Incorporating a Feature Tree Geometry into a Matcher for a Speech Recognizer

by Aaron Maldonado

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
~~May 1999~~
[June 1999]

Author ....
Department of Electrical Engineering and Computer Science
May 1999

Certified by ....
Professor Kenneth N. Stevens
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Incorporating a Feature Tree Geometry Into a Matcher for a Speech Recognizer
by Aaron Maldonado

Submitted to the
Department of Electrical Engineering and Computer Science

May 1999

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Abstract: The goal of this thesis is to incorporate a feature tree geometry into an existing matcher for a lexical access system. Proposed by Professors Kenneth N. Stevens and Samuel Jay Keyser in their paper *Feature Geometry and the Vocal Tract*, the feature tree geometry is a hierarchical structure for representing the features of a phoneme. The four components involved in incorporating the feature tree geometry into the lexical access system are examined individually: the development of a computational model, the invocation of rules to account for assimilation in running speech, the production of morphemic stems, in particular, plurals, and the process of word retrieval. The last component is not implemented with the feature tree, but the implications of a tree-based matcher are explored. Initially, a brief overview of the lexical access system is presented for the reader's background. Each component is discussed first from a theoretical standpoint and then its implementation is presented and discussed. This paper concludes with a comparison of the original matcher's performance to the performance of the matcher based on the feature tree geometry. Performance results indicate that the computation time of the matcher fitted with the feature tree geometry decreased to 40% of the original matcher's execution time. Overall, this paper represents an initial probe into using alternative lexical representations in a lexical access system to obtain a desired level of performance.

Thesis Supervisor: Kenneth N. Stevens
Title: Clarence Joseph LeBel Professor of Electrical Engineering

To my Lord and Savior, Jesus Christ,
who is my refuge and strength. Words
cannot express my gratitude, for with you all
things are truly possible.

To my family for their love, prayers, and
support throughout my five years of education
at MIT. Your counsel is a well-spring of life.

To Ken for being a kind, supportive, and inspiring
mentor and teacher. Such a combination of knowledge
and humility is rare at MIT.

To Elizabeth for her tremendous advice and support.
Thank-you for taking time to answer so many questions.
Your labeling tutorial is a work of art, and I can assure
you that your future is sprinkled with success.

To the Sunday Night staff and congregation at Park
Street Church for two and a half years of encouragement
and inspiration. Danny and Gordon, thank-you so much. May
God continue to bless each and every one of you.

To my family away from home, Chi Alpha. To Mike O.,
Liz, Melissa, Tony, Keith, Andre, Andres, Rojo, Ryota, Ted,
David, and Dedric--each day God is glorified in you.
Thank-you for your love and support.

To all my friends, and brothers and sisters in the Christian
Community and abroad at MIT, thank-you for your encouragement
and prayers. It is my prayer that you would continue to bear fruit,
and shine like the stars God has made you to be.

To my academic advisor William T. Peake, thank-you for
encouragement and wisdom. Your commitment, dedication, and
refusal to let prior academic performance affect my confidence
in my ability to succeed at MIT are greatly appreciated.

To everyone around the lab for your help and
encouragement without which this project could not
be possible. Thank-you, Arlene, for making the transition
from student to research assistant a lot easier.

TABLE OF CONTENTS

I Introduction

Lexical access refers to the process of retrieving words from a lexicon to recreate a spoken utterance. In the system proposed here, the fundamental unit of recognition is the segment, an element derived from a speech signal which represents an underlying phoneme and which is identified by a unique set of binary features. The lexicon consists of a series of words with each word having a pronunciation made up of a string of phonemes. The ability of a system to access the lexicon and hypothesize an output in an efficient manner is dependent on the unit of lexical representation, or, in this case, the representation of a phoneme.[1] This raises an important question: how do you represent a phoneme such that the process of lexical access is performed in an efficient manner?

The features associated with a phoneme have traditionally been represented as an array, or matrix, of feature values.[2] While it is possible to implement a system for lexical access using a matrix of features, there is evidence which suggests that alternative lexical representations may provide computational benefits and more closely resemble the way listeners extract information from speech.[3] One such lexical representation is the feature tree geometry proposed by Keyser and Stevens (1994). Psychological evidence has suggested that lexical items stored in memory are represented by a distinct set of hierarchical features, thus echoing the need for a tree-type structure.[4] Furthermore, linguists have often used tree geometries to account for a variety of speech phenomena, such as co-referentiality and syllable structure.[5]

The system proposed here for lexical access involves two levels of processing: a front-end signal processor, and a back-end matcher. The task of the signal processor can be subdivided into three components: landmark detection, feature detection, and conversion. Given a speech signal, the signal processor must identify landmarks in the signal corresponding to vowels, glides, and consonants. Once the landmarks have been identified, values are assigned to a set of binary features for each landmark. The assignment of a plus or minus to a feature is based on the presence of certain acoustic cues around a particular landmark. The final step is to convert the landmarks into segments, which can then be used as input to the matcher.

Matching involves three tasks: initialization, applying rules to expand the lexicon, and feature matching to produce word candidates. The initialization phase of the matching process consists of several procedures for reading in data, representing phonemes in terms of the feature tree geometry, and expanding the lexicon with morphemes. After initialization, the lexicon can be further expanded by applying a set of tree-based rules to the input sentence. These rules account for variations in word pronunciation that might have occurred as a result of speech modifications. Once the lexicon has been fully expanded, the feature trees are converted back into feature matrices and a feature-by-feature match of the segments from the input sentence is performed against the phonemic representation of each word in the lexicon.

This project demonstrates how one might incorporate a feature tree geometry into a matcher for a lexical access system. On a higher level, it provides a framework for how various lexical representations might be used in parallel to produce an efficient system for lexical access. It is motivated by observations that the feature tree geometry proposed by Keyser and Stevens can, in certain instances, produce results more efficiently than a matrix of features.

II Background

2.1 Lexical Access Project

The lexical access project is an ongoing endeavor of the Speech Communications Group at MIT. The aim of this project is to create a knowledge-based, rule-governed speech recognition system which is modeled after the way humans produce speech. Unlike modern speech recognition systems which rely primarily on statistical analysis and complex markov models for detection, this system relies on a set of acoustic cues in speech which provide information about the articulatory movements in the vocal tract.[6]

Figure 2.1 is a block diagram of the system. First, the landmarks are identified; that is, the time where a landmark is thought to be present is recorded and preliminary information about the type of sound is obtained. Landmarks can either be classified as vowel, glide, or consonant. Vowels are produced with no narrowing of the vocal tract and have spectral energy contained in frequency bands called formants. The time associated with a vowel landmark is marked near the place where the amplitude of the first formant (F1) is a maximum.[7] The first column of table 2.1 lists the vowels in English. Diphthongs, such as /ai/, /au/, and /oi/, exhibit both vowel and glide properties. In this case, the first half of the landmark is marked as a vowel and the portion near the end of the sound is marked as a glide.

Landmarks for glides are chosen at times where the signal amplitude is a minimum.[7] Since glides are produced with an intermediate narrowing of the vocal tract, they do not exhibit abrupt changes in spectrum from a preceding vowel. The four English glides are listed in the second column of table 2.1.

Consonants are produced with an extreme narrowing or a complete closure of the vocal tract which is then released. This closing and opening of the vocal tract produces abrupt spectral

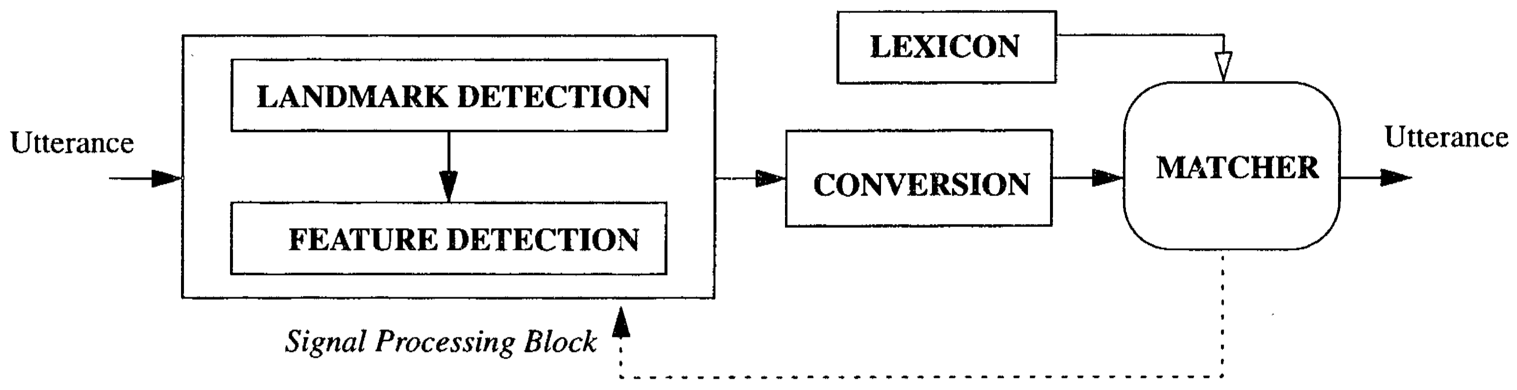

FIGURE 2.1 Block diagram of system.

changes resulting in a pair of boundaries for the consonant. Each boundary is marked with a separate time. There are four acoustically different types of consonants: affricate, fricative, sonorant, and stop consonants. See columns three through six of table 2.1.

Once a landmark has been detected, further processing is done in the vicinity of the landmark to identify its underlying features. (See section 2.2.2 for a discussion of features.) Then, the detected landmarks are converted into segments. While a converter has not been implemented yet, the issues involved in the process of conversion have been addressed in detail in Zhang (1998).[8] According to Zhang, the converter: 1) must be able to combine multiple landmarks which imply the same segment, 2) account for missing landmarks as a result of deletion in a CC environment, and 3) know how and when to utilize non-landmark information such as voicing onset time.

When the converter has finished transforming landmarks into segments, the output is sent to the matcher. After a period of initialization, the matcher attempts to recreate what was said by applying assimilation rules to the phonemic representation of words in the lexicon and comparing the result against the output of the converter. The output of the matcher consists of complete sentences, a series of words which matched the input from beginning to end, and incomplete sentences, the words which matched the input until a mismatch occurred.

The dotted line from the matcher to the signal processing block is a feedback path. Feeding back information from a preliminary matching can help improve the detection stage especially in impoverished speech. This feedback line has not yet been implemented.

Vowels	Glides	Consonants			
		affricates	fricatives	sonorants	stops
/iy/ beat	/h/ hat	/ch/ church	/f/ far	/l/ life	/b/ bug
/ih/ bit	/w/ wait	/dj/ judge	/dh/ then	/m/ man	/d/ dog
/ey/ bait	/y/ yell		/s/ sink	/n/ new	/g/ get
/eh/ bet	/r/ red		/sh/ should	/ng/ thing	/k/ cat
/ae/ bat			/th/ think		/p/ pat
/aa/ father			/v/ vile		/t/ tar
/ao/ bought			/z/ zoo		
/ow/ go			/zh/ garage		
/ah/ but					
/uh/ foot					
/uw/ boot					
/rr/ bird					
/er/ mother					
/ex/ about					

Table 2.1 A listing of the English vowels, glides and consonants recognized by our system.

2.2 Basic Terminology of Lexical Access

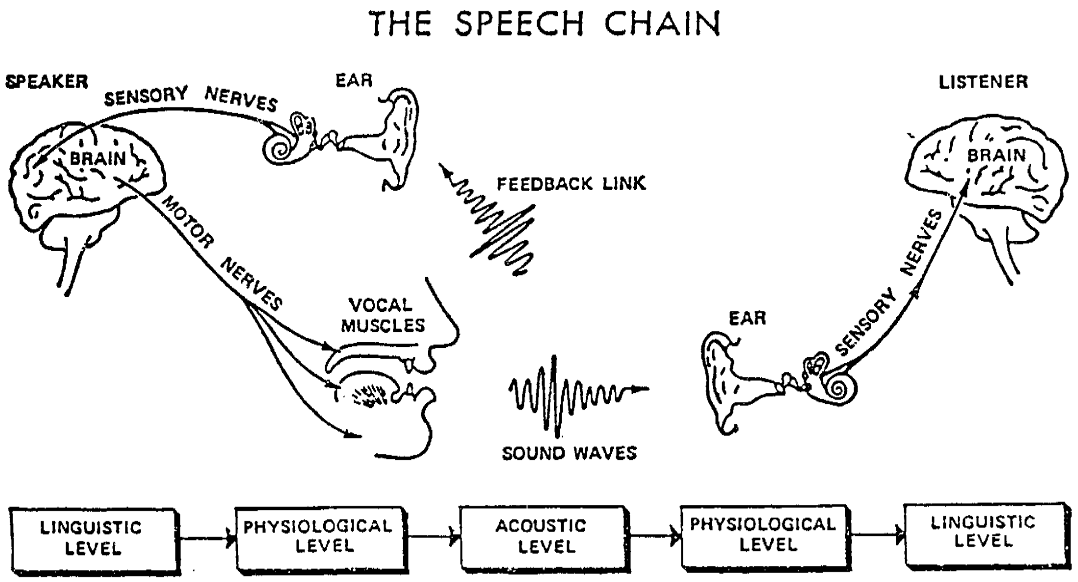

Figure 2.2 The speech chain.

According to the "speech chain" model of speech perception and production (figure 2.2), a speaker first arranges his thoughts and puts them into linguistic form.[9] This consists of accessing words and phrases stored in memory and arranging them in the correct order. As the brain performs this process, it sends impulses along motor nerves which are connected to the lips, tongue, and other articulators of the vocal tract. Propagation of these impulses to the various articulators causes them to move and causes sound sources to be generated from the modulation of airflow. The result is the radiation of sound.

The propagating sound wave travels to the ear of the listener, where it is received and processed by a complex filtering system. Upon reception, sensory nerves connected to the cochlea are activated and the information is sent to the brain where it is parsed into words. The information is then stored in memory, where it can be accessed at a later time.

Evidence has shown that at the linguistic level of speech perception and production there is a more fundamental unit of representation than words, the phoneme.[9] Phonemes are the units of speech which serve to distinguish one utterance from another in a language. In our system, phonemes refer to the lexically specified units of speech which are the underlying form of segments, bundles of features acquired from a speech signal. The segment is the fundamental unit of recognition. Utterances are recreated by accessing a lexicon which consists of words represented by a string of phonemes.

2.2.1 Anatomy of the Vocal Tract

The various phonemes produced by a speaker are a result of the manipulation of vocal tract articulators. There are six articulators in the vocal tract: the larynx, the lips, the soft palate, the tongue blade, the tongue body, and the vocal folds. Of these six there are three that are often specified as primary articulators (the lips, the tongue blade, and the tongue body). They are the ones over which a person expresses cognizant influence during the speech process. Figure 2.3 is a diagram of the human vocal tract; it is divided into four functional regions.

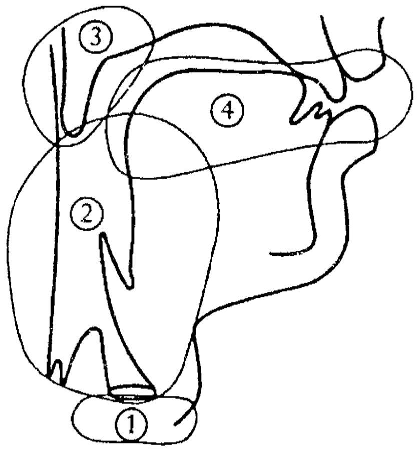

Figure 2.3 Segmenting the vocal tract.

Region 1 corresponds to the vocal folds. The vocal folds are two membranes that extend from the thyroid cartilage at the front of the neck to the arytenoid cartilages at the back. During voiced speech (speech characterized by vocal fold vibration), the arytenoid cartilages are pressed together resulting in a narrowing of the space between the vocal folds, known as the glottis. Closure of the glottis results in a cessation of airflow into the oral cavity and a build up of air pressure behind the vocal fold constriction. When the subglottal pressure exceeds the supraglottal pressure, the vocal folds are forced open and a puff of air is released. This flow of air causes the vocal folds to vibrate in a relaxation oscillation. Once the air is released, the change in pressure causes the vibrating vocal folds to close and the process is repeated. Thus, the vocal folds periodically interrupt the normal flow of air from the lungs to the mouth by producing a sequence of air puffs whose frequency is determined by the rate at which glottis is opened and closed.[10]

Region 2 contains the laryngeal and pharyngeal pathways leading to the oral cavity. The larynx is a cartilaginous structure in which many ligaments are connected, including the vocal folds. It is not rigid and can move vertically during speech production and in swallowing. The pharynx is a tube-like orifice connected to the apex of the larynx and it extends to the base of the oral cavity leading into the mouth. During the production of certain sounds as in /iy/, the walls of the pharynx expand. Manipulation of the cross-sectional area within the pharyngeal region contributes to acoustic filtering of the sound source.[10]

The portion of the vocal tract marked region 3 corresponds to the soft palate (or velum). The soft palate is a flap of tissue which can be lowered or raised. When the velum is lowered, air coming from the lungs through the pharynx can escape through the nasal cavity. If the velum is raised, the route through the nasal cavity is closed and the expelled air is directed along the mouth. Nasal sounds are produced by lowering the soft palate so that the pharyngeal airways are coupled with the nasal cavities.

The lips, the tongue blade, and the tongue body are the articulators which comprise region 4. The lips can be protruded, rounded and spread. Each of these actions has an influence on the acoustic properties of the output. Since the lower lip is connected to the mandible, the lips are displaced whenever the jaw is lowered or raised. The tongue is a muscular mass of tissue which is raised, lowered, moved forward, and moved backward during speech production. Often the tongue is raised against the roof of the mouth to form closures in the oral cavity behind which pressure is built up or to form narrow constrictions which generate turbulent noise sources. Although the tongue body and the tongue blade are anatomically connected, they can be manipulated independently of one another to a certain degree.

2.2.2 *Discussion of Features*

Features are used to describe segments that are implemented by the speaker. Each feature has a corresponding set of acoustic correlates.[11] Table 2.2 lists all of the features our lexical access system currently uses and their possible values. If there is acoustic evidence for the presence of a particular feature, it is assigned a value of plus (+). A minus (-) can be assigned to a feature for one of the following reasons: 1) to emphasize the absence of a feature, like the nasal feature in the English /r/ and /l/, or 2) to represent the opposite meaning of a plus feature designation. The context of a minus sign is made clear by the feature to which it is attached. For example, a [- back] vowel is the same as a front vowel and [- round] means "without lip rounding" but [-low] does not imply [+high] tongue body position.

There are three distinctive categories of features. The first category, the *articulator-free* features, relays information about the type of sound and the manner of articulation without reference to a particular articulator.[7] The articulator-free features, also known as manner features, make up the first two rows of table 2.2. *vowel*, *glide*, and *consonant* are features which distinguish between the possible types of phonemes. Only one of these features can be attributed to a phoneme and its value is always positive. The remaining manner features are binary and only apply to consonants.

Manner features serve to distinguish between the different types of consonants as described in table 2.3. The feature *sonorant* indicates whether or not there is a build up of pressure behind a constriction in the oral cavity. A [+ sonorant] consonant means that there is no pressure build up and is characterized by a continuation of low frequency energy at its closure; [-sonorant] consonants show a decrease in low frequency energy into the closure. [- continuant] indicates that there is a complete closure in the oral cavity and manifests itself in the spectrum as an abrupt decrease or increase of high frequency energy; [+ continuant] means that there is only a partial closure in the vocal tract and it is characterized by a continuantion of high frequency energy throughout the closure. The feature *strident* applies to fricatives which exhibit exceptionally strong frication noise.

Articulator-Free	Articulator	Articulator-Bound	Values
vowel			+
glide			+
consonant			+
sonorant			+/-
continuant			+/-
strident			+/-

Table 2.2 A list of the features used in lexical access and their possible values.

Articulator-Free	Articulator	Articulator-Bound	Values
	vocal folds	stiff slack	+/- +/-
	glottis	spread constricted	+/- +/-
	pharynx	advanced tongue root constricted tongue root	+/- +/-
	soft palate	nasal	+/-
	body	high low back	+/- +/- +/-
	blade	anterior distributed lateral rhotic	+/- +/- +/- +/-
	lips	round	+/-

Table 2.2 A list of the features used in lexical access and their possible values

Consonant Type	Sonorant	Strident	Continuant
affricate	-		+ and -
fricative	-	+ or -	+
sonorant	+		-
stop	-		-

Table 2.3 A listing of the manner features which distinguish the four consonant types.

Articulator features, the second category, identify which articulator is used to produce a sound. They include *blade*, *body*, *lips*, *pharynx*, and *soft palate*. The first three features correspond to the primary articulators of consonants and specify whether a sound was made with the tongue blade, tongue body, or lips respectively. *pharynx* and *soft palate* are referred to as secondary articulators; that is, they play an active role in the production of the sound but they are not the primary place of articulation. The [+ pharynx] feature is used to describe differences between tense and lax sounds. [+ soft palate] is used to indicate that the soft palate was lowered, as in nasal consonants. While secondary places of articulation can be marked with a + or -, their effect on articulation is often indicated by marking their associated articulator-bound features.[7]

Articulator-bound features are the third distinctive feature category. These features specifically describe how an articulator is used to produce a sound. The articulator-bound features associated with the tongue blade are *anterior*, *distributed*, *lateral*, and *rhotic*. [+ anterior] specifies that the tongue tip makes contact with the anterior portion of the alveolar ridge. [+ distributed] means that a broad portion of the tongue makes contact with the alveolar ridge. [+ lateral] applies to /l/ and implies that the tongue is situated in the oral cavity such that air can flow around the sides of the tongue. [+ rhotic] applies to /r/ and is equivalent to [+ lateral] but with a different shaping of the tongue blade.

Features associated with the tongue body are *high*, *low*, and *back*. These three features describe the placement of the tongue body in the oral cavity. While *high* and *low* are a measure of vertical displacement, the *back* feature expresses horizontal displacement. [- high; - low] means an intermediate vertical height and [- back] means a fronted tongue body position.

Other articulator-bound distinctions include [+ nasal] for the nasal consonants, [+ round] for consonants made with rounded lips, *advanced tongue root* and *constricted tongue root* for tense/lax sounds, *constricted glottis* and *spread glottis* for sounds made with the glottis like /h/, and *stiff vocal folds* and *slack vocal folds* for voiced and unvoiced consonants respectively.

2.2.3 Feature Tree Geometry

Historically, the phonemes have been represented as a matrix of features. For example, the phoneme /t/ is represented as a matrix of features in table 2.4. Some linguists have challenged this phonological representation, citing that features are hierarchical and should therefore be represented in a tree structure. In their paper *Feature Geometry and the Vocal Tract*, professors Samuel Jay Keyser and Kenneth N. Stevens present a feature tree geometry which defines a hierarchy of features based on the anatomy of the vocal tract.

Figure 2.3 is the basis for the feature tree geometry of Keyser and Stevens. This figure demonstrates how the vocal tract can be partitioned into four regions of independent activity. Regions 2,3, and 4 correspond to supraglottal structures whose positions can be manipulated by the contraction of arrays of muscles.[12] Articulators in regions 2, 3, and 4 form constrictions in the vocal tract which provide acoustic filtering to sound sources and also serve to generate sources in the vicinity of the constrictions. Region 1 corresponds to the vocal folds, which can be slackened and stiffened by the adjustment of intrinsic laryngeal muscles. Stiffening and slackening of the vocal folds changes the fundamental frequency of vocal-fold vibration and can serve to inhibit or facilitate glottal vibration for obstruent consonants, but control of vocal-fold stiffness never serves to create a constriction in the airway.[12] The fact that the primary role of the vocal folds is not to form constrictions differentiates region 1 from regions 2, 3, and 4. A tree which expresses this distinction is shown in figure 2.4.

symbol	/t/
consonant sonorant continuant	+ - -
stiff vocal folds slack vocal folds	+ -
blade	+
anterior distributed	+ -

Table 2.4 Matrix representation of the segment /t/.

Region 4 can be sub-divided according to the articulators in the oral cavity: the lips, the tongue body, and the tongue blade. Since the tongue blade and the tongue body are connected, their movements are not completely independent. The lips, on the other hand, can be manipulated independently of the tongue. The region 4 node can therefore be extended with two sub-nodes, one representing the lips and one representing the tongue (referred to as the *lingual* node). The lingual node can then be sub-divided into two branches for the tongue blade and the tongue body. In a similar fashion, regions 1, 2 and 3 can be represented according to their respective articulators. Figure 2.5 shows the trees for all four regions.

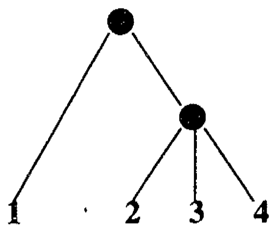

Figure 2.4 Tree structure showing distinctive regions.

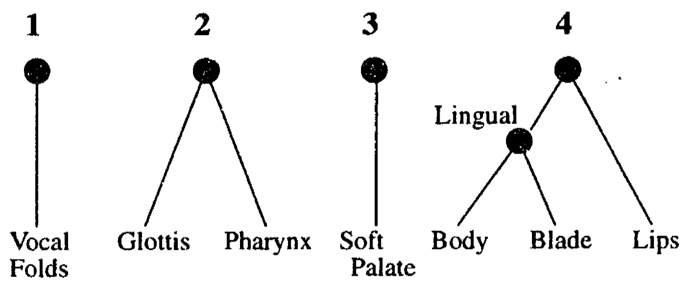

Figure 2.5 Tree structures representing regions 1, 2, 3, and 4.

