palavras são separadas em Unidades de Análise Fonética (UAF) que podem conter mais de uma vogal e o autômato define, baseando-se no contexto, como as UAF são transformadas em seqüências fonéticas (gerando uma ou duas sílabas).

Como mostra a Figura 10, o processo de tradução grafema-fonema foi dividido em duas tarefas: a divisão da palavra em seqüências de UAF (realizada pelo *Separador*) e a definição da representação fonética das UAF levando em consideração o contexto (realizada pelo Autômato Adaptativo). A definição da representação fonética é dividida em duas fases, a primeira em que as UAF são reconhecidas (*Reconhecedor*) e são criadas regras para análise do contexto em que essa UAF se encontra (chamadas de Função Adaptativa alteram o *Transdutor*) e a segunda em que a análise é realizada e são encontradas as representações fonéticas possíveis para uma palavra (*Transdutor*).

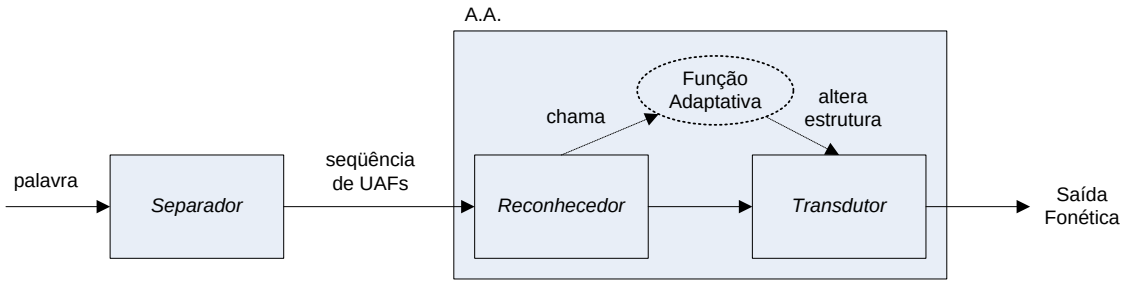

Figura 10 – Esquema de funcionamento do módulo de tradução grafema-fonema.

Inicialmente, a tarefa de separação das palavras era realizada pelo autômato, assim como no trabalho de Alfenas et al (2004). Como a simplificação gerada pela divisão de palavras em UAF (no lugar de sílabas) não há necessidade de utilizar adaptatividade para separar as palavras, portanto é possível criar um bloco responsável por essa tarefa sem relação com o autômato.

3.2 Descrição Funcional

A cada UAF reconhecida, as ações adaptativas executadas criam no *Transdutor* um bloco de transições formando uma estrutura fechada, denominada *elo*. As transições que formam os *elos* garantem a continuidade da execução do *Transdutor*, quando esse é chamado pelo *Reconhecedor*,

por meio de leitura e escrita de símbolos na cadeia de entrada. A Figura 11 apresenta um esquema simplificado do elo e a interação com os elos adjacentes.

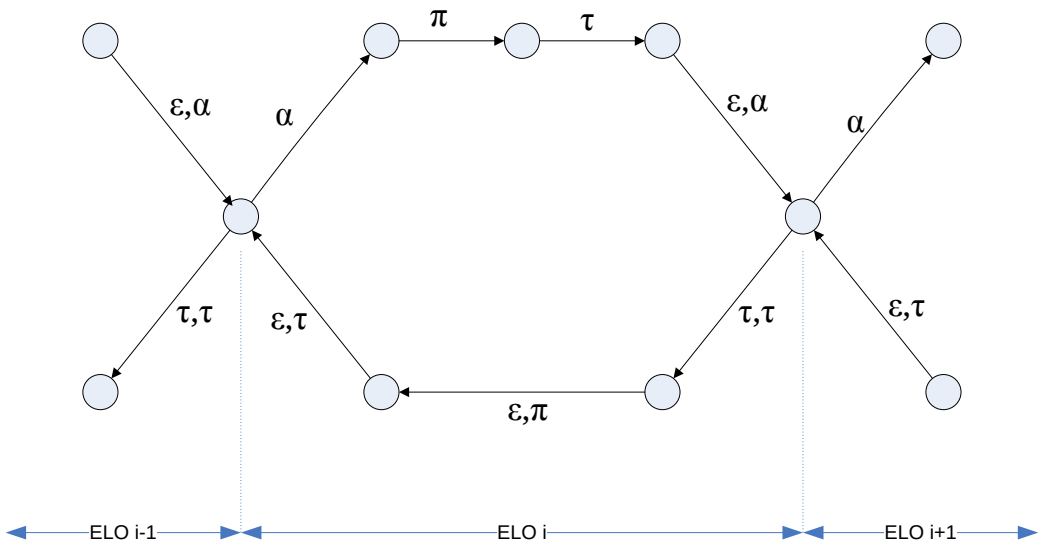

Figura 11 – Um elo e sua interação com os elos vizinhos.

3.2.1 Execução do Elo

Durante a execução do *Transdutor*, ocorrem duas passagens por cada um dos elos. Na primeira passagem ocorre o reconhecimento da tonicidade da UAF, são definidas as influências que as UAF recebem da UAF posterior e a tonicidade da UAF anterior. Na segunda passagem, ocorre o reconhecimento das influências anterior e posterior, a geração do som e a definição da influência sobre a UAF posterior.

Na primeira passagem, o elo recebe a cadeia de entrada com um símbolo τ no início, que é utilizado como parâmetro para reconhecer a regra de tonicidade que deve usar. Ele consome o símbolo inicial e insere três novos símbolos na pilha: um símbolo τ e um símbolo π para uso próprio na segunda passagem e um símbolo τ que a UAF anterior[2] utiliza como parâmetro para desencadear processo semelhante.

[2] Na primeira passagem a execução vai da última UAF em direção à primeira e na segunda percorre o caminho inverso. Essa característica é necessária para definir as regras de tonicidade e escrever a saída fonética na ordem correta.

Na segunda passagem, o elo recebe a cadeia com um símbolo α no início, que é utilizado como parâmetro para desencadear o processo de definição do som. Os dois símbolos seguintes a esse símbolo são os símbolos (π e τ) que foram escritos na primeira passagem. O elo consome esses dois símbolos (na transição em que consome o símbolo τ escreve a seqüência fonética na saída) e escreve um símbolo α para que a UAF seguinte possa repetir o processo.

Além das transições dos elos existem mais três transições nessa sub-máquina que são fundamentais ao funcionamento do processo. A transição que liga o estado I ao *estado inicial*[3] do último elo é executada sem consumo de símbolos e desencadeia o processo escrevendo o símbolo τ apropriado[4], a transição cíclica no estado AUX que consome o último símbolo τ e escreve um símbolo α invertendo o sentido da execução e a transição que liga o *estado inicial* do último elo ao estado final consumindo o último símbolo α.

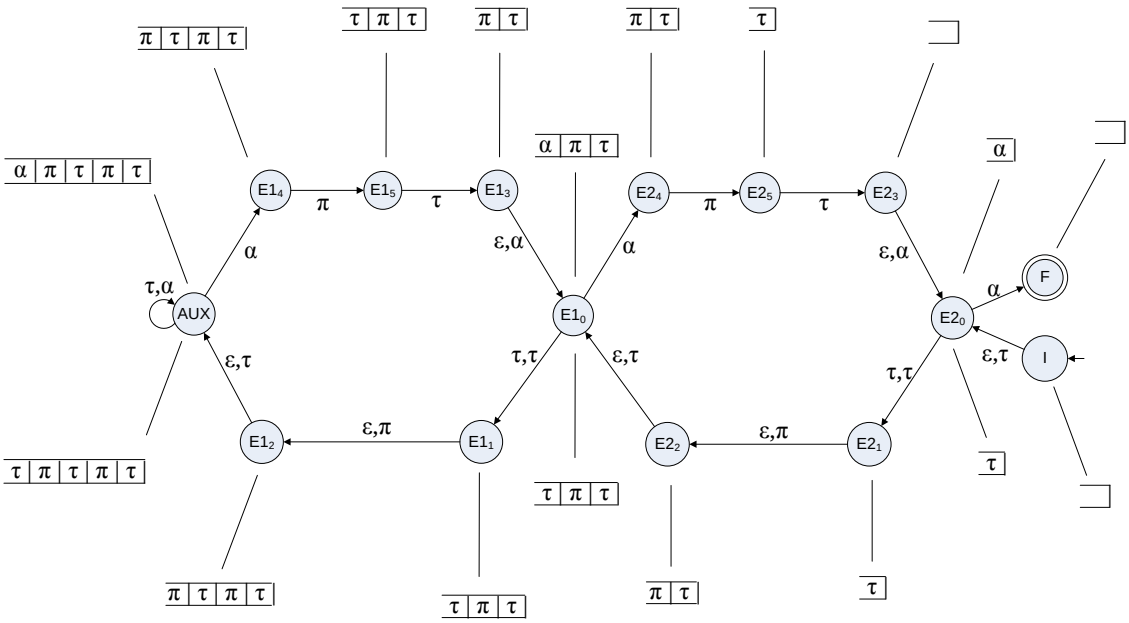

Figura 12 – Evolução da cadeia de entrada durante a execução do Transdutor

[3] *estado inicial* é o estado de entrada da primeira passagem e saída da segunda passagem pelo elo. Possui esse nome por ser o primeiro a ser criado na seqüência de chamadas de função adaptativas. O *estado inicial* do elo anterior é a saída da primeira passagem e entrada da segunda passagem. As transições de marcação do elo ligam *estados iniciais* consecutivos.

[4] Em palavras não acentuadas o valor apropriado é τ_F indicando que deve ser usada a regra padrão de tonicidade para aquela UAF quando ela é final. Para palavras acentuadas esse valor é τ_A indicando que essa UAF é obrigatóriamente átona, independente de sua característica.

A Figura 12 apresenta um esquema de funcionamento do processo de execução do *Transdutor* junto da evolução da cadeia de símbolos durante esse processo. Por essa figura, pode-se perceber que um elo consome apenas os símbolos que ele mesmo gera e os símbolos iniciais da cadeia vindos do vizinho. Dessa forma garante-se que um elo devolve ao seu vizinho posterior a mesma configuração de cadeia de entrada, exceto pelo primeiro símbolo que contém a informação trocada entre eles.

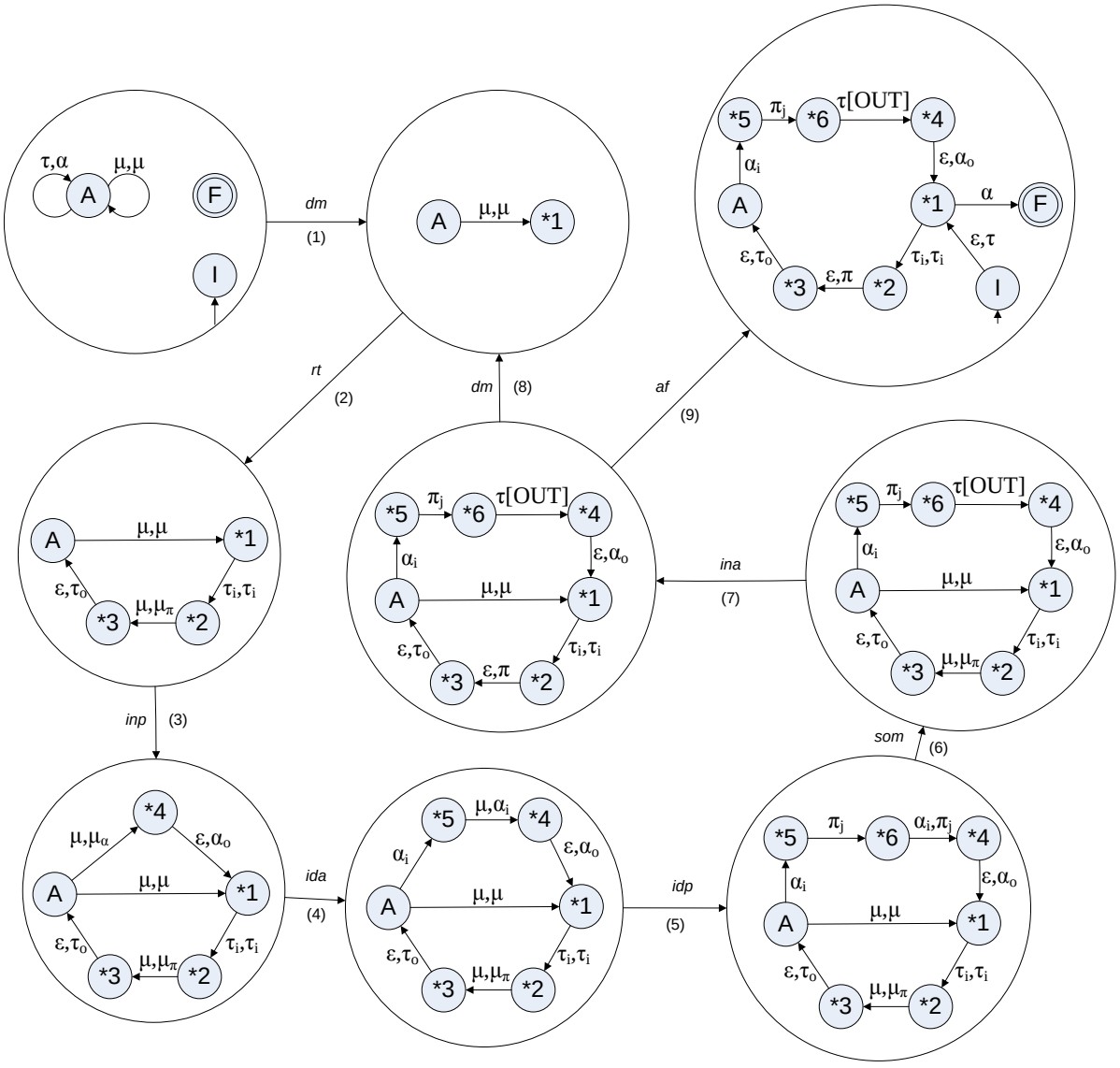

Figura 13 – Diagrama da seqüência de chamadas de função adaptativa.

3.2.2 Criação do Elo

A criação de um elo depende das ações adaptativas executadas quando ocorre o reconhecimento de uma UAF. A Figura 13 apresenta um esquema simplificado do processo de criação de um elo no *Transdutor* por meio da execução da ação adaptativa associada à transição de reconhecimento de uma UAF. Nessa figura, o caminho (1) a (6) representa a criação do primeiro elo, o ciclo (7) a (6) representa a criação dos elos subseqüentes e (9) representa os ajustes finais que preparam o *Transdutor* para execução.

As ações adaptativas associadas ao reconhecimento de UAF são compostas por seqüências de ações adaptativas menores, realizadas por meio de chamadas de funções adaptativas. Na Figura 13 as alterações realizadas em cada uma dessas chamadas de funções adaptativas são representadas de forma simplificada por meio das setas.

As repetições de chamadas de uma função que ocorrem na criação de um elo e permitem a criação de diversos caminhos, cada um analisando uma possibilidade contextual, são omitidas na figura. Essas repetições podem ocorrer para os passos (2), (4), (5) e (6).

É importante entender de forma genérica como se processa a criação de um elo no *Transdutor*. Detalhes das funções adaptativas utilizadas, seu uso para criação de uma ação adaptativa e o processo de criação de um elo serão apresentados com mais detalhes no decorrer das próximas seções.

3.3 Descrição Detalhada

Nesta seção, os dispositivos apresentados na Figura 10 são descritos de forma detalhada. A seção encontra-se dividida em duas partes: a primeira apresenta as regras para criação das UAF utilizadas pelo *Separador de Palavras*, enquanto a segunda parte apresenta o autômato adaptativo composto por *Reconhecedor* e *Transdutor*.

3.3.1 Separador de Palavras

O *Separador de Palavras*, a seguir denominado *Separador*, é o primeiro dos dispositivos a ser executado. Esse dispositivo recebe como entrada uma palavra e gera como saída uma seqüência de UAF. O *Separador* pode ser implementado utilizando diferentes técnicas, desde que o

Reconhecedor seja capaz de manipular a seqüência de UAF gerada pelo *Separador*. Por essa razão, é mais urgente definir um padrão para a seqüência de UAF gerada do que definir a técnica utilizada para criar essas UAF.

A divisão atômica das palavras utilizada nesse trabalho foi baseada nas regras de divisão silábica definida para palavras da língua portuguesa apresentadas em Cipro Neto e Infante (1997). A divisão atômica segue as seguintes regras:

1. Os dígrafos[5] *ch*, *lh*, *nh*, *gu*, *qu* pertencem a uma única UAF;
2. As letras que formam os dígrafos *rr*, *ss*, *sc*, *sç*, *xs* e *xc* devem ser separadas;
3. Encontros consonantais[6] que ocorrem em UAF internos (de uma palavra) devem ser separados, exceto aqueles em que a segunda letra é *l* ou *r*;
4. Grupos consonantais que iniciam palavras não são separáveis.
5. Encontros de símbolos que podem representar vogais e semivogais devem ser mantidos na mesma UAF se: há duas consoantes e elas são diferentes, ou se possuírem três consoantes terminadas em *iu*. Em outros casos, quando há três ou mais símbolos, a separação deve dar preferência às combinações terminadas em *i* e *u* que formam ditongos crescentes.

O *Separador* insere automaticamente um símbolo que indica o final da palavra no final da cadeia, que funciona como marcador de final da seqüência de entrada. A transição do *Reconhecedor* que lê esse símbolo executa uma ação adaptativa que prepara o *Transdutor* para execução e leva o *Reconhecedor* ao estado em que ocorre a chamada do *Transdutor*.

3.3.2 Autômato Adaptativo

O autômato adaptativo gera um conjunto de representações fonéticas para representar a cadeia de entrada proveniente do *Separador*. Nesta seção são descritos o conjunto de símbolos utilizados

[5] *Dígrafo: letras que representam um fonema. Podem ser consonantais ou vocálicos.*
[6] *Encontro Consonantal: agrupamento de duas ou mais consoantes sem vogal intermediária*

pelo autômato adaptativo, as sub-máquinas que compõem o autômato, as funções adaptativas utilizadas e as regras para construir as ações adaptativas que alteram a topologia do autômato.

3.3.2.1 Símbolos

Os símbolos utilizados pelo autômato são diferenciados em seu uso. Os símbolos encontram-se agrupados em três conjuntos, levando em consideração o que representam no tratamento da palavra de entrada e como são tratados pelo autômato.

a) Unidades de Análise Fonética – UAF (símbolos do Reconhecedor)

Esses símbolos representam as UAF reconhecidas pelo *Separador*. Eles são utilizados exclusivamente pelo *Reconhecedor*, são consumidos pelas transições do *Reconhecedor* e não são reescritos na cadeia de entrada. No decorrer do trabalho, esses símbolos são representados pela letra grega σ acompanhada de um índice que indique a UAF representada (ex: a palavra *casa* é separada em duas UAF, a primeira composta pelas letras *ca* e a segunda pelas letras *sa*, que são representadas como σ_{ca} e σ_{sa} respectivamente).

b) Símbolos Contextuais (do *Transdutor*)

Esses símbolos são utilizados pelo *Transdutor* para definir o som associado a cada UAF em função dos outras UAF que compõem a palavra. O *Transdutor* escreve e consome todos os símbolos desse tipo na cadeia de entrada durante sua execução. Os símbolos utilizados pelo *Transdutor* foram divididos em três classes:

Classe τ: os símbolos dessa classe são utilizados pelo autômato para definir tonicidade do som de saída e definir a necessidade de separar em dois sons uma UAF com encontros vocálicos. Existem quatro símbolos para a representação de tonicidade, que são:

- τ_T: indica que uma UAF contém o som tônico.
- τ_A: indica que uma UAF contém apenas sons átonos.
- τ_F: indica que uma UAF é a última de uma palavra não acentuada.
- τ_*: coringa – permite a leitura de qualquer símbolo dessa classe.

Classe α: os símbolos dessa classe são utilizados para representar a influência que uma UAF recebe da UAF anterior. Essa influência é determinada pela letra final da UAF anterior e pode gerar como resultado uma alteração no som da UAF seguinte. Os símbolos da classe α são:

- α_V: indica que a UAF anterior é terminada em vogal.
- α_*: coringa – permite a leitura de qualquer símbolo dessa classe.

Classe π: os símbolos dessa classe são utilizados para representar a influência que uma UAF recebe da UAF posterior. Essa influência é determinada pela letra inicial da UAF posterior e pode gerar como resultado uma alteração no som da UAF anterior. Os símbolos da classe π são:

- π_F: indica que a UAF é a última da palavra.
- π_R: indica que a UAF precede uma UAF iniciada em *r*.
- π_S: indica que a UAF precede uma UAF iniciada em som fricativo alveolares ou pós-alveolares.
- π_V: indica que a UAF precede uma UAF iniciada em consoante sonora.
- π_U: indica que a UAF precede uma UAF iniciada em consoante surda.
- π_N: indica que a UAF precede uma UAF iniciado em consoante nasal.
- π_*: coringa – permite a leitura de qualquer símbolo dessa classe.

c) Símbolos de Marcação

Esses símbolos não são escritos nem consumidos da cadeia de entrada em nenhum momento. Eles são utilizados pelas chamadas de função adaptativa como marcas para encontrar transições, denominadas *transições de marcação*. Essas transições indicam estados de referência para a inserção de transições de escrita e consumo dos símbolos contextuais no *Tradutor* e são apagadas quando não são necessárias. Os símbolos de marcação são representados pelo símbolo μ acompanhados de um índice que indica o tipo de marcação.

3.3.2.2 Reconhecedor

O *Reconhecedor* lê as UAF geradas pelo *Separador* e chama funções adaptativas que alteram o *Transdutor* de forma a contemplar as características da UAF. Quando chega ao final da seqüência de UAF chama o *Transdutor*.

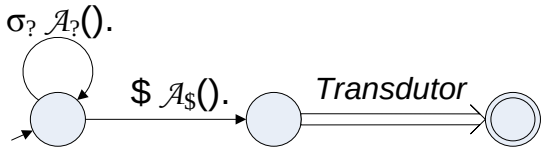

Figura 14 – Esquema simplificado do *Reconhecedor*

A Figura 14 apresenta um esquema simplificado do *Reconhecedor*. Nessa figura, as transições que consomem as UAF são representadas por uma única transição que consume o símbolo σ?. A utilização dessas transições dispara a execução de ações adaptativas (representada por 𝒜?()), que alteram o *Transdutor* de forma a representar a UAF lida. O símbolo de final de cadeia é representado por $, e seu consumo dispara a execução de uma ação adaptativa (representada por 𝒜$()) que gera as últimas alterações necessárias à execução do *Transdutor*. Após a leitura desse símbolo há uma chamada para a sub-máquina *Transdutor* e seu retorno é para o único estado final do autômato, indicando que a palavra foi reconhecida.