The different regions can be combined to form a composite tree as shown in figure 2.6. At the top of the hierarchy are the *vowel*, *glide*, and *consonant* nodes. When a tree is constructed for a phoneme, only one of these nodes is specified as the dominant node. The dominant node is marked with an open circle while the remaining two classifier nodes are represented by dark circles. Inherently, the feature tree is a picture of the vocal tract showing the progression from the vocal folds through the pharyngeal airways into the oral cavity and terminating at the lips.

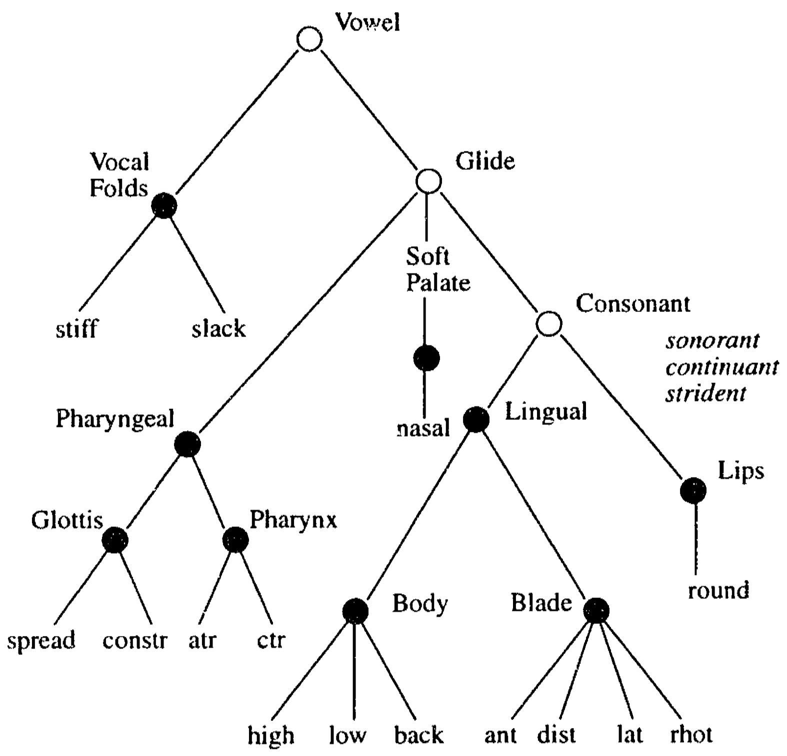

Figure 2.6 Composite tree with terminal features.

Attached to the end of every node at the base of the tree are the features associated with the various articulators. The primary articulator for a segment is specified by adding a plus or minus to the features underneath a primary articulator node (blade, body, or lips). Secondary articulators are specified in the same manner. The articulator-free features are not attached to any node because they are not associated with an articulator. Since the manner features apply only to consonants, they are placed beside the consonant node. Figure 2.7 is an example of a composite tree for the features listed in table 2.4.

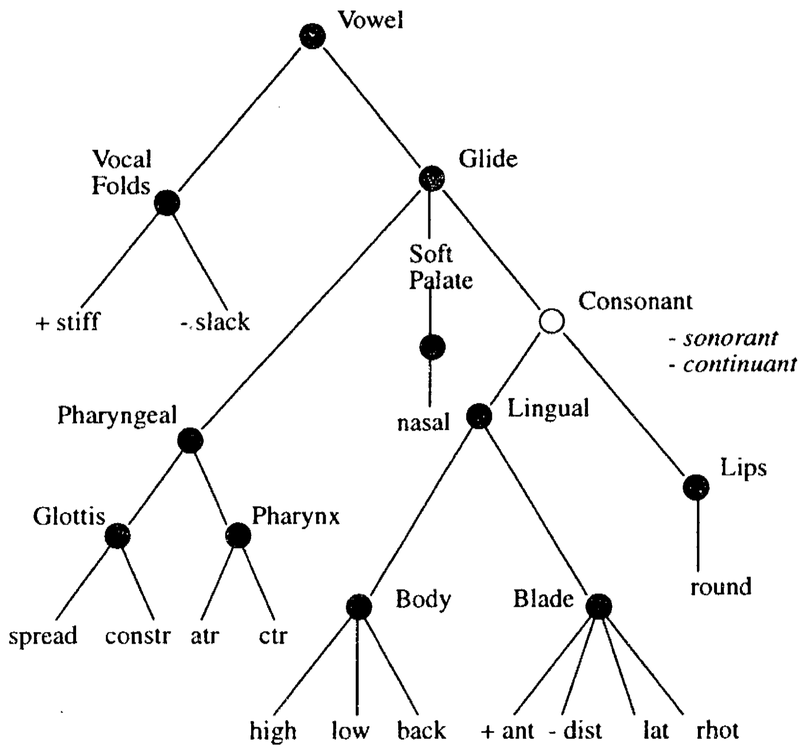

Figure 2.7 Composite tree for /t/ showing dominant node and distinctive features from table 2.4.

2.3 Overview of Original Matcher

Once landmarks and their associated features have been identified from the acoustic information in the signal, they need to be matched against the lexicon taking into account possible linguistic modifications introduced by the speaker. The matcher, which is called *matchG3*, takes as input the segments from the acoustic analysis, a file containing a series of rules which are to be applied during the matching process, a file containing a list of the standard phonemes from table 2.1 and their corresponding features, and a lexicon consisting of words and pronunciations.[8]

The matcher is written in *C++*. This language is appropriate for the task of accessing the lexicon because the lexicon, the rule set, and the input sentence can be represented in terms of classes which inherit from one another. There are ten classes which form the foundation for the matching process, and they are represented in figures 2.8 and 2.9. In these figures, inner boxes represent the recipient in a *has-a* relationship (i.e. class _segment *has a* class _bundle); arrows represent inheritance, or an *is-a* relationship, with the class being pointed to inheriting from the class at the origin. The classes _pronunciation, _sentence, and _rule all inherit from the _segments class, which *has* an array of elements of class _segment. This hierarchy echoes the idea that the segment is the fundamental unit of recognition; it is the smallest and most distinctive unit of speech from which all utterances are constructed.

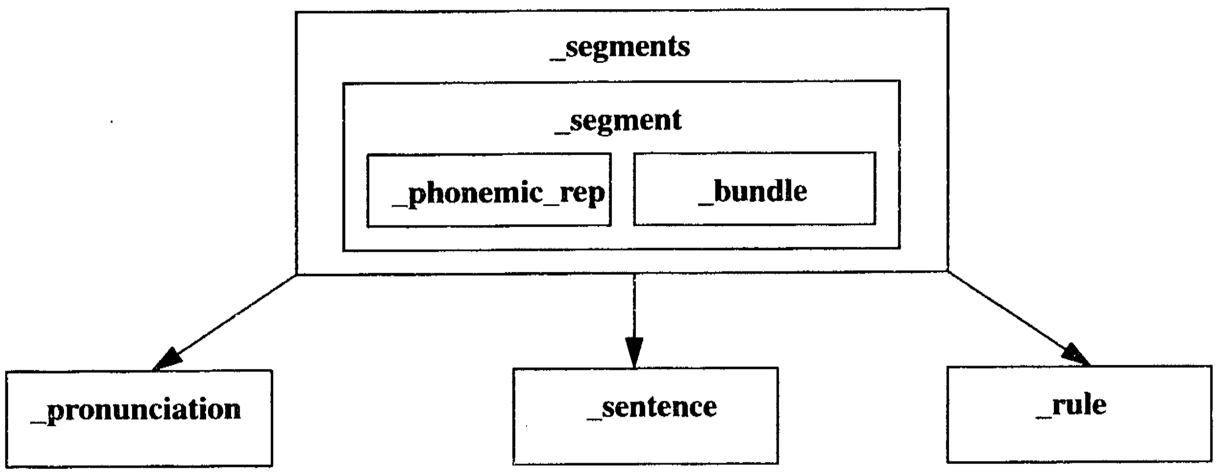

Figure 2.8 Class hierarchy. Arrows represent inheritance; inner boxes represent a has-a relationship. This structure displays that the segment is the fundamental unit of recognition.

Figure 2.9 displays the hierarchy of classes related to the lexicon. *array* is a template for a generic array class. Classes _lexicon, _lex_word, and _rules all inherit from *array* which means that they are arrays of classes. The segmented arrows of figure 2.9 indicate which classes _lexicon, _lex_word and _rules are arrays of. For example, _lexicon is an array of _lex_word objects, but each _lex_word object is an array of _pronunciation objects. This hierarchy demonstrates that the lexicon is a collection of words in which each word may be represented by one or more standard pronunciations, which is just a collection of phonemes.

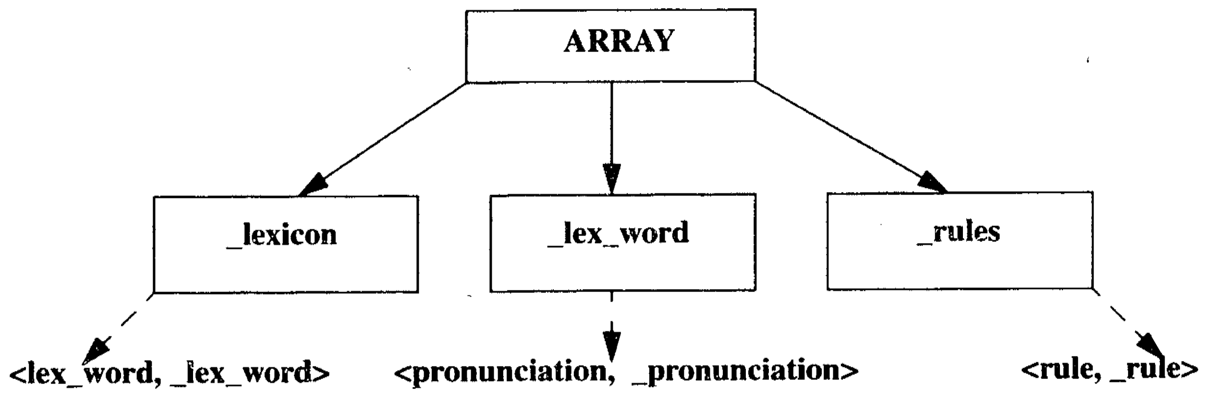

Figure 2.9 Class hierarchy demonstrating the structure of the lexicon.

The first stage of the matching process is initialization. Initialization involves reading the lexicon, the rule set, the input sentence, and the standard set of phonemes with features into feature matrices which can be manipulated during the matching process. While this occupies a large portion of the code, the details of its implementation are not relevant to the purpose of this thesis and will not be discussed. Once initialization is completed, the lexicon is expanded by applying the rule set and adding new pronunciations to words that could have arisen as a result of some fluent speech modification at word boundaries.[13] Prior to this expansion, the lexicon only contains standard pronunciations. If every speaker spoke carefully and slowly, there would not be a need to apply rules to the lexicon, but speech modifications are a given in casual speech. For example, a speaker might say the phrase "bap man" when he actually meant to say "bat man". This process is called assimilation because the /t/ in "bat" assimilates to a /p/ by the taking the place of articulation of the /m/ in "man". Rules are statements of the contexts in which neighboring segments can change. They also specify how the change is made and which features are affected. The assimilation in "bap man" can be captured by a rule and, if it applies to the input sentence, will result in the addition of the pronunciation [b ae p] to the word "bat" in the lexicon.

Rule invocation and matching are best demonstrated by an example. Suppose the output of the acoustic analysis produces a sequence of phonemes corresponding to the phrase "another ape back in power" as demonstrated in figure 2.10. Here, the output is represented as a series of phonetic symbols even though the signal processor produces a series of feature bundles and not symbols. The numbers beneath each symbol are index positions into the array in which the output is stored. Suppose, also, that the lexicon only contains the following words: [x n] ("an"), [x n ah dh x r] ("another"), [ey p] ("ape"), [b ae k] ("back"), [ih n] ("in"), [p aa w rr] ("power"), and [n aa y n t iy n] ("nineteen"). Finally, suppose there is one rule which states that if a word ends in a /d/,

$\boxed{\text{x}}$	$\boxed{\text{n}}$	$\boxed{\text{ah}}$	$\boxed{\text{dh}}$	$\boxed{\text{x}}$	$\boxed{\text{r}}$	$\boxed{\text{ey}}$	$\boxed{\text{p}}$	$\boxed{\text{b}}$	$\boxed{\text{ae}}$	$\boxed{\text{k}}$	$\boxed{\text{ih}}$	$\boxed{\text{n}}$	$\boxed{\text{p}}$	$\boxed{\text{aa}}$	$\boxed{\text{w}}$	$\boxed{\text{rr}}$
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
x	n															
x	n	ah	dh	x	r											
ey	p															
b	ae	k														
ih	n															
p	aa	w	rr													
n	aa	y	n	t	iy	n										

Figure 2.10 Hypothesized output of the acoustic analysis represented in terms of phonetic symbols.

an /n/, or a /t/ and is followed by a consonant made with the lips, then the place of articulation of the segment ending the word becomes [+ lips].

First, a temporary copy of the lexicon is made so new pronunciations can be added without altering the base lexicon. Then, imagine taking every pronunciation of every word in the lexicon and lining it up against the phonemic representation in figure 2.10 starting at index 0. Only the word "nineteen" satisfies the rule because it ends in an /n/ and is followed by a /p/. As a result, a new pronunciation is constructed for the word "nineteen" according to the specifications of the rule, and the pronunciation is added to the temporary lexicon.

After all the rules have been applied to every word in the lexicon, the features of the phonemes in each pronunciation are matched against the segments of the input sentence starting at index 0. Figure 2.10 shows that two words match, "an" and "another". Since no other words match at the initial index, the temporary lexicon is destroyed and the matches are stored in a tree. The tree is grown by successive iterations of the process just described. For example, in the next pass, a new temporary lexicon is created and the rules are applied to the lexicon starting at index 2 as a result of the number of phonemes in the match "an". Eventually, this branch will fail to generate a match and the tree position for "another" will be grown. With the index set to 6, "ape" will match. This process will continue until "another ape back in power" is reconstructed. Once the matching tree has been fully grown, the matcher displays all complete and partial matches.

III A Computational Model of the Feature Tree Geometry

Code: see def.H, treeClasses.H, node.H, and node.C in Appendix B

3.1 Theory and Implementation

The development of a computational model for the feature tree was guided by six observations: 1) nodes can be modeled as classes, 2) terminal nodes have a set of features, 3) the ideal geometry of figure 2.6 defines a set of hierarchical relationships, 4) a branch emanating from one node may be pointed to a node from an adjacent tree (see Chapter IV), 5) manner features are not attached to an articulation node and apply only to consonants, and 6) speech sounds are classified by the specification of a dominant node. With a computational model of the feature tree geometry, it is possible to represent phonemes as feature trees rather than as feature matrices. Representing phonemes by a feature tree geometry has several theoretical implications, but it also has practical implications related to computational efficiency which will be explored in subsequent chapters.

In any type of object-oriented application, the first responsibility of the programmer is to identify the objects. Once the objects have been identified, classes containing member variables and member functions can be written for each object. Member variables define specific attributes of an object; member functions are a mechanism for manipulating member variables and for adding relevant functionality to a class. For example, a **ball** class might have two member variables called *x* and *y* which specify the ball's planar coordinates. Member functions of the ball class might include *void set_x (float x_coordinate)* and *void set_y(float y_coordinate)*, which can be used to set the x and y coordinates of the ball, *float get_x(void)* and *float get_y(void)*, which return the x and y coordinates, and *void bounce(float height, float forces[ ])* , which fluctuates the x and y coordinates according the parameters in *forces[ ]* and prints them to the standard output to give the impression of a bouncing ball.

The nodes of the feature tree correspond to objects because nodes are abstractions for concepts and physical matter to which attributes can be ascribed. If nodes are modeled as classes, then the *has-a* requirement of observation two implies that features can be modeled as internal variables of terminal nodes. Classes representing terminal nodes therefore need member functions which allow feature values to be set and returned. The binary nature of features requires a three-state feature value distinction; that is, feature variables, which are declared as integer values, can be set equal to the pre-defined values POS, NEG, or NOVAL, mnemonics for the integers -99, -98, and -97.

Every tree is composed of twelve nodes resulting in twelve class declarations for the model: class _Vowel, class _Glide, class _Consonant, class _VocalFolds, class _Soft_Palate, class _Pharyngeal, class _Glottis, class _Pharynx, class _Lips, class _Lingual, class _Blade, and class _Body. Figure 3.1 illustrates the interactions between these classes. Classes are interconnected according to the hierarchy defined by the geometry of figure 2.6. Observation two justified making features internal variables of terminal nodes because the *has-a* relationship is characteristic of member variables. In a similar way, parent nodes have links, or branches, to subsequent children nodes. These links can also be modeled as member variables of the parent class. However, observation four requires that a branch from a parent to child node not be permanent, since it could possibly be altered by a rule. Instead of parent classes *having* a child node as a member variable, it must *have* a pointer to the child node. Classes which contain pointers to subsequent nodes will have member functions which allow the pointer handle to be set and returned. Using a combination of these two functions, a pointer can be reassigned to a node on a neighboring tree.

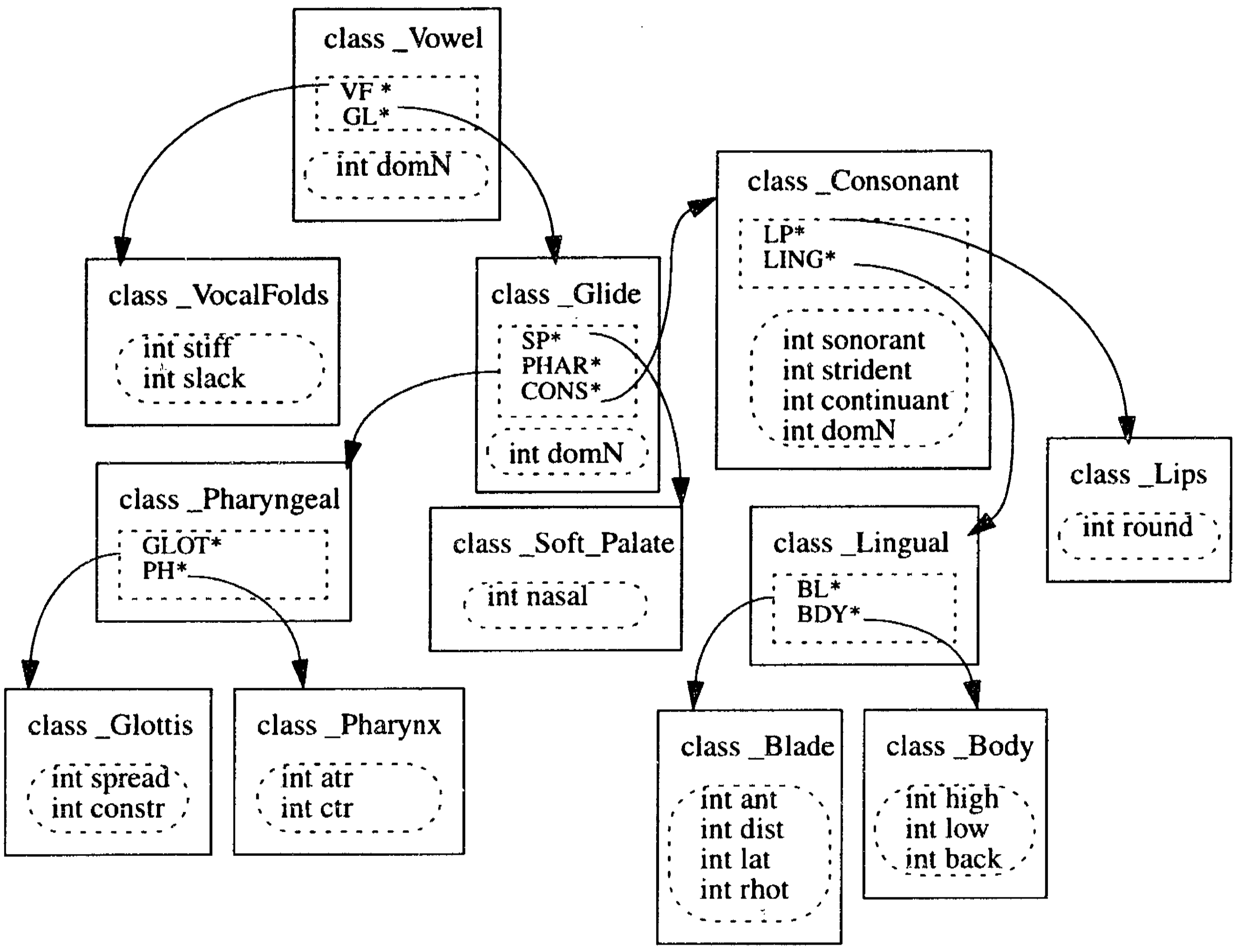

Figure 3.1 Diagram showing the interaction between classes. Inner structures represent the internal variables of a class (boxes contain pointer variables and circles contain scalar variables).

Only those features which are associated with an articulator are attached to a terminal node. Since the manner features describe how a sound was made independent of which articulator was manipulated, they are not associated with any specific articulation node. Manner features do, however, describe attributes of consonants and can be modeled as integer member variables of the consonant class. As a result, the consonant class will have "set" and "get" member functions for both types of member variables, pointer and scalar.

The last element of the model is the specification of a dominant node. In the ideal tree of figure 2.6, dominant nodes are specified by marking the vowel, glide, or consonant node with an open circle. The dominant node identifies which part of the vocal tract is active in producing a particular sound. As such, dominance is a unique characteristic which can be modeled in a class by a scalar variable. When a vowel object, for instance, is declared to be the dominant object in a tree, its dominant node variable is set to POS. The other dominant node variables in the glide and consonant objects are set to NOVAL, the default state.

3.2 Supporting Functions

During the initialization phase of the original matching process, the standard template file, containing the phonemes used in representing words, is read in and each phoneme is represented as a matrix, or an array, of features. An excerpt from the standard template file is shown in listing 3.1. In the new matcher, phonemes are represented in terms of matrices and trees. At all times two concurrent representations are made available for individual processing purposes.

```
<
Time: (nil)
Symbol: t
Prosody: (nil)
Release or Closure: unspecified
Features:
+ consonant
- continuant
- sonorant
+ blade
+ anterior
- distributed
- constricted_glottis
- slack_vocal_folds
>
```

Listing 3.1 Excerpt from standard template file.

In order to represent phonemes as trees, the following functions are required: *int determineDominant(segment data), void makeTree(Vowel treeRoot, segment data),* and *void setDomi-*

nant(int domNode, Vowel tree). For each matrix of features that is created from the standard template, a tree is also created using the information from the matrix. The first step in the process of creating a tree from a matrix is to create an array of pointers to vowel objects. The length of the array is the number of phonemes in the template file; each index of the array is a pointer to a tree structure for a different phoneme. Then the vowel pointers are assigned to vowel objects which are initialized by the default constructor of the vowel class. The default constructor will set the internal variables of the vowel object to NULL or NOVAL depending on whether the variable is a pointer or a scalar.

After the array of trees is initialized, the dominant node of each tree is identified. This can be determined by checking a tree's corresponding matrix of features to see which one of the features--*vowel*, *glide*, or *consonant*--is present and equal to POS. If, for instance, the feature *consonant* is found to be positive, then the integer value corresponding to the *consonant* feature is stored in the variable *domNode* and used as input to the *setDominant* function. *setDominant* will set the appropriate dominant node of a fully connected tree according to the feature stored in *domNode*.

Before the dominant node can be set, the rest of the tree must be instantiated by the function *makeTree*. *makeTree* takes as input a matrix of features and the vowel object for the current segment. The function works similar to a jig-saw puzzle in that classes corresponding to different nodes are instantiated and then the individual components are connected according to the hierarchy depicted in figure 3.1. The tree is constructed from the bottom up. First the node corresponding to the vocal folds is created. Then, the body class followed by the lingual class are instantiated. The two pointers in the lingual object are then set to the body and blade objects. Next, the lips class is instantiated. Once the consonant class is instantiated, its pointers are set to the lingual and lip objects. This concludes the consonant branch. Afterwards, the soft palate class, the glide class and the pharyngeal class are instantiated, the pointers in the glide object are set to the consonant, soft palate and pharyngeal objects. Then, the glottis and pharynx class are instantiated and connected to the pharyngeal object. Finally, the pointers in the vowel object are set to the vocal fold and glide objects to produce the structure in figure 2.6.

3.3 Using the Model

The geometry proposed in Keyser and Stevens is composed of three elements: nodes, branches, and features. In terms of the model developed in section 3.1, these elements correspond to classes, pointers, and integer variables. Each class includes accessor functions which allow the state of an object's member variables to be manipulated. In the tree of figure 2.6, an inferior node is reached by traversing a path from the root, or vowel, node to subsequent levels of the tree. For instance, in order to reach the blade node, the branch connecting the vowel, glide, consonant, and lingual nodes is traversed.