3.3.2.3 Transdutor

Em sua execução o *Transdutor* gera as seqüências de saídas fonéticas aceitas para representar uma palavra. A execução do *Transdutor* inicia-se com a cadeia de entrada vazia a partir da chamada de sub-máquina executada pelo *Reconhecedor*, portanto o *Transdutor* deve ser capaz de escrever na cadeia todos os símbolos que irá consumir.

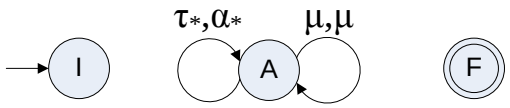

Figura 15 – Configuração inicial do *Transdutor*

A Figura 15 apresenta a configuração inicial do *Transdutor*, antes de ser alterado por funções adaptativas chamadas pelo *Reconhecedor*. A transição marcada é utilizada pelas funções

adaptativas para determinar onde as alterações devem ser realizadas nessa sub-máquina. A transição que consome um símbolo τ qualquer e escreve o símbolo α_* é a transição que inverte o sentido de execução como explicado na seção 3.2.1.

3.3.2.4 Funções Adaptativas

As funções adaptativas representam padrões para execução das ações adaptativas utilizadas para alterar a estrutura do autômato a cada UAF reconhecida. A função de reconhecimento de acentuação altera a estrutura do *Reconhecedor* enquanto as outras alteram a estrutura do *Transdutor*. A seguir são listadas as funções adaptativas desse autômato junto de suas descrições, um diagrama explicando seu funcionamento e as regras de utilização.

a) Influência na UAF Anterior (*ina*)

Essa função substitui, no último elo criado, as transições marcadas com o símbolo μ_π por uma transição que escreve um símbolo da classe π, que indica a influência que a UAF exerce sobre a UAF anterior. Essa função é chamada uma vez após o reconhecimento de cada UAF.

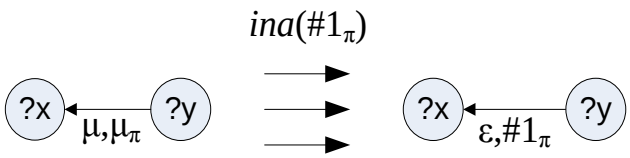

Figura 16 – Diagrama de alterações da função "Influência na Anterior"

O símbolo da classe π a ser escrito pela transição criada na chamada de *ina* é definido por passagem de parâmetro dessa função. O símbolo é definido, na maioria dos casos, em função do primeiro caractere da UAF e em alguns casos depende também do segundo caractere. A Tabela 5 apresenta as regras utilizadas para definir o valor utilizado em função dos caracteres iniciais.

Quando ocorre a leitura do símbolo indicador de final de cadeia pelo *Reconhecedor*, há uma chamada de *ina* que utiliza como parâmetro o símbolo π_F que indica que a UAF analisada anteriormente é a última da palavra.

Tabela 5 – Influência Posterior Gerada

Caracteres Iniciais	Influência Posterior Gerada
r	π_R
x, s, ç, j, z, *c* ou *g* seguidos de *e* ou *i*	π_S
b, d, l, v, *g* seguido de *a, o* ou *u*	π_V
f, p, q, t, *c* seguido de *a, o* ou *u*	π_U
m, n	π_N
h, vogais	π_*

b) Desloca a Marcação de Elo (*dm*)

Essa função apaga a transição de marcação do elo (que indica o local em que o elo deve ser adicionado) e cria uma nova transição de marcação do elo que tenha como estado de origem o estado de destino da transição apagada, e como estado de destino um novo estado gerado pela função. Essa função é chamada uma vez após o reconhecimento de cada UAF.

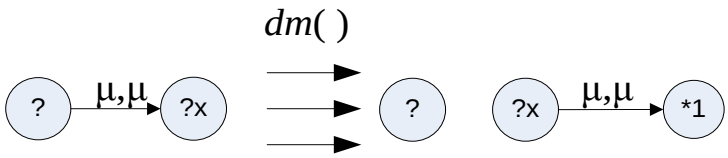

Figura 17 – Diagrama de alterações da função "Desloca Marcação de Elo"

c) Regras de Tonicidade (*rt*)

Essa função cria transições que representam as regras de tonicidade, que são utilizadas para reconhecer a tonicidade da UAF e definir a tonicidade da UAF anterior, para a UAF que está sendo adicionada ao *Transdutor*. A transição de marcação do elo é utilizada como referência para inserir as novas transições.

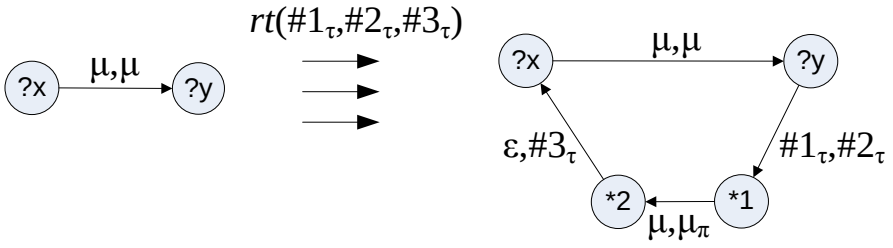

Figura 18 – Diagrama de alterações da função "Regras de Tonicidade"

Cada regra criada é formada por duas transições: a primeira reconhece o símbolo de tonicidade recebido como entrada (#1) e escreve um símbolo (#2) que é utilizado porteriormente na definição do som da UAF e a segunda define um símbolo de tonicidade (#3) para ser utilizado como entrada do mesmo processo na UAF anterior nessa mesma palavra. Além disso, uma terceira transição marcada com μ_π é adicionada para marcar o *ponto de entrada*[7] da função *ina* na construção do elo referente à próxima UAF.

Tabela 6 – Regras de tonicidade para criação dos elos

Tipo de UAF	Descrição	Chamadas de Função
Átonas Finais	UAF terminadas em *a, e, o, as, es, os, em, ens, am*	$rt(\tau_A, \tau_A, \tau_A)$ $rt(\tau_T, \tau_T, \tau_A)$ $rt(\tau_F, \tau_A, \tau_T)$
Acentuadas	UAF com *á, é, í, ó, ú, â, ê* e *ô*	$rt(\tau_A, \tau_T, \tau_A)$
Tônicas Finais	Outros	$rt(\tau_A, \tau_A, \tau_A)$ $rt(\tau_T, \tau_T, \tau_A)$ $rt(\tau_F, \tau_T, \tau_A)$

A Tabela 6 apresenta as chamadas da função adaptativa *rt* utilizadas na criação dos elos tendo como base o tipo de UAF analisada. As UAF se encontram divididas em três grupos: as *Átonas Finais* que têm som átono em final de palavra e indicam à UAF anterior que esta é tônico, as *Acentuadas* que forçam as outras UAF da palavra a serem átonas e as *Tônicas Finais* que são tônicas em final de palavras não acentuadas.

d) Influência na UAF posterior (*inp*)

Essa função cria regras que definem como uma UAF influencia a próxima. A transição de marcação de elo também é utilizada como referência por essa função que, sem apagar a transição de referência, cria uma nova transição de marcação que indica o ponto de entrada para a função adaptativa *ida*, e uma transição que escreve um símbolo da classe α que indica a influência que essa UAF exerce na UAF posterior. Essa função é chamada uma vez após o reconhecimento de cada UAF.

[7] Local de referência marcado por uma *transição de marcação* onde uma função insere novas transições.

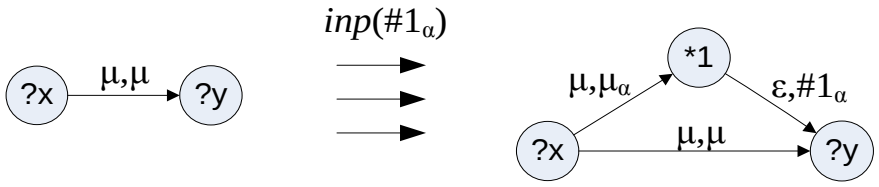

Figura 19 – Diagrama de alterações da função "Influência na Posterior"

Assim como as chamadas de *ina*, as chamadas da função adaptativa *inp* utilizam apenas um parâmetro. A definição da influência anterior que essa UAF exerce sobre a UAF seguinte depende apenas do último símbolo: utiliza-se α_V quando o último símbolo é uma vogal e α_* em outros casos.

Tabela 7 – Influência Anterior Gerada

Caractere Final	Influência Anterior Gerada
Vogal	α_V
Outros	α_*

e) Influência da UAF anterior (*ida*)

Essa função define regras para verificar a influência da UAF anterior na UAF cujo elo está sendo criado. A chamada dessa função utiliza a marcação criada pela função *inp* como ponto de entrada, e insere uma transição que consome um símbolo da classe α e uma transição de marcação utilizada como ponto de entrada de *idp*. Após o reconhecimento de uma UAF, essa função deve ser chamada uma vez para cada símbolo da classe α que influencie o som da UAF reconhecida.

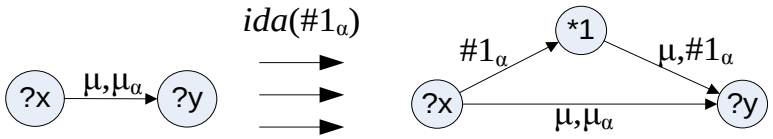

Figura 20 – Diagrama de alterações da função "Influência da Anterior"

A função adaptativa *ida* utiliza apenas um parâmetro, um símbolo da classe α, que indica o símbolo consumido pela transição a ser criada. Todas as UAF possuem uma chamada para *ida* passando como parâmetro o símbolo α_*, que indica a *regra padrão* na qual não há alteração sobre o som original da letra. As UAF iniciadas em *r* e *s* possuem uma segunda chamada para essa função adaptativa passando como parâmetro α_V.

Tabela 8 – Influências Anteriores Analisadas

Caractere Inicial	Influências Anteriores Analisadas
r e *s*	α_*, α_V
outros	α_*

f) Influência da UAF posterior (*idp*)

Essa função define regras para verificar a influência da UAF posterior na UAF cujo elo está sendo criado. Utilizando a transição criada por uma das chamadas da função adaptativa *ina* como ponto de entrada, as chamadas dessa função inserem uma transição que consome um símbolo da classe π e uma transição de marcação que indica o par (α, π). Após o reconhecimento de uma UAF, essa função deve ser chamada uma vez para cada par (α, π) de símbolos que influenciem o som dessa UAF.

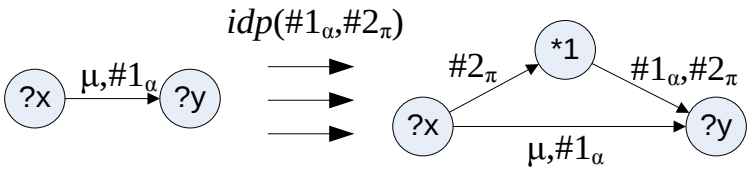

Figura 21 – Diagrama de alterações da função "Influência da Posterior"

As chamadas da função *idp* utilizam dois parâmetros, um símbolo da classe α que indica a influência anterior analisada (utilizado para encontrar a transição de marcação que indica onde o bloco de influência posterior deve ser colocado) e um símbolo da classe π que indica a influência posterior a ser analisada. Assim como ocorre no caso das influências anteriores, há uma chamada para *idp* que utiliza o símbolo π_*, que indica a *regra padrão* na qual não há alteração sobre o som original da letra. A Tabela 9 define os símbolos da classe π passados como parâmetro das outras chamadas da função *idp* tendo por base a estrutura da UAF.

Tabela 9 – Influências Posteriores Analisadas

Caracteres Finais	Influências Posteriores Analisadas
s, *z*	π_V, π_U, π_S
r	π_R
x	π_S
Vogais	π_N, π_F
em, *ens*, *am*	π_F
Vogais + *s*	π_V, π_U, π_S, π_F

g) Definição do Som (*som*)

Essa função cria as transições que escrevem símbolos fonéticos na saída. As chamadas desta função adaptativa utilizam como ponto de entrada a transição de marcação criada por *idp* e inserem transições que são executadas a partir do consumo de um símbolo da classe τ e escrevem seqüências fonéticas na saída.

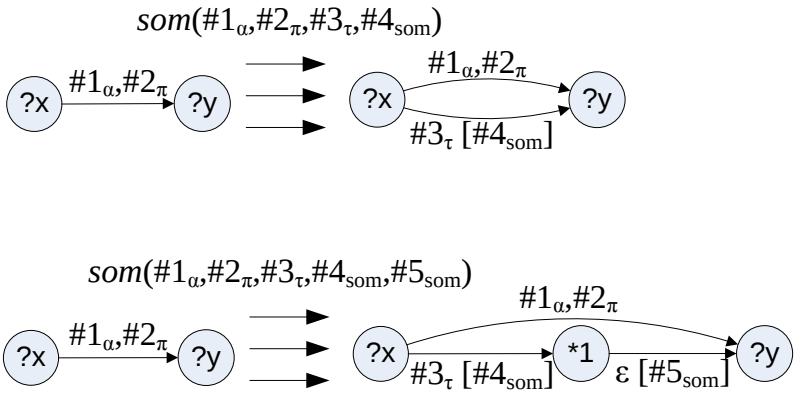

Figura 22 – Diagrama de alterações da função "Definição do Som"

Existem duas variações dessa função, uma que adiciona apenas um som para a UAF e uma que divide a UAF em dois sons. A função que escreve um som gera apenas uma transição que consome o símbolo τ e escreve a seqüência fonética na saída enquanto a variação que gera dois sons cria duas transições, uma que consome o símbolo τ e escreve a primeira seqüência fonética na saída e uma segunda transição vazia que escreve a segunda seqüência fonética na saída.

As chamadas da função adaptativa *som* utilizam quatro ou cinco parâmetros. Os dois primeiros parâmetros são símbolos das classes α e π utilizados para encontrar a transição de marcação que define onde as novas transições devem ser inseridas. O terceiro parâmetro é um símbolo de tonicidade que é consumido para gerar a seqüência fonética que representa a UAF. O quarto parâmetro e o quinto (quando existe) são seqüências de símbolos fonéticos que representam a UAF analisada quando recebe as influências e a tonicidade definidas nos três primeiros parâmetros.

Tipicamente, dentro de um bloco de influências posteriores, dois símbolos da classe τ são utilizados para indicar a tonicidade, τ_A e τ_T. Essa regra só não é válida nos seguintes casos:

1. UAF acentuadas: apenas o símbolo τ_T é utilizado (a UAF é sempre tônica).

2. UAF átonas finais: no bloco de influência posterior π_F apenas o símbolo τ_A é utilizado (esse bloco é utilizado quando a UAF é final e nessa situação essas UAF são átonas).

Os parâmetros que representam as seqüências fonéticas geradas são determinados a partir das características das UAF, da tonicidade e das influências recebidas. Para definir a seqüência fonética adequada a UAF é dividida em três partes: prefixo, núcleo e sufixo. O núcleo consiste do grupo de vogais da UAF e sua presença é obrigatória. O prefixo e o sufixo são estruturas opcionais e são formados por consoantes, encontros consonantais e dígrafos antes (prefixo) e depois (sufixo) do núcleo.

O som do prefixo é definido a partir das seguintes regras:

1. As letras *r* e *s* podem sofrer influência anterior. Precedidos de vogais (α_V) possuem os sons [ɾ] e [s], precedidos de outros símbolos (α_*) possuem os sons [ɣ] e [z].

2. As letras *t* e *d* transformam-se em fricativos ([tʃ] e [dʒ] respectivamente) quando anteriores aos sons [i] e [ɪ] (inclui UAF terminadas em *e* e *es* em posição final).

3. As letras *c* e *g* são transformadas em fricativas alveolares ([s] e [z] respectivamente) antes de *e*, *i*, *é*, *ê* e *í*.

4. As letras *j*, *q*, *ç* são representadas por [ʒ], [k] e [s]. A letra *x* é representada pelos sons [s], [z], [ʃ] e [ks].

5. Os dígrafos *ch*, *lh*, *nh*, *gu* e *qu* são representados por [ʃ], [ʎ], [ɲ], [g] e [k].

6. Em encontros consonantais com *r* na segunda posição, sua representação é [ɾ].

7. Os outros sons são invariantes e representados foneticamente pelo mesmo caractere utilizado na língua portuguesa.

Tabela 10 – Representação sonora para núcleos de um símbolo

suffix	-						*s*				*m, n*		other	
	π_*		π_N		π_F		π_*		π_F					
	τ_A	τ_T	τ_A	τ_T	τ_A	τ_T	τ_A	τ_T	τ_A	τ_T	τ_A	τ_T	τ_A	τ_T
a	[a]	[a]	[a]	[ɐ̃]	[ɐ]	-	[a]	[a]	[ɐ]	-	[ɐ̃]	[ɐ̃]	[a]	[a]
e	[e]	[e],[ɛ]	[e]	[ẽ]	[ɪ]	-	[e]	[e],[ɛ]	[ɪ]	-	[ẽ]	[ẽ]	[e]	[e],[ɛ]
i	[i]	[i]	[i]	[ĩ]	[ɪ]	[i]	[i]	[i]	[ɪ]	[i]	[ĩ]	[ĩ]	[i]	[i]
o	[o]	[o],[ɔ]	[o]	[õ]	[ʊ]	-	[o]	[o],[ɔ]	[ʊ]	-	[õ]	[õ]	[o]	[o],[ɔ]
u	[u]	[u]	[u]	[ũ]	[ʊ]	[u]	[u]	[u]	[ʊ]	[u]	[ũ]	[ũ]	[u]	[u]
á	-	[a]	-	-	-	[a]	-	[a]	-	[a]	-	-	-	[a]
â	-	[ɐ̃]	-	[ɐ̃]	-	[ɐ̃]	-	[ɐ̃]	-	[ɐ̃]	-	[ɐ̃]	-	[ɐ̃]
é	-	[ɛ]	-	-	-	[ɛ]	-	[ɛ]	-	[ɛ]	-	[ẽ]	-	[ɛ]
ê	-	[e]	-	[ẽ]	-	[e]	-	[e]	-	[e]	-	[ẽ]	-	[e]
í	-	[i]	-	[ĩ]	-	[i]	-	[i]	-	[i]	-	[ĩ]	-	[i]
ó	-	[ɔ]	-	[ɔ]	-	[ɔ]	-	[ɔ]	-	[ɔ]	-	[ɔ]	-	[ɔ]
ô	-	[o]	-	[õ]	-	[o]	-	[o]	-	[o]	-	[õ]	-	[o]
ú	-	[u]	-	[ũ]	-	[u]	-	[u]	-	[u]	-	[ũ]	-	[u]

O som do núcleo é definido da seguinte forma:

1. Se o núcleo possui um caractere: verificar Tabela 10.

2. Se o núcleo possui dois caracteres:

    a. *ai*, ei, oi*, au*, eu, ou*, ui, iu, ãe*, ãe*, õe** são representados por [aɪ̯], [eɪ̯], [oɪ̯], [aʊ̯], [eʊ̯], [oʊ̯], [uɪ̯], [iʊ̯], [ɐ̃ɪ̯], [ɐ̃ʊ̯], [õɪ̯].

    b. E terminar em *a*, *e* ou *o*: verificar Tabela 11.

3. Se o núcleo possui três caracteres:

    a. E termina em *aiu*: Se π_F e τ_T quebrar em [a]+[iʊ̯], quebrar em [aɪ̯]+[u] em outros casos. Terminados em *eiu*, *oiu*, *uiu*: seguir a mesma regra substituindo [a] por [e], [o] e [u] respectivamente.

    b. E iniciados em *gu* ou *qu* e terminados em um dos ditongos marcados com asterisco. Criar um tritongo transformando *u* em [ʊ̯] e utilizando a mesma regra para o ditongo.

Tabela 11 – Representação sonora para núcleos de dois símbolos

	π_*		π_N		π_F	
	τ_A	τ_T	τ_A	τ_T	τ_A	τ_T
ia	[i̯a]	[i]+[a]	[i̯a]	[i]+[a]	[i̯ɐ]	[i]+[a]
ie	[i̯e]	[i]+[e]	[i̯e]	[i]+[e]	[i̯ɪ]	[i]+[e]
io	[i̯o]	[i]+[o]	[i̯o]	[i]+[o]	[i̯ʊ]	[i]+[o]
ua	[u̯a]	[u]+[a]	[u̯a]	[u]+[a]	[u̯ɐ]	[u]+[a]
ue	[u̯e]	[u]+[e]	[u̯e]	[u]+[e]	[u̯ɪ]	[u]+[e]
uo	[u̯o]	[u]+[o]	[u̯o]	[u]+[o]	[u̯ʊ]	[u]+[o]
ae	[a]+[e]	[a]+[e]	[a]+[e]	[a]+[ẽ]	[aɪ̯]	[aɪ̯]
ao	[a]+[o]	[a]+[o]	[a]+[o]	[a]+[õ]	[au̯]	[au̯]
ea	[e]+[a]	[e]+[a]	[e]+[a]	[e]+[ɐ̃]	[i̯ɐ]	[e]+[ɐ]
eo	[e]+[o]	[e]+[o]	[e]+[o]	[e]+[õ]	[i̯ʊ]	[eu̯]
oa	[o]+[a]	[o]+[a]	[o]+[a]	[o]+[ɐ̃]	[u̯ɐ]	[o]+[ɐ]
oe	[o]+[e]	[o]+[e]	[o]+[e]	[o]+[ẽ]	[u̯ɪ]	[oɪ̯]

O som do sufixo é definido a partir das seguintes regras:

1. *l* é representado pela semi-vogal [u̯].

2. *n*, *m* são representados por liberações nasais [ⁿ] e [ᵐ].

3. *s* e *z* finais são representados por [z] quando seguidos de consoantes sonoras (π_V), [s] quando seguidos de consoantes surdas (π_U) ou em UAF finais (π_F) e não tem som quando precedem outro som fricativo (π_S).

4. *x* final não tem som quando precede um som fricativo (π_S) e é representado por [s] em outros casos.

5. *r* final não tem som quando precede outro *r* (π_R) e é representado por [ɾ] em outros casos.

h) Ajustes Finais (*af*)

Essa função gera os ajustes finais para que a execução da sub-máquina *Transdutor* ocorra com sucesso. Essa função é chamada pelo *Reconhecedor* quando ocorre a leitura do símbolo que indica o final da seqüência de UAF ($). A chamada desta função adaptativa apaga a transição marcada e liga os estados inicial e final ao resto da sub-máquina.