In the model of the feature tree geometry, pointers are traversed using the accessor functions of parent nodes. For example, the value of the feature *anterior* can be determined using the following statement: *tree->getGlide( )->getConsonant( )->getLingual( )->getBlade( )->getAnt( )*. *tree* is a pointer to the vowel object. The vowel object has a member function called *getGlide( )* which it uses to return a handle to the glide object. Then the pointer to the glide object is dereferenced and the function *getConsonant( )*, which is a member function of the glide class, is called. *getConsonant( )* returns the pointer connecting the glide object to the consonant object. This pointer is dereferenced and the *getLingual( )* function in the consonant class is called. *getLingual( )* returns the pointer to the lingual object. With this pointer, the handle to the blade object can be obtained via the *getBlade( )* function. The blade object contains a member function called *getAnt( )* which returns the value of the member variable *anterior.*

Feature values are initialized in a similar way except that the last function called is the *set* function for a particular feature. The anterior feature in the blade object, for example, is initialized with the statement: *tree->getGlide( )->getConsonant( )->getLingual( )->getBlade( )->set-Ant(POS).* Classes which have pointers as member variables also have functions which allow the pointers to be set and returned. Branches can be pointed to nodes on adjacent trees as a result of speech modifications which occur in casual speech. A branch from one tree can be pointed to a node on another tree with the following line of code: *treeA->getGlide( )->getConsonant( )->get-Lingual( )->setBlade(treeB->getGlide( )->getConsonant( )->getLingual( )->getBlade( )).* Here, *treeA* and *treeB* are pointers to the vowel objects of two neighboring segments. This statement sets the pointer connecting the lingual and blade nodes of treeA to the blade node of treeB.

IV A Method for Applying Linguistic Rules Via the Tree Geometry

Code: see def.H, node.H, node.C, and *construct_pronunciation* in Appendix B

4.1 Assimilation and the Feature Tree Geometry

Assimilation is the process by which a sound adapts to an adjacent sound. The fact that words can change internally and across word boundaries poses a problem for speech recognition. How is the matcher supposed to know a person meant to say "balloon before" if he says "balloom before"? Fortunately, many of the modifications that occur in speech are governed by rules.

In the original matcher, when a rule is applied and it is found that an alternative pronunciation should be generated, the features in the matrix of the target segment are changed according to the specification of the rule (e.g. the /n/ in "balloon" becomes an /m/). Each feature change requires at least one operation for setting the new value of the feature. If a rule happens to apply to a sentence many times and involves the alteration of a number of features, the resulting computational lag can be significant. This coupled with the fact that every rule in the rule set could potentially have the same effect warrants the search for a more efficient method of applying rules.

The feature tree potentially provides a natural way of applying rules to account for assimilation with fewer operations.[12] (Results are listed in Chapter VII.) The situation is illustrated in figure 4.1. The tree on the left corresponds to the segment /n/ and the tree on the right corresponds to the segment /b/. The rule which specifies how the /n/ can become an /m/ in "balloon before" is the same rule as for the phrase "bap man" in section 2.3. If the rule were applied using feature matrices, it would require the segment /n/ to acquire the following features from the /b/ segment: *lips*, *blade*, *dorsum*, *round*, *anterior*, *distributed*, *high*, *low*, and *back*. This would require at least nine operations for setting these feature values in the /n/ segment. With the segments represented in terms of trees, an equivalent way of expressing this rule is to point the branch from the glide node of the /n/ tree to the consonant node of the /b/ tree. This change is illustrated in figure 4.1 with a dotted arrow. By using pointer redirection, approximately eight operations are saved.

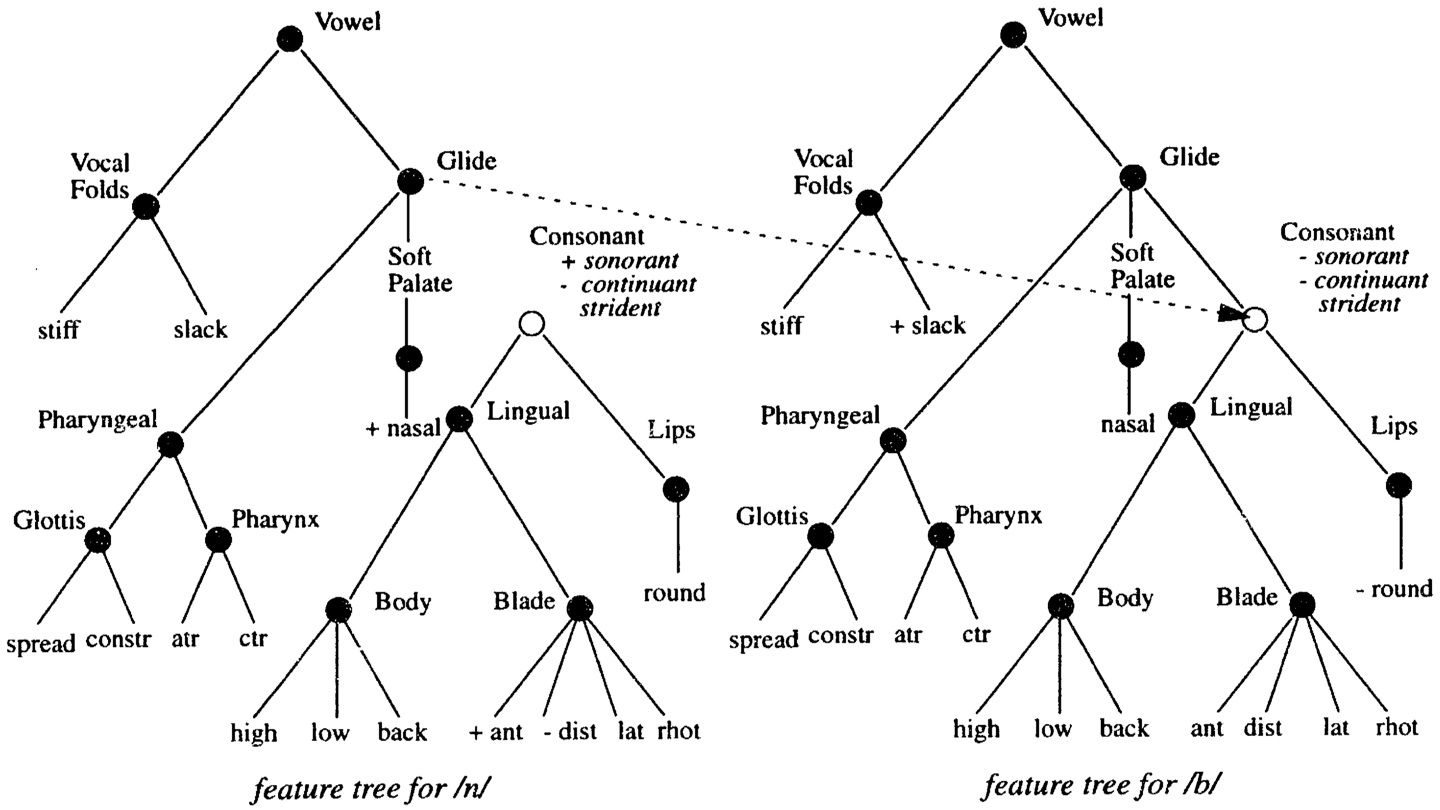

Figure 4.1 Handling assimilation via the feature tree in the phrase "balloon before".

In the example of figure 4.1, swapping nodes captured all of the features specified by the rule except the manner features. While the value of *continuant* does not change, the value for *sonorant* changes from POS to NEG. Since the manner features are not grouped under a node of articulation, they must explicitly be set. The rule which governs this modification can therefore be performed in two operations: one which does pointer redirection between adjacent trees, and one which explicitly sets the feature sonorant to POS. However, this second operation can be saved by exploiting the fact that all [+ consonant] segments which are also [+ nasal] are automatically [+ sonorant]. In general, knowledge of feature dependencies can help reduce the total number of operations required when applying rules.

4.2 Specifying a Rule File Format

One of the inputs to the matcher is a rule set. A rule set consists of the individual rules which are applied during the matching process. Listing 4.1 in appendix A is the file which specifies the place of articulation rule for alveolars followed by labials. (This rule is actually more general and can apply for alveolars followed by velars as well.) Rule files are broken up into fragments of code which identify the orientation, the contexts, and the surface change for a rule. The orientation is associated with the first fragment and is defined in the "Type" heading. For modifications which occur across word boundaries, the orientation specifies the relative position of the segment being changed. For example, the rule in listing 4.1 is "left-sided" meaning that the segment to the left of the word boundary is the one which is modified.

Contexts are underspecified segments. They are used to identify those features which a segment must have in order for a rule to be applicable. The rule in appendix A has three contexts: C1-C2-C3. The first context specifies the last phoneme of a lexical item and is identified by the features [+ consonant], [- continuant], [+ blade], [+ anterior] and [- distributed]. C2 represents a word boundary and C3 corresponds to a segment from the input sentence which has the features [+ consonant] and [+ lips]. The surface change of listing 4.1 identifies C1 as the target and C3 as the donor. As such, C1 receives the value of *lips*, *blade*, *dorsum*, *round*, *anterior*, *distributed*, *high*, *low*, and *back* from C3.

While this format is suitable for working with feature matrices, it does not reflect the aim of using the feature tree to apply rules; that is, it does not allow for the specification of nodes in the surface change. In order for the program to correctly interpret the rule file, there must also be a flag which identifies the rule as tree-based or matrix-based so the appropriate function calls can be made. The current matcher incorporates these specifications into the rule handler.

Listing 4.2 in appendix A shows the rule in listing 4.1 configured for use with feature trees. The flag which identifies the lexical representation can take on one of two values, MATRIX or TREE. It is specified in the "Release or Closure:" heading of the first code fragment. In this file format, nodes are distinguished from features by capital letters (see *def.H* in appendix B for examples). Therefore, the item "CONSONANT" in the surface change of listing 4.2 refers to the consonant node and not the *consonant* feature. The surface change requires that C1 take the consonant node of C3 and that the *sonorant* feature be set to POS, although in this case specification of *sonorant* is redundant.

In this new format, the body of a surface change consists of a list of nodes followed by features. Each element of the surface change is prefixed with a modifier which can take on any one of the following values : '+', '-', 'X', or a context such as C1 or C2. Pluses and minuses are used to indicate feature values and do not apply to nodes. Context values such as C1 and C2 apply to both features and nodes and identify underspecified segments across a word boundary. 'X' is the symbol used to indicate a blank feature or an empty tree. Empty trees can be used to clear the terminal values of an articulation node, as in the rule which governs nasalization of the /th/ segment in phrases like "in the" (listing 4.3 in Appendix A). This rule uses an empty sub-tree to clear the terminal values of the vocal fold node as illustrated in figure 4.2.

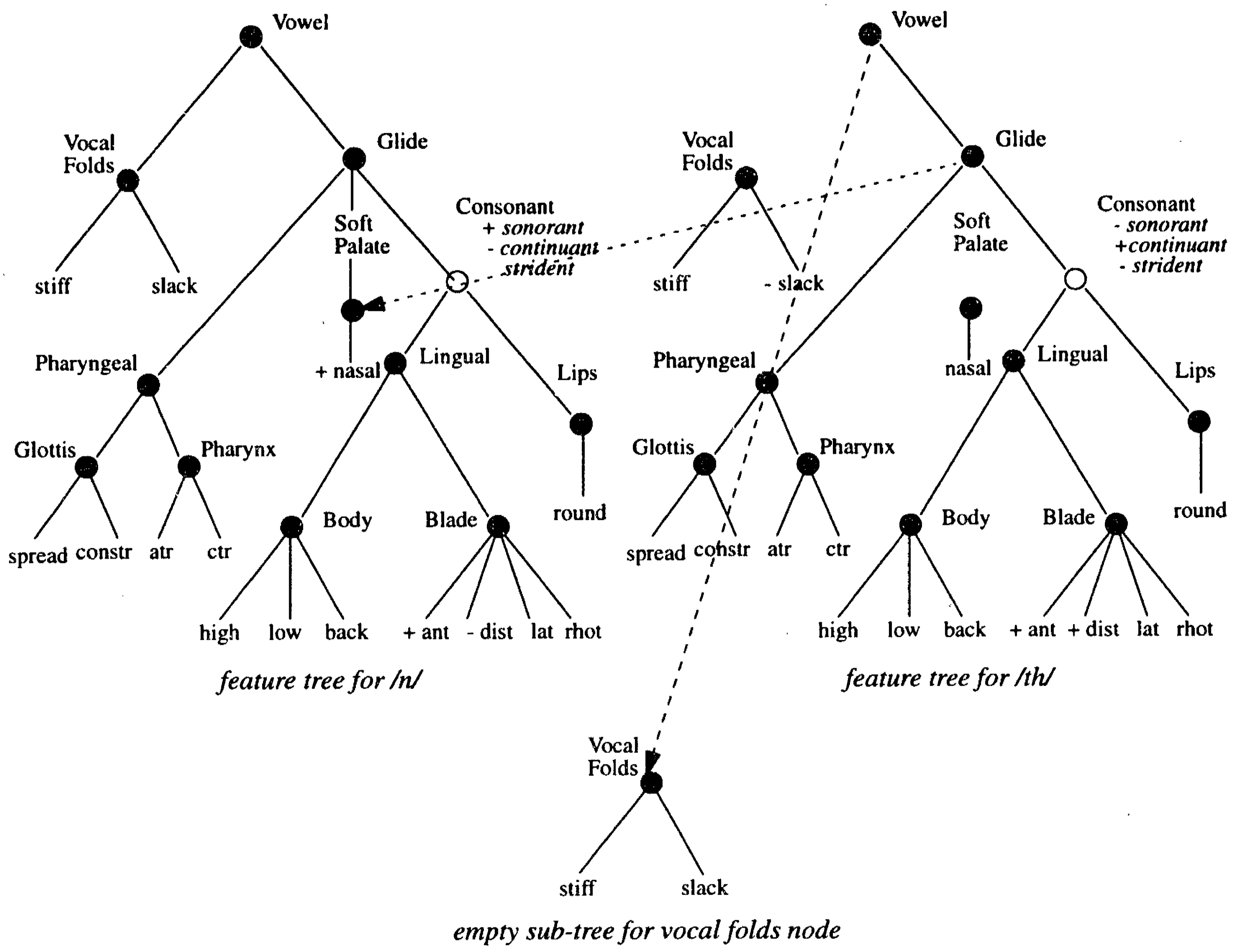

Figure 4.2 Assimilation in the phrase "in the". The terminal features of the vocal folds node are cleared with the use of an empty tree.

4.3 Methods and Functions for Applying Word-Boundary Rules

Once a rule is found to apply, an alternative pronunciation is constructed for the target word via the *construct_pronunciation* function. A pronunciation can be constructed using features matrices or feature trees. The method chosen depends on the value of the identifier flag in the rule file. Using the feature tree to apply rules requires three functions: *void copyTree(Vowel treeA, Vowel treeB)*, *void makeChange(Vowel treeA, Vowel treeB, int node)*, and *segment tree2seg(Vowel tree, bundle b, segment seg)*.

First, a copy of the target feature tree is made by the *copyTree* function. During the initialization phase of the program, an array of trees is created from the standard template file and from the input sentence; that is, there exists one tree for every phoneme in the standard template and one tree for every segment in the input sentence. Before a pointer can be moved, the tree must be copied to ensure that the underlying form is not lost and the standard template is not contaminated. The copy of the target tree is then passed to the *makeChange* function as the parameter *treeA*. *treeB* is the variable name for the pointer to the feature tree which is adjacent to *treeA*. *makeChange* is essentially a switch statement which does case selection on the various node types. The integer variable *node* determines which node of *treeB* is transferred to *treeA*.

After the surface change is implemented, the target tree is transformed into a matrix of features by *tree2seg* and an alternative pronunciation is created by concatenating the unchanged segments of the target word to the altered segment. This is illustrated in figure 4.3 for the word “balloon” in “balloon before”. It is necessary to convert the target tree into a feature matrix because the matcher has not yet been configured to match with trees. This conversion also ensures that continuity is maintained between the two lexical representations.

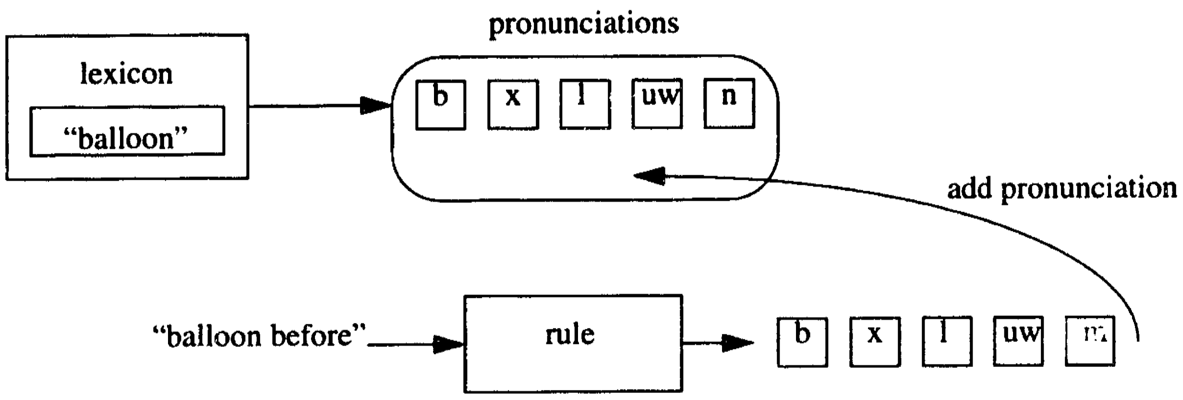

Figure 4.3 Adding a pronunciation to the lexicon.

V Morphemes

Code: see morphemes.H, morphemes.C, and match3.C in Appendix B

5.1 Discussion and Motivation

The smallest linguistic unit in a language which conveys meaning is called a morpheme. Morphemes are grouped into two classes: free and bound. A free morpheme is one that may stand alone as an independent linguistic form, such as "man", "go", and "call", or it may combine with other morphemes as in "manly", "going", and "called".[14] A bound morpheme is one that must always appear as part of a combination form. While bound morphemes are parts of complex words, they are not words themselves. Affixes are bound morphemes which are directly attached to words. Examples of affixes include "-ly", "-ing", "-ed", "de-", and "in-". If an affix is attached to the beginning of a word, it is called a prefix. Suffixes refer to those affixes which are attached to the end of a word.

Morphological analysis is important for speech recognition systems because it can reduce redundancy in the lexicon thereby saving memory. For example, if a system is equipped with methods for generating complex forms from free morphemes, then separate lexical entries are not needed for words such as "submission", "submissions", "submitting", and "submitted". These variants could be derived from a lexicon which just has the word "submit". Complex words are generated by applying morphological and phonological rules to base words in the lexicon. In this system, morphological expansion of the lexicon occurs during the initialization phase of the program so that morpheme-derived words are also candidates for word-boundary rules.

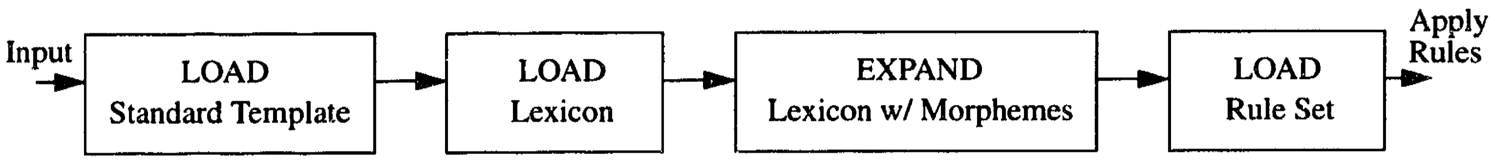

Figure 5.1 Initialization including morphological expansion.

Calling the function *morphemes(lexicon lex)* initiates morphological expansion of the lexicon. *morphemes* is just a container for a series of other function calls which perform specific types of expansion. Currently, only two functions have been written: *plurals(lexicon lex)*, which generates new lexical entries by adding a plural ending (either "-s" or "-es") to base words in the lexicon, and *expand_ion(lexicon lex)*, which adds new items to the lexicon by attaching the "-ion" suffix to base words. Listing 5.1 shows the lexicon format that was used before morphological expansion was introduced into the matcher. Every lexical item consists of a label and a set of standard pronunciations, which is a string of phonemes. When a word in the lexicon is expanded, a new pronunciation is created for each standard pronunciation of the base word along with a new label reflecting the change. Each new pronunciation is then added to the lexicon as a separate entry.

```
LexiconName: M3Lex_Version_1_7/11/1998
<
   a          [ah]
   able       [ey b x l], [ey b l]
   about      [x b aa w t]
   above      [x b ah v]
   ache       [ey k]
   add        [ae d]
   again      [x g eh n]
   all        [ao l]
   among      [x m ah ng]
   an         [ae n], [x n]
   and        [ae n d], [x n d]
   another    [x n ah dh x r]
   ...
   ...
   ...
>
```

Listing 5.1 Excerpt from original lexicon.

The next section will discuss the algorithm and methods used to generate plurals in detail, but the algorithm is briefly discussed here to demonstrate that the lexical format used in listing 5.1 is sufficient but not efficient. Plurals can be generated by the following general rule: 1) if the phonemic representation of a word ends in a segment which is [+ blade] and [+ strident], then create a new pronunciation by adding a /x/ plus a /z/ to the end of the original pronunciation, 2) or else, add a /z/ to the last segment of the original pronunciation if it is voiced or an /s/ if it is unvoiced. The problem with this approach is that it will generate a plural counterpart for every word in the lexicon, irrespective of its part of speech; that is, the function which generates plurals should only be applied to nouns. This implies that some measure of grammatical knowledge--more specifically, a word's part of speech--needs to be incorporated into the lexicon.

Listing 5.2 is an excerpt from a new lexicon which includes information about the possible parts of speech for each lexical item. This format uses eleven lexical specifiers to indicate a word's part of speech. They are indicated in table 5.1. Providing each word with a part of speech restricts the number of words which can be expanded, thus saving memory. It also requires that generated words be given a part of speech so subsequent expansion rules can be properly applied. The current approach is to just mark the plural form with an [Np] to indicate a plural noun. Given this, the rule governing the generation of plurals can be restated in the following manner: 1) if a word is a noun, [N], 2) and the phonemic representation of that word ends in a segment which is [+ blade] and [+ strident], then create a new pronunciation by adding a /x/ plus a /z/ to the end of the original pronunciation, 3) or else, add a /z/ to the last segment of the original pronunciation if it is voiced or an /s/ if it is unvoiced, 4) then give the new pronunciation an [Np] part of speech.

Lexical Marker	Part of Speech
N	noun
V	verb
ADJ	adjective
ADV	adverb
PN	pronoun
PP	preposition
CJ	conjunction
AR	article
Np	plural noun
Vp	past tense of verb
NS	an undetermined category

Table 5.1 Lexical markers and their definitions.

```
LexiconName: M3Lex_Version_1_7/11/1998

<                ...
   [N V ADV]    say         [s ey]
   [N V]        seal        [s iy l]
   [V]          see         [s iy]
   [V]          seek        [s iy k]
   [V]          seize       [s iy z]
   [V]          shake       [sh ey k]
   [PN]         she         [sh iy]
   [N V]        shoe        [sh uw]
   [ADJ]        short       [sh ao r t]
   [N]          shortage    [sh ao r t x dj]
   [N V]        show        [sh ow]
   [N V]        sing        [s ih ng]
                ...
                ...
 >              ...
```

Listing 5.2 Excerpt from new lexicon.

Another concern regarding morphological expansion of the lexicon is the order in which the expansion rules are applied. If the plural rule is applied to the lexicon followed by the rule for the "-ion" suffix, then the word "submission" will be generated for "submit" but there will be no chance of generating "submissions". Since there are only two rules at the moment, this problem can be avoided by applying the plural rule last, but as more rules are added, the ordering will become increasingly important.

5.2 Generating Plurals using the Feature Matrix

Figure 5.2 illustrates the structure of the lexicon. The lexicon is an array of words in which each word is an array of pronunciations. Each pronunciation is an array of segments which is represented in terms of a feature matrix and a feature tree. The function *plurals* was written to exploit the properties of the feature matrix.

plurals takes as input a pointer to the lexicon structure of figure 5.2. It then loops through all the words in the lexicon array looking for those which are marked as a noun. For every word that is lexically specified as a noun, the function loops through all of that word's pronunciations checking if the last phoneme is specified as [+ blade] and [+ strident] in its feature matrix. If the last phoneme is found to meet this criterion, meaning that it is a /ch/, /dj/, /s/, /sh/, /zh/, or /z/, then a new pronunciation is constructed for the plural form of the base word, as demonstrated in figure 5.3. First, the original pronunciation is copied into a new pronunciation object. Then, two more phonemes are added to the pronunciation: a schwa and a /z/. Once the plural pronunciation has been formed, a new word object is created and connected to the new pronunciation. Since each word has a label and part of speech associated with it, it is necessary to initialize these variables in the new word object. The label of the new word is the concatenation of the label for the base word and "+es". Therefore, a word like "bus" would become "bus+es". The part of speech is initialized to [Np], which indicates that word is a plural noun. Finally, the new word is added to the lexicon's array of words which makes it available for matching.

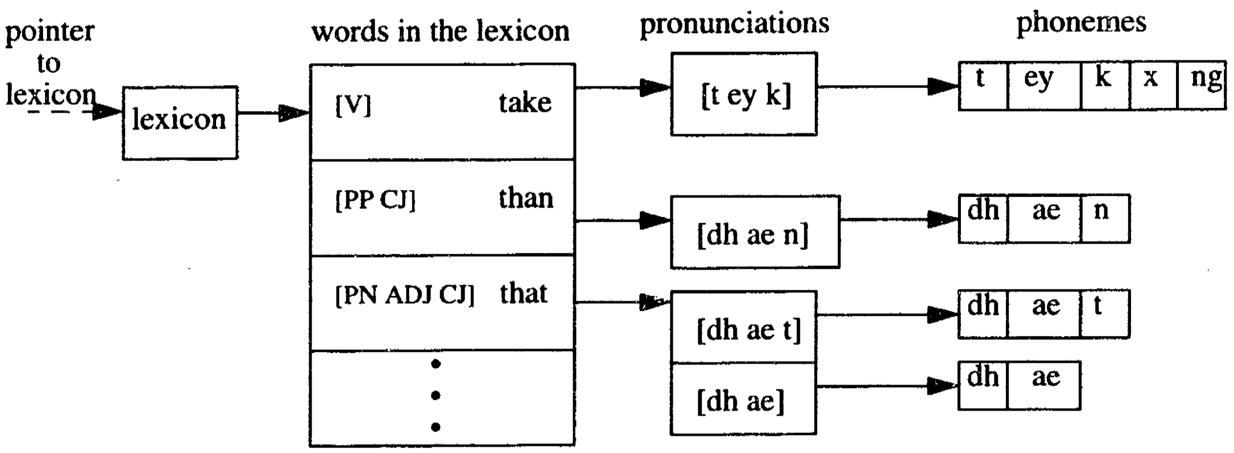

Figure 5.2 Structure of the lexicon.