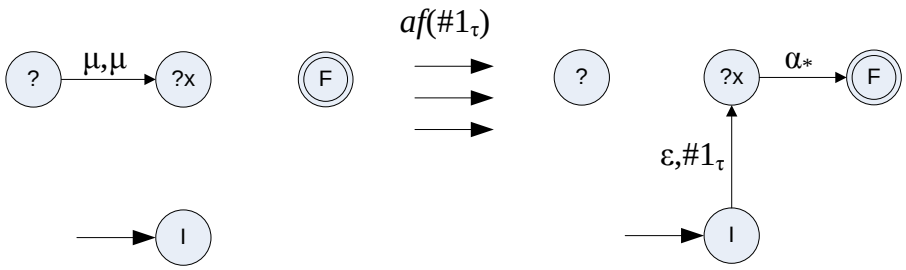

Figura 23 – Diagrama de alterações da função "Ajustes Finais"

i) Apagar Transição de Marcação (*am*)

Essa função é utilizada para apagar transições de marcação. Elas apagam as transições que consomem o símbolo passado no primeiro parâmetro e escrevem o símbolo passado no segundo parâmetro. Os símbolos são passados como parâmetros. Após o reconhecimento de uma UAF, essa função deve ser chamada para apagar todas as transições de marcação criadas durante a criação do respectivo elo, exceto pela transição de marcação do elo.

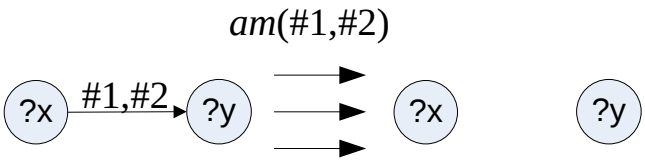

Figura 24 – Diagrama de alterações da função "Apagar Transição de Marcação".

Essa função deve ser utilizada com cuidado, pois ela apagará qualquer transição que consome e escreve os símbolos determinados pelos parâmetros. Nesse caso, há garantia de que só uma transição será apagada, pois a transição marcada é única para servir de ponto de entrada para as funções adaptativas.

j) Reconhecimento de Acentuação (*ra*)

Segundo as regras da língua portuguesa, uma palavra só pode ter um acento agudo ou circunflexo por palavra (nada impede que ocorram tremas ou tils junto desses acentos). Por essa razão, essa função apaga todas as transições de reconhecimento de UAF acentuadas quando a primeira é lida. Essa função também altera a transição que consome o símbolo final. Sua ação adaptativa é

substituida por uma ação em que a função *af* é chamada passando o parâmetro τ_A para que a sílaba acentuada seja a sílaba tônica.

3.3.2.5 Ações Adaptativas

As ações adaptativas associadas à leitura de uma UAF pelo *Reconhecedor* são compostas por seqüências de chamadas das funções adaptativas definidas no item anterior. Em relação às alterações realizadas, pode-se dizer que essas ações adaptativas (trabalhando com regras contextuais) estão para chamadas de função adaptativa do modelo original (trabalhando com transições) como chamadas das funções aqui definidas (trabalhando com regras contetuais) estão para uma ação elementar do modelo original (trabalhando com transições). As ações adaptativas associadas ao reconhecimento criam conjuntos de regras que representam uma UAF, enquanto as chamadas das funções adaptativas criam as partes elementares desses conjuntos que são as regras contextuais.

Essas ações adaptativas devem criar o elo de forma a representar a UAF lida. Para isso, é importante que as funções adaptativas sejam chamadas em uma ordem que permita a estruturação correta do elo. É importante que a sequência siga três regras:

1. As chamadas de *rt* devem ser posteriores à chamada de *ina*.

2. As chamadas de *inp*, *ida* e *idp* devem ser nessa ordem.

3. Todas as transições de marcação devem ser apagadas ao final da criação do elo.

A ordem apresentada nesta seção não é a única válida, mas permite a estruturação em blocos que facilitam a visualização de seu funcionamento. A seqüência é dividida em dois blocos: no primeiro bloco são criadas as regras de tonicidade e as transições que definem as influências que a UAF exerce sobre as UAF vizinhas, enquanto no segundo bloco são criadas as transições que definem as influências que a UAF recebe de suas vizinhas e as transiçoes que definem o som correto para a UAF analisada.

O primeiro bloco é formado por uma chamada de *ina* passando como parâmetro o símbolo π definido em função da primeira letra desse elo, uma chamada de *dm*, uma a três chamadas de *rt*

para definir as regras de tonicidade e uma chamada de *inp* passando como parâmetro o símbolo α definido em função da última letra dessa UAF.

O segundo bloco é dividido em blocos menores, chamados blocos de influência anterior. Um bloco de influência anterior inicia-se com uma chamada de *ida*, termina com uma chamada de *am* que apaga a transição de marcação criada pela chamada de *ida* e possui blocos de influência posterior definidos entre essas duas chamadas de função adaptativa. Os blocos de influência posterior, por sua vez, iniciam-se com chamadas de *idp*, terminam em chamadas de *am* que apagam as transições criadas pela respectiva chamada de *idp* e podem possuir chamadas de *som* (quantas forem necessárias).

Ao final da criação do último bloco de influência anterior há uma chamada para a função *am*, que apaga a transição de marcação criada pela função *inp* utilizada como referência nas chamadas da função *ida*.

As UAF que possuem vogais marcadas com acento agudo ou circunflexo, possuem uma chamada para *ra* que geram alterações no *Reconhecedor* impedindo a existência de duas vogais acentuadas na mesma palavra e no *Transdutor* indicando que a palavra análisada é acentuada.

3.4 Exemplos

Para facilitar o entendimento do funcionamento do autômato, a seguir são apresentados alguns exemplos. O primeiro exemplo apresenta a ação adaptativa referente à UAF σ_{sa}, o segundo exemplo ilustra a criação de um elo após o reconhecimento da mesma UAF, apresentando passo a passo as alterações realizadas pela ação adaptativa descrita e o terceiro exemplo apresenta ilustrações com a estrutura do *Transdutor* após o reconhecimento de algumas palavras, mostrando diversos tipos de elos e destacando as transições utilizadas no processo de execução.

3.4.1 Ação Adaptativa

A Figura 25 apresenta um exemplo de ação adaptativa acionada quando ocorre a leitura de uma UAF no *Reconhecedor*. Essa é a ação adaptativa associada à leitura da UAF σ_{sa}. Essa ação adaptativa é composta por uma seqüência de chamadas das função adaptativas descritas no item 3.3.2.4 utilizando os devidos parâmetros e na ordem correta.

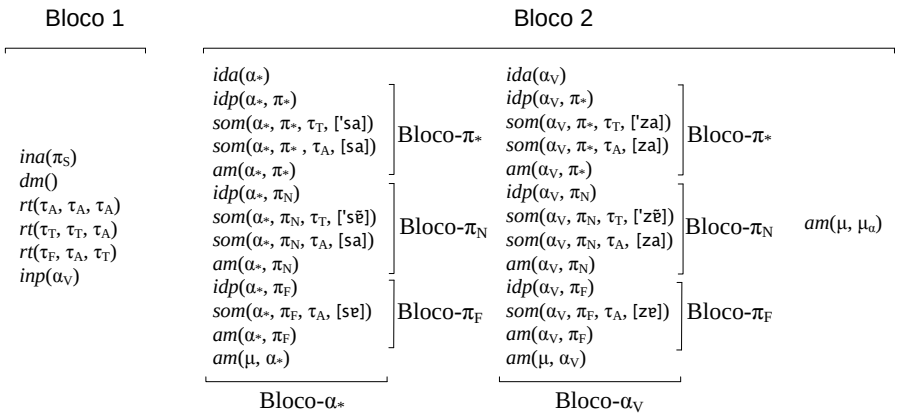

Figura 25 – Ação adaptativa $\mathcal{A}_{sa}$( ) associada à leitura de σ_{sa}

O primeiro bloco é composto por uma chamada de *ina* (π_S passado como parâmetro, pois inicia em fricativo), uma chamada para *dm*, três chamadas de *rt* (como determinado pela primeira linha da Tabela 6) e uma chamada de *inp* (α_V passado como parâmetro pois termina em vogal).

O segundo bloco é composto por dois blocos de influência anterior, padrão (α_*) e vogal (α_V), pois a UAF começa com a letra *s*. Para cada bloco de influência anterior há três blocos de influência posterior, padrão (π_*), nasal (π_N) e final (π_F), pois a UAF termina em vogal. Para os blocos de influência posterior padrão e nasal há duas chamadas da função adaptativa *som*, contemplando a possibilidade de sons tônicos (τ_T) e átono (τ_A), e para os blocos de influência posterior final há apenas uma possibilidade (τ_A) levando em consideração a segunda das exceções feitas sobre a tonicidade.

Os sons da letra *s* são diferenciados pelos blocos de influência anterior, [s] para o bloco-α_* e [z] para o bloco-α_V. Os sons da letra *a* são diferenciados pelos blocos de influência posterior e pela tonicidade, [a] para o bloco-π_* (ambas as tonicidades), [ẽ] (tônico) e [a] (átono) para bloco-π_N e [ɐ] para bloco π_F.

3.4.2 Alteração do Transdutor

Esta seção apresenta um exemplo das alterações que ocorrem no *Transdutor* durante as ações adaptativas acionadas no processo de reconhecimento das UAF que formam uma palavra. Neste exemplo são analisadas as alterações realizadas no reconhecimento da palavra *casa*, estudando passo a passo as alterações referentes à segunda UAF (σ_{sa}).

As ilustrações apresentadas nesta seção são compostas de uma representação completa do *Transdutor*, acompanhados do esquema simplificado (semelhante ao da Figura 12) sempre que há uma alteração significativa no *Transdutor*. Esse esquema destaca as estruturas criadas (preto), as que não foram criadas (cinza) e as criadas mais recentemente (vermelho).

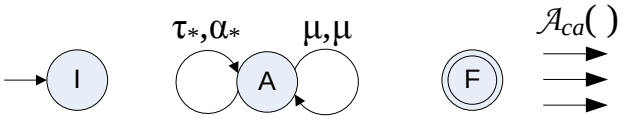

Figura 26 – Ação Adaptativa - Configuração Inicial do *Transdutor*

A Figura 26 apresenta a configuração inicial do Transdutor. Os estados `I` e `F` são omitidos das figuras seguintes que mostram as alterações da estrutura do *Transdutor* até que sejam utilizados por uma ação adaptativa.

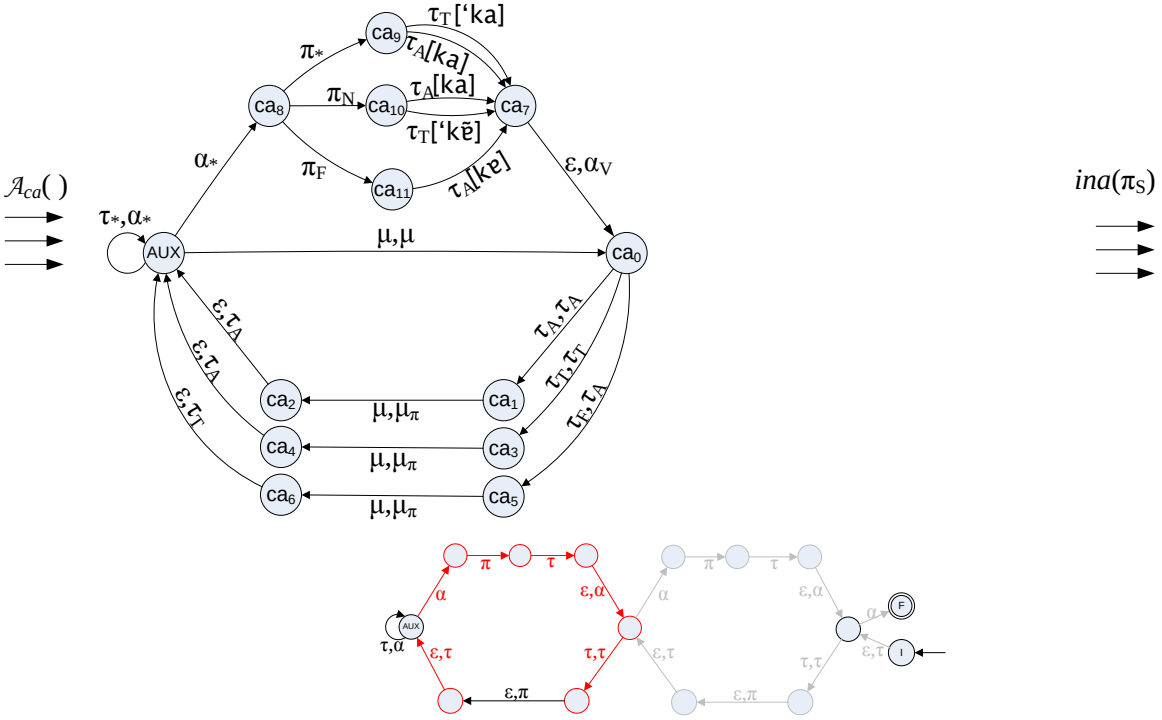

Figura 27 – Ação Adaptativa – Reconhecimento da UAF σ_{ca}

A Figura 27 apresenta as alterações que ocorrem a partir da configuração inicial para criar o elo que representa σ_{ca}. A ação adaptativa é representada por $\mathcal{A}_{ca}$( ) nessa figura. A partir da estrutura gerada, vamos analisar passo a passo a criação do elo que representa σ_{sa}.

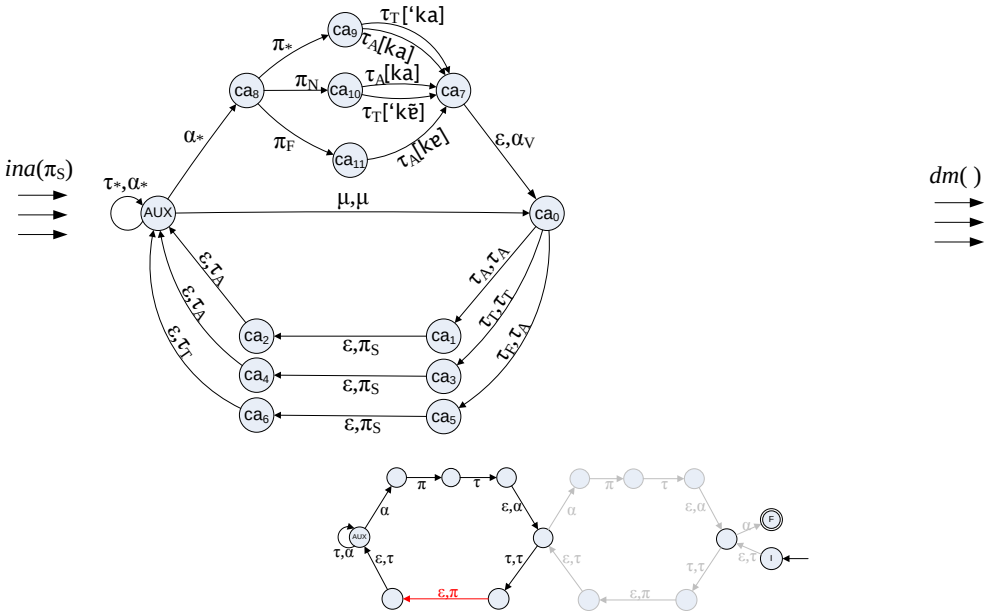

Figura 28 – Ação Adaptativa – Influência na UAF Anterior (*ina*)

A criação do elo que representa σ_{sa} se inicia por meio de uma chamada de *ina*, que substitui no elo que representa σ_{ca}, as transições marcadas com o símbolo μ_π por transições que escrevem o símbolo π_S, indicando que a UAF seguinte inicia-se com a letra *s*, como ilustrado na Figura 28, seguida de uma chamada de *dm*, apresentada na Figura 29 que desloca a transição de marcação do elo indicando onde o novo elo deve ser criado.

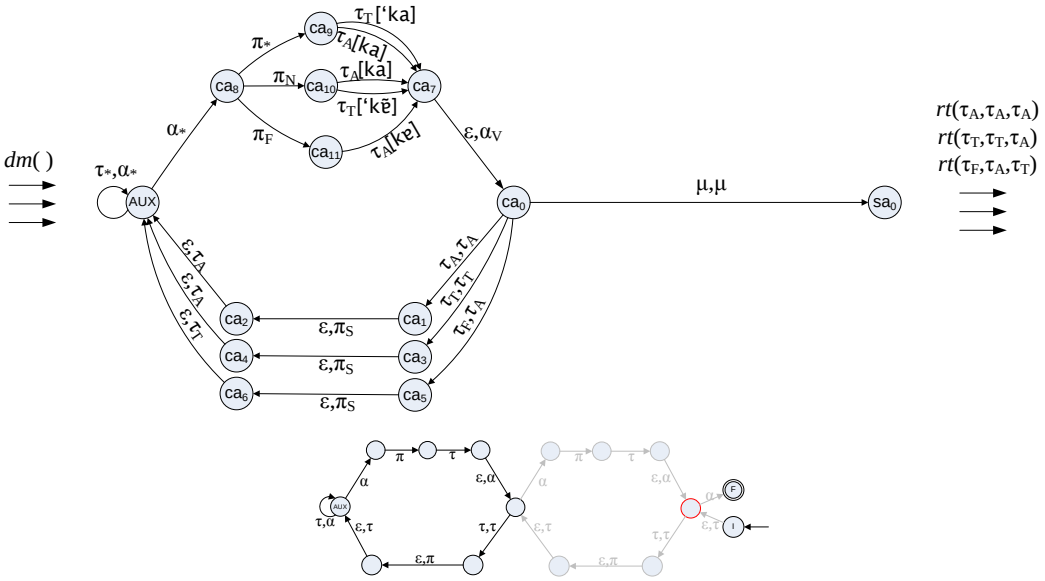

Figura 29 – Ação Adaptativa – Desloca Transição de Marcação do Elo (*dm*)

Sabendo onde o novo elo deve ser colocado, inicia-se a construção dessa estrutura. Esse processo se inicia através de chamadas de *rt*, ilustradas na Figura 30, que criam as regras de tonicidade referentes a σ_{sa} como definido pela primeira linha da Tabela 6 e novas transições marcadas pelo símbolo μ_π que devem ser substituidas na próxima chamada de *ina*.

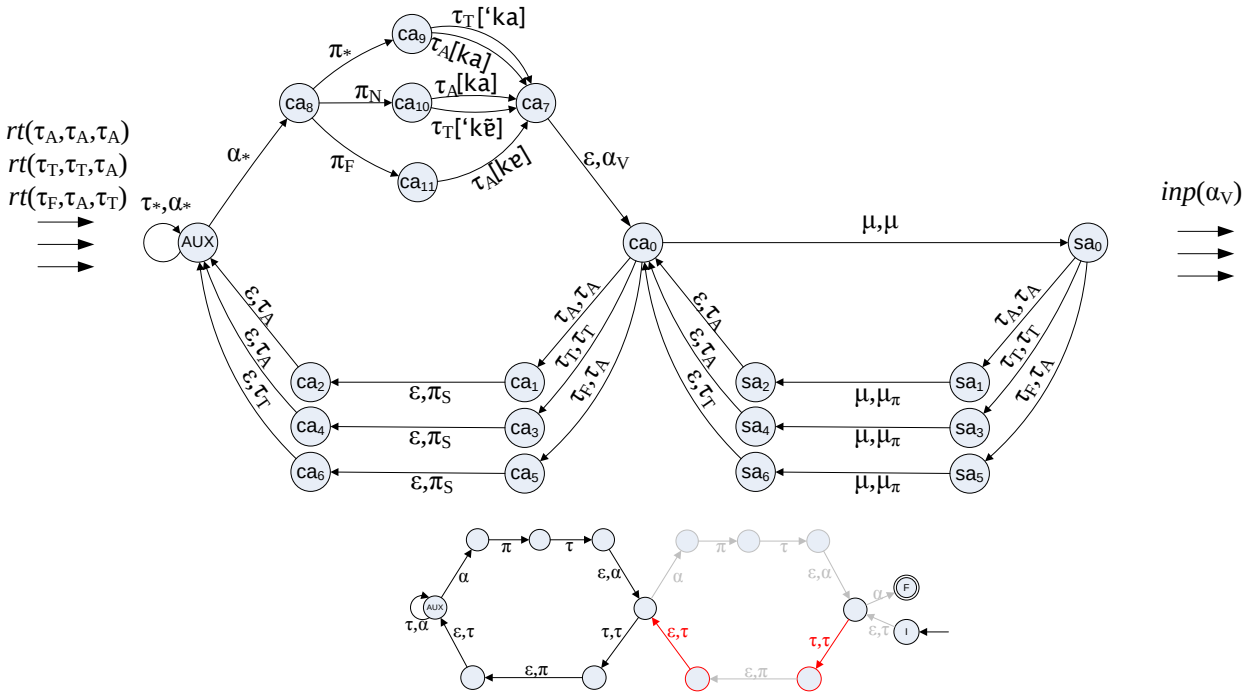

Figura 30 – Ação Adaptativa – Regras de Tonicidade (*rt*)

A chamada da função *inp* cria a transição que escreve o símbolo α_V na cadeia, indicando que essa UAF termina em vogal, e cria uma transição de marcação que indica o ponto de entrada para as chamadas de *ida*. A Figura 31 apresenta essa alteração e destaca transições de escrita de influência anterior no modelo simplificado.

A chamada de *idp* termina a construção do primeiro bloco, que define as influências que σ_{sa} exerce em seus vizinhos e suas regras de tonicidade. As próximas chamadas constroem o segundo bloco, onde há o reconhecimento das influências que a UAF recebe das UAF vizinhas e onde a saída fonética é gerada.