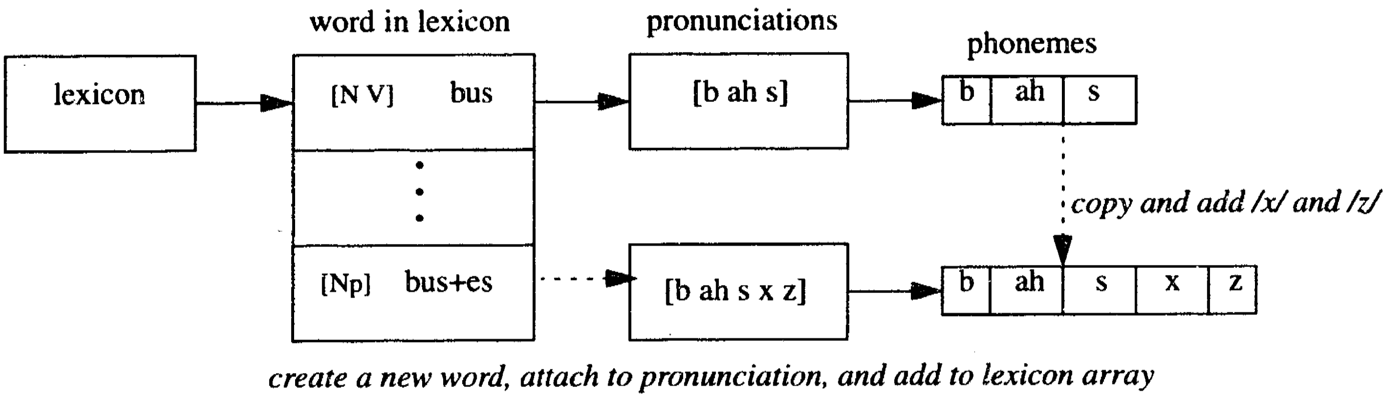

Figure 5.3 Adding plurals to the lexicon.

If the last segment of a pronunciation is not [+ strident] and [+ blade], then the plural is formed by adding an /s/, if the last phoneme is unvoiced, or a /z/, if it is voiced, to the original pronunciation. The mechanics of this change are similar to that of figure 5.3. However, the label of the new word is initialized by concatenating the label of the original word with "+s". For example, the label of the plural for "cap" would be "cap+s".

5.3 Generating Bound Forms using the Feature Tree

There are many words in the English language which can be expanded with the "-ion" suffix, such as "addict", "commit", "submit", and "transmit". Underlyingly the expanded form of these words can be considered to be the result of a series of transformation rules which 1) adds the suffix /yxn/ to the base word (spirantization rule), 2) changes the final /t/ to a /sh/ (palatization rule) and 3) deletes the suffix-initial /y/ (deletion rule) to form "submission".[14] In the matcher, the "-ion" expansion is implemented by using the feature tree geometry to change the final /t/ of a base word to a /sh/ and then adding a /x/ and an /n/.

First, the array of words is looped through to find all the words which are verbs and which end in /t/. While this criterion is very loose and will obviously generate erroneous pronunciations, it satisfies the need of our lexicon at the moment. Figure 5.4 demonstrates how the expansion is made once a word which meets this criterion is found. In the lexicon a /t/ is represented by the following features: [+ consonant], [- continuant], [- sonorant], [+ blade], [+ anterior], [- distributed], and [- slack vocal folds]. A /sh/ has the same features except that it is [+ strident] and the feature values under the blade node are reversed; that is, a /sh/ is [- anterior] and [+ distributed]. One way of transforming the /t/ to a /sh/ is change the pointer from the blade node of the /t/ tree to the blade node of a sub-tree with the correct features. The final step is to make the tree [+ strident].

Once the /t/ has been transformed to a /sh/, the *tree2seg* function is used to change the tree into a feature matrix and the new pronunciation is formed. Figure 5.5 demonstrates this process.

Before the new word is added to the lexicon, the segments /x/ and /n/ are added to the end of the pronunciation. The label for the new word consists of the original label plus the string "+ion". Therefore, "submit" becomes "submit+ion".

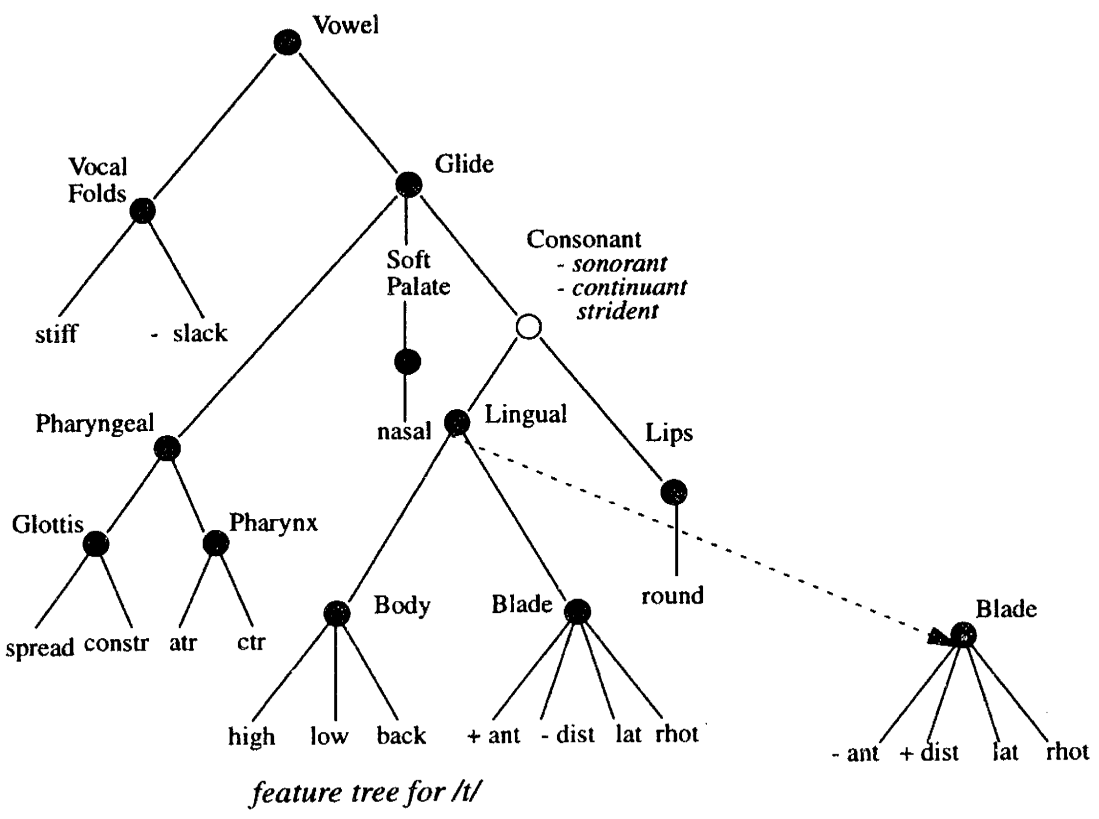

Figure 5.4 Expanding words with "-ion" via the feature tree.

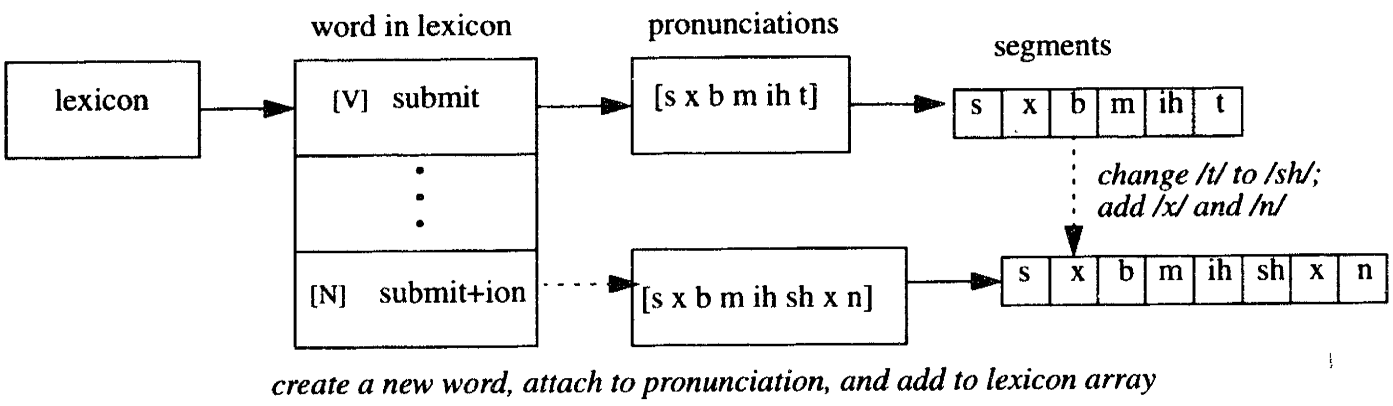

Figure 5.5 Expanding the lexicon with "-ion".

VI Implications of A Tree-Based Matcher

Code: see Appendix C for a more detailed review of performance specs

6.1 A Proposed Algorithm

Given that segments can be represented in terms of feature trees, how can matching be performed to exploit the structure of the tree geometry? Before this question can be answered, it is necessary to consider what makes a feature tree different from a matrix of features. Table 6.1 lists the most important attributes of the feature geometry which distinguish it from the feature matrix.

Attributes of the Feature Tree Geometry
1) specification of a dominant node
2) underspecification of features
3) inherent ordering of the features based on vocal tract geometry

Table 6.1 Feature Tree Geometry Attributes

Specification of the dominant node in a feature tree implies certain characteristics about the spectrum of a signal. For example, for a segment whose dominant node is the consonant node, there is an abrupt spectrum change associated with creating or releasing a constriction formed by the lips, blade, or tongue body [12]. Similarly, for a segment whose dominant node is the glide node, slow-varying spectral changes occur in a region of minimum amplitude.[12] Essentially, the dominant node classifies a speech sound as a vowel, glide, or consonant. The dominant node is therefore an important and defining characteristic of a feature tree and can be used to discriminate between trees.

An underspecified feature is one that does not need to be explicitly given a value because its state can be derived from the remaining features. Even before the feature tree was introduced into the matcher, underspecification was present (although it was not utilized) because not all features are independent. For instance, a segment that is [+ nasal] is also [+ sonorant], and if a segment is [+ sonorant] it must also be [- continuant]. The feature tree adds to the underspecification of features. If the dominant node is *vowel*, for example, then it is not necessary to specify the manner features or values for the *stiff* and *slack* vocal fold features. As such, they are not useful for discriminating between vowels. Another source of underspecification is the articulator features. In the feature tree, an articular is considered to be active if at least one of the terminal features of its corresponding node has a value. Therefore, if the feature anterior were marked with a '+', it is understood, in the case of a consonant, that the feature tree is also specified as [+ blade]. If two consonant trees were being matched and it was found that the both trees were [+ anterior], there would be no need to match the [+ blade] feature of the lexically specified segment.

Unlike the feature matrix, the feature tree is hierarchical and suggests a particular ordering of the features. Currently, when two segments are matched, there is no consideration given to the order in which the features should be compared. The feature tree is, in essence, a picture of the vocal tract. Depending on the type of sound being made, the vocal tract will take on a particular shape and certain features will play a more prominent role in describing that sound than others. This suggests that the order in which features are matched might be different depending on whether the segment is a vowel, a glide, or a consonant.

The aforementioned observations regarding the specification of a dominant node, underspecification of features, and feature ordering can be used to propose an algorithm for matching two segments represented in terms of the feature tree:

- First, check to see if two trees have the same dominant node.
  - -If the dominant nodes are equal, continue matching.
  - -If the dominant nodes are not equal, then consider it a mismatch
- If the dominant node is *vowel*, then match features in the following order:

high
low
back
round
advanced tongue root
constricted tongue root
anterior
distributed
nasal
rhotic
reduced

- If the dominant node is *glide*, then match features in the following order:

high
low
back
anterior
distributed
round
advanced tongue root
constricted tongue root
rhotic

- If the dominant node is *consonant*, then match features in the following order:

high
low
back
anterior
distributed
nasal
rhotic
round
strident
<check *sonorant* if *nasal* is not '+'>
sonorant
<check *continuant* if *sonorant* is not '+'>
continuant

The feature tree geometry inherently distinguishes between vowel, glide, and consonant. At its most fundamental level, the process of matching reduces to a feature-by-feature match of two segments. Unlike the feature matrix, the feature tree suggests that vowels, glides, and consonants should be handled differently. The algorithm presented here is an attempt to differentiate between the three classes of speech sounds on the basis of how the features are ordered; that is, the features which are the most variable among a particular class are matched first in order to quickly remove unlikely candidates. At the moment, there is no evidence which supports the idea that such an ordering of features will improve the matcher. In the future, a more sound algorithm will be investigated.

VII Performance Evaluation

A performance analysis was performed using the *gprof* profiler in UNIX. *gprof* produces an execution profile of a program; it provides a variety of statistical data on the execution time of each function and the program as a whole.[15] Table 7.1 is a comparison of the profiling results from the original matcher and the matcher which incorporates the feature tree geometry. These results do not account for the processing time spent on expanding the lexicon with morphemes. What is being compared in table 7.1 is the processing time required for the original matcher to recognize the utterance "Catch a balloom before Mary does.", given that the place of articulation rule must change "balloom" to "balloon", and the time it takes for the new matcher to recognize the same utterance via the feature tree. This experiment was also carried out on the utterance "The baby can go to him in the bus today." which makes use of the nasalization rule. Table 7.2 presents execution time data for specific function calls.

Both tables show that the matcher which incorporates the feature tree geometry is more efficient than the original matcher. In particular, table 7.1 shows that the computation time of the new matcher decreased to 40% of the original matcher's execution time. The computational savings are expected to increase for utterances involving multiple fluent speech modifications in which the surface change can be captured by the swapping nodes between adjacent trees.

Utterance	Total Execution Time (s)	Matcher
"Catch a balloom before Mary does."	1.01	Original
	.3	New
"The baby can go to him in the bus today"	1.29	Original
	.5	New

Table 7.1 A comparison of execution times from *gprof.*

Utterance	Function Name	Execution Time (ms)	Matcher
"Catch a balloom before Mary does."	*main*	345.77	Original
		102.03	New
	construct_pronunication	1.23	Original
		.02	New
	grow_aux	191.82	Original
		80.66	New
"The baby can go to him in the bus today."	*main*	394.14	Original
		150.97	New
	construct_pronunciation	1.38	Original
		.19	New
	grow_aux	219.58	Original
		128.96	New

Table 7.2 A comparison of execution times from *gprof*.

Appendix B contains the output of the *gprof* program for the cases mentioned above. *gprof* uses a variety of different fields to classify performance. While only two are reported here, the other fields support these findings and the reader is encouraged to look through the appendix for a more comprehensive presentation of performance results.

VIII Conclusion

The intent of this thesis has been to motivate and implement the inclusion of a specific feature tree geometry into a matcher for a lexical access system. The following is a summary of the motivations for the inclusion of the feature tree geometry of Keyser and Stevens:

- the tree expresses characteristics of the articulatory process which a matrix of features cannot express (e.g. independence of articulators, and the grouping of articulators which produce similar acoustic attributes)
- the tree imposes an ordering to the features which reflects the form of the vocal tract
- the feature tree can be extended to incorporate syllable information, which is also represented in terms of trees
- the tree facilitates the process of applying rules by allowing groups of features to be changed simultaneously
- the feature tree can be used to facilitate morphological expansion of the lexicon

Results from the performance analysis (Chapter VII) indicate that the feature tree is also computationally advantageous.

There are many word-boundary rules which can make use of the feature tree but which have not yet been implemented. An example is the rule which changes a word-final /t/ into a glottal stop. If this rule were found to apply to a /t/ segment, the consonant dominant node would be moved up to the glide node making the anterior and distributed features no longer distinctive. Adding the features [+ constricted glottis] and [+ stiff vocal folds] would complete the transformation of /t/ to a glottal stop.

Another place where the feature tree can be further exploited is in the morphological expansion of the lexicon. In figure 5.1, a block diagram of the first stages of the matching process is presented. There, morphological expansion occurs before rules are applied to the lexicon in order to make variants of the root word available for modification by a rule, if necessary. The problem with this is that every word which meets the criteria for expansion will be expanded, even if it will not be used in matching. Ultimately, morphological expansion should be handled by rules which selectively expand words in a cohort so that only those words which need to be expanded are expanded.

While the first steps at integrating a feature tree geometry into this matcher have been taken, there are many areas which remain open for future work. They include word-internal rules, tree matching, and incorporating syllable structure and syntax information. Currently, the matcher is configured to work with word-boundary rules only. Routines need to be developed which determine when internal rules apply and which apply the rules to words in the lexicon. An example of a word-internal modification is the flapping of the /t/ in the word "water". As for matching, consideration should be given to an algorithm which exploits the structure of the feature tree and not just the implications it has for the ordering of the features in a feature matrix. Also, efficient matching practices such as cohort-building and segment look-ahead should be investigated and employed to reduce the computational strain on the matching and rule invocation processes. Lastly, knowledge of syllable structure and syntax is necessary to distinguish phrases like "I scream" from "ice cream" which have the same phonemic representation but different syllable structures and meanings.

Bibliography

[1] Roman Jakobson, *Preliminaries to speech analysis : the distinctive features and their correlates*, Cambridge, Mass. : M.I.T. Press, 1963

[2] N. Chomsky and M. Halle, *The Sound Pattern of English*, Cambridge, Mass. : MIT Press, 1991

[3] D.H. Klatt, "Reviews of Selected Models of Speech Perception", *Lexical Representation and Process*, Cambridge, MA, The MIT Press, pp. 169-219, 1989

[4] M.H. Johnson, *Brain development and cognition : a reader*, Oxford, UK ; Cambridge, USA : Blackwell, 1993

[5] G.N. Clements and S.J. Keyser, *CV Phonology: A Generative Theory of the Syllable*, Cambridge, MA, The MIT Press, 1983

[6] K.N. Stevens, *Lexical Access From Features*, unpublished document, MIT, 1992

[7] J.Y. Choi, *Labeling with Features*, unpublished document, MIT, 1999

[8] Y. Zhang, *Toward Implementation of a Feature-based Lexical Access System*, MIT Master's Thesis, 1998

[9] P.B. Denes and E.N. Pinson, *Speech Chain: the Physics and Biology of Spoken Language*, (2nd Edition) Gardon City, NY, Anchor Press, 1993

[10] K.N. Stevens, *Acoustic Phonetics*, Cambridge, MA, MIT Press

[11] K.N. Stevens, *Acoustic correlates of some phonetic categories, Journal of Acoustical Society of America* 68: 836-842, 1980

[12] S.J. Keyser and K.N. Stevens, "Feature Geometry and the Vocal Tract", *Phonology*, Vol. 11, pp. 207-236, 1994.

[13] P. Li, *Feature Modifications and Lexical Access*, MIT Bachelor's Thesis, 1993

[14] S.J. Keyser and K.N. Stevens, unpublished document

[15] UNIX man pages for *gprof*

APPENDIX A

```
*************
*LISTING 4.1*
*************

Rule: place of articulation change

<
Segment: T
Type: L
Prosody: (nil)
Release Or Closure: MATRIX
Features:
>

<
Segment: C1
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ consonant
- continuant
+ blade
+ anterior
- distributed
>

<
Segment: C2
Type: #
Prosody: (nil)
Release Or Closure: unspecified
Features:
>

<
Segment: C3
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ consonant
+ lips
>

<
Segment: S1
Type: C1
Prosody: (nil)
Release Or Closure: unspecified
Features:
C3 lips
C3 blade
C3 dorsum
C3 round
C3 anterior
C3 distributed
C3 high
C3 low
C3 back
>
```

```
*************
*LISTING 4.2*
*************

Rule: place of articulation change

<
Segment: T
Type: L
Prosody: (nil)
Release Or Closure: TREE
Features:
>

<
Segment: C1
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ consonant
- continuant
+ blade
+ anterior
- distributed
>

<
Segment: C2
Type: #
Prosody: (nil)
Release Or Closure: unspecified
Features:
>

<
Segment: C3
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ consonant
+ lips
>

<
Segment: S1
Type: C1
Prosody: (nil)
Release Or Closure: unspecified
Features:
C3 CONSONANT
+ sonorant
>
```

```
*************
*LISTING 4.3*
*************

Rule: nasalization

<
Segment: T
Type: R
Prosody: (nil)
Release Or Closure: MATRIX
Features:
>

<
Segment: C1
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ nasal
>

<
Segment: C2
Type: #
Prosody: (nil)
Release Or Closure: unspecified
Features:
>

<
Segment: C3
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ consonant
+ continuant
- sonorant
+ blade
+ distributed
+ slack_vocal_folds
>

<
Segment: S1
Type: C3
Prosody: (nil)
Release Or Closure: unspecified
Features:
- continuant
+ sonorant
+ nasal
X slack_vocal_folds
X stiff_vocal_folds
>
```

```
Rule: nasalization

<
Segment: T
Type: R
Prosody: (nil)
Release Or Closure: TREE
Features:
>

<
Segment: C1
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ nasal
>

<
Segment: C2
Type: #
Prosody: (nil)
Release Or Closure: unspecified
Features:
>

<
Segment: C3
Type: (nil)
Prosody: (nil)
Release Or Closure: unspecified
Features:
+ consonant
+ continuant
- sonorant
+ blade
+ distributed
+ slack_vocal_folds
>

<
Segment: S1
Type: C3
Prosody: (nil)
Release Or Closure: unspecified
Features:
C1 SOFT_PALATE
X VOCAL_FOLDS
>
```