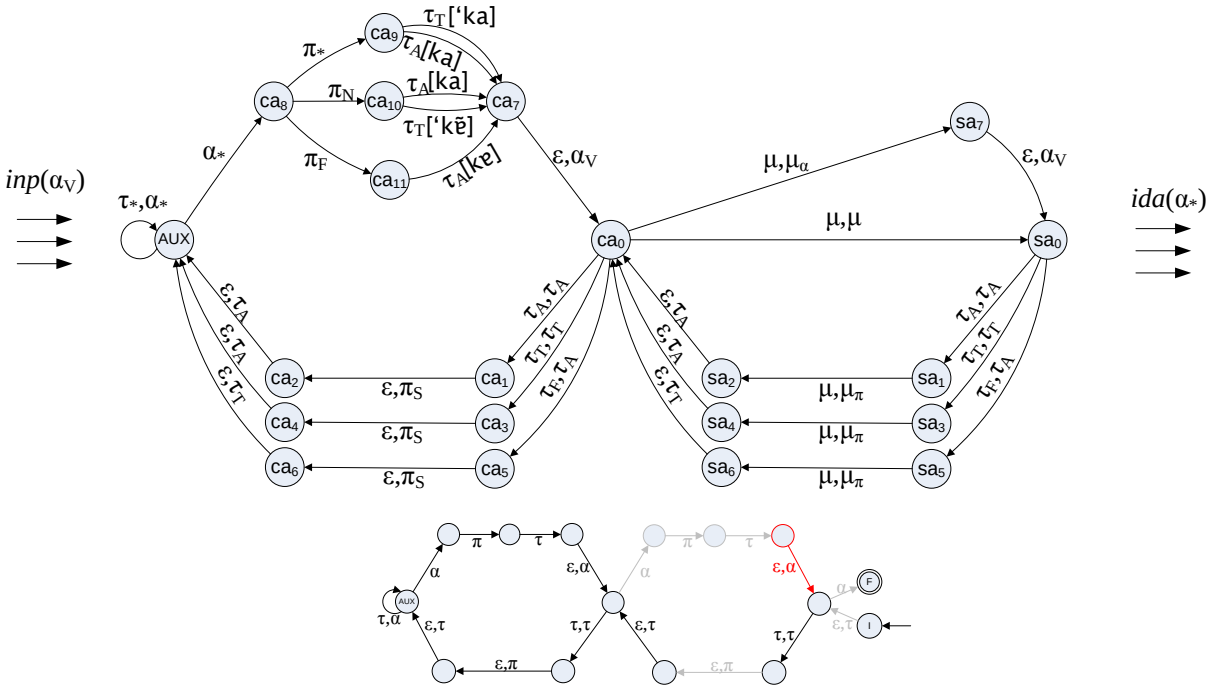

Figura 31 – Ação Adaptativa – Influência na UAF Posterior (*inp*)

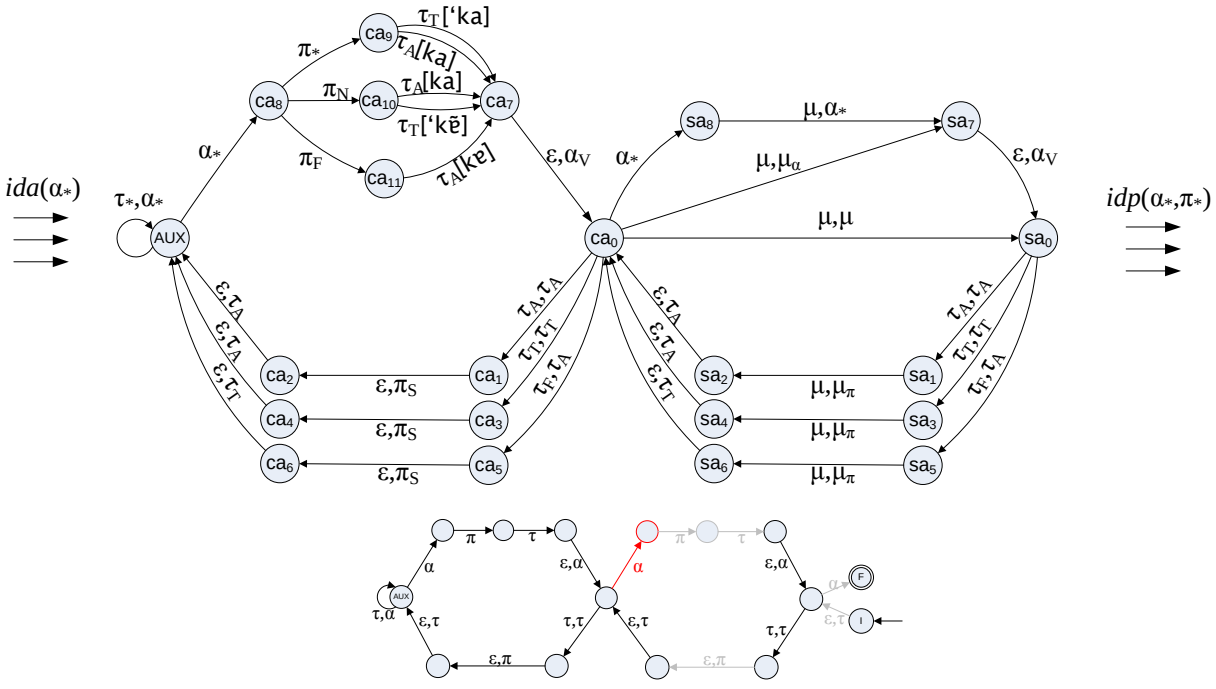

Figura 32 – Ação Adaptativa – Influência da UAF Anterior (*ida*)

A construção do segundo bloco se inicia com a chamada de *ida*, passando como parâmetro o símbolo α_*. A alteração realizada por essa ação adaptativa é apresentada na Figura 32 dando destaque à criação de uma regra de leitura de influência anterior. Além da criação dessa regra, cria-se uma transição de marcação que serve como ponto de entrada para as próximas chamadas de *idp*.

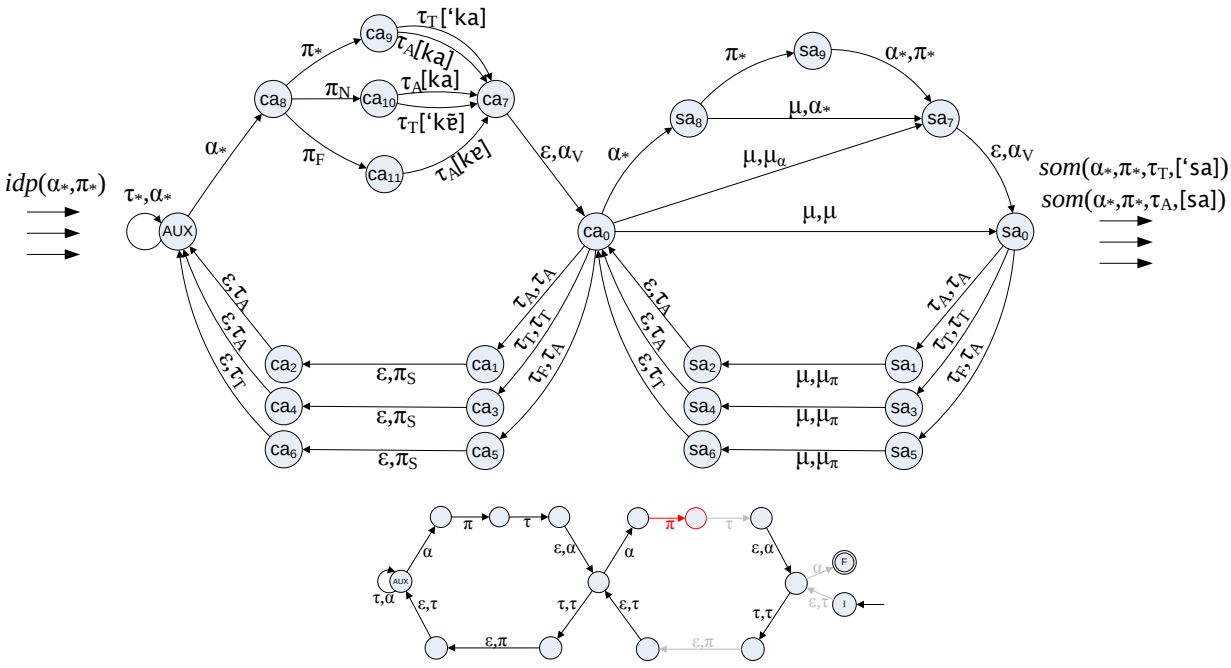

Figura 33 – Ação Adaptativa – Influência da UAF Posterior (*idp*)

A Figura 33 ilustra a alteração realizada na primeira chamada de *idp* dentro do bloco-α_*. Nessa ação adaptativa é criada a transição que consome o símbolo π_*, destacada no esquema simplificado, e uma transição de marcação utilizada como ponto de entrada para as chamadas da função *som*.

As alterações realizadas nas chamadas da função *som* são destacadas na Figura 34. Duas novas transições são criadas, cada uma por uma chamada dessa função, para o consumo dos símbolos τ_A e τ_T gerando os sons átono [sa] e tônico ['sa] respectivamente.

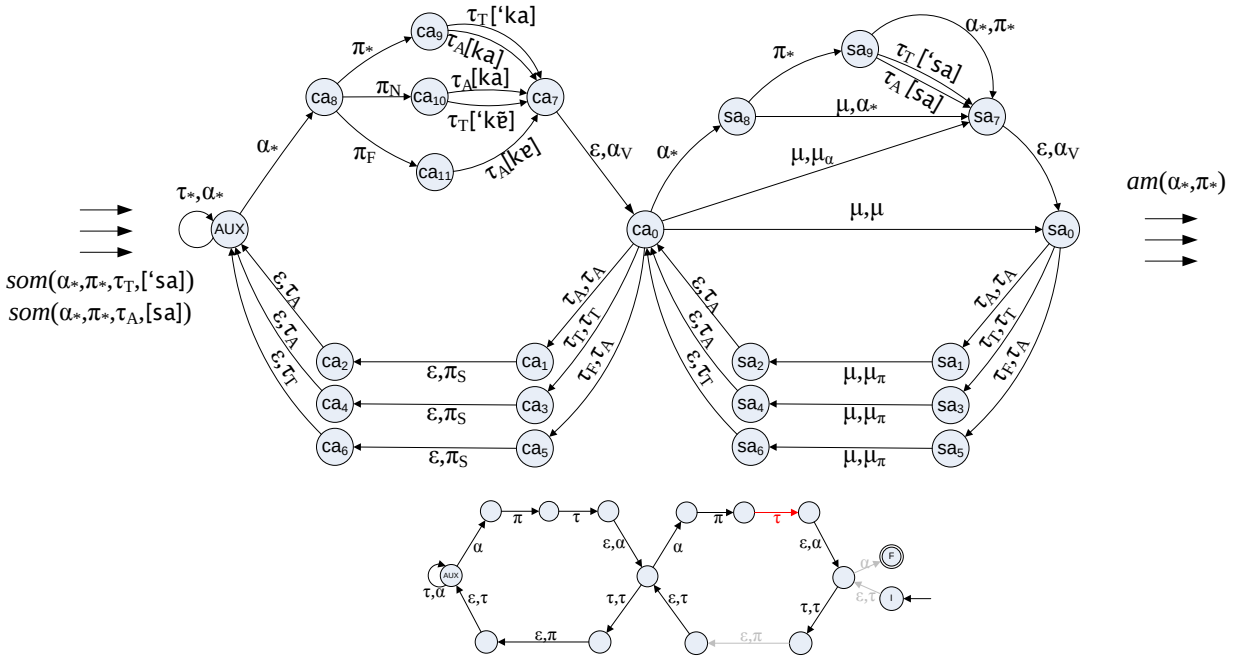

Figura 34 – Ação Adaptativa – Cria transições para gerar saída fonética (*som*)

A Figura 35 destaca a chamada de *am* apangando a transição de marcação utilizada pelas chamadas da função *som*.

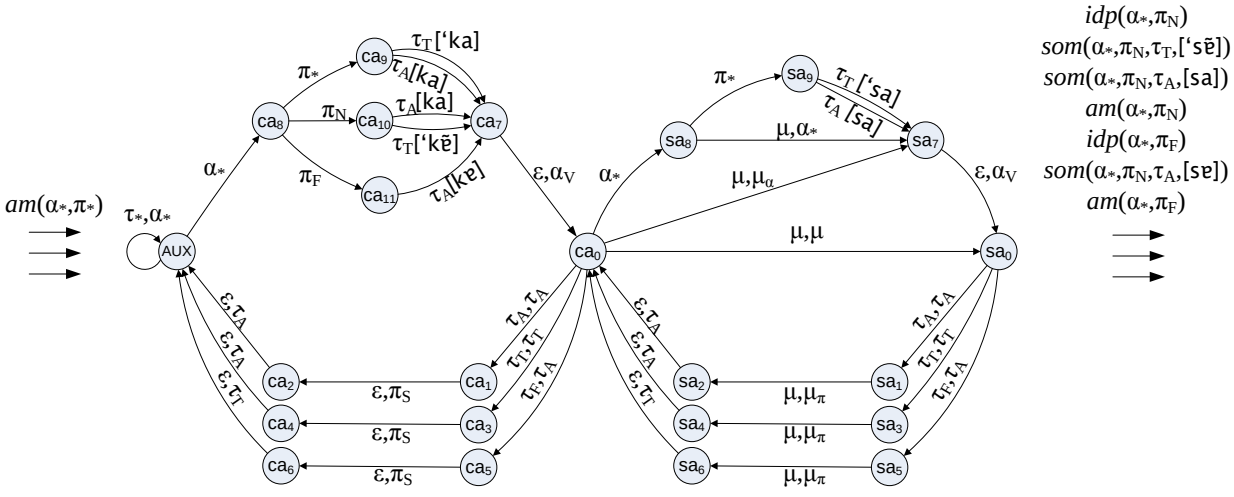

Figura 35 – Ação Adaptativa – apaga ponto de entrada de *som*

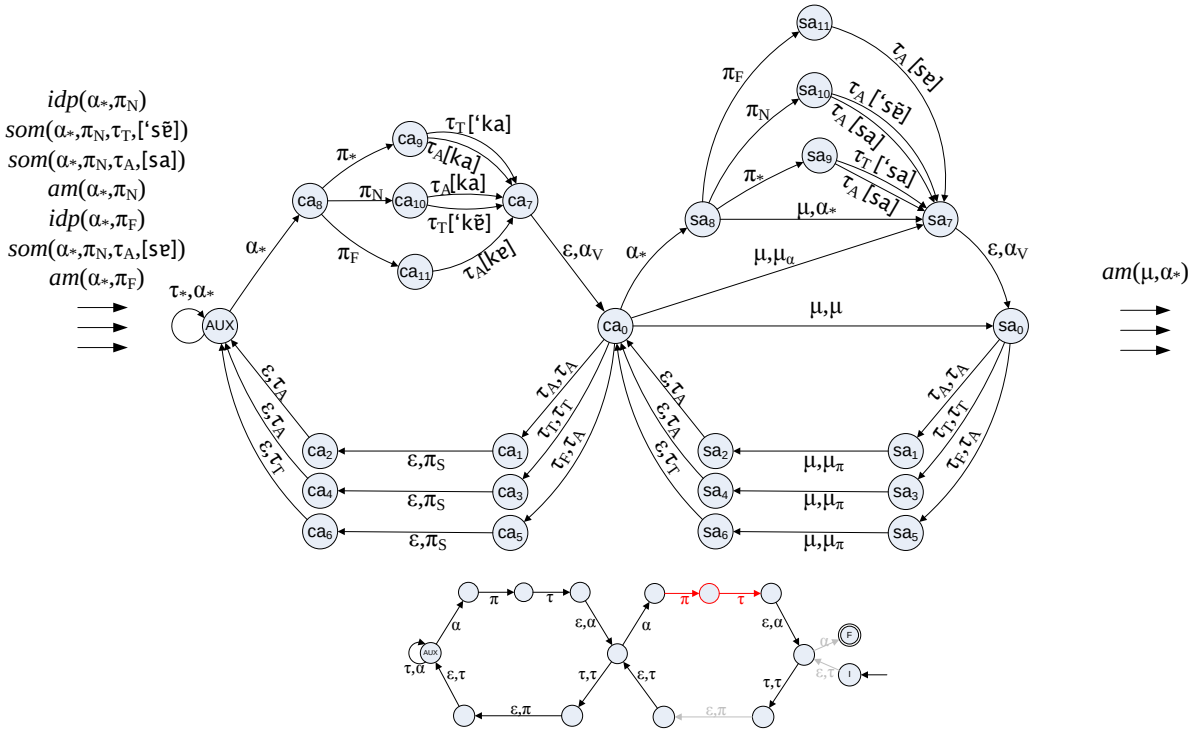

Figura 36 – Ação Adaptativa – Criação de Blocos-π

A Figura 36 destaca a repetição do processo de criação para blocos de influência posterior nasal (π_N) e final (π_F) e a Figura 37 ilustra a chamada de *am* que apaga a transição de marcação utilizada pelas chamadas de *idp*.

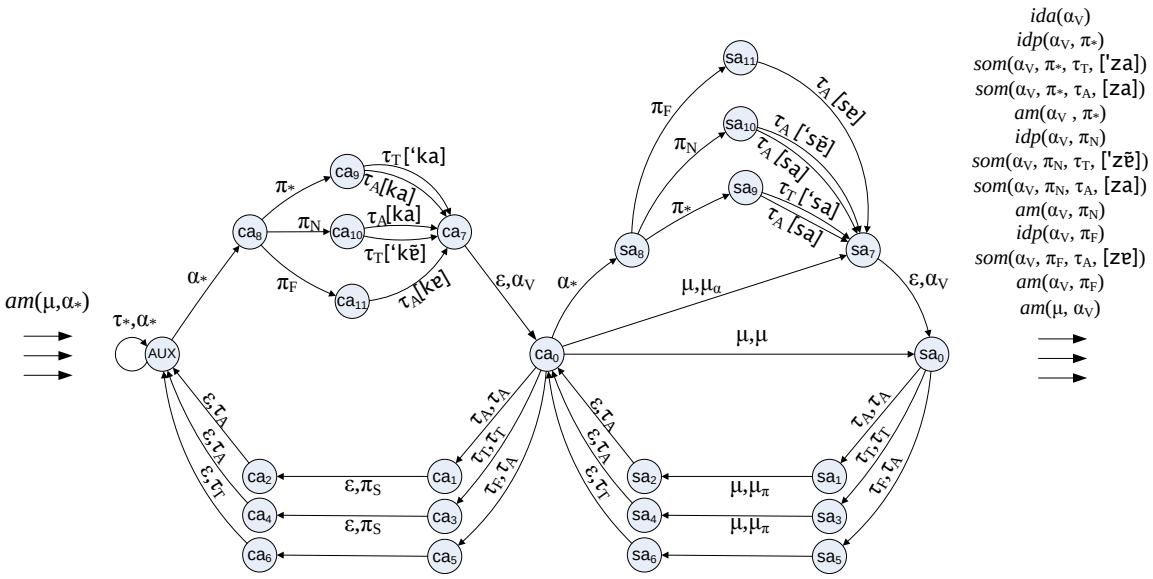

Figura 37 – Ação Adaptativa – Apaga ponto de entrada de *idp*

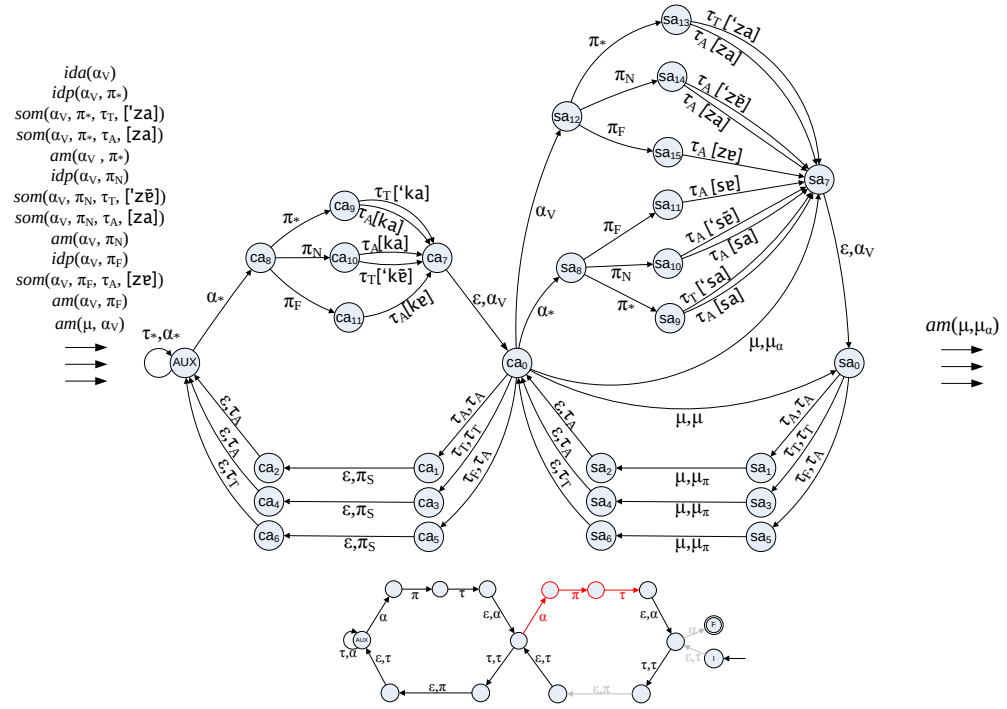

Figura 38 – Ação Adaptativa – Criação de Bloco-α

A Figura 38 apresenta a repetição do processo de criação para o bloco de influência anterior de uma vogal (α_V) e a Figura 39 ilustra a chamada de *am* que apaga a transição de marcação criada por *inp* e utilizada como ponto de entrada para as chamadas de *ida*. Nesse momento termina a execução da ação $\mathcal{A}_{sa}$, e o *Reconhecedor* pode ler outra UAF.