APPENDIX B

```
#define BOOLEAN int

// Standard Value Definitions

// features
enum {
  main_stress=0,
  reduced,
  vowel,
  glide,
  consonant,
  continuant,
  sonorant,
  strident,
  lateral,
  rhotic,
  lips,
  blade,
  dorsum,
  larynx,
  soft_palate,
  round,
  anterior,
  distributed,
  high,
  low,
  back,
  adv_tongue_root,
  const_tongue_root,
  nasal,

  stiff_vocal_folds,
  slack_vocal_folds,

  spread_glottis,
  constricted_glottis,

  //node types

  VOWEL,
  GLIDE,
  CONSONANT,
  LIPS,
  BLADE,
  BODY,
  LINGUAL,
  PHARYNX,
  GLOTTIS,
  PHARYNGEAL,
  SOFT_PALATE,
  VOCAL_FOLDS,

  NOF,
  separatortype
};


enum lexical_representation {MATRIX, TREE, noLexRep};
enum marker_value {stressed, unstressed, reduced2, nonspecified};
enum rc_value {release, closure, unspecified};
enum mode_value {phonemic, feature, template2, either, empty};
enum destination {disk, screen, printer};
enum heading_value {Time, Symbol, Segment, Type, Sentence, Template, Rule, Pronunciation
, NONE};
enum RunTimeStatus {success, failure};
enum {ST=-100,POS,NEG,NOVAL,FNOVAL,REVAL,ED};
enum rule_type {left, right, internal, external, notype};
```

```
enum exp_value {notexp,exp};
enum grammar {N, V, ADJ, ADV, PN, PP, CJ, AR, NS, Vp, Np} ;
enum output_mode {completed, incomplete};
enum case_value {case1=1,case2,case3,case4,case5,case6};
enum nop {NOP};


// Global Value

const int NoOfFeatures=NOF; // don't touch it
const int fv_start=ST,fv_end=ED; // don't touch it
const int main_start=vowel; // don't touch it
const int main_end=slack_vocal_folds; // don't touch it
const int main_node_start=VOWEL; // don't touch it
const int main_node_end=VOCAL_FOLDS; //don't touch it
const int MaxSentenceSize = 101; // max number of segs in a sentence
const int MaxWordSize = 20; // max number of segs in a word
const int MaxNoPhons = 102; // max number of phonemes in a word
const int MaxRuleSize = 20; // max number of segs in a rule
const int MaxRuleSetSize = 15; // max number of rules in a rule set
const int MaxLexiconSize = 10000; // max number of words in a lexicon
const int MaxMatchPerPos = 50; // max number of word matches per context
const int MaxContextSize = 20; // max number of segments in a particular sub-context
 like left or right side

//used for morphemes
const int MaxNoPOS = 11; // max number of Parts of Speech
const int s_index = 30; // change if new phones are added to standard template
const int z_index = 26;
const int schwa_index = 13;
const int n_index = 22;
const int sh_index = 31;
const int t_index = 38;
```

```
/*******************************************************
**                                                    **
**  Header File For TreeMaking Program                **
**                                                    **
**  Date: 1/04/99                                     **
**                                                    **
********************************************************/

class _Vowel;
class _Glide;
class _Consonant;
class _VocalFolds;
class _Soft_Palate;
class _Pharyngeal;
class _Glottis;
class _Pharynx;
class _Lips;
class _Lingual;
class _Blade;
class _Body;

//typedef class _Feature* Feature;
typedef class _Vowel* Vowel;
typedef class _Glide* Glide;
typedef class _Pharyngeal* Pharyngeal;
typedef class _Soft_Palate* Soft_Palate;
typedef class _Consonant* Consonant;
typedef class _VocalFolds* VocalFolds;
typedef class _Glottis* Glottis;
typedef class _Pharynx* Pharynx;
typedef class _Lips* Lips;
typedef class _Lingual* Lingual;
typedef class _Blade* Blade;
typedef class _Body* Body;


//*******************VOCAL_FOLDS****************************

class _VocalFolds{
public:
_VocalFolds(){
   stiff_vocal_folds=NOVAL;
   slack_vocal_folds=NOVAL;
}

_VocalFolds(int val1, int val2){
   stiff_vocal_folds=val1;
   slack_vocal_folds=val2;
}

~_VocalFolds(){
  // cout << "vocal folds deconstructor......" << endl;
}

int getStiff(){
   return stiff_vocal_folds;
}

void setStiff(int val){
  stiff_vocal_folds=val;
}

int getSlack(){
   return slack_vocal_folds;
}

void setSlack(int val){
   slack_vocal_folds=val;
}
private:
  int stiff_vocal_folds;
  int slack_vocal_folds;
};


//**********************SOFT_PALATE****************************

class _Soft_Palate{
public:
_Soft_Palate(){
    nasal=NOVAL;
  }

_Soft_Palate(int val){
    nasal=val;
  }

~_Soft_Palate(){
  // cout << "soft palate deconstructor" << endl;
}

  int getNasal(){
    return nasal;
  }

  void setNasal(int val){
    nasal=val;
  }

private:
  int nasal;
};


//*************************LIPS******************************

class _Lips{
public:
_Lips(){
    round=NOVAL;
  }

_Lips(int val){
    round=val;
  }

~_Lips(){
  // cout << "_lips deconstructor" << endl;
}

  int getRound(){
    return round;
  }

  void setRound(int val){
    round=val;
  }

private:
  int round;
};


//**********************BODY*******************************
```

```
class _Body{
public:
_Body(){
    low=NOVAL;
    high=NOVAL;
    back=NOVAL;
  }

_Body(int val1, int val2, int val3){
    low=val1;
    high=val2;
    back=val3;
  }

~_Body(){
  // cout << "_body() deconstrcutor..." << endl;
}

  int getLow(){
    return low;
  }

  int setLow(int val){
    low=val;
  }

  int getHigh(){
    return high;
  }

  int setHigh(int val){
    high=val;
  }

  int getBack(){
    return back;
  }

  int setBack(int val){
    back=val;
  }

private:
  int low;
  int high;
  int back;
};

//*********************BLADE*****************************

class _Blade{
public:
_Blade(){
    ant=NOVAL;
    distr=NOVAL;
    lat=NOVAL;
    rhotic=NOVAL;
  }

_Blade(int val1, int val2, int val3, int val4){
    ant=val1;
    distr=val2;
    lat=val3;
    rhotic=val4;
  }

~_Blade(){
  // cout << "_blade deconstructor...." << endl;
}

  int getAnt(){
    return ant;
  }

  int setAnt(int val){
    ant=val;
  }

  void setDistr(int val){
    distr=val;
  }

  int getDistr(){
    return distr;
  }

  int getLat(){
    return lat;
  }

  void setLat(int val){
    lat=val;
  }

  int getRhotic(){
    return rhotic;
  }

  void setRhotic(int val){
    rhotic=val;
  }

private:
  int ant;
  int distr;
  int lat;
  int rhotic;
};


//***************************LINGUAL*************************

class _Lingual{
public:
_Lingual(){
    blade=NULL;
    body=NULL;
  }

~_Lingual(){
  // cout << "_lingual() deconstructor......" << endl;
  if(blade!=NULL)
    delete blade;
  if(body!=NULL)
    delete body;
}

  void setBlade(Blade bladeArg){
    blade=bladeArg;
  }

  void setBody(Body bodyArg){
    body=bodyArg;
  }

  Body getBody(){
    return body;
  }

  Blade getBlade(){
    return blade;
  }

private:
  Blade blade;
  Body body;
};


//**********************CONSONANT************************

class _Consonant{
public:
_Consonant(){
    dominantNode=NOVAL;
    sonorant=NOVAL;
    strident=NOVAL;
    continuant=NOVAL;
    lips=NULL;
    lingual=NULL;
  }

_Consonant(int val1, int val2, int val3){
    dominantNode=NOVAL;
    sonorant=val1;
    strident=val2;
    continuant=val3;
    lips=NULL;
    lingual=NULL;
  }

~_Consonant(){
  // cout <<  " _consonant deconstructor" << endl;
  if(lips!=NULL)
    delete lips;
  if(lingual!=NULL)
    delete lingual;
}

  void setLingual(Lingual lingArg){
    lingual=lingArg;
  }

  void setLips(Lips lipArg){
    lips=lipArg;
  }

  void setManner(int val1, int val2, int val3){
    sonorant=val1;
    strident=val2;
    continuant=val3;
  }

  Lingual getLingual(){
    return lingual;
  }

  Lips getLips(){
    return lips;
  }

  int getSonorant(){
    return sonorant;
  }

  void setSonorant(int val){
    sonorant=val;
  }

  int getStrident(){
    return strident;
  }

  void setStrident(int val){
    strident=val;
  }

  int getContinuant(){
    return continuant;
  }

  int setContinuant(int val){
    continuant=val;
  }

  int getDominant(){
    return dominantNode;
  }

  void setDominant(int val){
    dominantNode=val;
  }

private:
  Lips lips;
  Lingual lingual;
  int dominantNode;
  int sonorant;
  int strident;
  int continuant;
};


//********************GLOTTIS*****************************

class _Glottis{
public:
_Glottis(){
    spread=NOVAL;
    constr=NOVAL;
  }

_Glottis(int val1,int val2){
    spread=val1;
    constr=val2;
  }

~_Glottis(){
  // cout << "_glottis deconstructor...." << endl;
}

  int getSpread(){
    return spread;
  }

  int setSpread(int val){
    spread=val;
  }

  int getConstr(){
    return constr;
  }

  int setConstr(int val){
    constr=val;
  }

private:
  int spread;
  int constr;
};


//*******************PHARYNX********************************

class _Pharynx{
public:

_Pharynx(){
    atr=NOVAL;
    ctr=NOVAL;
  }

_Pharynx(int val1, int val2){
    atr=val1;
    ctr=val2;
  }

~_Pharynx(){
  // cout << "_pharynx deconstructor..." << endl;
}

  int getAtr(){
    return atr;
  }

  void setAtr(int val){
    atr=val;
  }

  int getCtr(){
    return ctr;
  }

  void setCtr(int val){
    ctr=val;
  }

private:
  int atr;
  int ctr;
};


//*******************PHARYNGEAL*******************************

class _Pharyngeal{
public:
_Pharyngeal(){
    glottis=NULL;
    pharynx=NULL;
  }

~_Pharyngeal(){
  // cout << "_pharyngeal deconstructor......." << endl;
  if(glottis!=NULL)
    delete glottis;
  if(pharynx!=NULL)
    delete pharynx;
}

  void setGlottis(Glottis glotArg){
    glottis=glotArg;
  }

  void setPharynx(Pharynx pharArg){
    pharynx=pharArg;
  }

  Glottis getGlottis(){
    return glottis;
  }

  Pharynx getPharynx(){
    return pharynx;
  }

private:
  Glottis glottis;
  Pharynx pharynx;
};


//********************GLIDE************************************

class _Glide{
public:
  _Glide(){
    dominantNode=NOVAL;
    sftPalate=NULL;
    pharyngeal=NULL;
    consonant=NULL;
  }

 _Glide(int val){
    dominantNode=val;
    sftPalate=NULL;
    pharyngeal=NULL;
    consonant=NULL;
  }

  ~_Glide(){
    // cout << "_glide deconstructor..." << endl;
    if(sftPalate!=NULL)
      delete sftPalate;
    if(pharyngeal!=NULL)
      delete pharyngeal;
    if(consonant!=NULL)
      delete consonant;
  }


void setPharyngeal(Pharyngeal pharyngArg){
  pharyngeal=pharyngArg;
}

Pharyngeal getPharyngeal(){
  return pharyngeal;
}
```

```
void setSftP(Soft_Palate sftpArg){
  sftPalate=sftpArg;
}

Soft_Palate getSftP(){
  return sftPalate;
}

void setConsonant(Consonant consArg){
  consonant=consArg;
}

Consonant getConsonant(){
  return consonant;
}

int getDominant(){
  return dominantNode;
}

void setDominant(int val){
   dominantNode=val;
}

private:
  Soft_Palate sftPalate;
  Pharyngeal  pharyngeal;
  Consonant consonant;
  int dominantNode;
};


//************************VOWEL*****************************

class _Vowel{
public:
_Vowel(){
    dominantNode=NOVAL;
    glide=NULL;
    vclFolds=NULL;
}

~_Vowel(){
  // cout << "_vowel deconstructor....." << endl;
  if(glide!=NULL)
    delete glide;
  if(vclFolds!=NULL)
    delete vclFolds;
}


void setVcl(VocalFolds vclArg){
  vclFolds=vclArg;
}

void setGlide(Glide glideArg){
  glide=glideArg;
}

Glide getGlide(){
  return glide;
}

VocalFolds getVclFolds(){
  return vclFolds;
}

int getDominant(){
  return dominantNode;
}

void setDominant(int val){
  dominantNode=val;
}

private:
  Glide glide;
  VocalFolds vclFolds;
  int dominantNode;
};
```

```
int determineDominant(segment data);
void setDominant(int domNode, Vowel tree);
void makeTree(Vowel treeRoot);
void makeTree(Vowel treeRoot, segment seg);
void copyTree(Vowel treeA, Vowel treeB);
void makeChange(Vowel treeA, Vowel treeB, int node);
segment tree2seg(Vowel tree, bundle b, segment seg);
char* convert(int val);
```

```
//////////////////////////////////////////////////////////////////////
//  File: Node.C                                                    //
//  Author: Aaron Maldonado                                         //
//  Functions for using Trees                                       //
//////////////////////////////////////////////////////////////////////


int determineDominant(segment data){
  if(data->get_feature(vowel)==POS)
    return vowel;
  else if(data->get_feature(glide)==POS)
    return glide;
  else if(data->get_feature(consonant)==POS)
    return consonant;
  else
    exit(1);
}

void setDominant(int domNode, Vowel tree){
  if(domNode==vowel)
    tree->setDominant(POS);
  else if(domNode==glide)
    tree->getGlide()->setDominant(POS);
  else if(domNode==consonant)
    tree->getGlide()->getConsonant()->setDominant(POS);
  else
    exit(1);
}


void makeTree(Vowel treeDat, segment seg){

   treeDat->setVcl(new _VocalFolds(seg->get_feature(stiff_vocal_folds),seg->get_feature(
slack_vocal_folds)));

   Body tmpBody = new _Body(seg->get_feature(low), seg->get_feature(high), seg->get_feat
ure(back));

   Blade tmpBlade = new _Blade(seg->get_feature(anterior),seg->get_feature(distributed),
 seg->get_feature(lateral), seg->get_feature(rhotic));

   Lingual tmpLing = new _Lingual();

   tmpLing->setBody(tmpBody);
   tmpLing->setBlade(tmpBlade);

   Lips tmpLips = new _Lips(seg->get_feature(round));

   Consonant tmpCons = new _Consonant(seg->get_feature(sonorant), seg->get_feature(strid
ent), seg->get_feature(continuant));

   tmpCons->setLingual(tmpLing);
   tmpCons->setLips(tmpLips);

   Pharyngeal tmpPharyng = new _Pharyngeal();

   Glottis tmpGlottis = new _Glottis(seg->get_feature(spread_glottis), seg->get_feature(
constricted_glottis));

   Pharynx tmpPhar = new _Pharynx(seg->get_feature(adv_tongue_root),seg->get_feature(con
st_tongue_root));

   tmpPharyng->setGlottis(tmpGlottis);
   tmpPharyng->setPharynx(tmpPhar);

   Soft_Palate tmpPalate = new _Soft_Palate(seg->get_feature(nasal));

   Glide tmpGlide = new _Glide();

   tmpGlide->setPharyngeal(tmpPharyng);
   tmpGlide->setSftP(tmpPalate);
   tmpGlide->setConsonant(tmpCons);

   treeDat->setGlide(tmpGlide);
}

void makeTree(Vowel treeDat){

   treeDat->setVcl(new _VocalFolds());

   Body tmpBody = new _Body();

   Blade tmpBlade = new _Blade();

   Lingual tmpLing = new _Lingual();

   tmpLing->setBody(tmpBody);
   tmpLing->setBlade(tmpBlade);

   Lips tmpLips = new _Lips();

   Consonant tmpCons = new _Consonant();

   tmpCons->setLingual(tmpLing);
   tmpCons->setLips(tmpLips);

   Pharyngeal tmpPharyng = new _Pharyngeal();

   Glottis tmpGlottis = new _Glottis();

   Pharynx tmpPhar = new _Pharynx();

   tmpPharyng->setGlottis(tmpGlottis);
   tmpPharyng->setPharynx(tmpPhar);

   Soft_Palate tmpPalate = new _Soft_Palate();

   Glide tmpGlide = new _Glide();

   tmpGlide->setPharyngeal(tmpPharyng);
   tmpGlide->setSftP(tmpPalate);
   tmpGlide->setConsonant(tmpCons);

   treeDat->setGlide(tmpGlide);
}


void copyTree(Vowel treeA, Vowel treeB){
  //treeA->setDominant(treeB->getDominant());

  treeA->getGlide()->setDominant(treeB->getGlide()->getDominant());

  treeA->getGlide()->getConsonant()->setDominant(treeB->getGlide()->getConsonant()->
getDominant());

  treeA->getGlide()->getConsonant()->setSonorant(treeB->getGlide()->getConsonant()->
getSonorant());

  treeA->getGlide()->getConsonant()->setContinuant(treeB->getGlide()->getConsonant()
->getContinuant());

  treeA->getGlide()->getConsonant()->setStrident(treeB->getGlide()->getConsonant()->
getStrident());
```

```
  treeA->getGlide()->getConsonant()->getLingual()->getBlade()->setLat(treeB->getGlide()-
>getConsonant()->getLingual()->getBlade()->getLat());

  treeA->getGlide()->getConsonant()->getLingual()->getBlade()->setRhotic(treeB->getGlide
()->getConsonant()->getLingual()->getBlade()->getRhotic());

  treeA->getGlide()->getConsonant()->getLingual()->getBlade()->setAnt(treeB->getGlide()-
>getConsonant()->getLingual()->getBlade()->getAnt());

  treeA->getGlide()->getConsonant()->getLingual()->getBlade()->setDistr(treeB->getGlide(
)->getConsonant()->getLingual()->getBlade()->getDistr());

  treeA->getGlide()->getConsonant()->getLingual()->getBody()->setHigh(treeB->getGlide()-
>getConsonant()->getLingual()->getBody()->getHigh());

  treeA->getGlide()->getConsonant()->getLingual()->getBody()->setLow(treeB->getGlide()->
getConsonant()->getLingual()->getBody()->getLow());

  treeA->getGlide()->getConsonant()->getLingual()->getBody()->setBack(treeB->getGlide()-
>getConsonant()->getLingual()->getBody()->getBack());

  treeA->getGlide()->getConsonant()->getLips()->setRound(treeB->getGlide()->getConsonant
()->getLips()->getRound());

  treeA->getGlide()->getPharyngeal()->getPharynx()->setAtr(treeB->getGlide()->getPharyng
eal()->getPharynx()->getAtr());

  treeA->getGlide()->getPharyngeal()->getPharynx()->setCtr(treeB->getGlide()->getPharyng
eal()->getPharynx()->getCtr());

  treeA->getGlide()->getSftP()->setNasal(treeB->getGlide()->getSftP()->getNasal());

  treeA->getVclFolds()->setStiff(treeB->getVclFolds()->getStiff());
  treeA->getVclFolds()->setSlack(treeB->getVclFolds()->getSlack());

  treeA->getGlide()->getPharyngeal()->getGlottis()->setSpread(treeB->getGlide()->getPhar
yngeal()->getGlottis()->getSpread());

  treeA->getGlide()->getPharyngeal()->getGlottis()->setConstr(treeB->getGlide()->getPhar
yngeal()->getGlottis()->getConstr());
}


void makeChange(Vowel target, Vowel STD_tree, int node){
  switch(node){
  case (VOWEL):
    target=STD_tree;
    break;
  case(GLIDE):
    target->setGlide(STD_tree->getGlide());
    break;
  case(CONSONANT):
    target->getGlide()->setConsonant(STD_tree->getGlide()->getConsonant());
    break;
  case(LIPS):
    target->getGlide()->getConsonant()->setLips(STD_tree->getGlide()->getConsonant()->ge
tLips());
    break;
  case(BLADE):
    target->getGlide()->getConsonant()->getLingual()->setBlade(STD_tree->getGlide()->get
Consonant()->getLingual()->getBlade());
    break;
  case(BODY):
    target->getGlide()->getConsonant()->getLingual()->setBody(STD_tree->getGlide()->getC
onsonant()->getLingual()->getBody());
    break;
  case(LINGUAL):
    target->getGlide()->getConsonant()->setLingual(STD_tree->getGlide()->getConsonan
t()->getLingual());
    break;
  case(PHARYNX):
    target->getGlide()->getPharyngeal()->setPharynx(STD_tree->getGlide()->getPharyng
eal()->getPharynx());
    break;
  case(GLOTTIS):
    target->getGlide()->getPharyngeal()->setGlottis(STD_tree->getGlide()->getPharyng
eal()->getGlottis());
    break;
  case(PHARYNGEAL):
    target->getGlide()->setPharyngeal(STD_tree->getGlide()->getPharyngeal());
    break;
  case(SOFT_PALATE):
    target->getGlide()->setSftP(STD_tree->getGlide()->getSftP());
    break;
  case(VOCAL_FOLDS):
    target->setVcl(STD_tree->getVclFolds());
    break;
  default:
    exit(1);
  }
}

segment tree2seg(Vowel tree, bundle b, segment seg){

  int empty=1;

  if(tree->getDominant()==POS){
    b->set_feature(vowel, POS);
  }else if(tree->getGlide()->getDominant()==POS){
    b->set_feature(glide, POS);
  }else if(tree->getGlide()->getConsonant()->getDominant()==POS){
    b->set_feature(consonant, POS);
  }else exit(1);

  b->set_feature(continuant, tree->getGlide()->getConsonant()->getContinuant());
  b->set_feature(sonorant, tree->getGlide()->getConsonant()->getSonorant());
  b->set_feature(strident, tree->getGlide()->getConsonant()->getStrident());

  if(tree->getGlide()->getConsonant()->getLingual()->getBlade()->getLat()!=NOVAL){
    b->set_feature(lateral, tree->getGlide()->getConsonant()->getLingual()->getBlade
()->getLat());
    empty=0;
  }
  if(tree->getGlide()->getConsonant()->getLingual()->getBlade()->getAnt()!=NOVAL){
    b->set_feature(anterior, tree->getGlide()->getConsonant()->getLingual()->getBlad
e()->getAnt());
    empty=0;
  }
  if(tree->getGlide()->getConsonant()->getLingual()->getBlade()->getDistr()!=NOVAL){

    b->set_feature(distributed, tree->getGlide()->getConsonant()->getLingual()->getB
lade()->getDistr());
    empty=0;
  }
  if(tree->getGlide()->getConsonant()->getLingual()->getBlade()->getRhotic()!=NOVAL)
{
    b->set_feature(rhotic, tree->getGlide()->getConsonant()->getLingual()->getBlade(
)->getRhotic());
    empty=0;
  }
  if (!empty){
    b->set_feature(blade, POS);
  }
  empty=1;

  if(tree->getGlide()->getConsonant()->getLingual()->getBody()->getLow()!=NOVAL){
    b->set_feature(low, tree->getGlide()->getConsonant()->getLingual()->getBody()->getLo
w());
    empty=0;
  }
  if(tree->getGlide()->getConsonant()->getLingual()->getBody()->getHigh()!=NOVAL){
    b->set_feature(high, tree->getGlide()->getConsonant()->getLingual()->getBody()->getH
igh());
    empty=0;
  }
  if(tree->getGlide()->getConsonant()->getLingual()->getBody()->getBack()!=NOVAL){
    b->set_feature(back, tree->getGlide()->getConsonant()->getLingual()->getBody()->getB
ack());
    empty=0;
  }
  if (!empty){
    b->set_feature(dorsum, POS);
  }

  empty=1;

  if(tree->getGlide()->getConsonant()->getLips()->getRound()!=NOVAL && b->get_feature(co
nsonant)==POS){
    b->set_feature(round, tree->getGlide()->getConsonant()->getLips()->getRound());
    empty=0;
  }
  if (!empty){
    b->set_feature(lips, POS);
  }

  empty=1;

  if(tree->getGlide()->getPharyngeal()->getGlottis()->getSpread()!=NOVAL){
    b->set_feature(spread_glottis, tree->getGlide()->getPharyngeal()->getGlottis()->getS
pread());
    empty=0;
  }
  if(tree->getGlide()->getPharyngeal()->getGlottis()->getConstr()!=NOVAL){
    b->set_feature(constricted_glottis, tree->getGlide()->getPharyngeal()->getGlottis()-
>getConstr());
    empty=0;
  }
  if (!empty){
    b->set_feature(larynx, POS);
  }

  b->set_feature(nasal, tree->getGlide()->getSftP()->getNasal());
  b->set_feature(adv_tongue_root, tree->getGlide()->getPharyngeal()->getPharynx()->getAt
r());
  b->set_feature(const_tongue_root, tree->getGlide()->getPharyngeal()->getPharynx()->get
Ctr());
  b->set_feature(stiff_vocal_folds, tree->getVc!Folds()->getStiff());
  b->set_feature(slack_vocal_folds, tree->getVclFolds()->getSlack());

  seg->set_data(b);
  return(seg);
}

char* convert(int val){
  switch(val){
  case 0:
    return "main_stress";
  case 1:
    return "reduced";
  case 2:
    return "vowel";
  case 3:
    return "glide";
  case 4:
    return "consonant";
  case 5:
    return "continuant";
  case 6:
    return "sonorant";
  case 7:
    return "strident";
  case 8:
    return "lateral";
  case 9:
    return "rhotic";
  case 10:
    return "lips";
  case 11:
    return "blade";
  case 12:
    return "dorsum";
  case 13:
    return "larynx";
  case 14:
    return "soft_palate";
  case 15:
    return "round";
  case 16:
    return "anterior";
  case 17:
    return "distributed";
  case 18:
    return "high";
  case 19:
    return "low";
  case 20:
    return "back";
  case 21:
    return "adv_tongue_root";
  case 22:
    return "const_tongue_root";
  case 23:
    return "nasal";
  case 24:
    return "stiff_vocal_folds";
  case 25:
    return "slack_vocal_folds";
  case 26:
    return "spread_glottis";
  case 27:
    return "constricted_glottis";
  case -99:
    return "+";
  case -98:
    return "-";
  default:
    return "NOVAL";
  }
}
```

```
// constructs the alternative pronunciation
// IMPORTANT: assuming the contexts already matched.
pronunciation _rule::construct_pronunciation(pronunciation p, int pl, int pr, int rl1, i
nt rrl,
                                             sentence st, int sl, int sr, int rl2, int r
r2, int st_place)
{

  if(get_data(0)->get_bundle()->get_RULErc()==TREE){
    pronunciation new_p;
    segment seg = new _segment(feature);
    bundle b = new _bundle();
    new_p=new _pronunciation();

    Vowel STD_tree;
    Vowel donor_tree;
    Vowel copy_target;
    Vowel blank_tree;
    int whichNode;

    blank_tree = new _Vowel();
    makeTree(blank_tree);

    if(get_cv()==case3)
      {
        for (int i=0;i<pl;i++)
          new_p->add(p->get_data(i)->copy());


        STD_tree=STD->get_treeData(p->get_data(pl)->get_phonemic()->get_STD_index());
        copy_target = new _Vowel();
        donor_tree = new _Vowel();
        makeTree(copy_target);
        makeTree(donor_tree);
        copyTree(copy_target, STD_tree);


        int whichTree=st_place;
        int k=get_sc_r();

        //displayTree(st->get_treeData(whichTree));
        copyTree(donor_tree, st->get_treeData(whichTree));
        //displayTree(donor_tree);
        //exit(1);

        for(int j=get_sc_l(); j<=k; ++j)
          {
            for(int i=VOWEL; i<NOF; ++i)
              {
                if(get_data(j)->get_feature(i) > 0)
                  {
                    makeChange(copy_target, donor_tree, i);
                  }else if(get_data(j)->get_feature(i)==FNOVAL)
                    {
                      makeChange(copy_target, blank_tree, i);
                    }
              }
            if(get_data(j)->get_feature(sonorant)==POS || copy_target->getGlide()->getSf
tP()->getNasal()==POS)
              copy_target->getGlide()->getConsonant()->setSonorant(POS);
            else if(get_data(j)->get_feature(sonorant)==NEG)
              copy_target->getGlide()->getConsonant()->setSonorant(NEG);
            if(get_data(j)->get_feature(continuant)==POS)
              copy_target->getGlide()->getConsonant()->setContinuant(POS);
            else if(get_data(j)->get_feature(continuant)==NEG || copy_target->getGlide()
->getConsonant()->getSonorant()==POS)
              copy_target->getGlide()->getConsonant()->setContinuant(NEG);
            if(get_data(j)->get_feature(strident)==POS)
              copy_target->getGlide()->getConsonant()->setStrident(POS);
            else if(get_data(j)->get_feature(strident)==NEG)
              copy_target->getGlide()->getConsonant()->setStrident(NEG);
          }
        //displayTree(copy_target);
        new_p->add(tree2seg(copy_target, b, seg));

      }else if(get_cv()==case5)
        {
          STD_tree=STD->get_treeData(p->get_data(0)->get_phonemic()->get_STD_index()
);
          donor_tree = new _Vowel();
          copy_target = new _Vowel();
          makeTree(donor_tree);
          makeTree(copy_target);
          copyTree(copy_target, STD_tree);

          int whichTree=sl;
          int k=get_sc_r();

          copyTree(donor_tree, st->get_treeData(whichTree));

          for(int j=get_sc_l(); j<=k; ++j)
            {
              for(int i=VOWEL; i<NOF; ++i)
                {
                  if(get_data(j)->get_feature(i) > 0)
                    {
                      makeChange(copy_target, donor_tree, i);
                    }else if(get_data(j)->get_feature(i)==FNOVAL)
                      {
                        makeChange(copy_target, blank_tree, i);
                      }
                }
              if(get_data(j)->get_feature(sonorant)==POS || copy_target->getGlide()->g
etSftP()->getNasal()==POS)
                copy_target->getGlide()->getConsonant()->setSonorant(POS);
              else if(get_data(j)->get_feature(sonorant)==NEG)
                copy_target->getGlide()->getConsonant()->setSonorant(NEG);
              if(get_data(j)->get_feature(continuant)==POS)
                copy_target->getGlide()->getConsonant()->setContinuant(POS);
              else if(get_data(j)->get_feature(continuant)==NEG || copy_target->getGli
de()->getConsonant()->getSonorant()==POS)
                copy_target->getGlide()->getConsonant()->setContinuant(NEG);
              if(get_data(j)->get_feature(strident)==POS)
                copy_target->getGlide()->getConsonant()->setStrident(POS);
              else if(get_data(j)->get_feature(strident)==NEG)
                copy_target->getGlide()->getConsonant()->setStrident(NEG);
            }
          new_p->add(tree2seg(copy_target, b, seg));
          for (int i=pr+1;i<p->get_NoOfSegments();i++)
            new_p->add(p->get_data(i)->copy());
        }
    delete blank_tree, copy_target, donor_tree;
    return(new_p);
  }

  int i,j;
  segment *segs;
  segs= new segment [MaxContextSize];
  for (i=0;i<MaxContextSize;i++)
    segs[i]=NULL;


  // copy the matched segments from the original word pronunciation to the
  // proper context number in the rule.
  if (p!=NULL)
    {
      j=pl;
      for (i=rl1;i<=rr1;i++)
        segs[i]=p->get_data(j++);
    }

  // copy the matched segments from the sentence to the
  // proper context number in the rule.
  if (st!=NULL)
    {
      j=sl;
      for (i=rl2;i<=rr2;i++)
        segs[i]=st->get_data(j++);
    }

  // start constructing the pronunciation
  pronunciation new_p;
  new_p=new _pronunciation;

  if (cv==case1)
    {
      for (i=0;i<pl;i++)
        new_p->add(p->get_data(i)->copy());
      for (i=sc_l;i<=sc_r;i++)
        new_p->add(construct_segment(segs,i));
      for (i=pr+1;i<p->get_NoOfSegments();i++)
        new_p->add(p->get_data(i)->copy());
    }
  else if (cv==case2)
    {
      for (i=0;i<pl;i++)
        new_p->add(p->get_data(i)->copy());
      for (i=sc_l;i<=sc_r;i++)
        new_p->add(construct_segment(segs,i));
    }
  else if (cv==case3)
    {
      // copy the unchanged segments from
      // origional word pronunciation to the alternative pronunciation.

      for (i=0;i<pl;i++)
         new_p->add(p->get_data(i)->copy());
      // construct the segments that changes with the context
      for (i=sc_l;i<=sc_r;i++)
         new_p->add(construct_segment(segs,i));
    }
  else if (cv==case4)
    {
      for (i=0;i<pl;i++)
        new_p->add(p->get_data(i)->copy());
      for (i=sc_l;i<=sc_r;i++)
        new_p->add(construct_segment(segs,i));
    }
  else if (cv==case5)
    {
      // construct the segments that changes with the context
      for (i=sc_l;i<=sc_r;i++)
         new_p->add(construct_segment(segs,i));

      // copy the unchanged segments from
      // origional word pronunciation to the alternative pronunciation.
      for (i=pr+1;i<p->get_NoOfSegments();i++)
        new_p->add(p->get_data(i)->copy());
    }
  else if (cv==case6)
    {
      for (i=sc_l;i<=sc_r;i++)
         new_p->add(construct_segment(segs,i));
      for (i=pr+1;i<p->get_NoOfSegments();i++)
        new_p->add(p->get_data(i)->copy());
    }

  delete [] segs;
  return (new_p);

}
```

```
char* disMode(mode_value mv);
void morphemes(lexicon lex);
void plurals(lexicon lex);
void expand_ion(lexicon lex);
```

```
////////////////////////////////////////////////////////////////////
//  File: morphemes.C                                             //
//  Author: Aaron Maldonado                                       //
//  Functions for using MORPHEMES                                 //
////////////////////////////////////////////////////////////////////


void morphemes(lexicon lex){
  //expands words with /ion/ suffix
  expand_ion(lex);

  //expands with plurals
  plurals(lex);

  //other operations
}


void plurals(lexicon lex){
  int last;
  int loop1, loop2;
  int* partsOfSpeech;
  char *buffer;
  char *temp;
  segment seg;

  loop1=lex->get_NoOfData();

  for(int i=0; i<loop1; ++i)
    {
      partsOfSpeech=lex->get_data(i)->getPOS();
      if(partsOfSpeech[N]!=POS)
        continue;
      loop2=lex->get_data(i)->get_NoOfData();
      for(int j=0; j<loop2; j++)
        {
          last=lex->get_data(i)->get_data(j)->get_NoOfSegments()-1;

          if(
             lex->get_data(i)->get_data(j)->get_data(last)->get_feature(strident)==POS
             && lex->get_data(i)->get_data(j)->get_data(last)->get_feature(blade)==POS
)
            {
              lex_word new_entry = new _lex_word();
              pronunciation new_p;

              buffer=lex->get_data(i)->get_Label();
              temp=strcopy(buffer);
              strcat(temp,"es");
              //cout << temp << endl;
              new_entry->set_Label(temp);

              // get original pieces of pronunciation
              new_p=lex->get_data(i)->get_data(j)->copy();

              // add schwa
              seg=STD->get_data(schwa_index)->copy();
              seg->set_mode(either);
              new_p->add(seg);

              // add /z/
              seg=STD->get_data(z_index)->copy();
              seg->set_mode(either);
              new_p->add(seg);

              // add plural pronunciation to new lexical entry
              new_entry->add(new_p);

              // add new word to lexicon
              lex->add(new_entry);

              continue;
            }

          lex_word new_entry = new _lex_word();
          pronunciation new_p;

          // get original pieces of pronunciation
          new_p=lex->get_data(i)->get_data(j)->copy();

          seg = STD->get_data(s_index)->copy();

          buffer=lex->get_data(i)->get_Label();
          temp=strcopy(buffer);
          strcat(temp,"s");
          //cout << temp << endl;
          new_entry->set_Label(temp);

          if(lex->get_data(i)->get_data(j)->get_NoOfSegments()==1 ||
             lex->get_data(i)->get_data(j)->get_data(last)->get_feature(slack_vocal_
folds)!=NEG)
            seg->set_feature(slack_vocal_folds, POS);

          // make it have the right format for a pronunciation
          seg->set_mode(either);

          // add plural ending depending on voicing of previous consonant
          new_p->add(seg);

          // add plural pronunciation to new lexical entry
          new_entry->add(new_p);

          // add new word to lexicon
          lex->add(new_entry);
        }
    }
  if(buffer!=NULL)
    delete buffer;
  if(temp!=NULL)
    delete temp;
}


void expand_ion(lexicon lex){
  int loop1, loop2, last;
  char *buffer;
  char *temp;
  segment new_seg;
  int* partsOfSpeech;

  loop1=lex->get_NoOfData();

  for(int i=0; i<loop1; ++i)
    {
      partsOfSpeech=lex->get_data(i)->getPOS();
      if(partsOfSpeech[V]!=POS)
        continue;
      loop2=lex->get_data(i)->get_NoOfData();
      for(int j=0; j<loop2; j++)
        {
          last=lex->get_data(i)->get_data(j)->get_NoOfSegments()-1;
          if(lex->get_data(i)->get_data(j)->get_data(last)->get_feature(blade)==POS && l
ex->get_data(i)->get_data(j)->get_data(last)->get_feature(anterior)==POS && lex->get_dat
a(i)->get_data(j)->get_data(last)->get_feature(distributed)==NEG && lex->get_data(i)->ge
t_data(j)->get_data(last)->get_feature(constricted_glottis)==NEG && lex->get_data(i)->ge
t_data(j)->get_data(last)->get_feature(slack_vocal_folds)==NEG)
            {
              lex_word new_entry = new _lex_word();
              pronunciation new_p = new _pronunciation();

              buffer=lex->get_data(i)->get_Label();
              temp=strcopy(buffer);
              strcat(temp,"ion");
              cout << temp << endl;
              new_entry->set_Label(temp);

              // get original pieces of pronunciation
              for (int k=0;k<last;k++)
                new_p->add(lex->get_data(i)->get_data(j)->get_data(k)->copy());

              // add /sh/
              new_seg = STD->get_data(sh_index)->copy();
              new_seg->set_mode(either);
              new_p->add(new_seg);

              // add /x/
              new_seg = STD->get_data(schwa_index)->copy();
              new_seg->set_mode(either);
              new_p->add(new_seg);

              // add /n/
              new_seg = STD->get_data(n_index)->copy();
              new_seg->set_mode(either);
              new_p->add(new_seg);

              // add plural pronunciation to new lexical entry
              new_entry->add(new_p);

              // add new word to lexicon
              lex->add(new_entry);
            }
        }
    }
  if(buffer!=NULL)
    delete buffer;
  if(temp!=NULL)
    delete temp;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include <string.h>
#include <time.h>

//**Do not change the order of these include files

// definitions
#include "def.H"
// utilities used for classes
#include "util.H"
#include "util.C"
// templates
#include "Array.H"
// Main Headers
#include "../TreeRules/treeClasses.H"
#include "Classes.H"
#include "ClassFunc.H"
#include "node.H"
#include "morphemes.H"
// Main Code
#include "Classes.C"
#include "ClassFunc.C"
#include "node.C"
#include "morphemes.C"


void main(int argc, char *argv[])
{
  int i;
  char backup[500];

  // timing purpose
  time_t start;
  time_t finish;
  double interval;

  // filename variables
  const int sz=300;
  char ruleset_fn[sz], st_fn[sz], lexicon_fn[sz], sentence_dir[sz];

  // define defaluts
  strcpy(ruleset_fn,"rules.rs");
  strcpy(st_fn,"standard.label");
  strcpy(lexicon_fn,"lexicon.lex");
  strcpy(sentence_dir,".");

  // check for argument count
  // if out of range output syntax or if one of the help switch is invoked.
  if ((argc>5) ||
      ((argc==2) && (strsame(argv[1],"--help") ||
                     strsame(argv[1],"-help") ||
                     strsame(argv[1],"help") ||
                     strsame(argv[1],"info"))))

    {
      printf("\nSyntax: match3 [sentence-dir] [RuleSet] [Standard Template] [Lexicon]\n\
n");
      printf("  [xxxxx] are optional parameters.\n");
      printf("  If not specified they will be taken as:\n");
      printf("  [sentence-directory]: current directory\n");
      printf("  [RuelSet]: rules.rs\n");
      printf("  [Standard Template]: standard.label\n");
      printf("  [Lexicon]: lexicon.lex\n");
      printf("\n  Note: To specify the current directory, use a single '.'");
      cout << "\n\n";
      return 0;
    }

  if (argc>=2)
    strcpy(sentence_dir,argv[1]);
  if (argc>=3)
    strcpy(ruleset_fn,argv[2]);
  if (argc>=4)
    strcpy(st_fn,argv[3]);
  if (argc==5)
    strcpy(lexicon_fn,argv[4]);

  FILE* readptr;
  // Welcome msg
  cout << "\nWelcome to Match G3, system booting .....\n";

  // Global Init - STD
  cout << "\n-=-=-=-= SYSTEM INITIALIZATION =-=-=-=-\n\n";

  if (getenv("HOSTTYPE")!=NULL)
    cout << "Booting up on a " << getenv("HOSTTYPE") << " system ("<< getenv("HOST")
<< ") ...\n" << endl;


  // load Standard Template
  cout << "Loading Standard Template: " << st_fn << " ..... ";
  readptr=readfile(st_fn);;
  if (readptr==0) {cout<< "\nFile Not Found, system halt ...\n\n"; exit(1);}
  STD=new _sentence (readptr);
  closefile(readptr);
  cout << "Done.\n";

  // load Lexicon
  cout << "Loading Lexicon: " << lexicon_fn << " ..... ";
  readptr=readfile(lexicon_fn);
  if (readptr==0) {cout<< "\nFile Not Found, system halt ...\n\n"; exit(1);}
  lexicon lex= new _lexicon (readptr);
  closefile(readptr);
  cout << "Done.\n";


  // do morpheme expansion on lexicon
  cout << "Expanding Lexicon with Morphemes" << " ..... ";
  morphemes(lex);
  cout << "Done.\n";


  // Load Rule Set
  cout << "Loading Rule Set: " << ruleset_fn << " ..... ";
  readptr=readfile(ruleset_fn);
  if (readptr==0) {cout<< "\nFile Not Found, system halt ...\n\n"; exit(1);}
  rules rs = new _rules (readptr);
  closefile(readptr);
  cout << "Done.\n";

  cout << "\n=-=-=-=- INITIALIZATION COMPLETE -=-=-=-=\n\n";


  // Welcome User
  char user_name [100];
  if (getenv("USER")!=NULL)
    {
      strcpy(user_name,getenv("USER"));
      if (strsame(user_name,"nyker"))
        strcpy(user_name,"Yong");
      else if (strsame(user_name,"choi"))
        strcpy(user_name,"Elizabeth");
      else if (strsame(user_name,"stevens"))
        strcpy(user_name,"Ken");
      else if (strsame(user_name,"manuel"))
        strcpy(user_name,"Sharon");
      else if (strsame(user_name,"aaron"))
        strcpy(user_name,"Aaron");
      else if (strsame(user_name,"stef"))
        strcpy(user_name,"Stefanie");
      else if (strsame(user_name,"gow"))
        strcpy(user_name,"Dave");

      cout << "Welcome to Match G3, " << user_name << "!\n\n";
    }

  char sentence_fn[sz], filename[sz];
  sentence sent=NULL; tree t=NULL;
  char Message[] ="* Please enter the filename of a sentence to run. (w/o .label)\n* Typ
e 'help' at any time to list all available commands.\n", prompt[]="Match G3> ";

  cout << Message << prompt;
  cin >> sentence_fn;
  while (!(strsame(sentence_fn,"quit") || strsame(sentence_fn,"bye")))
    {
      if (strsame(sentence_fn,"help") || strsame(sentence_fn,"?"))
        {
          cout << "\nCommands:\nhelp - this message\nversion - displays the version numb
er\nfilename - runs the system with the sentence filename.label\nstatus - displays the m
atching tree status\nls - lists related system files in the current directory\nverbose -
 toggles the verbose mode on and off\nreload-template - loads a new set of template\nrel
oad-lexicon - loads a new set of lexicon\nreload-ruleset - loads a new set of rules\nreb
oot - reboots the system\nquit/bye - exits the system\n\n";
        }
      else if (strsame(sentence_fn,"version"))
        {
          cout << "\nMatch G3 Version 1.5 - 7/14/1998\n\n";
        }
      else if (strsame(sentence_fn,"status"))
        {
          // STD info
          if (STD->get_content()!=NULL)
            cout << "\nStandard Template: " << STD->get_content() << endl;
          else cout << "\nStandard Template: not specified" << endl;

          // Lexicon info
          if (lex->get_Label()!=NULL)
            cout << "Lexicon: " << lex->get_Label() << endl;
          else cout << "\nLexicon: not specified" << endl;

          // Rule Set info
          if (rs->get_Label()!=NULL)
            {
              cout << "Rule Set: " << rs->get_Label() <<endl;
              for (i=0;i<rs->get_NoOfData();i++)
                cout << "   Rule " << i+1 << ": " << rs->get_data(i)->get_content() << e
ndl;
            }
          else cout << "\nRule Set: not specified" << endl << endl;

          if (t!=NULL)
            {
              cout << endl;
              t->status();
              cout << endl;
            }
          else cout << "\nSentence is not loaded.\n\n";
        }
      else if (strsame(sentence_fn,"verbose"))
        {
          if (verbose==0)
            {
              verbose=1;
              cout << endl << "Verbose Mode Toggled On.\n\n";
            }
          else
            {
              verbose=0;
              cout << endl << "Verbose Mode Toggled Off.\n\n";
            }
        }
      else if (strsame(sentence_fn,"reload-template"))
        {
          strcpy(backup,st_fn);
          cout << "New Template (w/o .label): ";
          cin >> st_fn;
          strcat(st_fn,".label");
          readptr=readfile(st_fn);
          if (readptr!=0)
            {
              cout << "\nPurging old Matching Tree ..... ";
              if (sent!=NULL)
                {delete sent; sent=NULL;}
              if (t!=NULL)
                {delete t; t=NULL;}
              cout << "Done.\n";

              cout << "\nPurging old Standard Template ..... ";
              if (STD!=NULL)
                {delete STD; STD=NULL;}
              cout << "Done.\n";

              cout << "Loading new Standard Template: " << st_fn << " ..... ";
              STD=new _sentence (readptr);
              closefile(readptr);
              cout << "Done.\n\n";
            }
          else
            {
              cout << "\nFile Not Found. New template not loaded.\n\n";
              strcpy(st_fn,backup);
            }
        }
      else if (strsame(sentence_fn,"reload-lexicon"))
        {
          cout << "New Lexicon (w/o .lex): ";
          cin >> lexicon_fn;
          strcat(lexicon_fn,".lex");
          readptr=readfile(lexicon_fn);
          if (readptr!=0)
            {
              cout << "\nPurging old Lexicon ..... ";
              if (lex!=NULL)
                {delete lex; lex=NULL;}
              cout << "Done.\n";

              cout << "Loading new Lexicon: " << lexicon_fn << " ..... ";
              lex=new _lexicon (readptr);
              closefile(readptr);
              cout << "Done.\n\n";
            }
          else
            {
              cout << "\nFile Not Found. New lexicon not loaded.\n\n";
              strcpy(st_fn,backup);
            }
      }
    else if (strsame(sentence_fn,"reload-ruleset"))
      {
        cout << "New Rule Set (w/o .rs): ";
        cin >> ruleset_fn;
        strcat(ruleset_fn,".rs");
        readptr=readfile(ruleset_fn);
        if (readptr!=0)
          {
            cout << "Purging old rule set ..... ";
            if (rs!=NULL)
              {delete rs; rs=NULL;}
            cout << "Done.\n";

            cout << "Loading new Rule Set: " << ruleset_fn << " ..... ";
            rs=new _rules (readptr);
            closefile(readptr);
            cout << "Done.\n\n";
          }
        else
          {
            cout << "\nFile Not Found. New rule set not loaded.\n\n";
            strcpy(st_fn,backup);
          }
      }
    else if (strsame(sentence_fn,"reboot"))
      {
        // Clean up system
        cout << "\nSystem Going Down ....\n";

        cout << "Purging Match Tree ..... ";
        if (sent!=NULL)
          {delete sent; sent=NULL;}
        if (t!=NULL)
          {delete t; t=NULL;}
        cout << "Done.\n";

        cout << "Purging Standard Template ..... ";
        if (STD!=NULL)
          {delete STD; STD=NULL;}
        cout << "Done.\n";

        cout << "Purging Lexicon ..... ";
        if (lex!=NULL)
          {delete lex; lex=NULL;}
        cout << "Done.\n";

        cout << "Purging Rule Set ..... ";
        if (rs!=NULL)
          {delete rs; rs=NULL;}
        cout << "Done.\n";

        cout << "\nSystem Rebooting ....\n";

        // Global Init - STD
        cout << "\n-=-=-=-= SYSTEM INITIALIZATION =-=-=-=-\n\n";

        // host detection
        if (getenv("HOSTTYPE")!=NULL)
          cout << "Booting up on a " << getenv("HOSTTYPE") << " system ("<< getenv("HO
ST") << ") ...\n" << endl;

        // load Standard Template
        cout << "Loading Standard Template: " << st_fn << " ..... ";
        readptr=readfile(st_fn);
        if (readptr==0) {cout<< "\nFile Not Found, system halt ...\n\n"; exit(1);}
        STD=new _sentence (readptr);
        closefile(readptr);
        cout << "Done.\n";

        // load Lexicon
        cout << "Loading Lexicon: " << lexicon_fn << " ..... ";
        readptr=readfile(lexicon_fn);
        if (readptr==0) {cout<< "\nFile Not Found, system halt ...\n\n"; exit(1);}
        lex= new _lexicon (readptr);
        closefile(readptr);
        cout << "Done.\n";

        // do morpheme expansion on lexicon
        cout << "Expanding Lexicon with Morphemes" << " ..... ";
        morphemes(lex);
        cout << "Done.\n";

        // Load Rule Set
        cout << "Loading Rule Set: " << ruleset_fn << " ..... ";
        readptr=readfile(ruleset_fn);
        if (readptr==0) {cout<< "\nFile Not Found, system halt ...\n\n"; exit(1);}
        rs = new _rules (readptr);
        closefile(readptr);
        cout << "Done.\n";

        cout << "\n=-=-=-=- INITIALIZATION COMPLETE -=-=-=-=\n\n";
      }
    else if (strsame(sentence_fn,"ls"))
      {
        // Note the following is UNIX only
        cout << "\nSentences:\n";
        system("ls *.label");

        cout << "\nRule Sets:\n";
        system("ls *.rs");

        cout << "\nRules:\n";
        system("ls *.rul");

        cout << "\nLexicon:\n";
        system("ls *.lex");

        cout << endl;
      }
    else
      {
        // start timer
        start=time(NULL);

        if (sent!=NULL)
          {delete sent; sent=NULL;}
        if (t!=NULL)
          {delete t; t=NULL;}

        strcpy(filename,sentence_dir);
        strcat(filename,"/");
        strcat(filename,sentence_fn);
        strcat(filename,".label");


        // Load sentence
        cout << "Loading Sentence: " << filename << " ..... ";
        readptr=readfile(filename);
        if (readptr==0)
          {
            cout << "\nfile not found.\n\n";
            // ask for input
            cout << prompt;
            cin >> sentence_fn;

            continue;
          }

        sent = new _sentence (readptr);
        closefile(readptr);

        cout << "Done.\n";


        // Grow matching tree
        cout << "Growing Matching Tree ... ";
        t = new _tree (sent);
        t->grow(rs,lex);
        //exit(1);
        finish=time(NULL);
        cout << "Done. [" << difftime(finish,start) << " sec]\n";


        // print status if verbose on
        if (verbose)
          {
            cout << "\nTree Status:\n";
            t->status();
            cout << endl;
          }

        // Output Answer
        cout << "\n-=-=-=-= ANSWER =-=-=-=-\n";
        t->output(screen);
      }

    // ask for input
    cout << prompt;
    cin >> sentence_fn;
  }
  cout << "\nGoodbye, " << user_name << ". Thank you for using Match G3.\n" << endl;
}
```