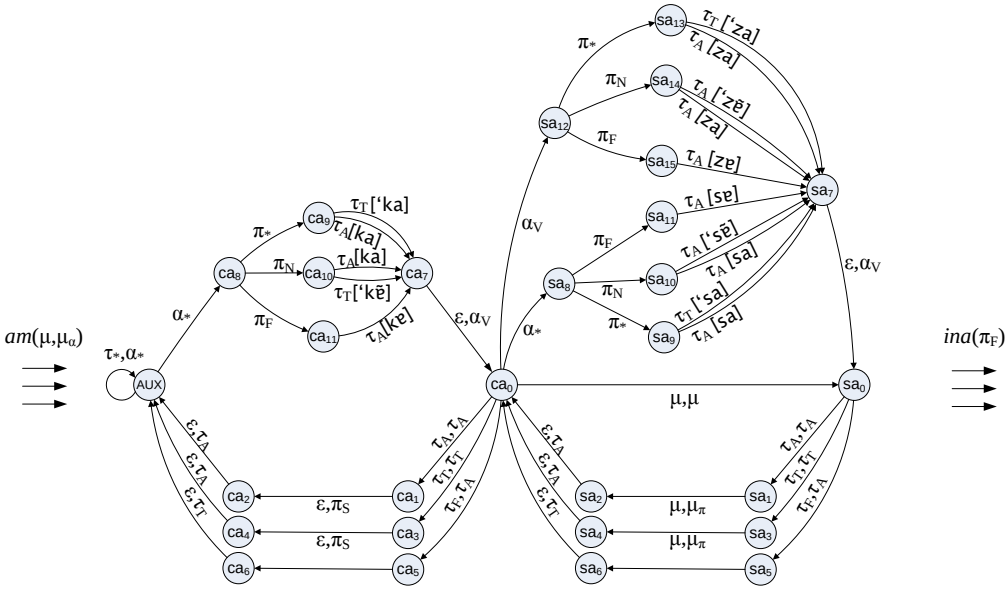

Figura 39 – Ação Adaptativa – Apaga ponto de entrada de *ida*

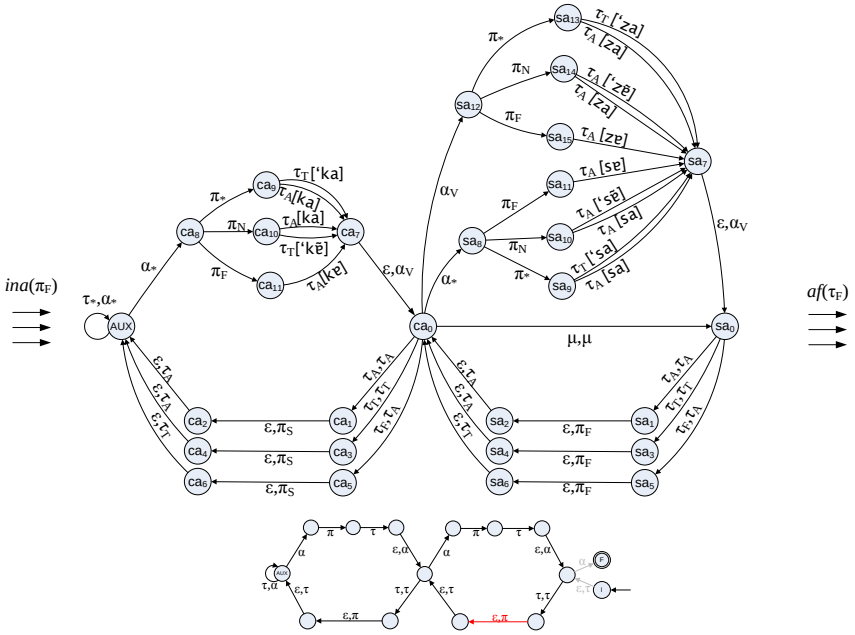

Figura 40 – Ação Adaptativa – Influência Posterior na UAF Final

No caso da palavra analisada, como σ_{sa} é a UAF final, o *Reconhecedor* lê o símbolo de final de cadeia e realiza chamadas para a função *ina*, ilustrada na Figura 40, que prepara substitui as transições marcadas por μ_π por transições que escrevem π_F e *af*, ilustrada na Figura 41, que preparam liga os estados inicial (I) e final (F) tornando o *Transdutor* pronto para execução.

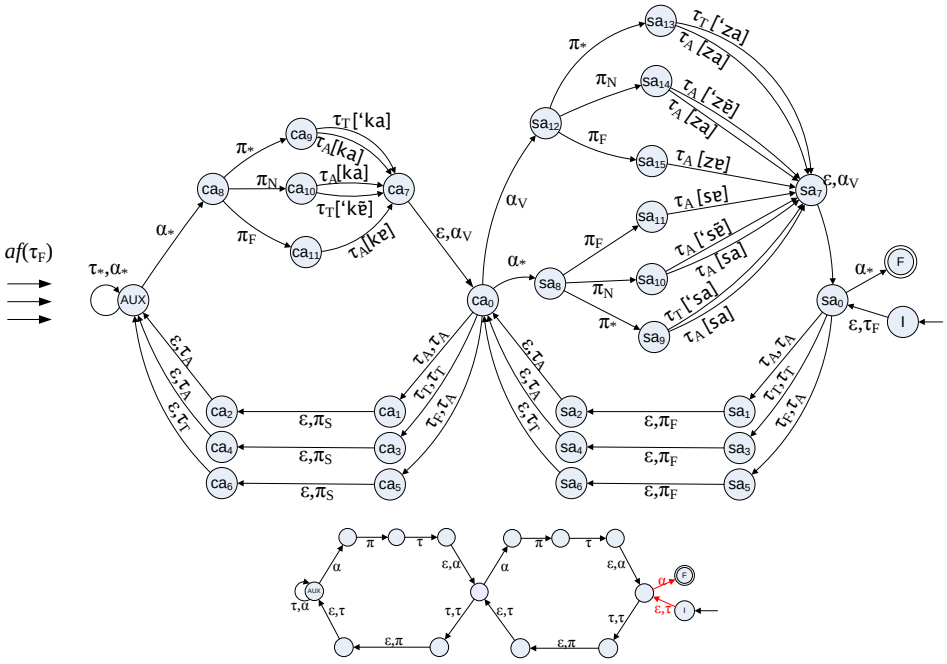

Figura 41 – Ação Adaptativa – Ajustes Finais

3.4.3 Palavras

Esta seção apresenta ilustrações com a representação da estrutura do *Transdutor* após o termino da leitura da cadeia de entrada executada pelo *Reconhecedor*. Utilizando essas figuras, pode-se entender como ocorre o processo de interação entre os elos que determina a representação fonética para uma palavra. Nessas figuras, são destacadas em vermelho as transições utilizadas durante a execução. As palavras *casa*, *colher*, *sabia* e *sábia* foram escolhidas como exemplos.

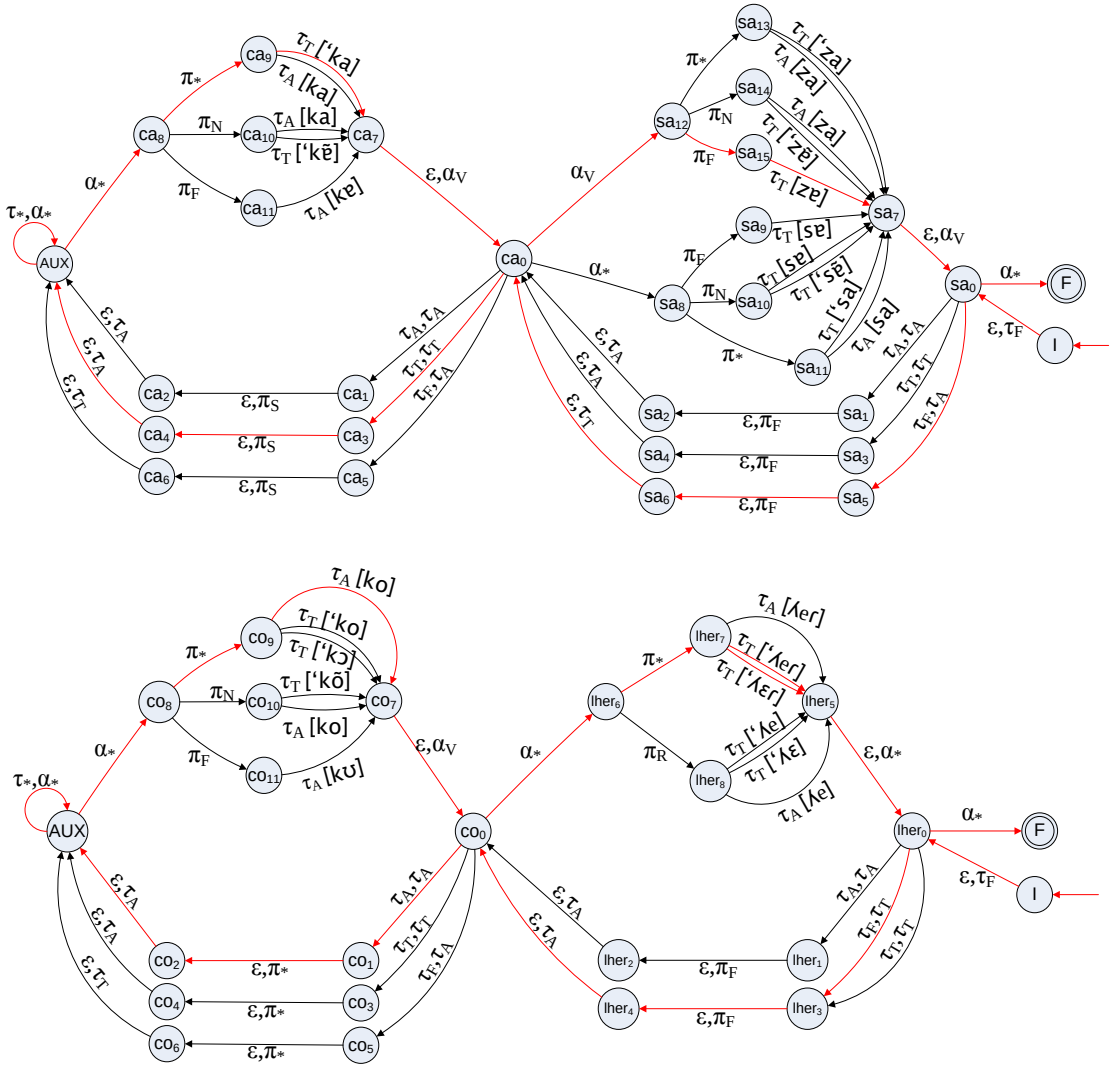

Figura 42 – Execução do *Transdutor* – *casa* e *colher*

Na palavra *casa* pode-se perceber a existência de uma sílaba átona final que desencadeia o processo para a sílaba anterior ser tônica, a influência anterior da vogal sobre o som da letra *s* gerando som [z] e a influência posterior que faz com que a letra *a* de σ_{sa} seja representada pelo

fonema [s]. A palavra *colher* apresenta um exemplo de palavra em que a sílaba final é tônica. Por essa razão, a letra *e* pode ser representada pelos sons [e] e [ɛ] e a palavra analisada possui dois sons diferentes[8]. No caso analisado os dois sons são válidos. A saída com o som [e] representa o verbo *colher* e a saída com o som [ɛ] representa o substantivo *colher*.

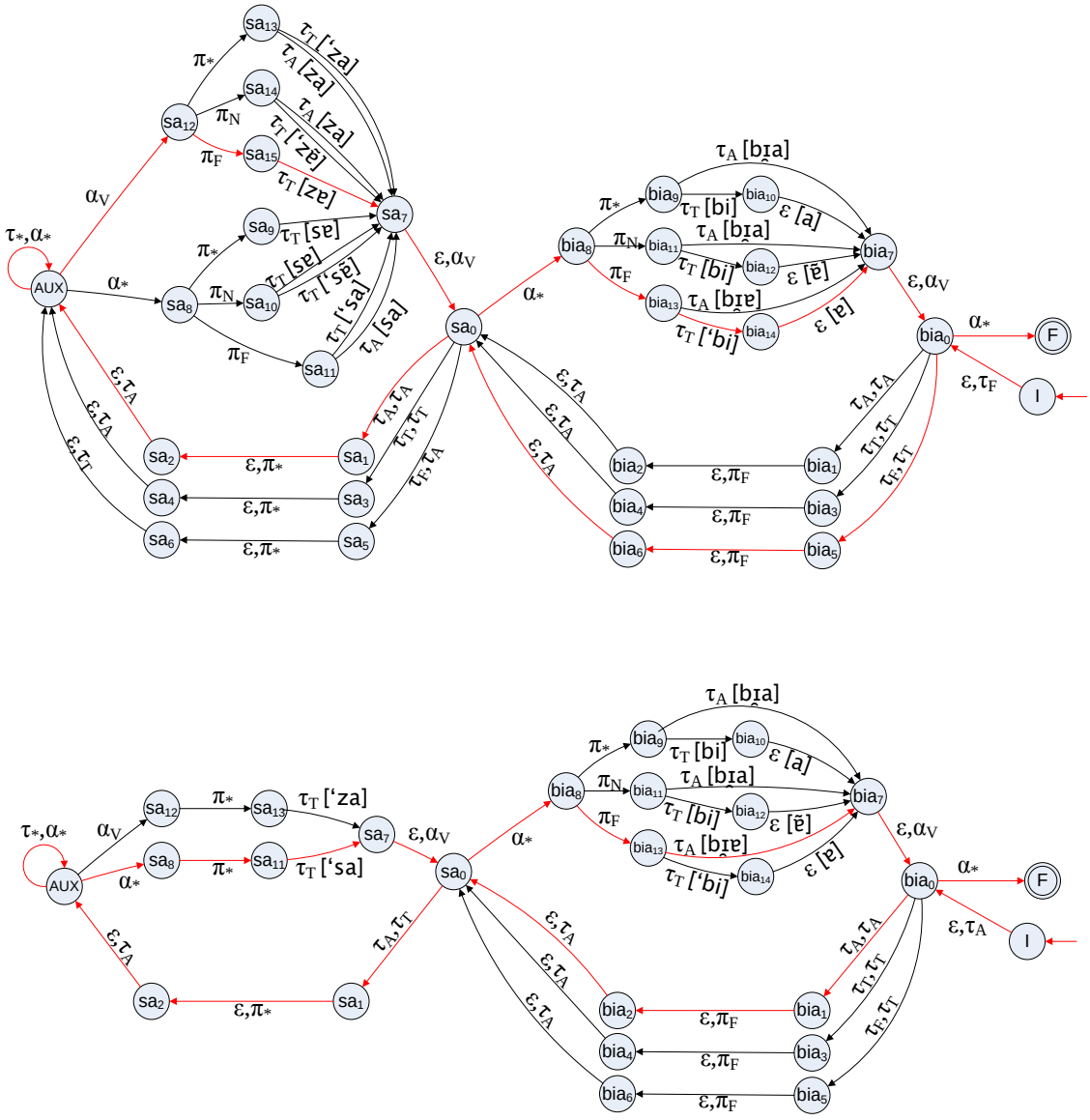

Figura 43 – Execução do *Transdutor – sabia* e *sábia*

[8] O som [ɛ] pode ser usado para 'e' átono (cafezinho), mas isso foi omitido do modelo pois o número de saídas aumentaria significativamente (paralelpipedo teria oito representações), na maioria dos casos o valor utilizado é [e] e não foi encontrada uma regra para escolher uma dentre as representações geradas.

A terceira e a quarta palavra escolhidas são diferenciadas entre si pelo acento gráfico. A comparação dos dois exemplos permite analisar como a alteração gerada pela sílaba acentuada, fazendo com que a função *af* escreva o símbolo τ_A no lugar de τ_F altera a representação fonética da palavra. Como resultado dessa alteração, $\sigma_{sá}$ passa a ser tônico e o resto da palavra tem som átono.

Essas duas palavras também podem ser utilizadas para analisar o processo de criação de ditongos por influência de uma sílaba acentuada. Na palavra *sabia* a UAF σ_{bia} se divide em duas partes sendo uma delas átona e outra tônica, enquanto na palavra *sábia* essa UAF não é dividida e a letra *i* é representada por uma semi-vogal.

4 Máquina Virtual

Para verificar a eficácia do modelo de autômato desenvolvido é necessário utilizar um software capaz de simular o funcionamento desse modelo. Seria então desejável um software que simulasse os autômatos adaptativos, que deveria ser utilizado como base para o desenvolvimento do software de tradução texto-voz.

Houve então a necessidade de escolher entre utilizar o Adaptools (PISTORI; JOSÉ NETO, 2003) como base para o desenvolvimento desse tradutor texto-voz, ou desenvolver um novo software para a simulação de autômatos adaptativos. Devido a algumas características de implementação do Adaptools (como a impossibilidade de escrever na cadeia de entrada e de trabalhar com autômatos não determinísticos) e escolhas de projeto, decidiu-se por criar um novo software para a simulação desses dispositivos.

4.1 Decisões de Implementação

Após a decisão de criar um novo simulador de autômatos houve a necessidade de definir as características desse simulador. Inicialmente, pensava-se em um simulador para diversos dispositivos que desacoplasse a camada adaptativa, as ações semânticas e a característica de determinismo do tipo de dispositivo utilizado. Essa idéia foi descartada devido à dificuldade de encontrar uma arquitetura que contemplasse essas características mantendo o devido desacoplamento entre os objetos e sem a necessidade de burlar regras de boas práticas de desenvolvimento.

Decidiu-se por criar apenas uma API que pudesse ser utilizada como base para a simulação de autômatos adaptativos sem a criação de interfaces gráficas. A linguagem Java foi escolhida para o desenvolvimento e o decidiu-se por seguir o paradigma da orientação a objetos para a criação de uma arquitetura para representação dos autômatos adaptativos.

Além dessas características, era necessário que o simulador permitisse a simulação de autômatos não determinísticos e a escrita de ações semânticas na seqüência de saída, e contemplasse as regras de funcionamento do autômato como definidas por Neto (1993).

4.2 Estrutura do Software

A arquitetura utilizada para representar a estrutura dos autômatos adaptativos se baseia em três elementos: as classes *Automaton* e *Configuration* e a interface *Action*. Nesses três elementos se encontram as regras de funcionamento do simulador. As outras classes representam estruturas de dados utilizadas para definir o comportamento do simulador. A Figura 44 apresenta a estrutura estática do simulador.

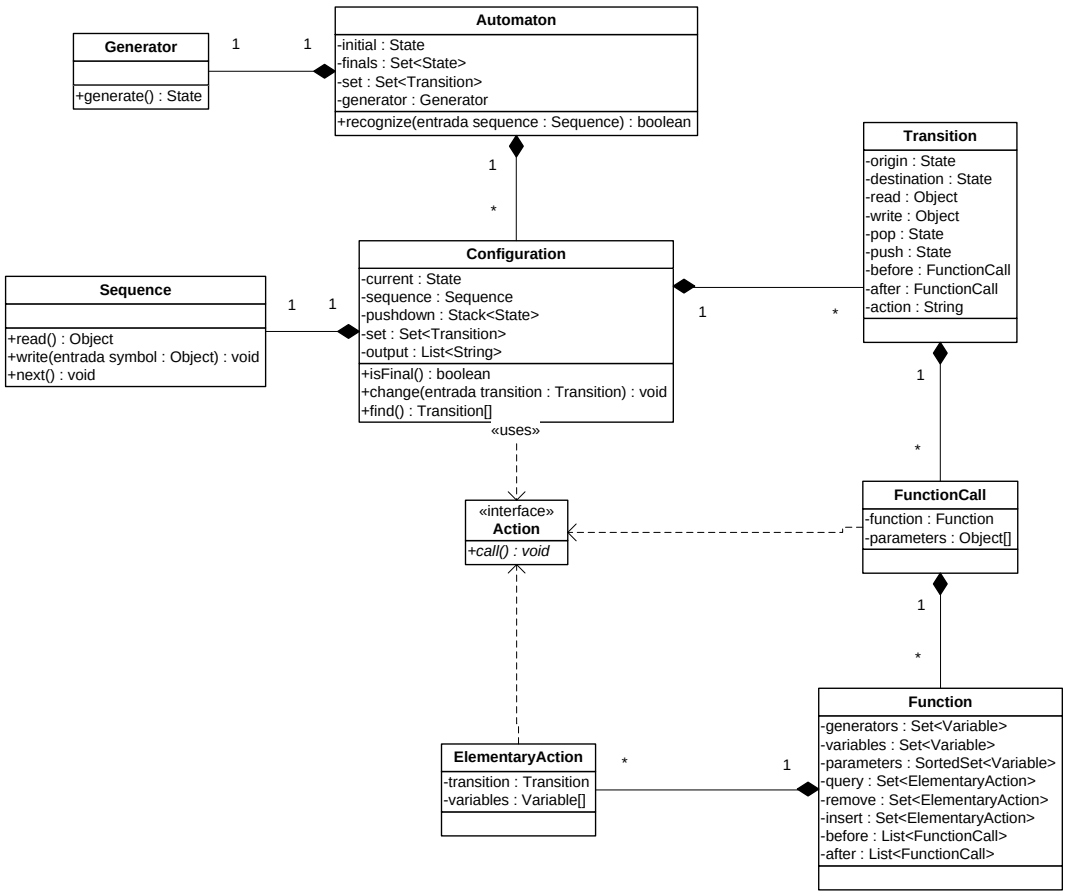

Figura 44 – Diagrama de classes do simulador de Autômatos Adaptativos

4.2.1 Automaton

A classe *Automaton* possui o método *recognize* que dispara o processo de reconhecimento de uma seqüência de símbolos e retorna o resultado na forma de um valor *boolean*. Durante o processo de reconhecimento, *Automaton* armazena um conjunto de configurações que são alteradas a cada passo da execução.

4.2.2 Configuration

A classe *Configuration* representa uma configuração do autômato durante o processo de execução. Em uma configuração são encontrados o estado corrente do autômato (*current*), a seqüência de símbolos analisada (*sequence*), a pilha onde são armazenados os estados de retorno (*pushdown*), o conjunto de transições (*set*) e a seqüência de saída onde são armazenadas as ações semânticas (*output*).

Os métodos da classe *Configuration* são utilizados durante o reconhecimento para verificar se a configuração é final (*isFinal*), buscar as transições que podem ser executadas a partir da configuração (*find*) e alterar a configuração (*change*).

4.2.3 Action

A classe *Action* representa ações adaptativas utilizadas para alterar o conjunto de transições de uma configuração do autômato. A execução da ação adaptativa é disparada pela chamada do método *call*.

4.2.4 Transition

A classe *Transition* representa uma transição dos autômatos adaptativos. Na transição são armazenadas utilizadas para alterar a estrutura de uma configuração do autômato. As informações armazenadas em uma transição são: estado de origem (*origin*), estado destino (*destination*), símbolo lido da cadeia (*read*), símbolo escrito na cadeia (*write*), estado desempilhado (*pop*), estado empilhado (*push*), chamada de função adaptativa anterior (*before*), chamada de função adaptativa posterior (*after*) e ação semântica (*action*).

4.2.5 FunctionCall

A classe *FunctionCall* representa um padrão para realizar uma chamada de função adaptativa. O padrão de chamada armazena a função adaptativa a ser chamada (*function*) e a lista de valores a serem associados aos parâmetros dessa função (*parameters*).

4.2.6 Function

A classe *Function* representa uma função adaptativa, utilizada para alterar o conjunto de transições do autômato. Uma função adaptativa armazena padrões para realizar chamadas de

outras funções adaptativas e padrões para realizar ações adaptativas elementares, além de um conjuntos de variáveis que indicam qual o significado das variáveis utilizadas na função.