APPENDIX C

```
****************************************************************
*       ORIGINAL MATCHER RESULTS FOR:                          *
*                                                              *
*       "Catch a balloom before Mary does."                    *
*                                                              *
****************************************************************

#pragma ident   "@(#)gprof.flat.blurb   1.8     93/06/07 SMI"


flat profile:

 %          the percentage of the total running time of the
time        program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self       the number of seconds accounted for by this
seconds     function alone.  This is the major sort for this
            listing.

calls       the number of times this function was invoked, if
            this function is profiled, else blank.

 self       the average number of milliseconds spent in this
ms/call     function per call, if this function is profiled,
            else blank.

 total      the average number of milliseconds spent in this
ms/call     function and its descendents per call, if this
            function is profiled, else blank.

name        the name of the function.  This is the minor sort
            for this listing. The index shows the location of
            the function in the gprof listing. If the index is
            in parenthesis it shows where it would appear in
            the gprof listing if it were to be printed.
```

```
granularity: each sample hit covers 4 byte(s) for 0.99% of 1.01 seconds

  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 52.5      0.53     0.53                             internal_mcount [1]
 12.9      0.66     0.13  1594628     0.00     0.00  get_feature__7_bundlei [20]
  7.9      0.74     0.08   881063     0.00     0.00  get_feature__8_segmenti   <cycle 1>
[14]
  7.9      0.82     0.08      900     0.09     0.30  feature2index__9_sentenceP7_bundle [
4]
  2.0      0.84     0.02    22019     0.00     0.00  _smalloc [31]
  2.0      0.86     0.02     7981     0.00     0.00  _memcpy [32]
  2.0      0.88     0.02     5320     0.00     0.00  _memset [33]
  2.0      0.90     0.02     3445     0.01     0.01  feature_match__5_ruleP8_segmentT1 [2
5]
  2.0      0.92     0.02                             mcount (447)
  1.0      0.93     0.01   102343     0.00     0.00  _return_zero [42]
  1.0      0.94     0.01    30589     0.00     0.00  _malloc_unlocked [37]
  1.0      0.95     0.01    25077     0.00     0.00  realfree [38]
  1.0      0.96     0.01    20336     0.00     0.00  _free_unlocked [35]
  1.0      0.97     0.01     1747     0.01     0.02  matched__5_ruleP9_sentenceiiii [28]
  1.0      0.98     0.01     1698     0.01     0.02  matched__5_ruleP14_pronunciationiiii
 [29]
  1.0      0.99     0.01        6     1.67     1.67  _open [44]
  1.0      1.00     0.01                             __throw [36]
  1.0      1.01     0.01                             execute_cfa_insn [41]
  0.0      1.01     0.00    51170     0.00     0.00  _mutex_lock [786]
  0.0      1.01     0.00    51169     0.00     0.00  mutex_unlock [87]
  0.0      1.01     0.00    41985     0.00     0.00  strcmp [88]
  0.0      1.01     0.00    34326     0.00     0.00  get_data__9_segmentsi [89]
  0.0      1.01     0.00    30463     0.00     0.00  malloc [24]
  0.0      1.01     0.00    29230     0.00     0.00  cleanfree [64]
  0.0      1.01     0.00    26363     0.00     0.00  get_feature__13_phonemic_repi      <c
ycle 1> [90]
  0.0      1.01     0.00    20336     0.00     0.00  free [34]
  0.0      1.01     0.00    19831     0.00     0.00  get_symbol__7_bundle [91]
  0.0      1.01     0.00    19451     0.00     0.00  get_symbol__8_segment [92]
  0.0      1.01     0.00    19371     0.00     0.00  strsame__FPcT0 [93]
  0.0      1.01     0.00    16378     0.00     0.00  get_NoOfSegments__9_segments [94]
  0.0      1.01     0.00     9526     0.00     0.00  exact_match__Fii [95]
  0.0      1.01     0.00     8332     0.00     0.00  check_STD__Fv [96]
  0.0      1.01     0.00     8306     0.00     0.00  strlen [97]
  0.0      1.01     0.00     6574     0.00     0.00  get_data__t5Array2ZP9_lex_wordZ9_lex
_wordi [98]
  0.0      1.01     0.00     6110     0.00     0.00  _._8_segment [787]
  0.0      1.01     0.00     6110     0.00     0.00  __13_phonemic_repi12marker_value [78
8]
  0.0      1.01     0.00     6110     0.00     0.00  __8_segment10mode_value [789]
  0.0      1.01     0.00     6110     0.00     0.00  copy__13_phonemic_rep [99]
  0.0      1.01     0.00     6110     0.00     0.00  copy__8_segment [100]
  0.0      1.01     0.00     4608     0.00     0.00  get_NoOfData__t5Array2ZP5_ruleZ5_rul
e [101]
  0.0      1.01     0.00     4588     0.00     0.00  strcopy__FPc [46]
  0.0      1.01     0.00     3288     0.00     0.06  construct_pronunciation__5_ruleP14_p
ronunciationP9_sentencei [10]
  0.0      1.01     0.00     3288     0.00     0.00  get_data__t5Array2ZP5_ruleZ5_rulei [
102]
  0.0      1.01     0.00     3278     0.00     0.00  get_NoOfData__t5Array2ZP14_pronuncia
tionZ14_pronunciation [103]
  0.0      1.01     0.00     3189     0.00     0.00  strcpy [104]
  0.0      1.01     0.00     2229     0.00     0.00  is_alphabet__Fc [105]
  0.0      1.01     0.00     2128     0.00     0.00  __tf13RunTimeStatus [790]
  0.0      1.01     0.00     1967     0.00     0.00  t_delete [106]
  0.0       1.01     0.00     1915     0.00     0.14  compress__8_segment [5]
  0.0       1.01     0.00     1788     0.00     0.00  bundle_match__FP8_segmentT0 [47]
  0.0       1.01     0.00     1754     0.00     0.00  get_data__t5Array2ZP14_pronunciation
Z14_pronunciationi [107]
  0.0       1.01     0.00     1748     0.00     0.00  _._14_pronunciation [791]
  0.0       1.01     0.00     1748     0.00     0.00  _._9_segments [792]
  0.0       1.01     0.00     1748     0.00     0.00  matching__5_treeP14_pronunciationP9_
sentencei [48]
  0.0       1.01     0.00     1698     0.00     0.00  derive_main_stress__13_phonemic_rep
<cycle 1> [108]
  0.0       1.01     0.00     1698     0.00     0.00  derive_reduced__13_phonemic_rep   <c
ycle 1> [109]
  0.0       1.01     0.00     1644     0.00     0.00  __14_pronunciation3nop [793]
  0.0       1.01     0.00     1644     0.00     0.00  __9_segments3nop [794]
  0.0       1.01     0.00     1644     0.00     0.00  copy__14_pronunciation [58]
  0.0       1.01     0.00     1542     0.00     0.00  get_NoOfData__t5Array2ZP9_lex_wordZ9
_lex_word [110]
  0.0       1.01     0.00     1536     0.00     0.00  _._9_lex_word [795]
  0.0       1.01     0.00     1536     0.00     0.00  _._t5Array2ZP14_pronunciationZ14_pro
nunciation [796]
  0.0       1.01     0.00     1536     0.00     0.00  __9_lex_word3nop [797]
  0.0       1.01     0.00     1536     0.00     0.00  __t5Array2ZP14_pronunciationZ14_pron
unciation [798]
  0.0       1.01     0.00     1536     0.00     0.12  apply_rules__9_lex_wordP6_rulesP9_se
ntencei [11]
  0.0       1.01     0.00     1536     0.00     0.00  copy__9_lex_word [50]
  0.0       1.01     0.00     1536     0.00     0.00  get_expanded__9_lex_word [111]
  0.0       1.01     0.00     1525     0.00     0.00  _ungetc_unlocked [799]
  0.0       1.01     0.00     1525     0.00     0.00  ungetc [112]
  0.0       1.01     0.00     1292     0.00     0.00  get_word__FP4FILE [57]
  0.0       1.01     0.00     1015     0.00     0.14  __8_segment10mode_valueT1P4FILE13hea
ding_valueT4 [16]
  0.0       1.01     0.00      942     0.00     0.00  __13_phonemic_repP4FILE [800]
  0.0       1.01     0.00      942     0.00     0.00  phoneme2index__9_sentencePc [113]
  0.0       1.01     0.00      939     0.00     0.00  t_splay [114]
  0.0       1.01     0.00      900     0.00     0.30  __13_phonemic_repP7_bundle [6]
  0.0       1.01     0.00      888     0.00     0.00  check_fvalue__Fi [115]
  0.0       1.01     0.00      888     0.00     0.00  set_feature__7_bundleii [116]
  0.0       1.01     0.00      888     0.00     0.14  set_feature__8_segmentii [23]
  0.0       1.01     0.00      532     0.00     0.00  __tcf_0 [801]
  0.0       1.01     0.00      513     0.00     0.00  string2feature_index__FPc [117]
  0.0       1.01     0.00      458     0.00     0.00  add__9_segmentsP8_segment [54]
  0.0       1.01     0.00      458     0.00     0.00  set_mode__8_segment10mode_value [55]
  0.0       1.01     0.00      410     0.00     0.00  get_time__7_bundle [118]
  0.0       1.01     0.00      380     0.00     0.00  __7_bundleP13_phonemic_rep [53]
  0.0       1.01     0.00      380     0.00     0.00  expand__8_segment [51]
  0.0       1.01     0.00      380     0.00     0.00  get_Header__7_bundle [119]
  0.0       1.01     0.00      380     0.00     0.00  get_STD_index__13_phonemic_rep [120]
  0.0       1.01     0.00      380     0.00     0.00  get_Value__7_bundle [121]
  0.0       1.01     0.00      380     0.00     0.00  get_bundle__8_segment [122]
  0.0       1.01     0.00      380     0.00     0.00  get_marker__13_phonemic_rep [123]
  0.0       1.01     0.00      380     0.00     0.00  get_prosody__7_bundle [124]
  0.0       1.01     0.00      380     0.00     0.00  get_rc__7_bundle [125]
  0.0       1.01     0.00      380     0.00     0.00  set_stress__7_bundle12marker_value [
126]
  0.0       1.01     0.00      380     0.00     0.00  transfer_features__7_bundleP7_bundle
 [52]
  0.0       1.01     0.00      378     0.00     0.00  add__t5Array2ZP14_pronunciationZ14_p
ronunciationP14_pronunciation [127]
  0.0       1.01     0.00      368     0.00     0.00  get_stress__7_bundle [128]
  0.0       1.01     0.00      362     0.00     0.00  _sbrk [66]
  0.0       1.01     0.00      362     0.00     0.00  _sbrk_unlocked [802]
  0.0       1.01     0.00      298     0.00     0.00  get_line__FP4FILE [62]
  0.0       1.01     0.00      278     0.00     0.52  __9_segmentsi10mode_valueT2P4FILE13h
eading_value [15]
```

```
  0.0       1.01     0.00      274     0.00     0.52  __14_pronunciationP4FILE [19]
  0.0       1.01     0.00      256     0.00     0.55  __9_lex_wordP4FILE [18]
  0.0       1.01     0.00      256     0.00     0.00  __t5Array2ZP14_pronunciationZ14_pron
unciationi [803]
  0.0       1.01     0.00      256     0.00     0.00  add__t5Array2ZP9_lex_wordZ9_lex_word
P9_lex_word [129]
  0.0       1.01     0.00      256     0.00     0.00  set_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciationPc [65]
  0.0       1.01     0.00      256     0.00     0.00  strcat [130]
  0.0       1.01     0.00      223     0.00     0.00  file_get_item__FP4FILEPc [61]
  0.0       1.01     0.00      181     0.00     0.00  _morecore [67]
  0.0       1.01     0.00      150     0.00     0.00  heading2string__F13heading_value [13
1]
  0.0       1.01     0.00      146     0.00     0.00  get_state__8_segment [132]
  0.0       1.01     0.00      115     0.00     0.00  _realbufend [804]
  0.0       1.01     0.00      113     0.00     0.00  atoi [133]
  0.0       1.01     0.00      104     0.00     0.00  __14_pronunciation [805]
  0.0       1.01     0.00      104     0.00     0.00  __9_segmentsi10mode_value [806]
  0.0       1.01     0.00      104     0.00     0.00  change_subtitle__9_segments13heading
_valueT1 [134]
  0.0       1.01     0.00      104     0.00     1.23  construct_pronunciation__5_ruleP14_p
ronunciationiiiiP9_sentenceiiii [21]
  0.0       1.01     0.00      104     0.00     1.21  construct_segment__5_rulePP8_segment
i [22]
  0.0       1.01     0.00      104     0.00     0.00  set_title__9_segments13heading_value
 [135]
  0.0       1.01     0.00       73     0.00     0.02  __7_bundleP4FILE13heading_valueT2 [5
6]
  0.0       1.01     0.00       73     0.00     0.01  file_get_rorc__7_bundleP4FILE [63]
  0.0       1.01     0.00       69     0.00     0.00  nvmatch [136]
  0.0       1.01     0.00       49     0.00     0.00  .udiv [137]
  0.0       1.01     0.00       48     0.00     0.00  _fwrite_unlocked [79]
  0.0       1.01     0.00       48     0.00     0.00  fwrite [138]
  0.0       1.01     0.00       36     0.00     0.00  get_NoOfData__t5Array2ZP5_wordZ5_wor
d [139]
  0.0       1.01     0.00       32     0.00     0.00  _write [807]
  0.0       1.01     0.00       32     0.00     0.00  _xflsbuf [808]
  0.0       1.01     0.00       30     0.00     0.00  __flsbuf [809]
  0.0       1.01     0.00       30     0.00     0.00  get_data__t5Array2ZP5_wordZ5_wordi [
140]
  0.0       1.01     0.00       30     0.00     0.00  get_time__8_segment [141]
  0.0       1.01     0.00       24     0.00     0.00  get_NoOfSegments__5_word [142]
  0.0       1.01     0.00       20     0.00     0.00  ___errno [810]
  0.0       1.01     0.00       20     0.00     0.00  _return_negone [811]
  0.0       1.01     0.00       20     0.00     0.00  thr_main [143]
  0.0       1.01     0.00       16     0.00     0.00  _fflush_u [812]
  0.0       1.01     0.00       12     0.00     0.00  __filbuf [69]
  0.0       1.01     0.00       12     0.00     0.00  _read [813]
  0.0       1.01     0.00       12     0.00     0.00  get_word__5_word [144]
  0.0       1.01     0.00        8     0.00     0.00  _findbuf [68]
  0.0       1.01     0.00        8     0.00     0.00  _ioctl [814]
  0.0       1.01     0.00        8     0.00     0.00  _isatty [815]
  0.0       1.01     0.00        8     0.00     0.00  _setbufend [816]
  0.0       1.01     0.00        7     0.00     0.00  _lseek64 [817]
  0.0       1.01     0.00        7     0.00    27.40  grow_aux__5_treeP6_rulesP8_lexiconi
[7]
  0.0       1.01     0.00        6     0.00     0.00  _._8_lexicon [818]
  0.0       1.01     0.00        6     0.00     0.00  _._t5Array2ZP9_lex_wordZ9_lex_word [
819]
  0.0       1.01     0.00        6     0.00     0.00  __5_wordPci [72]
  0.0       1.01     0.00        6     0.00     0.00  __8_lexicon3nop [820]
  0.0       1.01     0.00        6     0.00     0.00  __t5Array2ZP5_wordZ5_wordi [821]
  0.0       1.01     0.00        6     0.00     0.00  __t5Array2ZP9_lex_wordZ9_lex_word3no
p [822]
  0.0       1.01     0.00        6     0.00     0.00  _cerror [823]
  0.0       1.01      0.00        6      0.00      0.00   _close [824]
  0.0       1.01      0.00        6      0.00      1.67   _endopen [43]
  0.0       1.01      0.00        6      0.00      0.00   _findiop [825]
  0.0       1.01      0.00        6      0.00      0.00   _fstat64 [826]
  0.0       1.01      0.00        6      0.00      0.00   add__5_treeP5_wordi [145]
  0.0       1.01      0.00        6      0.00      0.00   add__t5Array2ZP5_wordZ5_wordP5_word
[146]
  0.0       1.01      0.00        6      0.00     30.98   apply_rules__8_lexiconP6_rulesP9_sen
tencei [12]
  0.0       1.01      0.00        6      0.00      0.00   closefile__FP4FILE [70]
  0.0       1.01      0.00        6      0.00      0.43   copy__8_lexicon [49]
  0.0       1.01      0.00        6      0.00      0.00   fclose [71]
  0.0       1.01      0.00        6      0.00      0.00   fflush [147]
  0.0       1.01      0.00        6      0.00      1.67   fopen [39]
  0.0       1.01      0.00        6      0.00      0.00   get_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciation [148]
  0.0       1.01      0.00        6      0.00      1.67   readfile__FPc [40]
  0.0       1.01      0.00        5      0.00      0.00   getenv [78]
  0.0       1.01      0.00        2      0.00      0.52   __5_ruleP4FILE [59]
  0.0       1.01      0.00        2      0.00      0.52   __9_sentenceP4FILE [60]
  0.0       1.01      0.00        2      0.00      0.00   _flushlbf [85]
  0.0       1.01      0.00        2      0.00      0.00   _rw_rdlock [827]
  0.0       1.01      0.00        2      0.00      0.00   _time [828]
  0.0       1.01      0.00        2      0.00      0.00   add__t5Array2ZP5_ruleZ5_ruleP5_rule
[149]
  0.0       1.01      0.00        2      0.00      0.00   atexit [82]
  0.0       1.01      0.00        2      0.00      0.00   close__5_rule [150]
  0.0       1.01      0.00        2      0.00      0.00   output_aux__5_tree11destinationP4FIL
EPci11output_mode [151]
  0.0       1.01      0.00        2      0.00      0.00   rw_unlock [152]
  0.0       1.01      0.00        1      0.00      0.00   .urem [153]
  0.0       1.01      0.00        1      0.00      0.00   __5_treeP9_sentence [829]
  0.0       1.01      0.00        1      0.00      4.38   __6_rulesP4FILE [45]
  0.0       1.01      0.00        1      0.00    141.87   __8_lexiconP4FILE [17]
  0.0       1.01      0.00        1      0.00      0.00   __t5Array2ZP5_ruleZ5_rulei [830]
  0.0       1.01      0.00        1      0.00      0.00   __t5Array2ZP9_lex_wordZ9_lex_wordi [
831]
  0.0       1.01      0.00        1      0.00      0.00   _exithandle [84]
  0.0       1.01      0.00        1      0.00      0.00   _profil [832]
  0.0       1.01      0.00        1      0.00      0.00   _wrtchk [80]
  0.0       1.01      0.00        1      0.00      0.00   difftime [154]
  0.0       1.01      0.00        1      0.00      0.00   exit [83]
  0.0       1.01      0.00        1      0.00    191.82   grow__5_treeP6_rulesP8_lexicon [8]
  0.0       1.01      0.00        1      0.00    191.82   grow_aux2__5_treeP6_rulesP8_lexiconi
 [9]
  0.0       1.01      0.00        1      0.00    345.77   main [3]
  0.0       1.01      0.00        1      0.00      0.00   output__5_tree11destinationP4FILE [1
55]
```

```
*********************************************************
*       NEW MATCHER RESULTS FOR:                        *
*                                                       *
*       "Catch a balloom before Mary does."             *
*                                                       *
*********************************************************



#pragma ident   "@(#)gprof.flat.blurb   1.8     93/06/07 SMI"


flat profile:

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.
```

```
granularity: each sample hit covers 4 byte(s) for 3.33% of 0.30 seconds

  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call name
 53.3      0.16     0.16                             internal_mcount [1]
  6.7      0.18     0.02   176854     0.00     0.00  get_feature__8_segmenti   <cycle 1>
[12]
  6.7      0.20     0.02   144601     0.00     0.00  get_feature__7_bundlei [20]
  6.7      0.22     0.02     3445     0.01     0.02  feature_match__5_ruleP8_segmentT1 [1
0]
  3.3      0.23     0.01    67645     0.00     0.00  _return_zero [29]
  3.3      0.24     0.01    33821     0.00     0.00  _mutex_lock [32]
  3.3      0.25     0.01    19371     0.00     0.00  strsame__FPcT0 [31]
  3.3      0.26     0.01     6006     0.00     0.00  copy__13_phonemic_rep [30]
  3.3      0.27     0.01     6006     0.00     0.00  copy__8_segment [21]
  3.3      0.28     0.01      942     0.01     0.02  phoneme2index__9_sentencePc [22]
  3.3      0.29     0.01                             _mcount (1508)
  3.3      0.30     0.01                             mcount (526)
  0.0      0.30     0.00    42840     0.00     0.00  get_data__9_segmentsi [76]
  0.0      0.30     0.00    33820     0.00     0.00  mutex_unlock [77]
  0.0      0.30     0.00    32345     0.00     0.00  get_feature__13_phonemic_repi     <c
ycle 1> [78]
  0.0      0.30     0.00    31922     0.00     0.00  _malloc_unlocked [41]
  0.0      0.30     0.00    31591     0.00     0.00  malloc [27]
  0.0      0.30     0.00    31582     0.00     0.00  cleanfree [79]
  0.0      0.30     0.00    27926     0.00     0.00  strcmp [80]
  0.0      0.30     0.00    22326     0.00     0.00  _smalloc [58]
  0.0      0.30     0.00    19347     0.00     0.00  get_symbol__7_bundle [81]
  0.0      0.30     0.00    19347     0.00     0.00  get_symbol__8_segment [82]
  0.0      0.30     0.00    16273     0.00     0.00  get_NoOfSegments__9_segments [83]
  0.0      0.30     0.00    11858     0.00     0.00  exact_match__Fii [84]
  0.0      0.30     0.00    10788     0.00     0.00  realfree [85]
  0.0      0.30     0.00    10084     0.00     0.00  getGlide__6_Vowel [86]
  0.0      0.30     0.00     7052     0.00     0.00  check_STD__Fv [87]
  0.0      0.30     0.00     6790     0.00     0.00  getConsonant__6_Glide [88]
  0.0      0.30     0.00     6574     0.00     0.00  get_data__t5Array2ZP9_lex_wordZ9_lex
_wordi [89]
  0.0      0.30     0.00     6110     0.00     0.00  __8_segment10mode_value [871]
  0.0      0.30     0.00     6006     0.00     0.00  __13_phonemic_repi12marker_value [87
2]
  0.0      0.30     0.00     4608     0.00     0.00  get_NoOfData__t5Array2ZP5_ruleZ5_rul
e [90]
  0.0      0.30     0.00     3664     0.00     0.00  getLingual__10_Consonant [91]
  0.0      0.30     0.00     3448     0.00     0.00  strcopy__FPc [36]
  0.0      0.30     0.00     3288     0.00     0.02  construct_pronunciation__5_ruleP14_p
ronunciationP9_sentencei [7]
  0.0      0.30     0.00     3288     0.00     0.00  get_data__t5Array2ZP5_ruleZ5_rulei [
92]
  0.0      0.30     0.00     3278     0.00     0.00  get_NoOfData__t5Array2ZP14_pronuncia
tionZ14_pronunciation [93]
  0.0      0.30     0.00     2229     0.00     0.00  is_alphabet__Fc [94]
  0.0      0.30     0.00     2118     0.00     0.00  getPharyngeal__6_Glide [95]
  0.0      0.30     0.00     2104     0.00     0.00  getBlade__8_Lingual [96]
  0.0      0.30     0.00     2049     0.00     0.00  strcpy [97]
  0.0      0.30     0.00     1849     0.00     0.00  strlen [98]
  0.0      0.30     0.00     1788     0.00     0.00  bundle_match__FP8_segmentT0 [33]
  0.0      0.30     0.00     1754     0.00     0.00  get_data__t5Array2ZP14_pronunciation
Z14_pronunciationi [99]
  0.0      0.30     0.00     1748     0.00     0.00  matching__5_treeP14_pronunciationP9_
sentencei [34]
  0.0      0.30     0.00     1747     0.00     0.02  matched__5_ruleP9_sentenceiiii [13]
  0.0      0.30     0.00     1698     0.00     0.00  derive_main_stress__13_phonemic_rep
<cycle 1> [100]
```

```
  0.0       0.30     0.00     1698     0.00     0.00  derive_reduced__13_phonemic_rep   <c
ycle 1> [101]
  0.0       0.30     0.00     1698     0.00     0.02  matched__5_ruleP14_pronunciationiii
 [14]
  0.0       0.30     0.00     1644     0.00     0.00  __14_pronunciation3nop [873]
  0.0       0.30     0.00     1644     0.00     0.00  __9_segments3nop [874]
  0.0       0.30     0.00     1644     0.00     0.01  copy__14_pronunciation [26]
  0.0       0.30     0.00     1623     0.00     0.00  _free_unlocked [875]
  0.0       0.30     0.00     1623     0.00     0.00  free [40]
  0.0       0.30     0.00     1560     0.00     0.00  getBody__8_Lingual [102]
  0.0       0.30     0.00     1542     0.00     0.00  get_NoOfData__t5Array2ZP9_lex_wordZ9
_lex_word [103]
  0.0       0.30     0.00     1536     0.00     0.00  __9_lex_word3nop [876]
  0.0       0.30     0.00     1536     0.00     0.00  __t5Array2ZP14_pronunciationZ14_pron
unciation [877]
  0.0       0.30     0.00     1536     0.00     0.04  apply_rules__9_lex_wordP6_rulesP9_se
ntencei [8]
  0.0       0.30     0.00     1536     0.00     0.01  copy__9_lex_word [25]
  0.0       0.30     0.00     1536     0.00     0.00  get_expanded__9_lex_word [104]
  0.0       0.30     0.00     1525     0.00     0.00  _ungetc_unlocked [878]
  0.0       0.30     0.00     1525     0.00     0.00  ungetc [105]
  0.0       0.30     0.00     1264     0.00     0.00  get_word__FP4FILE [39]
  0.0       0.30     0.00     1232     0.00     0.00  check_fvalue__Fi [106]
  0.0       0.30     0.00     1232     0.00     0.00  set_feature__7_bundleii [107]
  0.0       0.30     0.00     1119     0.00     0.00  compress__8_segment [108]
  0.0       0.30     0.00     1078     0.00     0.00  getGlottis__11_Pharyngeal [109]
  0.0       0.30     0.00     1052     0.00     0.00  getVclFolds__6_Vowel [110]
  0.0       0.30     0.00     1040     0.00     0.00  getPharynx__11_Pharyngeal [111]