Há duas listas de padrões para chamadas de funções, uma a ser executada antes das ações elementares (*before*) e outra a ser executada depois dessas ações (*after*) e três conjuntos de ações adaptativas elementares, para busca de transições sem remoção (*query*), busca de transições com remoção (*remove*) e inserção de transições (*insert*).

As variáveis podem ser armazenadas em três conjuntos: o conjunto de geradores (*generators*) que são definidos com um novo estado, o conjunto de variáveis (*variables*) que são resolvidas pelas ações elementares e o conjunto de parâmetros (*parameters*) que armazena as variáveis cujos valores são definidos na chamada da função.

4.2.7 ElementaryAction

A classe *ElementaryAction* representa padrões para a execução de ações adaptativas elementares. Um objeto dessa classe armazena uma transição que armazena os campos com valores conhecidos dessa ação (*transition*) e um array de variáveis que armazena as variáveis (*variables*) utilizadas para representar os campos cujos valores são desconhecidos antes da chamada da função adaptativa que contém essa ação.

4.2.8 Sequence

A classe *Sequence* representa uma seqüência de objetos, utilizada como cadeia de entrada para o autômato. Esse objeto permite o armazenamento de objetos de qualquer classe e possui métodos para leitura do símbolo apontado pelo cursor (*read*), escrita de símbolos (*write*) e movimentação do cursor para apontar o próximo símbolo da seqüência (*next*).

4.2.9 Generator

A classe *Generator* permite que sejam gerados novos estados para o autômato nas ações adaptativas elementares de inserção. Chamadas para o método *generate* geram os novos estados.

4.2.10 State

A classe *State* representa estados na estrutura dos autômatos adaptativos. Objetos dessa classe funcionam apenas para marcar o estado corrente do autômato e não possuem métodos que afetam o simulador. Seu único atributo é o nome, armazenado na forma de *String*.

4.2.11 Variable

A classe *Variable* representa variáveis utilizadas nos padrões de ações adaptativas elementares e nas funções adaptativas. Essa classe não possui métodos que afetem o funcionamento do simulador e seu único atributo é o nome, armazenado na forma de *String*.

```
finals ≔ conjunto vazio
queue ≔ fila vazia
configuration ≔ configuração inicial
step ≔ VERDADEIRO
accepted ≔ FALSO
enquanto step faça
    se |queue| ≠ 0 então
        configuration ← queue
    se configuration é aceita então
        accepted ≔ VERDADEIRO
        configuration → finals
        step ≔ ALL & |queue|≠ 0
    senão
        set ≔ conjunto de transições que podem ser executadas
        se |set| = 1 então
            altera configuration utilizando a transição encontrada
            se |queue|≠ 0 então
                configuration → queue
        senão
            se |set| > 1 então
                para transition em set faça
                    copy ≔ cópia de configuration
                    altera copy utilizando transition
                    copy → queue
            senão
                step ≔|queue|≠ 0
                configuration → finals
```

Figura 45 – Algoritmo para simulação dos Autômatos Adaptativos

4.3 Algoritmo

O algoritmo de simulação da execução dos autômatos é definido no método *recognize*. De forma simplificada, pode-se dizer que o algoritmo consiste em buscar, analisar e alterar configurações até que não haja mais novas configurações para serem analisadas. Enquanto a execução é determinística (só há uma transição que pode ser executada para a configuração) a configuração é alterada em função da transição, se a execução passa a ser não-determinística são criadas cópias da configuração que são alteradas e armazenadas para serem analisadas posteriormente.

O algoritmo é apresentado em pseudo-código na Figura 45. O conjunto *finals* armazena as configurações de aceitação e aquelas em que não há possibilidade de executar nenhuma transição. A fila *queue* armazena as configurações geradas a partir da execução de uma transição. A variável *configuration* armazena a próxima configuração a ser testada. As variáveis booleanas *step* e *accepted* indicam se deve haver um novo passo no processo de execução e se a cadeia de entrada foi aceita respectivamente.

Nessa descrição, aparece a variável ALL, não declarada no algoritmo. Essa variável funciona como um valor determinado na instanciação do autômato que determina se o algoritmo deve percorrer todos os caminhos mesmo depois de encontrar uma configuração que indica aceitação da cadeia de entrada. Seu valor deve ser VERDADEIRO para executar transduções não determinísticas em que há necessidade de conhecer mais de um resultado. Em outros casos seu valor pode ser FALSO e o algoritmo pára assim que reconhece uma configuração de aceitação.

5 Tradutor texto-voz

A partir do modelo apresentado no capítulo 3 foi criado um software de tradução texto-voz para textos escritos na língua portuguesa. O tradutor texto-voz é formado por um módulo que traduz o texto de entrada em uma seqüência de fonemas e um sintetizador de fala que gera o som a ser executado baseado na saída do tradutor.

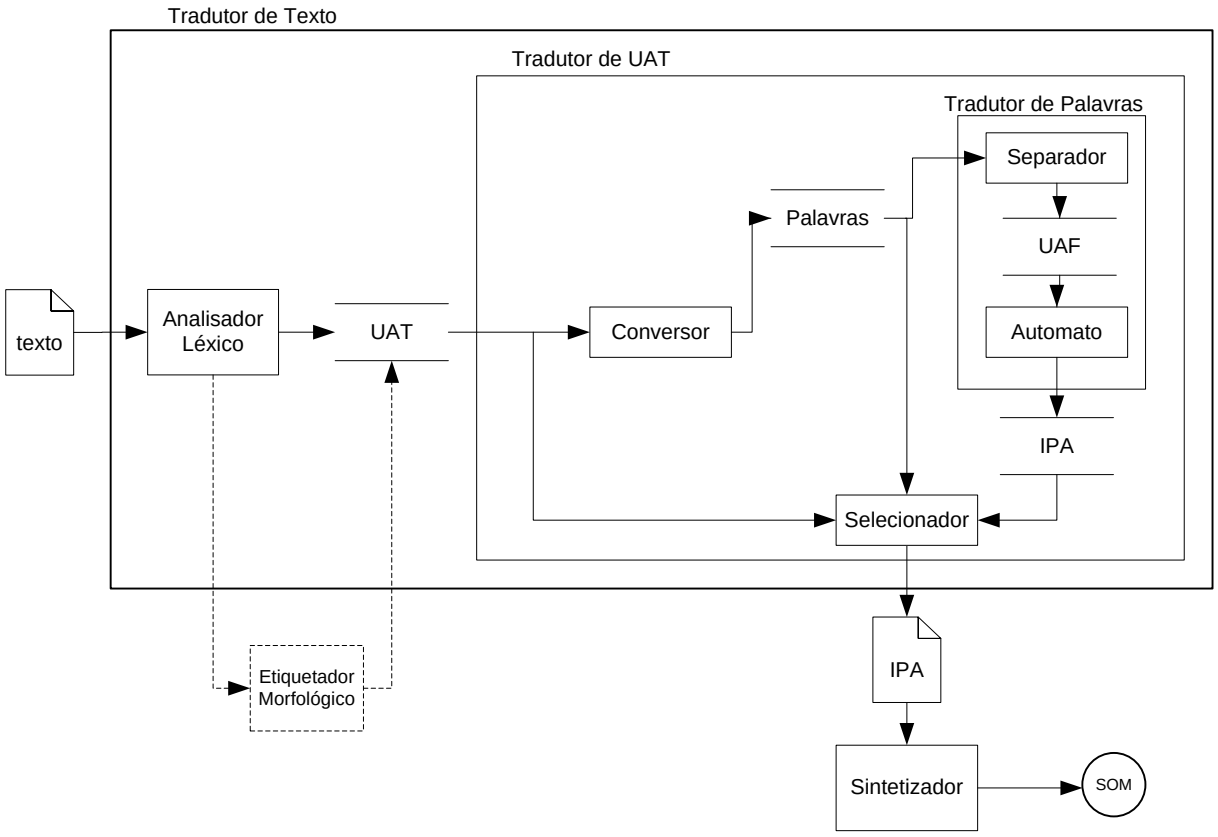

Figura 46 – Esquema de funcionamento do tradutor texto-voz

A Figura 46 ilustra o funcionamento do tradutor texto-voz dividindo-o em blocos. O arquivo de entrada é submetido ao Analisador Léxico que separa o texto em Unidades de Análise de Texto (UAT) e gera etiquetas para cada uma das UAT reconhecidas. O Analisador Léxico pode chamar um Etiquetador Morfológico externo que gera um segundo conjunto de etiquetas para auxiliar o processo de tradução grafema-fonema.

As UAT obtidas nessa primeira fase são submetidas ao tradutor de UAT. Ao entrar nesse bloco, a UAT passa por um processo de verificação que define se essa se encontra na forma textual e pode

ser traduzida pelo tradutor de palavras. Se a forma não é textual, a UAT é submetida a um conversor que a transforma em uma seqüência de palavras. As palavras são passadas ao tradutor de palavras baseado no modelo apresentado no capítulo 3.

A palavra, a etiqueta gerada pelo etiquetador morfológico (se houver), e o conjunto de traduções obtido para a palavra são submetidos a um bloco que escolhe uma das saídas como correta nos casos em que mais de uma representação é obtida. As representações escolhidas como corretas são armazenadas em um arquivo que é utilizado pelo sintetizador como entrada para o processo de síntese.

5.1 Tradutor de Texto

O tradutor de textos é o responsável pela tradução grafema-fonema de textos escritos na língua portuguesa. Nessa fase identificam-se as UAT, essas UAT são classificadas, definindo dessa forma o seu comportamento no texto, e por fim as UAT são traduzidas para a forma fonética levando em consideração a classificação obtida.

5.1.1 Analisador Léxico

O Analisador Léxico é responsável pelo primeiro processamento ao qual o texto de entrada é submetido. Esse módulo trasforma o texto em uma seqüência de UAT e gera um conjunto de etiquetas para cada uma das UAT reconhecidas. As etiquetas geradas por esse módulo são posteriormente utilizadas para definir o processo adequado de tradução da UAT para a forma fonética.

O funcionamento adequado desse bloco depende de uma definição adequada do que é uma UAT. Durante o desenvolvimento do software, notou-se que para o processo de tradução texto-voz baseado nessa arquitetura, uma UAT é um elemento de texto que possui uma forma padronizada de leitura[9]. Como resultado, deve-se definir o conjunto de UAT que o analisador reconhece e um método de tradução de cada tipo de UAT para sua forma fonética.

O conjunto de UAT existente é vasto, contém números, datas, siglas, e-mails, endereço de internet, CEP, etc. Eventualmente uma UAT pode ser representada de formas diferentes e possuir mais de um padrão de leitura. Por essa razão, foram definidos alguns padrões de escrita utilizados

[9] Em um texto, a estrutura “R$ 11 mil” é lida como o texto “onze mil reais”, logo essa estrutura é uma UAT.

com certa freqüência e de simples leitura para que a fala gerada não fosse interrompida com a perda desses elementos. Foram definidas etiquetas para representar as seguintes UAT:

- *Palavras*: seqüências de letras minúsculas iniciadas por uma maiúscula ou minúscula.
- *Números*: números nos formatos inteiros (ordinais e cardinais), decimal e porcentagem.
- *Datas*: datas no formato DD/MM/AAAA ou DD/MM/AA
- *Siglas*: seqüências de letras maiúsculas separadas por pontos ou seqüências de letras maiúsculas que não representem números romanos.
- *Moedas*: Representação de valores monetários em reais na forma "R$ ##,##".

5.1.2 Etiquetador Morfológico

O etiquetador morfológico é o responsável por gerar as etiquetas com a classificação morfológica das palavras encontradas no texto de entrada. Essa funcionalidade não é essencial ao funcionamento do tradutor, mas seu uso facilita o processo de desambiguação e escolha de sons para determinadas palavras homógrafas[10], melhorando o resultado final.

A versão do programa de tradução grafema-fonema apresentada utiliza as etiquetas do corpus Tycho Brahe (CHPTB), na tentativa de resolver os casos em que há múltiplos resultados para palavras com dúvidas geradas por *e* e *o* em posição tônica. Em uma análise simples, esses foram os casos identificados que permitiam melhoria de resultados a partir do resultado de um etiquetador morfológico.

O tradutor texto-voz permite a chamada de diferentes etiquetadores morfológicos, por meio de alterações do arquivo *tagger.bat*. O etiquetador utilizado deve ser capaz de receber um arquivo contendo um texto etiquetado e gerar como saída um arquivo contendo o mesmo texto etiquetado pelo conjunto de etiquetas do corpus Tycho Brahe.

[10] Palavras escritas de mesma forma e que tem sons diferentes dependendo do seu significado.

5.1.3 Tradutor de UAT

As UAT geradas pelo Analisador Léxico são utilizadas como entrada do tradutor de UAT. A função deste bloco é definir uma representação fonética para as UAT. Esse processo é dividido em três fases: inicialmente a UAT é transformada em uma seqüência de palavras, na segunda fase as UAT são submetidas ao tradutor de palavras que gera um conjunto de palavras e na terceira parte o conjunto de representações fonéticas obtido, a palavra e a etiqueta são passados por um método de escolha que define qual dos valores do conjunto obtido é o correto.

5.1.3.1 Conversores

Os conversores são objetos que realizam a tradução de UAT que não são formados exclusivamente de palavras em uma seqüência de palavras. Há um conversor para cada uma das classificações geradas pelo Analisador Léxico, exceto para palavas que não necessitam de conversão antes da tradução. Em muitos casos o processo de conversão gera mais de uma palavra, que são submetidas uma a uma ao tradutor.

5.1.3.2 Tradutor de Palavras

O tradutor de palavras é a implementação do modelo de tradutor grafema-fonema apresentado no capítulo 3. Assim como no modelo, o processo é dividido em duas fases: na primeira as palavras são separadas em seqüências de UAF enquanto na segunda fase essa seqüência é utilizada para gerar seqüências fonéticas que representem a palavra analisada.

a) Separador

A criação do *Separador* foi baseada no pacote de expressões regulares da linguagem JAVA – java.util.regex. Foi criado um padrão de expressões regulares que contempla as regras apresentadas no item **Erro! Fonte de referência não encontrada.**, exceto pela regra de eparação de vogais. Os resultados obtidos são passados por um método auxiliar que separa as vogais em função das regras definidas para a separação de vogais. As palavras são analisadas e as UAF reconhecidas são armazenadas (na forma String) em uma lista. Essa lista é saída do *Separador* e serve como entrada do módulo Autômato.

b) Autômato

Assim como ocorre no modelo do autômato, este módulo foi criado para receber a saída do *Separador* e gerar a seqüência de fonemas como sua saída. O funcionamento do módulo se divide nas fases de reconhecimento das UAF e tradução como ocorre no modelo, porém a estrutura utilizada para realizar esse processo foi alterada de modo a diminuir o tempo necessário para implementar o *Reconhecedor*.

O *Reconhecedor* foi substituído por um laço que lê as Strings da seqüência de entrada criada pelo *Separador*, cria as chamadas de funções adaptativas levando em consideração as características das Strings que representam as UAF e dispara essa seqüência de chamadas que altera o conjunto de transições do *Transdutor*. Quando ocorre a leitura do indicador de final de cadeia, prepara o *Transdutor* para execução, ligando os estados inicial e final ao resto da estrutura e dispara a execução desse autômato.

O *Transdutor* foi implementado como um autômato finito com capacidade de escrever símbolos na cadeia e com a capacidade de definir uma ação semântica para cada transição utilizada representada por uma String. Seu conjunto de transições é alterado no laço que representa o *Reconhecedor* e o processo de execução é disparado quando ocorre a leitura do símbolo de final de cadeia. Durante a execução, as representações fonéticas são geradas por meio da escrita de ações semânticas que representam a forma de leitura das UAF contidas na seqüência de entrada.

5.1.3.3 Selecionador

Esse módulo seleciona uma das representações fonéticas geradas pelo autômato adaptativo para ser armazenada no arquivo de saída, e ser utilizado posteriormente pelo sintetizador. A escolha de regras adequadas de seleção é fundamental para melhorar o resultado no processo de tradução grafema-fonema de textos.

A idéia de selecionar uma representação correta no conjunto é semelhante ao processo realizado pelo ser humano quando ele encontra palavras que geram ambigüidade em um texto. Esse processo é dificultado significativamente, pois há muitas palavras que não seguem regras e as regras existentes têm exceções.

Foram analisados dois conjuntos de regras de seleção. O primeiro baseia-se apenas na saída fonética gerada escolhendo a representação mais provável para o elemento que gerou dúvida, enquanto o segundo utiliza o resultado do etiquetador combinado a regras baseadas na estrutura da palavra. O objetivo dessas duas análises foi comparar o resultado obtido pelos dois métodos para verificar se vale a pena utilizar algum método adicional de seleção.

a) Mais Provável

As regras de seleção para escolha da sonoridade mais provável para uma letra levaram em consideração uma contagem realizada sobre a própria amostra que indicou que os sons mais prováveis eram:

1. [ɛ] para a letra *e* tônica.

2. [o] para a letra *o* tônica.

3. [ʃ] para a letra *x*.

b) Regras e Etiquetador

Para definir o som das letras *e* e *o* em posição tônica e da letra *x* em início de UAF, foram criadas regras baseadas na estrutura da palavra e da classificação morfológica obtida para a palavra. Apesar de haver diversas exceções, foi uma forma de melhorar os resultados. Foram utilizadas as seguintes regras:

1. O som da letra *x* foi definido como [z] depois de prefixo *e* (ex: *exato, exame*), [ʃ] depois de prefixos *en* (ex: *enxada, enxuto*) e *me* (ex: *mexicano, mexer*) e depois de ditongos decrescentes (ex: *caixa, frouxo*) e [s] em outros casos.

2. O som da letra *e* foi definido como [ɛ] exceto nos casos de verbos terminados em *erem, eram, esse, essem, era* (ex: *comerem, comeram,* etc) e pronomes e demonstrativos terminados em *ele* (ex: *ele, dele, aquele).*

3. Para o som de *o* considerou o som [o] exceto nos casos de verbos com final *a, e, o, em, am* (ex: *cobra, cobre,* etc*)* e palavras terminadas em *ol* (ex: *espanhol*).

5.2 Sintetizador

O sintetizador de fala realiza a leitura do arquivo gerado pelo tradutor grafema-fonema, que contém a representação fonética do texto de entrada, e gera uma saída sonora que representa o texto de entrada em sua forma falada. Nesse software, foi utilizado o sintetizador de fala desenvolvido por Koike et al. (2007). O sintetizador de fala foi incorporado ao software de tradução grafema-fonema baseado em autômatos adaptativos e apresentado em Shibata e Koike (2008).

O software desenvolvido nesse trabalho se baseia no processo de síntese por concatenação, utilizando sílabas como unidades de concatenação. Devido à quantidade de sílabas existentes, os arquivos de áudio são armazenados na forma MP3 com o objetivo de reduzir o tamanho do software.

Durante o processo de síntese de fala os arquivos são convertidos para o padrão WAV, que pode ser manipulado mais facilmente que os arquivos MP3. Este artifício é utilizado para alterar a entonação de uma unidade de fonação por meio da amplificação de seu sinal, gerando dessa forma a versão tônica da unidade de fonação

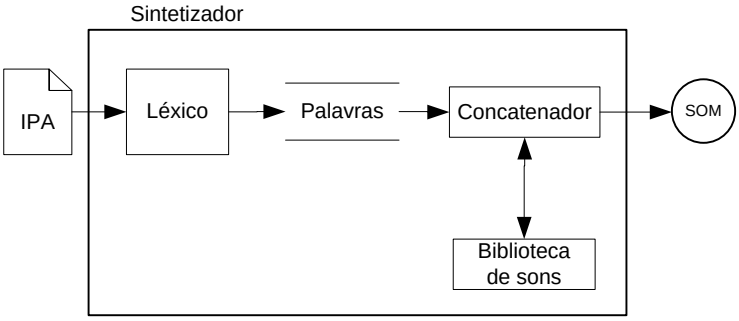

Figura 47 – Esquema do Sintetizador de Fala

A Figura 47 apresenta uma ilustração do funcionamento do sintetizador. O processamento do arquivo de entrada se inicia pelo Analisador Léxico que reconhece palavras e unidades de concatenação necessárias para criar o som que representa as palavras. As palavras são passadas ao bloco de concatenação, que obtém os arquivos que representam as unidades necessárias do inventário e utiliza-os para gerar o som que representa a palavra.

6 Testes e Resultados

Este capítulo apresenta os resultados obtidos no processo de tradução grafema-fonema para a língua portuguesa utilizando autômatos adaptativos. A avaliação foi dividida em duas partes: a primeira relacionada à tradução de palavras pelo autômato e a segunda à tradução de palavras pelo software. A seguir são descritas a metodologia utilizada para obter as amostras de teste e realizar os testes, os métodos escolhidos para classificar os resultados e os resultados obtidos para a tradução de palavras pelo autômato e pelo software.

6.1 Metodologia

Para analisar o resultado obtido nos processos de tradução grafema-fonema, foi necessário obter uma quantidade significativa de palavras da língua portuguesa de modo a retratar o máximo de características das palavras e o máximo de combinações dessas características. A amostra obtida foi submetida ao software, que gerou o conjunto de saídas e definiu uma das saídas como a mais provável. Os resultados foram analisados e classificados para serem posteriormente analisados.

6.1.1 Amostra

O inventário de palavras foi obtido a partir de textos jornalísticos obtidos da internet relacionados a cinco temas: política, esportes, tecnologia, economia e cultura. As palavras encontradas nesses textos foram todas submetidas ao processo de tradução e seu resultado foi analisado posteriormente.

A coleção de palavras obtidas inicialmente continha aproximadamente 9000 palavras. Dessas palavras, foram removidos todos os nomes de pessoas e as palavras estrangeiras que não são obrigadas a seguir as regras de tradução grafema-fonema da língua portuguesa além de palavras que continham erros de digitação. As palavras separadas por hífen, exceto nos casos de ênclise, foram separadas em duas palavras[11]. Após esse tratamento na coleção inicial foi obtida uma coleção de aproximadamente 7800 palavras, que foi usada para analisar os resultados da tradução de palavras pelo autômato.

Os mesmos textos foram submetidos ao processo de classificação morfológica pelo etiquetador VLMC Tagger e, nos casos em que a execução do etiquetador não foi falha, as palavras foram

[11] Não foram encontradas mesóclises na amostra, construção raramente utilizada na língua portuguesa do Brasil.

utilizadas para analisar a saída do software tradutor de palavras. A coleção obtida nesse caso, formada pelos pares de palavra e etiqueta, continha aproximadamente 9000 elementos.

6.1.2 Testes

O processo de testes foi dividido em duas fases, a primeira para testar a eficácia do modelo de autômato no processo de tradução de grafemas para um conjunto de representações fonéticas aceitas, e a segunda para testar a eficácia do método de tradução escolhido, escolhe um dos resultados do autômato supondo que essa seja a representação utilizada por uma pessoa lendo o texto de entrada.

A primeira fase do teste consistiu em gerar o conjunto de representações fonéticas aceitas para uma palavra utilizando o modelo de autômato definido para esta tarefa. Nesta fase, as palavras contidas na amostra foram submetidas ao software simulador de autômatos e foi gerado um conjunto de representações fonéticas para cada palavra analisada.

A segunda fase do teste consistiu em escolher uma das saídas obtidas no processo de tradução realizado pelo autômato, indicando qual será a saída gerada pelo tradutor de textos, da forma mais precisa possível. Na segunda fase, foram utilizados como entradas os pares formados por palavras e etiquetas e os resultados do autômato. Em função da etiqueta ou de características da palavra de entrada, uma das representações fonéticas foi escolhida como correta.

Para os testes realizados nessa fase, as etiquetas de classificação morfológica associadas às palavras foram obtidas utilizando o VLMC Tagger, um etiquetador morfológico baseado em cadeias de Markov de tamanho variável que utiliza o conjunto de etiquetas do corpus Tycho Brahe.

Para verificar o impacto que as regras utilizadas geram na escolha da forma correta, foi realizado um procedimento de testes semelhante, que escolhe como representação fonética a representação que contém a sonoridade mais provável para a letra em que a dúvida foi gerada. Neste caso, o etiquetador e características da palavra não foram levados em consideração.

6.2 Resultados do Autômato

Esta seção apresenta os resultados obtidos no processo de tradução grafema-fonema utilizando autômatos adaptativos. Os resultados são analisados em função do número de traduções obtidas, números de traduções esperadas e se as traduções obtidas são pertinentes. A primeira parte apresenta a classificação utilizada na análise dos resultados do autômato e a segunda apresenta os resultados obtidos para o processo de tradução grafema-fonema realizado pelo modelo de autômato adaptativo apresentado.

6.2.1 Classificação

Devido ao problema da existência de diferentes representações fonéticas para determinados grafemas, a classificação dos resultados obtidos ao submeter uma palavra ao tratamento do autômato deve levar em consideração, além do conjunto de saídas obtido, o conjunto de saídas válidas para a palavra de entrada e a relação entre os dois conjuntos.

Foram classificadas como *corretas* as palavras em que se obteve apenas uma tradução e essa tradução equivale à que seria utilizada por uma pessoa falando. Foram classificadas como *erro* as palavras em que foi gerado um conjunto de saídas que continha apenas traduções errôneas (que afetam significativamente a inteligibilidade da saída fonética).

Para a classificação do resultado obtido para as palavras que não se encontram em nenhum desses dois grupos foram criadas duas classificações. A classificação *falha* define que algum símbolo da palavra retornou apenas o resultado incorreto por limitação das regras utilizadas no autômato. A classificação *dúvida* define se a tradução de algum símbolo retornou um conjunto de sons, incluindo aquele que é considerado correto. As classificação *falha* e *dúvida* são divididas em sub-grupos indicando o que caracterizou a classificação. Há alguns casos em que uma palavra pode ser classificada em mais de uma dessas categorias.

O grupo *falha* foi dividido nas seguintes categorias: monossílabo (quando ocorre uma falha em um monossílabo tônico e sua saída é átona), ditongo gerado (um ditongo não esperado que foi gerado), ditongo não gerado (um ditongo esperado que não foi gerado), ditongo erro (quando há ditongo na palavra mas é diferente do que foi gerado), *a* átono pré-nasal (essa letra recebe o som

a quando ã era o esperado), *e* átono e *e* tônico (som [e] quando [ɛ] era o esperado), *o* átono e *o* tônico (som [o] quando [ɔ] era o esperado).

O grupo *dúvida* foi dividido nos seguintes grupos: e1 (quando a letra *e* em posição tônica gera dois sons e um é válido), e2 (quando a letra *e* em posição tônica gera dois sons e os dois são válidos), o1 (quando a letra *o* em posição tônica gera dois sons e um é válido), o2 (quando a letra *o* em posição tônica gera dois sons e os dois são válidos) e *x* (quando a letra *x* aparece na palavra).

Além disso, foram destacadas em um grupo chamado *distorção* algumas palavras cuja tradução (ou traduções) é correta segundo as regras, mas que podem ser faladas com algumas alterações por nativos da cidade de São Paulo.

6.2.2 Resultados

A apresentação dos resultados do autômato foi dividida em três tabelas. Nas três tabelas são apresentadas o número de ocorrências de palavras em cada classificação e exemplos de palavras que foram colocadas nessa classificação. As tabelas contendo a análise completa se encontram armazenada no CD incluso neste volume.

Tabela 12 – Ocorrências de palavras consideradas *falhas*

	Ocorrências	Exemplo	Representação Gerada	Representação Esperada
Monossílabo	2	Mas	[mɐs]	['mas]
Ditongo Gerado	84	Piorar	[pi̯o.'ɾar]	[pi.o.'ɾar]
Ditongo Omitido	50	Graduação	[grã.du̯a.'sɐ̃ʊ]	[grã.du.a.'sɐ̃ʊ]
Ditongo Incorreto	1	Circuito	[siɾ.'kui̯.tʊ]	[siɾ.'ku̯i.tʊ]
A átono pré-nasal	35	Camisa	[kɐ̃.'mi.zɐ]	[ka.'mi.zɐ]
E átono	11	Aeroporto	[a.e.ɾo.'poɾ.tʊ]	[a.ɛ.ɾo.'poɾ.tʊ]
O átono	2	Remotamente	[ɣe.mo.ta.'mẽⁿ.tʃɪ]	[ɣe.mɔ.ta.'mẽⁿ.tʃɪ]
E tônico	10	Vielas	[vi.'e.lɐs]	[vi.'ɛ.lɐs]
O tônico	18	Valiosos	[va.li.'o.zʊs]	[va.li.'ɔ.zʊs]
Total	213			

A Tabela 12 apresenta as ocorrências de palavras classificadas como *falhas*. Nessa tabela, a apresentação se encontra dividida por categorias que podem levar uma palavra a ser classificada como incorreta, o número de ocorrências de palavras em cada categoria e um exemplo de palavra

que se encaixa na categoria seguido da representação gerada pelo autômato e pela representação esperada. A Tabela 13 apresenta as palavras classificadas como *dúvida* seguindo o mesmo padrão utilizado na Tabela 12. Na última linha das duas tabelas são apresentados o número total de ocorrências de alterações que classifiquem como *falhas* ou *dúvida*s.

Tabela 13 – Ocorrências de palavras classificadas como *dúvida*

	Ocorrências	Exemplo	Representação Gerada	Representação Esperada
X	125	Vexame	[ve.'ksɐ̃.mɪ], [ve.'sɐ̃.mɪ], [ve.'zɐ̃.mɪ], [ve.'ʃɐ̃.mɪ]	[ve.'ʃɐ̃.mɪ]
E1	625	Vetos	['vɛ.tʊs], ['ve.tʊs]	['vɛ.tʊs]
E2	30	Sede	['se.ʤɪ], ['sɛ.ʤɪ]	['se.ʤɪ], ['sɛ.ʤɪ]
O1	517	Sucessor	[su.se.'soɾ], [su.se.'sɔɾ]	[su.se.'soɾ]
O2	34	Lobos	['lo.bʊs], ['lɔ.bʊs]	['lo.bʊs], ['lɔ.bʊs]
Total	1331			

A Tabela 14 apresenta os resultados obtidos para os testes do autômato. A tabela segue o mesmo padrão utilizado para a Tabela 12 e a Tabela 13, porém os exemplos das categorias *falha* e *dúvida* são omitidos, pois são apresentados nessas tabelas. O valor total é definido pela soma das classificações *correto*, *falha*, *dúvida*, *erro* e *distorção* subtraídas dos valores em que há duas ocorrências de alterações em uma mesma palavra.

Tabela 14 – Resultados dos testes para os autômatos adaptativos

Classificação	Ocorrências	Exemplo	Gerada	Esperada
Correto	6192	Casa	['ka.zɐ]	['ka.zɐ]
Falha	213		Vide Tabela 12	
Dúvida	1331		Vide Tabela 13	
Erro	10	Porque	['por.kɪ], ['pɔr.kɪ]	[por.'ke]
Distorção	72	Fortemente	[foɾ.te.'mɐ̃ⁿ.ʧɪ]	[foɾ.ʧɪ.'mɐ̃ⁿ.ʧɪ]
2 ocorrências	21	Manobra	[ma.'nɔ.brɐ], [ma.'no.brɐ]	[mɐ̃.'nɔ.brɐ]
Total	7797			

6.3 Resultados do Tradutor

Esta seção apresenta a forma de classificação e os resultados obtidos para o processo de tradução grafema-fonema de palavras realizado pelo conjunto formado pelo autômato e a lógica auxiliar de seleção.

6.3.1 Classificação

A classificação dos resultados obtidos pelo tradutor de textos é mais simples que a classificação utilizada para os resultados do autômato. Essencialmente, deve-se analisar se a tradução gerada para uma palavra é correta ou não levando em consideração o contexto.

Para isso foram considerados todos os pares de palavras e etiquetas obtidos e analisou-se a tradução gerada como saída para esses pares. O experimento foi realizado com duas regras para resolução de dúvidas: (1) escolhendo o valor mais provável e (2) analisando características das palavras e a etiqueta utilizada.

Na análise dos resultados do processo de tradução realizado pelo conjunto formado pelo modelo de autômato e o método de escolha, são considerados corretas as traduções dos pares em que: a tradução do autômato para a palavra foi *correta*, a tradução do autômato foi classificada como *dúvida* e o processo de seleção escolheu a representação correta e a tradução foi classificada como *falha* por falta de uma das representações esperadas, mas a representação gerada equivale à saída da palavra para a classe morfológica determinada pelo etiquetador.

Tabela 15 – Resolução de Dúvidas

Resultado	(1) Mais Provável	(2) Inferência
Correto	866	1094
Falhas	764	536
Não Resolvidas	6	6
Desprezadas	1	1

6.3.2 Resultados

A Tabela 15 apresenta o resultado dos dois processos utilizados para resolver as dúvidas geradas no processo de tradução utilizado pelo autômato. Para as palavras que receberam outras classificações, o resultado é igual independente do método de inferência. Foram analisadas 7235 pares cuja classificação é *correta* (desses 68 com distorção), 235 pares cuja classificação é *falha* (desses 233 continuaram incorretos e 2 foram resolvidos) e 12 pares cuja tradução foi definida como erro.

A Tabela 16 apresenta os resultados obtidos no processo de tradução. Foram classificadas como *corretas* aquelas palavras cuja representação fonética gerada foi equivalente à esperada, ou continha uma variação classificada como *distorção*[12]. As *incorretas* são aquelas que contêm alguma variação em relação à saída esperada (gerada no processo de tradução realizado pelo autômato ou causada por falhas no processo de decisão da saída correta nos casos de dúvida). O memorial de cálculo se apresenta no CD anexo a este volume.

Tabela 16 – Resultados dos Testes para o bloco tradutor

Classificação	(1) Mais Provável	(2) Inferência
Correto	8103 (89,04%)	8331 (91,55%)
Incorretas	997 (10,96%)	769 (8,45%)
Total	9100 (100%)	9100 (100%)

[12] Essa análise é pessimista, já que algumas palavras que tem duas saídas aceitas como corretas foram classificadas como incorretas. A palavra “camisa” utilizada como exemplo de “a átono pré nasal” é um exemplo disso.

7 Conclusão

Este trabalho apresenta os resultados obtidos durante o processo de pesquisa por um modelo de autômatos adaptativos para auxiliar o processo de tradução texto-voz de palavras da língua portuguesa.

Como resultados do trabalho foram obtidos um modelo de autômatos adaptativos para tradução grafema-fonema de palavras da língua portuguesa, um software para simulação de autômatos adaptativos desenvolvido sobre o paradigma da orientação a objetos e um software para tradução grafema-fonema de textos da língua portuguesa.

Em Koike et al. (2007), trabalho desenvolvido paralelamente foi criado um sintetizador de sons baseado na saída fonética gerada pelo tradutor grafema-fonema de textos desenvolvido neste trabalho.

7.1 Análise dos Resultados

Esta seção apresenta as conclusões obtidas a partir da análise dos resultados dos testes e com o aprendizado obtido sobre o processo de tradução grafema-fonema utilizando autômatos adaptativos.

7.1.1 Autômato Adaptativo

Neste trabalho foi criado um modelo de autômato capaz de tratar um conjunto significativo de características da língua portuguesa ao realizar a tradução grafema-fonema de suas palavras. Esse modelo explora a capacidade de tratar sensibilidade a contexto do dispositivo para definir a sílaba tônica da palavra e a sonoridade associada a cada letra em função da sua localização na palavra.

Os resultados obtidos para o autômato são bons, levando à representação fonética correta em 80% dos casos, gerando um conjunto de soluções que contém a solução correta em aproximadamente 15% dos casos e gerado conjunto de soluções que não continham a solução esperada nos 5% restantes.

7.1.1.1 Não Determinismo

Neste trabalho foram apresentados diversos casos que mostram que, devido à existência de palavras homógrafas com significados e sonoridades diferentes, um modelo de autômato

adaptativo determinístico não é suficiente para resolver a questão da tradução grafema-fonema para a língua portuguesa baseado apenas no grafema. O trabalho mostra como utilizar o não determinismo no modelo para gerar mais de uma representação quando necessário e se utiliza dessa capacidade para melhorar o resultado sobre um modelo determinístico.

O primeiro experimento considerando a solução mais provável, assim como o trabalho de Alfenas et al. (2004), equivale a uma solução determinística, enquanto o segundo experimento que usa o resultado do etiquetador morfológico se aproveita dos resultados do autômato não determinístico para melhorar os resultados.

7.1.1.2 Flexibilidade

O modelo apresentado representa características básicas da língua portuguesa (as regras para definir a sílaba tônica e as regras de influência que uma sílaba recebe da anterior e da posterior) no processo de escrita e leitura de símbolos. As características sonoras de cada UAF são definidas como parâmetros para as chamadas de função adaptativa.

Essa separação entre o conjunto de regras utilizado e as características associadas à UAF permite que o modelo seja utilizado para variações da língua portuguesa pela alteração das regras das características sonoras associadas a cada UAF, sem necessidade de alterar o resto do modelo. Eventualmente, outras línguas podem ser tratadas com alterações das regras de influências e tonicidade.

7.1.1.3 Completeza

O conjunto de regras que determina as características sonoras de uma UAF pode ser alterado, para satisfazer características sonoras não analisadas neste trabalho. Isso inclui a possibilidade de substituir regras escolhendo alguns sons em detrimento dos utilizados no decorrer deste trabalho, adicionar regras para obter conjuntos de resultados mais amplos ou remover regras para obter um conjunto de resultados menor.

Diminuir o conjunto de regras tende a gerar conjuntos de resultados menos amplos, podendo inclusive levar a modelos não determinísticos em que cada palavra traduzida gera um único resultado e não há necessidade de um método de escolha auxiliar. O aumento do conjunto de regras leva à criação de conjuntos de resultados maiores, podendo eventualmente conter a saída

correta para todas as palavras analisadas mas tendo como contraponto a necessidade de um método de seleção auxiliar mais preciso.

Como as regras que não foram adicionadas neste trabalho são relacionadas às diversas variações sonoras que uma UAF pode ter em posição átona pré-tônica, a adição dessas regras acarretaria no aumento de não determinismos e levaria a conjuntos de resultados maiores. Isso significa que as palavras em que a saída obtida não foi a esperada (*incorretos*, *inválidos* ou *distorção*) poderiam ter a saída correta como parte do conjunto obtido (*dúvida*). Por outro lado, palavras cujo resultado foi classificado como *correto* poderiam passar pelo mesmo processo e as palavras classificadas como *dúvida* teriam conjuntos de resultados ainda maiores.

Sabendo que 5% das palavras da amostra tiveram resultados diferentes dos esperados, e a **possível** melhora desse grupo acarretaria na queda dos resultados obtidos para os 95% restantes das palavras analisadas, optou-se por não implementar essas regras.

7.1.2 Tradução de Palavras

Neste trabalho foi utilizado um método de tradução baseado em um modelo de autômato adaptativo não determinístico. O autômato foi modelado de forma a gerar apenas uma saída nos casos em que isso foi possível ou nos casos em que foi considerado complexo o processo de resolução de dúvidas. Para os outros casos, o autômato gerou mais de uma saída e a decisão de qual dessas é a correta foi delegada a um método de seleção auxiliar.

7.1.2.1 Resolução de Dúvidas

Foram testados dois modelos simples para a solução de dúvidas: a primeira levando em consideração a sonoridade mais provável para a letra que gerou dúvida e a segunda utilizando características da palavra além de sua classificação morfológica determinada pelo etiquetador para definir a melhor solução quando possível e utilizando as maiores probabilidades quando nenhuma regra era encontrada.

7.1.3 Tradutor de Textos

Os resultados obtidos para o tradutor grafema-fonema de textos, apesar de não terem sido analisados formalmente, são satisfatórios. A saída sonora gerada pode ser entendida por nativos

da língua. A precisão desse tradutor depende de diversos fatores além da precisão do conjunto formado pelo autômato, etiquetador morfológico e pela lógica adicional para resolver dúvidas.

A tradução de um texto deve levar em consideração a tradução correta de estruturas mais complexas (como números, moedas e datas) que os seres humanos são capazes de ler com facilidade, mas tornam o processo de tradução extremamente complexo para um computador. Eventualmente podem aparecer estruturas que são híbridas (em “R$ 14 mil” o número aparece dividido entre número e texto), estruturas ambíguas (“XL” pode ser uma sigla ou um número romano), entre outras estruturas que podem causar problema.

Além disso, o resultado da tradução dessas estruturas na forma fonética e o resultado da tradução de palavras pelo autômato tende a ser independente, pois se há a necessidade de transformá-las na forma fonética é mais eficiente fazê-lo de forma direta do que transformá-las em seqüências de palavras para depois transformar essas seqüências em seqüências fonéticas.

Considerando essas razões, optou-se por não realizar nenhuma análise sobre o tradutor de textos e transformá-lo em uma ferramenta de aplicação desse modelo de autômatos adaptativos e do sintetizador texto-voz desenvolvido em Koike et al. (2007).

7.1.4 Metodologia para Trabalho com Autômatos

Outro resultado importante obtido durante o trabalho foi a criação de uma metodologia para utilização dos autômatos adaptativos, que permite uma implementação linear (vide Apêndice A) para análise de sensibilidade ao contexto. Apesar de haver diversos trabalhos com aplicações de autômatos adaptativos, não foram encontrados trabalhos que especifiquem metodologias para criação de autômatos e formas específicas de trabalhar com a dependência de contexto.

O método utilizado nesse trabalho cria blocos de transições que trabalham em conjunto para determinar a saída correta para a palavra analisada. Os blocos se comportam de forma semelhante quanto à leitura e escrita de símbolos sempre lendo apenas um símbolo vindo do bloco vizinho, gerando todos os outros símbolos necessários (para ele mesmo) e mais um símbolo para seu outro vizinho. O que diferencia os blocos são as ações semânticas geradas e o símbolo que passa para um dos vizinhos em função do símbolo que recebeu do outro viinho.

7.1.4.1 Sub-Máquinas

O autômato é dividido em duas sub-máquinas, uma responsável pela leitura da cadeia de entrada e alteração de uma segunda sub-máquina, que é executada assim que a cadeia de entrada se exaure e é capaz de executar a partir da cadeia vazia. Durante o processo de execução da segunda sub-máquina não há chamadas de funções adaptativas e todas as relações de dependência de contexto são definidas pela escrita de símbolos na cadeia (que serão lidos em um momento adequado).

7.1.4.2 Funções Adaptativas

As ações adaptativas realizam transformações pequenas, geralmente com uma busca ou remoção de uma transição e inserção de uma seqüência de transições entre os estados marcados por essa transição. As chamadas de função adaptativa posterior e anterior não são utilizadas, pois as funções são chamadas em seqüência após a leitura de cada símbolo da cadeia de entrada. As ações adaptativas utilizam-se dos símbolos de marcação para realizar a busca de estados, e podem se utilizar de estados de marcação utilizados para realizar a busca de transições.

7.2 Trabalhos Relacionados

O trabalho apresentado neste volume abre frente para novos trabalhos relacionados ao uso de autômatos adaptativos para a tradução grafema-fonema de textos escritos em linguagens naturais. A seguir são apresentados alguns exemplos de trabalhos que podem ser realizados utilizando a mesma temática.

7.2.1 Outras Linguagens

Pode ser analisada a viabilidade de uso de autômatos adaptativos para a tradução grafema-fonema em outras linguagens naturais, incluindo modelos semelhantes ao modelo de autômato apresentado neste trabalho (ainda que não seja comprovado, devido às características de construção do autômato, este parece ser mais adequado às línguas latinas que outras linguagens).

7.2.2 Variações do Português

Podem ser criados modelos de autômatos adaptativos para outras variações da língua portuguesa. Essencialmente, a estrutura de tratamento de tonicidade das sílabas, verificação de ditongos e

tritongos e influências de sílabas sobre as sílabas adjacentes tende a se manter o mesmo, exigindo alterações nos sons gerados para as sílabas.

7.2.3 Melhorias no Processo de Escolha

Pode-se desenvolver uma heurística mais precisa para a escolha da saída correta de palavras que permitem mais de uma interpretação fonética, utilizando outras classificações morfológicas além dos verbos e considerar o tempo verbal na análise. A análise etimológica também pode ajudar nesse processo, já que palavras derivadas tendem a ter características semelhantes às que as originaram. Isso resolve parte dos problemas ocorridos nos casos de *a* átonos pré-nasais e ditongos pré-tônicos anteriormente citados.

7.2.4 Análise de Relação Inter-UAF

Uma análise da relação entre as UAF e como uma UAF tende a afetar o som de outra pode melhorar o resultado do processo de tradução baseado em autômatos adaptativos. Neste trabalho, considerou-se que letras nasais posteriores às letras *e* e *o* tônicas tendem a fechar o som dessas letras. Apesar de não ser verdade sempre, a regra vale na grande maioria dos casos. Regras semelhantes podem ser criadas através da análise de pares de sílabas e verificando se existem UAF que alteram sistematicamente o som de outras UAF.

7.2.5 Tradutor Fonema-Grafema

O autômato desenvolvido pode ser utilizado como base para desenvolver um autômato adaptativo com a funcionalidade inversa, que receberia entradas fonéticas e geraria como saída os grafemas que geram tal entrada. Esse autômato pode ser posteriormente utilizado no processo de tradução de voz para texto se associado a um módulo que transforme sons em seqüências de fonemas.

7.3 Considerações Finais

Este trabalho apresenta uma evolução no estudo da tradução grafema-fonema na língua portuguesa utilizando autômatos adaptativos apresentado inicialmente como um exemplo do Adaptools e posteriormente estudado por Alfenas et al (2004). Neste trabalho o método utilizado por Alfenas et al (2004) foi melhorado e complementado utilizando uma metodologia padronizada para o tratamento da sensibilidade a contexto inerente às línguas naturais.

Como resultado final do trabalho foi gerado um modelo de autômato adaptativo capaz de realizar a tradução grafema-fonema de palavras da língua portuguesa, um software que utiliza esse modelo de autômato como base para realizar a tradução grafema-fonema de textos na língua portuguesa e um software capaz de simular autômatos adaptativos.

O trabalho abre novas frentes de estudo relacionadas à utilização dos autômatos adaptativos para o tratamento da língua portuguesa ou até mesmo de outras linguagens naturais, sendo que os resultados de alguns desses trabalhos podem contribuir para a melhoria do processo de tradução grafemas-fonemas de textos na língua portuguesa apresentado.

Os resultados obtidos no processo de tradução mostram que o uso de autômatos adaptativos para modelar a tradução grafema-fonema na língua portuguesa é viável, chegando ao resultado correto em aproximadamente 80% dos casos e obtendo conjuntos que contém a tradução correta em outros 15%. Nos casos restantes, a grande maioria não chegou a todos os resultados por limitação imposta às regras e em apenas oito casos foram encontradas traduções inválidas (em palavras que não seguem as regras ou monossílabos tônicos).

A combinação do autômato com métodos auxiliares para definir apenas uma saída, obteve precisão superior a 90% levando em consideração a classificação morfológica da palavra. Esse resultado é extremamente satisfatório e mostra que o método pode de fato ser utilizado para reduzir o dicionário de dados de um tradutor texto-voz.

REFERÊNCIAS BIBLIOGRAFICAS

ALFENAS, Daniel A.; CASTRO, Allan Jones B.; SHIBATA, Danilo P.; SOEJIMA, Henrique T. **Sintetizador Texto-Voz baseado em autômatos adaptativos.** 98p. Trabalho de Conclusão de Curso – Escola Politécnica, Universidade de São Paulo. São Paulo, 2004.

ALMEIDA JR., Jorge Rady. **STAD – Uma ferramenta para representação e simulação de sistemas através de Statecharts Adaptativos.** 205p. Tese de Doutorado – Escola Politécnica, Universidade de São Paulo. São Paulo, 1995.

BARBOSA, Plínio A.; ALBANO, Eleonora C. Brazilian Portuguese. Illustrations of the IPA. Journal of the International Phonetic Association, v. 34, n. 2, p. 227-232, 2004.

BARBOSA, Plínio A. et al. Aiuruetê: a high-quality concatenative text-to-speech sustem for Brazilian Portuguese with demisyllabic analysis-based units and a hierarchical model of rhythm production. In: EUROSPEECH, 1999, Budapeste, Hungria. **Proceedings of the Eurospeech '99.** Budapeste, Hungria: v. 5, p. 2059-2062, 1999.

BRILL, Eric. Unsupervised Learning of Disambiguation Rules Part of Speech Tagging. In: THIRD WORKSHOP ON VERY LARGE CORPORA. **Proceedings of the Third Workshop on Very Large Corpora.** EUA: p. 1-13, 1995.

CIPRO NETO, Pasquale; INFANTE, Ulisses. **Gramática da Língua Portuguesa**. 1ª Edição. São Paulo: Editora Scipione, 1997. 583p.

ETIQUETADOR Morfológico Online para português brasileiro (QTAG). Disponível em QTAG <http://www2.lael.pucsp.br/corpora/etiquetagem/index.html>. Acesso em 21 ago. 2006.

GAMALLO: Home Page. Disponível em <http://gramatica.usc.es/~gamallo/index.html>. Acesso em: 21 ago. 2006;

IWAI, Margarete K. **Um formalismo gramatical adaptativo para linguagens dependentes de contexto.** 190p. Tese de Doutorado – Escola Politécnica, Universidade de São Paulo. São Paulo, 2000.

HIPA – IPA. **Handbook of the International Phonetic Association:** A Guide to the Use of the International Phonetics Alphabet. EUA: Cambrige University Press, 1999. 214p.

IPA – Internacional Phonetic Association. Disponível em <http://www.arts.gla.ac.uk/ipa/>. Acesso em 12 de Novembro de 2007.

JOSÉ NETO, João. **Introdução à Compilação.** Rio de Janeiro: LTC, 1987. 222p.

JOSÉ NETO, João. **Contribuição à Metodologia de Construção de Compiladores.** 130p. Tese de Livre Docência – Escola Politécnica, Universidade de São Paulo. São Paulo, 1993.

JOSÉ NETO, João. Adaptive Automata for Context -Sensitive Languages. SIGPLAN NOTICES, v. 29, n. 9, p. 115-124, 1994.

JOSÉ NETO, João. Adaptive Rule-Driven Devices - General Formulation and Case Study. In: CONFERENCE ON IMPLEMENTATION AND APPLICATION OF AUTOMATA, 6., 2001, Pretória, África do Sul. **Implementation and Application of Automata, 6th International Conference**. Lecture Notes in Computer Science, 2002, v. 2494, p. 234-250.

JOSÉ NETO, João; MORAES, Miryam de. Formalismo adaptativo aplicado ao reconhecimento de linguagem natural. In: CONFERENCIA IBEROAMERICANA EN SISTEMAS, CIBERNÉTICA E INFORMÁTICA, 2002, Orlando, EUA. **Anais da conferencia iberoamericana en sistemas, cibernática e informática.**

JOSÉ NETO, João. Autômatos em Engenharia de Computação - uma Visão Unificada. In: PRIMERA SEMANA DE CIENCIA Y TECNOLOGÍA DE LA SOCIEDAD CHOTANA DE CIENCIAS Y LA RED MUNDIAL DE CIENTÍFICOS PERUANOS, 2003, Ciudade de Chota, Perú.

KEPLER, Fábio N. **Um etiquetador morfo-sintático baseado em Cadeias de Markov de tamanho variável.** 58p. Dissertação de Mestrado – Instituto de Matemática e Estatística, Univesidade de São Paulo. São Paulo, 2005.

KOIKE, Fabio R.; OLIVEIRA, Luiz R. E.; SANTOS, Ricardo H. S.; RIZZO Tiago M. **Síntese de Fala com apoio de Técnicas Adaptativas.** 57p. Trabalho de Conclusão de Curso – Faculdade Engenheiro Celso Daniel, Centro Universitário Fundação Santo André. Santo André, 2007.

KOTELLY, Blade. **The Art and Business of Speech Recognition**: Creating the Nobel Voice. Boston, EUA: Addison-Wesley, 2003. 184p.

LEMMETTY, Sami. **Review of Speech Synthesis Technology.** 104p. M. Sc. Thesis – Laboratory of Acoustics and Audio Signal Processing, Hensinki University of Technology. Helsinque, Finlândia, 1999.

LEWIS, Harry R.; PAPADIMITRIOU, Christos H. **Elementos da Teoria da Computação.** Tradução de Edson Furmankiewicz. 2ª Edição. Porto Alegre: Bookman. 2000. 344p.

LIBRAS. Disponível em: <www.libras.org.br>. Acesso em 18 ago. 2006.

MANNING, Chris; SCHÜTZE, Hinrich. **Foundations of Statistical Natural Language Processing**. Cambridge, EUA: MIT Press, 1999. 620 p.

MENEZES, Carlos Eduardo D. de. **Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos.** 114p. Tese de Doutorado – Escola Politécnica, Universidade de São Paulo. São Paulo, 2000.

MENEZES, Carlos Eduardo D. de; JOSÉ NETO, João. Um método híbrido para a construção de etiquetadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos. In:

CONFERENCIA IBEROAMERICANA EN SISTEMAS, CIBERNÉTICA E INFORMÁTICA, 2002, Orlando, EUA. **Anais da conferencia iberoamericana en sistemas, cibernática e informática.**

PISTORI, Hemerson. **Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.** Edição Revisada. 2003, 174 p. Tese de Doutorado – Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

PISTORI, Hemerson; JOSÉ NETO, João. **AdapTools: Aspectos de Implementação e Utilização.** Boletim Técnico PCS – Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

PISTORI, Hemerson, JOSÉ NETO, João. An Experiment on Handshape Sign Recognition using Adaptive Technology: Preliminary Results. In: XVII BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE – SBIA'04. São Luis, 2004.

RATNAPARKHI, Adwait. A Maximum Entropy Model for Part-of-Speech Tagging. In: CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING. **Proceedings of the Conference on Empirical Methods in Natural Language Processing**. Somerset, EUA: Association for Computational Linguistics, 1996. p. 133-142.

ROCHA, Ricardo L. A. Tecnologia Adaptativa Aplicada ao Processamento Computacional de Língua Natural. Revista IEEE Latin America, v. 5, n. 7, p. 544-551, 2007.

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

SCHROEDER, M. A Brief History of Synthetic Speech. Speech Communication, v. 13, n. 1, p. 231-237, 1993.

SILVA, Adelaide H. P; ALBANO, Eleonora. Brazilian Portuguese Rhotics and the Phonetics/Phonology Boundary. **Proceedings ICPhS'99**. San Francisco, EUA: University of California at Berkley, p. 2211-2214, 1999.

SHIBATA, Danilo P.; KOIKE, Fábio R. Tradutor Texto-Voz baseado em Autômatos Adaptativos. In: WORKSHOP DE TECNOLOGIAS ADAPTATIVAS (WTA), 2., 2008, São Paulo.

SHIBATA, Danilo P.; ROCHA, Ricardo L. A. An Adaptive Automata based method to improve the output of text-to-speech translators, In: Congress of Logic Applied to Technology, 6., 2007, Santos. 1 CD-ROM.

SHUTT, J. N. **Recursive Adaptable Grammar.** 140p. M. Sc. Thesis – Computer Science Departament, Worcester Polytecnic Institute. Massachusetts, EUA, 1993.

TETSCHNER, W. **Voice Processing**. 2ª Edição. EUA: Artech House Boston-London. 1993. 511p.

CHPTB – Corpus Histórico do Português Tycho Brahe. Disponível em: <http://www.ime.usp.br/~tycho/corpus/>. Acesso em 28 de Janeiro de 2008.

VISL – Tree Structure. Disponível em: <http://visl.sdu.dk/>. Acesso em 21 ago. 2006.

WEINGESSEL, Andreas. **Speech Recognition.** 110p. Diplomarbeit – Technischen Universität Wien. Viena, Áustria, 1994.

WHITLEY M. S. Rhotic representation: problems and proposals. Journal of the International Phonetic Association, v. 33, n. 1, p. 81-86, 2003.

ZUFFO, F., PISTORI, H. **Tecnologia Adaptativa e Síntese de Voz: Primeiros Experimentos.** In: Workshop de Software Livre, 5., Porto Alegre, 2004. **Anais do V Workshop de Software Livre.**

GLOSSÁRIO

Bloco de Influência Anterior (Bloco-α): bloco de transições utilizadas quando uma UAF recebe uma influência anterior específica.

Bloco de Influência Posterior (Bloco-π): bloco de transições utilizadas quando uma UAF recebe uma influência posterior específica.

Elo: estrutura formada por transições do *Transdutor* que representa uma UAF.

Estado Inicial: primeiro estado criado na estrutura de um elo. A transição de marcação do elo sempre aponta dois estados iniciais de elos adjacentes, do último elo criado e do elo que está sendo criado.

Influência Anterior: influência que uma UAF recebe da UAF anterior e gera para a UAF posterior. Representadas por símbolos α.

Influência Posterior: influência que uma UAF recebe da UAF posterior e gera para a UAF anterior. Representadas por símbolos π.

Ponto de Entrada: local de referência em que as novas transições (geradas por ações elementares de inserção de uma função adaptativa) devem ser posicionadas. O ponto de entrada é delimitado por dois estados e são encontrados por meio de busca de transições de marcação que apontam esses estados como origem e destino.

Regra de Tonicidade: uma regra característica da UAF, utilizada para ler a sua tonicidade e definir a tonicidade da UAF anterior.

Regra Padrão: regra para o consumo de um símbolo de influência (anterior ou posterior) em que o símbolo utilizado é o coringa, e não há alteração sobre o som original da letra.

Transição Padrão: transição que representa a regra padrão em um elo.

Transição de Marcação/Referência: transição utilizada como referência para definir o ponto de entrada de uma função.

Unidade de Análise Fonética (UAF): entidade representada por uma seqüência de letras utilizada como base para a análise fonética por meio dos autômatos adaptativos. O *Separador* transforma uma palavra em uma seqüência de UAF, e o autômato traduz essa seqüência em uma seqüência de símbolos fonéticos.

Unidade de Análise de Texto (UAT): entidade de texto utilizada para análise fonética, tipicamente tem um padrão de leitura (ex: palavras, números, etc).

APÊNDICE A

Neste apêndice apresentamos um estudo sobre a complexidade computacional do método utilizado. O estudo apresentado é dividido em duas partes em que são definidos o fator de crescimento do espaço ocupado em memória pelo autômato e o fator de crescimento do tempo de execução da tradução de uma seqüência, ambos em função do tamanho da seqüência analisada.

Memória

O processo de análise da memória utilizada pelo autômato é bastante simples. A cada UAF reconhecida há a execução de uma ação adaptativa que transforma a estrutura do *Transdutor*. As alterações geradas por essas ações adaptativas dependem do tipo de UAF, mas independem da posição da UAF na cadeia de entrada, ou seja, a análise de uma determinada UAF **sempre** gera o mesmo número de transições no autômato.

Sabendo o número de transições gerados por uma UAF independe da posição da mesma na cadeia, podemos dizer que há uma UAF que gera um número máximo de transições determinado pela constante ΔP. A partir da constante P_0, que indica as transições existentes na configuração inicial e as criadas durantes os ajustes previos à execução do *Transdutor*, podemos dizer que o número de transições após o reconhecimento da i-ésima UAF é limitada superiormente por:

$$|P(i)| = i \cdot \Delta P + P_0$$

Podemos dizer então, que para uma cadeia de tamanho n o número de transições criadas é limitado superiormente pelo número de transições após a leitura do último símbolo:

$$|P(n)| = n \cdot \Delta P + P_0$$

Com raciocínio análogo podemos verificar que o crescimento do número de estados segue o mesmo processo, logo, após a leitura do último símbolo da cadeia temos o número de estados limitado superiormente por:

$$|Q(n)| = n \cdot \Delta Q + Q_0$$

Como esses dois conjuntos são os únicos alterados durante as ações adaptativas, podemos dizer que o espaço ocupado em memória cresce linearmente tendo por base o tamanho da cadeia de entrada analisada.

$$Memória(n) \in O(n)$$

Tempo de Execução

O tempo de execução do sistema depende do tempo necessário para alterar a configuração do autômato utilizando uma transição ($t_T(i)$) e do tempo necessário para executar alterações na estrutura após o reconhecimento de uma UAF ($t_A(i)$). O tempo gasto para realizar as alterações de configurações, desconsiderando as ações adaptativas associadas, é constante:

$$t_T(i) = T_T$$

O tempo necessário para realizar alterações na estrutura do autômato, por sua vez, depende das alterações que são realizadas. Supondo um conjunto simples, baseado em uma lista ordenada por ordem de inserção dos elementos, podemos dizer o tempo de execução das ações elementares cresce linearmente. Nos casos das buscas e remoções, devido à necessidade de iterar sobre o conjunto para verificar quais transições satisfazem o padrão, e no caso das inserções, devido à necessidade de iterar sobre o conjunto para verificar se a nova transição não se encontra no conjunto. Dessa forma, podemos dizer que o tempo necessário para executar uma ação adaptativa associada ao reconhecimento de uma UAF é limitado superiormente por:

$$t_A(i) = k_1 \cdot |P(i)| + k_0$$

Podemos verificar uma parcela dependente do tamanho do conjunto (k_1) e uma parcela constante (k_0) associada a tarefas como a criação de novos estados, associação de valores a variáveis e chamadas das ações adaptativas cujo tempo de execução é constante. As constantes determinam fatores relacionados ao tempo gasto por cada uma das tarefas e o número de vezes que cada uma delas é executada. Com a manipulação dos valores verificamos que o tempo de uma ação é proporcional à posição da UAF.

$$t_A(i) = \underbrace{(k_1 \cdot \Delta P)}_{\Delta A} \cdot i + \underbrace{(k_1 \cdot P_0 + k_0)}_{A_0}$$

$$t_A(i) = \Delta A \cdot i + A_0$$

As transições de reconhecimento de UAF possuem uma ação adaptativa associada, logo o tempo necessário para executar uma dessas ações adapatativas é limitado superiormente por:

$$t_{UAF}(i) = T_T + t_A(i)$$

O tempo de execução do *Reconhecedor* é dado pela somatória do tempo de execução das transições que reconhecimento de UAF, somado a uma parcela constante referente à transição de final de cadeia e sua ação adaptativa e da execução da transição de chamada para a sub-máquina *Transdutor*. O tempo de execução do *Reconhecedor* é limitado superiormente por:

$$t_{Reconhecedor}(n) = k_{R_0} + \sum_{i=1}^{n} t_{UAF}(i) = k_{R_0} + \sum_{i=1}^{n} \left(T_T + t_A(i)\right)$$

$$= k_{R_0} + n \cdot T_T + \sum_{i=1}^{n} (\Delta A \cdot i + A_0) = k_{R_0} + n \cdot (T_T + A_0) + \Delta A \cdot \sum_{i=1}^{n} i$$

$$= k_{R_0} + n \cdot (T_T + A_0) + \Delta A \cdot \frac{n \cdot (n+1)}{2} = k_{R_0} + \left(T_T + A_0 + \frac{\Delta A}{2}\right) \cdot n + \frac{\Delta A}{2} \cdot n^2$$

Analisando cuidadosamente as funções adaptativas utilizadas, percebemos que as transições utilizadas nas ações elementares de busca e remoção são sempre as *transições de marcação* e as novas transições criadas pelas ações elementares de inserção podem ser criadas de modo a nunca repetir uma transição já existente. Sendo assim, não há necessidade de iterar o conjunto para buscar as transições nem para verificar a existência de uma transição antes de adicioná-la ao conjunto. Isso torna o tempo de execução das ações adaptativas constante na prática, o que por sua vez torna o tempo de execução do *Reconhecedor* linear.

$$t_A(i) = T_A$$

$$t_{UAF}(i) = T_{UAF} = T_T + T_A$$

$$t_{Reconhecedor}(n) = k_{R_0} + \sum_{i=1}^{n} t_{UAF}(i) = k_{R_0} + \sum_{i=1}^{n} T_{UAF} = k_{R_0} + T_{UAF} \cdot n$$

Durante a execução do *Transdutor*, para gerar uma representação fonética, para cada elo são executadas três transições na ida (regras de tonicidade), quatro ou cinco (quando há dois sons) na volta além das transições inicial, final e de inversão de sentido. O tempo do execução do *Transdutor* é limitado superiormente por:

$$t_{Transdutor}(n) = 3 \cdot T_T + \sum_{i=1}^{n} (3 \cdot T_T + 5 \cdot T_T) = 3 \cdot T_T + 8 \cdot T_T \cdot n$$

Considerando os não determinismos que há no processo de volta, e que o número máximo de não determinismos por elo é N, a transição final pode ser executada N^n vezes, e as transições de volta N^i vezes para o i-ésimo elo. Considerando essas alterações, o tempo de execução do *Transdutor* passa a ser limitado superiormente por:

$$\begin{aligned} t_{Transdutor}(n) &= (2 + N^n) \cdot T_T + \sum_{i=1}^{n} \left(3 \cdot T_T + 5 \cdot T_T \cdot N^i\right) \\ &= 2 \cdot T_T + T_T \cdot N^n + 3 \cdot T_T \cdot n + 5 \cdot T_T \cdot \sum_{i=1}^{n} N^i \\ &= 2 \cdot T_T + 3 \cdot T_T \cdot n + T_T \cdot N^n + 5 \cdot T_T \cdot \frac{N(N^n - 1)}{N - 1} \\ &= \left(1 - \frac{5 \cdot N}{N - 1}\right) \cdot T_T + 3 \cdot T_T \cdot n + \left(1 + \frac{5 \cdot N}{N - 1}\right) \cdot N^n \end{aligned}$$

Com essa análise podemos concluir que tempo de execução do *Reconhecedor* aumenta quadraticamente em relação ao tamanho da cadeia sem nenhum artifício para aumentar a velocidade das ações adaptativas elementares. Utilizando uma implementação específica para esse modelo o tempo de execução das ações elementares pode tornar-se constante, o que torna o tempo de execução do *Reconhecedor* linear.

$$t_{Reconhecedor}(n) \in \begin{cases} O(n^2), & \text{implementação genérica} \\ O(n), & \text{implementação específica} \end{cases}$$

O tempo de execução do *Transdutor* depende do número de não determinismos utilizados no processo de execução. Seu crescimento é linear quando a execução do autômato é determinística, mas varia exponencialmente quando há não determinismos na execução.

$$t_{Transdutor}(n) = \begin{cases} O(k^n), & \text{não determinístico} \\ O(n), & \text{determinístico} \end{cases}$$

Aplicação do Estudo ao Tradutor-Grafema Fonema

Para o autômato definido neste trabalho, o número máximo de não determinismos em um elo é oito, mas na prática isso só ocorre em um dos elos (elo tônico com vogal *e* ou *o* iniciado em *x*). Nos outros elos o número de não determinismos não pode passar de quatro (iniciados em *x*), portanto podemos dizer que na pior das hipóteses a execução do *Transdutor* é proporcional a 4^n.

Além disso, o número de UAF de uma palavra não costuma ser muito grande[13] e a letra *x* que gera os não determinismos não é muito freqüente na língua portuguesa. Isso faz com que grande parte das palavras gere apenas uma saída (na amostra mais de 80%), e das restantes a maioria gere duas, quatro ou oito representações fonéticas.

[13] A palavra *inconstitucionalissimamente*, listada comumente como a maior da língua portuguesa tem 11 UAF. Com esse número de Unidades de Análise Fonética, é possível gerar 8388608 representações fonéticas no pior caso. Apenas uma representação fonética é gerada para essa palavra.