  0.0       0.30     0.00     1015     0.00     0.02  __8_segment10mode_valueT1P4FILE13hea
ding_valueT4 [19]
  0.0       0.30     0.00      942     0.00     0.02  __13_phonemic_repP4FILE [23]
  0.0       0.30     0.00      612     0.00     0.00  getLips__10_Consonant [112]
  0.0       0.30     0.00      598     0.00     0.00  _sbrk [42]
  0.0       0.30     0.00      598     0.00     0.00  _sbrk_unlocked [879]
  0.0       0.30     0.00      544     0.00     0.00  getSftP__6_Glide [113]
  0.0       0.30     0.00      514     0.00     0.00  __6_Vowel [880]
  0.0       0.30     0.00      485     0.00     0.00  string2feature_index__FPc [114]
  0.0       0.30     0.00      467     0.00     0.00  setConsonant__6_GlideP10_Consonant [
115]
  0.0       0.30     0.00      458     0.00     0.00  add__9_segmentsP8_segment [116]
  0.0       0.30     0.00      458     0.00     0.00  set_mode__8_segment10mode_value [117
]
  0.0       0.30     0.00      416     0.00     0.00  getSonorant__10_Consonant [118]
  0.0       0.30     0.00      404     0.00     0.00  getRound__5_Lips [119]
  0.0       0.30     0.00      387     0.00     0.00  setSftP__6_GlideP12_Soft_Palate [120
]
  0.0       0.30     0.00      387     0.00     0.00  setVcl__6_VowelP11_VocalFolds [121]
  0.0       0.30     0.00      378     0.00     0.00  add__t5Array2ZP14_pronunciationZ14_p
ronunciationP14_pronunciation [122]
  0.0       0.30     0.00      375     0.00     0.00  __11_Pharyngeal [881]
  0.0       0.30     0.00      375     0.00     0.00  __6_Glide [882]
  0.0       0.30     0.00      375     0.00     0.00  __8_Lingual [883]
  0.0       0.30     0.00      375     0.00     0.00  setBlade__8_LingualP6_Blade [123]
  0.0       0.30     0.00      375     0.00     0.00  setBody__8_LingualP5_Body [124]
  0.0       0.30     0.00      375     0.00     0.00  setGlide__6_VowelP6_Glide [125]
  0.0       0.30     0.00      375     0.00     0.00  setGlottis__11_PharyngealP8_Glottis
[126]
  0.0       0.30     0.00      375     0.00     0.00  setLingual__10_ConsonantP8_Lingual [
127]
  0.0       0.30     0.00      375     0.00     0.00  setLips__10_ConsonantP5_Lips [128]
  0.0       0.30     0.00      375     0.00     0.00  setPharyngeal__6_GlideP11_Pharyngeal
 [129]
  0.0       0.30     0.00      375     0.00     0.00  setPharynx__11_PharyngealP8_Pharynx
[130]
  0.0       0.30     0.00      350     0.00     0.00  getConstr__8_Glottis [131]
  0.0        0.30     0.00      324     0.00     0.00  getAnt__6_Blade [132]
  0.0        0.30     0.00      324     0.00     0.00  getDistr__6_Blade [133]
  0.0        0.30     0.00      324     0.00     0.00  getNasal__12_Soft_Palate [134]
  0.0        0.30     0.00      312     0.00     0.00  __10_Consonant [884]
  0.0        0.30     0.00      312     0.00     0.00  __11_VocalFolds [885]
  0.0        0.30     0.00      312     0.00     0.00  __12_Soft_Palate [886]
  0.0        0.30     0.00      312     0.00     0.00  __5_Body [887]
  0.0        0.30     0.00      312     0.00     0.00  __5_Lips [888]
  0.0        0.30     0.00      312     0.00     0.00  __6_Blade [889]
  0.0        0.30     0.00      312     0.00     0.00  __8_Glottis [890]
  0.0        0.30     0.00      312     0.00     0.00  __8_Pharynx [891]
  0.0        0.30     0.00      312     0.00     0.00  getAtr__8_Pharynx [135]
  0.0        0.30     0.00      312     0.00     0.00  getBack__5_Body [136]
  0.0        0.30     0.00      312     0.00     0.00  getContinuant__10_Consonant [137]
  0.0        0.30     0.00      312     0.00     0.00  getCtr__8_Pharynx [138]
  0.0        0.30     0.00      312     0.00     0.00  getDominant__10_Consonant [139]
  0.0        0.30     0.00      312     0.00     0.00  getDominant__6_Glide [140]
  0.0        0.30     0.00      312     0.00     0.00  getHigh__5_Body [141]
  0.0        0.30     0.00      312     0.00     0.00  getLat__6_Blade [142]
  0.0        0.30     0.00      312     0.00     0.00  getLow__5_Body [143]
  0.0        0.30     0.00      312     0.00     0.00  getRhotic__6_Blade [144]
  0.0        0.30     0.00      312     0.00     0.00  getSlack__11_VocalFolds [145]
  0.0        0.30     0.00      312     0.00     0.00  getSpread__8_Glottis [146]
  0.0        0.30     0.00      312     0.00     0.00  getStiff__11_VocalFolds [147]
  0.0        0.30     0.00      312     0.00     0.00  getStrident__10_Consonant [148]
  0.0        0.30     0.00      312     0.00     0.00  makeTree__FP6_Vowel [149]
  0.0        0.30     0.00      312     0.00     0.00  setContinuant__10_Consonanti [150]
  0.0        0.30     0.00      312     0.00     0.00  setSonorant__10_Consonanti [151]
  0.0        0.30     0.00      299     0.00     0.00  _morecore [43]
  0.0        0.30     0.00      298     0.00     0.00  get_line__FP4FILE [46]
  0.0        0.30     0.00      278     0.00     0.08  __9_segmentsi10mode_valueT2P4FILE13h
eading_value [15]
  0.0        0.30     0.00      274     0.00     0.08  __14_pronunciationP4FILE [18]
  0.0        0.30     0.00      256     0.00     0.08  __9_lex_wordP4FILE [17]
  0.0        0.30     0.00      256     0.00     0.00  __t5Array2ZP14_pronunciationZ14_pron
unciationi [892]
  0.0        0.30     0.00      256     0.00     0.00  add__t5Array2ZP9_lex_wordZ9_lex_word
P9_lex_word [152]
  0.0        0.30     0.00      256     0.00     0.00  set_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciationPc [51]
  0.0        0.30     0.00      256     0.00     0.00  strcat [153]
  0.0        0.30     0.00      238     0.00     0.00  setDominant__10_Consonanti [154]
  0.0        0.30     0.00      223     0.00     0.00  file_get_item__FP4FILEPc [45]
  0.0        0.30     0.00      216     0.00     0.00  setDominant__6_Glidei [155]
  0.0        0.30     0.00      214     0.00     0.00  t_delete [156]
  0.0        0.30     0.00      208     0.00     0.00  copyTree__FP6_VowelT0 [157]
  0.0        0.30     0.00      208     0.00     0.00  get_treeData__9_segmentsi [158]
  0.0        0.30     0.00      208     0.00     0.00  setAnt__6_Bladei [159]
  0.0        0.30     0.00      208     0.00     0.00  setAtr__8_Pharynxi [160]
  0.0        0.30     0.00      208     0.00     0.00  setBack__5_Bodyi [161]
  0.0        0.30     0.00      208     0.00     0.00  setConstr__8_Glottisi [162]
  0.0        0.30     0.00      208     0.00     0.00  setCtr__8_Pharynxi [163]
  0.0        0.30     0.00      208     0.00     0.00  setDistr__6_Bladei [164]
  0.0        0.30     0.00      208     0.00     0.00  setHigh__5_Bodyi [165]
  0.0        0.30     0.00      208     0.00     0.00  setLat__6_Bladei [166]
  0.0        0.30     0.00      208     0.00     0.00  setLow__5_Bodyi [167]
  0.0        0.30     0.00      208     0.00     0.00  setNasal__12_Soft_Palatei [168]
  0.0        0.30     0.00      208     0.00     0.00  setRhotic__6_Bladei [169]
  0.0        0.30     0.00      208     0.00     0.00  setRound__5_Lipsi [170]
  0.0        0.30     0.00      208     0.00     0.00  setSlack__11_VocalFoldsi [171]
  0.0        0.30     0.00      208     0.00     0.00  setSpread__8_Glottisi [172]
  0.0        0.30     0.00      208     0.00     0.00  setStiff__11_VocalFoldsi [173]
  0.0        0.30     0.00      208     0.00     0.00  setStrident__10_Consonanti [174]
  0.0        0.30     0.00      150     0.00     0.00  heading2string__F13heading_value [17
5]
```

```
  0.0        0.30      0.00        146      0.00       0.00   get_state__8_segment [176]
  0.0        0.30      0.00        120      0.00       0.00   _realbufend [893]
  0.0        0.30      0.00        116      0.00       0.00   get_cv__5_rule [177]
  0.0        0.30      0.00        116      0.00       0.00   makeChange__FP6_VowelT0i [178]
  0.0        0.30      0.00        104      0.00       0.00   _._10_Consonant [894]
  0.0        0.30      0.00        104      0.00       0.00   _._11_Pharyngeal [895]
  0.0        0.30      0.00        104      0.00       0.00   _._11_VocalFolds [896]
  0.0        0.30      0.00        104      0.00       0.00   _._12_Soft_Palate [897]
  0.0        0.30      0.00        104      0.00       0.00   _._5_Body [898]
  0.0        0.30      0.00        104      0.00       0.00   _._5_Lips [899]
  0.0        0.30      0.00        104      0.00       0.00   _._6_Blade [900]
  0.0        0.30      0.00        104      0.00       0.00   _._6_Glide [901]
  0.0        0.30      0.00        104      0.00       0.00   _._6_Vowel [902]
  0.0        0.30      0.00        104      0.00       0.00   _._8_Glottis [903]
  0.0        0.30      0.00        104      0.00       0.00   _._8_Lingual [904]
  0.0        0.30      0.00        104      0.00       0.00   _._8_Pharynx [905]
  0.0        0.30      0.00        104      0.00       0.00   __14_pronunciation [906]
  0.0        0.30      0.00        104      0.00       0.00   __7_bundle [907]
  0.0        0.30      0.00        104      0.00       0.00   __9_segmentsi10mode_value [908]
  0.0        0.30      0.00        104      0.00       0.00   change_subtitle__9_segments13heading
_valueT1 [179]
  0.0        0.30      0.00        104      0.00       0.02   construct_pronunciation__5_ruleP14_p
ronunciationiiiiP9_sentenceiiiii [35]
  0.0        0.30      0.00        104      0.00       0.00   getDominant__6_Vowel [180]
  0.0        0.30      0.00        104      0.00       0.00   get_RULErc__7_bundle [181]
  0.0        0.30      0.00        104      0.00       0.00   get_STD_index__13_phonemic_rep [182]
  0.0        0.30      0.00        104      0.00       0.00   get_bundle__8_segment [183]
  0.0        0.30      0.00        104      0.00       0.00   get_phonemic__8_segment [184]
  0.0        0.30      0.00        104      0.00       0.00   get_sc_l__5_rule [185]
  0.0        0.30      0.00        104      0.00       0.00   get_sc_r__5_rule [186]
  0.0        0.30      0.00        104      0.00       0.00   set_data__8_segmentP7_bundle [187]
  0.0        0.30      0.00        104      0.00       0.00   set_title__9_segments13heading_value
 [188]
  0.0        0.30      0.00        104      0.00       0.00   tree2seg__FP6_VowelP7_bundleP8_segme
nt [54]
  0.0        0.30      0.00         82      0.00       0.00   t_splay [189]
  0.0        0.30      0.00         73      0.00       0.01   __7_bundleP4FILE13heading_valueT2 [3
8]
  0.0        0.30      0.00         69      0.00       0.00   nvmatch [190]
  0.0        0.30      0.00         63      0.00       0.00   __10_Consonantiii [909]
  0.0        0.30      0.00         63      0.00       0.00   __11_VocalFoldsii [910]
  0.0        0.30      0.00         63      0.00       0.00   __12_Soft_Palatei [911]
  0.0        0.30      0.00         63      0.00       0.00   __5_Bodyiii [912]
  0.0        0.30      0.00         63      0.00       0.00   __5_Lipsi [913]
  0.0        0.30      0.00         63      0.00       0.00   __6_Bladeiiii [914]
  0.0        0.30      0.00         63      0.00       0.00   __8_Glottisii [915]
  0.0        0.30      0.00         63      0.00       0.00   __8_Pharynxii [916]
  0.0        0.30      0.00         63      0.00       0.00   determineDominant__FP8_segment [52]
  0.0        0.30      0.00         63      0.00       0.00   file_get_rorc__7_bundleP4FILE [48]
  0.0        0.30      0.00         63      0.00       0.00   makeTree__FP6_VowelP8_segment [44]
  0.0        0.30      0.00         63      0.00       0.00   setDominant__FiP6_Vowel [191]
  0.0        0.30      0.00         52      0.00       0.00   .udiv [192]
  0.0        0.30      0.00         51      0.00       0.00   _fwrite_unlocked [71]
  0.0        0.30      0.00         51      0.00       0.00   fwrite [193]
  0.0        0.30      0.00         36      0.00       0.00   get_NoOfData__t5Array2ZP5_wordZ5_wor
d [194]
  0.0        0.30      0.00         33      0.00       0.00   _write [917]
  0.0        0.30      0.00         33      0.00       0.00   _xflsbuf [918]
  0.0        0.30      0.00         31      0.00       0.00   __flsbuf [919]
  0.0        0.30      0.00         30      0.00       0.00   get_data__t5Array2ZP5_wordZ5_wordi [
195]
  0.0        0.30      0.00         30      0.00       0.00   get_time__7_bundle [196]
  0.0        0.30      0.00         30      0.00       0.00   get_time__8_segment [197]
  0.0        0.30      0.00         25      0.00       0.00   setDominant__6_Voweli [198]
  0.0        0.30      0.00         24      0.00       0.00   get_NoOfSegments__5_word [199]
  0.0        0.30      0.00        20      0.00      0.00   ___errno [920]
  0.0        0.30      0.00        20      0.00      0.00   _return_negone [921]
  0.0        0.30      0.00        20      0.00      0.00   thr_main [200]
  0.0        0.30      0.00        16      0.00      0.00   _fflush_u [922]
  0.0        0.30      0.00        12      0.00      0.00   __filbuf [57]
  0.0        0.30      0.00        12      0.00      0.00   _read [923]
  0.0        0.30      0.00        12      0.00      0.00   get_word__5_word [201]
  0.0        0.30      0.00        10      0.00      0.00   file_get_RULErorc__7_bundleP4FILE [5
3]
  0.0        0.30      0.00         8      0.00      0.00   _findbuf [59]
  0.0        0.30      0.00         8      0.00      0.00   _ioctl [924]
  0.0        0.30      0.00         8      0.00      0.00   _isatty [925]
  0.0        0.30      0.00         8      0.00      0.00   _setbufend [926]
  0.0        0.30      0.00         7      0.00      0.00   _lseek64 [927]
  0.0        0.30      0.00         6      0.00      0.00   _._8_lexicon [928]
  0.0        0.30      0.00         6      0.00      0.00   _._t5Array2ZP9_lex_wordZ9_lex_word [
929]
  0.0        0.30      0.00         6      0.00      0.00   __5_wordPci [63]
  0.0        0.30      0.00         6      0.00      0.00   __8_lexicon3nop [930]
  0.0        0.30      0.00         6      0.00      0.00   __t5Array2ZP5_wordZ5_wordi [931]
  0.0        0.30      0.00         6      0.00      0.00   __t5Array2ZP9_lex_wordZ9_lex_word3no
p [932]
  0.0        0.30      0.00         6      0.00      0.00   _cerror [933]
  0.0        0.30      0.00         6      0.00      0.00   _close [934]
  0.0        0.30      0.00         6      0.00      0.00   _endopen [935]
  0.0        0.30      0.00         6      0.00      0.00   _findiop [936]
  0.0        0.30      0.00         6      0.00      0.00   _fstat64 [937]
  0.0        0.30      0.00         6      0.00      0.00   _open [938]
  0.0        0.30      0.00         6      0.00      0.00   add__5_treeP5_wordi [202]
  0.0        0.30      0.00         6      0.00      0.00   add__t5Array2ZP5_wordZ5_wordP5_word
[203]
  0.0        0.30      0.00         6      0.00      9.04   apply_rules__8_lexiconP6_rulesP9_sen
tencei [9]
  0.0        0.30      0.00         6      0.00      0.00   closefile__FP4FILE [60]
  0.0        0.30      0.00         6      0.00      3.30   copy__8_lexicon [24]
  0.0        0.30      0.00         6      0.00      0.00   fclose [61]
  0.0        0.30      0.00         6      0.00      0.00   fflush [204]
  0.0        0.30      0.00         6      0.00      0.00   fopen [205]
  0.0        0.30      0.00         6      0.00      0.00   get_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciation [206]
  0.0        0.30      0.00         6      0.00     13.44   grow_aux__5_treeP6_rulesP8_lexiconi
[4]
  0.0        0.30      0.00         6      0.00      0.00   readfile__FPc [207]
  0.0        0.30      0.00         5      0.00      0.00   getenv [62]
  0.0        0.30      0.00         2      0.00      0.08   __5_ruleP4FILE [49]
  0.0        0.30      0.00         2      0.00      0.08   __9_sentenceP4FILE [50]
  0.0        0.30      0.00         2      0.00      0.00   _flushlbf [74]
  0.0        0.30      0.00         2      0.00      0.00   _rw_rdlock [939]
  0.0        0.30      0.00         2      0.00      0.00   _rw_unlock [940]
  0.0        0.30      0.00         2      0.00      0.00   _time [941]
  0.0        0.30      0.00         2      0.00      0.00   add__t5Array2ZP5_ruleZ5_ruleP5_rule
[208]
  0.0        0.30      0.00         2      0.00      0.00   atexit [64]
  0.0        0.30      0.00         2      0.00      0.00   atoi [209]
  0.0        0.30      0.00         2      0.00      0.00   close__5_rule [55]
  0.0        0.30      0.00         2      0.00      0.00   output_aux__5_tree11destinationP4FIL
EPci11output_mode [210]
  0.0        0.30      0.00         1      0.00      0.00   .urem [211]
  0.0        0.30      0.00         1      0.00      0.00   __5_treeP9_sentence [942]
  0.0        0.30      0.00         1      0.00      0.16   __6_rulesP4FILE [47]
  0.0        0.30      0.00         1      0.00     21.05   __8_lexiconP4FILE [16]
  0.0        0.30      0.00         1      0.00      0.00   __t5Array2ZP5_ruleZ5_rulei [943]
  0.0        0.30      0.00         1      0.00      0.00   __t5Array2ZP9_lex_wordZ9_lex_wordi [
944]
  0.0        0.30      0.00         1      0.00      0.00   _exithandle [66]
```

```
   0.0        0.30      0.00         1      0.00      0.00  _memcpy [945]
   0.0        0.30      0.00         1      0.00      0.00  _profil [946]
   0.0        0.30      0.00         1      0.00      0.00  _wrtchk [72]
   0.0        0.30      0.00         1      0.00      0.00  difftime [212]
   0.0        0.30      0.00         1      0.00      0.00  exit [65]
   0.0        0.30      0.00         1      0.00     80.66  grow__5_treeP6_rulesP8_lexicon [5]
   0.0        0.30      0.00         1      0.00     80.66  grow_aux2__5_treeP6_rulesP8_lexiconi
 [6]
   0.0        0.30      0.00         1      0.00    102.03  main [3]
   0.0        0.30      0.00         1      0.00      0.00  output__5_tree11destinationP4FILE [2
13]
```

```
*******************************************************************
*       ORIGINAL MATCHER RESULTS FOR:                             *
*                                                                 *
*       "The baby can go to him in the bus today."                *
*                                                                 *
*******************************************************************

#pragma ident   "@(#)gprof.flat.blurb   1.8     93/06/07 SMI"


flat profile:

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.
```

```
granularity: each sample hit covers 4 byte(s) for 0.78% of 1.29 seconds

  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call name
 58.1      0.75     0.75                             internal_mcount [1]
 16.3      0.96     0.21  1775624     0.00     0.00  get_feature__7_bundlei [12]
  8.5      1.07     0.11  1047939     0.00     0.00  get_feature__8_segmenti   <cycle 1>
[8]
  4.7      1.13     0.06      900     0.07     0.36  feature2index__9_sentenceP7_bundle [
4]
  1.6      1.15     0.02    42490     0.00     0.00  realfree [32]
  1.6      1.17     0.02    34793     0.00     0.00  _free_unlocked [27]
  1.6      1.19     0.02                             _brk_unlocked [33]
  1.6      1.21     0.02                             mcount (449)
  0.8      1.22     0.01   166095     0.00     0.00  _return_zero [44]
  0.8      1.23     0.01    83046     0.00     0.00  _mutex_lock [41]
  0.8      1.24     0.01    83045     0.00     0.00  mutex_unlock [37]
  0.8      1.25     0.01    47840     0.00     0.00  _malloc_unlocked [36]
  0.8      1.26     0.01    14659     0.00     0.00  exact_match__Fii [38]
  0.8      1.27     0.01       35     0.29     0.29  _write [42]
  0.8      1.28     0.01                             .LL514 [39]
  0.8      1.29     0.01                             find_fde [40]
  0.0      1.29     0.00    61561     0.00     0.00  get_data__9_segmentsi [91]
  0.0      1.29     0.00    47811     0.00     0.00  get_feature__13_phonemic_repi     <c
ycle 1> [92]
  0.0      1.29     0.00    47716     0.00     0.00  malloc [28]
  0.0      1.29     0.00    46507     0.00     0.00  cleanfree [64]
  0.0      1.29     0.00    44268     0.00     0.00  strcmp [93]
  0.0      1.29     0.00    35217     0.00     0.00  _smalloc [78]
  0.0      1.29     0.00    34793     0.00     0.00  free [26]
  0.0      1.29     0.00    29169     0.00     0.00  get_NoOfSegments__9_segments [94]
  0.0      1.29     0.00    19799     0.00     0.00  get_symbol__7_bundle [95]
  0.0      1.29     0.00    19459     0.00     0.00  get_symbol__8_segment [96]
  0.0      1.29     0.00    19371     0.00     0.00  strsame__FPcT0 [97]
  0.0      1.29     0.00    12903     0.00     0.00  check_STD__Fv [98]
  0.0      1.29     0.00    11895     0.00     0.00  get_data__t5Array2ZP9_lex_wordZ9_lex
_wordi [99]
  0.0      1.29     0.00    10721     0.00     0.00  __8_segment [787]
  0.0      1.29     0.00    10721     0.00     0.00  __13_phonemic_repi12marker_value [78
8]
  0.0      1.29     0.00    10721     0.00     0.00  __8_segment10mode_value [789]
  0.0      1.29     0.00    10721     0.00     0.00  copy__13_phonemic_rep [100]
  0.0      1.29     0.00    10721     0.00     0.00  copy__8_segment [101]
  0.0      1.29     0.00    10127     0.00     0.00  strlen [102]
  0.0      1.29     0.00     8941     0.00     0.00  _memcpy [790]
  0.0      1.29     0.00     8448     0.00     0.00  get_NoOfData__t5Array2ZP5_ruleZ5_rul
e [103]
  0.0      1.29     0.00     7130     0.00     0.00  strcopy__FPc [50]
  0.0      1.29     0.00     6895     0.00     0.01  feature_match__5_ruleP8_segmentT1 [2
5]
  0.0      1.29     0.00     6028     0.00     0.03  construct_pronunciation__5_ruleP14_p
ronunciationP9_sentencei [13]
  0.0      1.29     0.00     6028     0.00     0.00  get_data__t5Array2ZP5_ruleZ5_rulei [
104]
  0.0      1.29     0.00     5960     0.00     0.00  _memset [791]
  0.0      1.29     0.00     5928     0.00     0.00  get_NoOfData__t5Array2ZP14_pronuncia
tionZ14_pronunciation [105]
  0.0      1.29     0.00     4407     0.00     0.00  strcpy [106]
  0.0      1.29     0.00     3718     0.00     0.00  t_delete [107]
  0.0      1.29     0.00     3455     0.00     0.00  derive_main_stress__13_phonemic_rep
<cycle 1> [108]
  0.0      1.29     0.00     3455     0.00     0.00  derive_reduced__13_phonemic_rep   <c
ycle 1> [109]
```

```
  0.0       1.29     0.00     3455     0.00     0.01  matched__5_ruleP14_pronunciationiiii
 [30]
  0.0       1.29     0.00     3440     0.00     0.01  matched__5_ruleP9_sentenceiiii [31]
  0.0       1.29     0.00     3138     0.00     0.00  get_data__t5Array2ZP14_pronunciation
Z14_pronunciationi [110]
  0.0       1.29     0.00     3126     0.00     0.00  _._14_pronunciation [792]
  0.0       1.29     0.00     3126     0.00     0.00  _._9_segments [793]
  0.0       1.29     0.00     3125     0.00     0.01  matching__5_treeP14_pronunciationP9_
sentencei [34]
  0.0       1.29     0.00     3022     0.00     0.01  bundle_match__FP8_segmentT0 [35]
  0.0       1.29     0.00     3014     0.00     0.00  __14_pronunciation3nop [794]
  0.0       1.29     0.00     3014     0.00     0.00  __9_segments3nop [795]
  0.0       1.29     0.00     3014     0.00     0.00  copy__14_pronunciation [59]
  0.0       1.29     0.00     2827     0.00     0.00  get_NoOfData__t5Array2ZP9_lex_wordZ9
_lex_word [111]
  0.0       1.29     0.00     2816     0.00     0.00  _._9_lex_word [796]
  0.0       1.29     0.00     2816     0.00     0.00  _._t5Array2ZP14_pronunciationZ14_pro
nunciation [797]
  0.0       1.29     0.00     2816     0.00     0.00  __9_lex_word3nop [798]
  0.0       1.29     0.00     2816     0.00     0.00  __t5Array2ZP14_pronunciationZ14_pron
unciation [799]
  0.0       1.29     0.00     2816     0.00     0.07  apply_rules__9_lex_wordP6_rulesP9_se
ntencei [14]
  0.0       1.29     0.00     2816     0.00     0.00  copy__9_lex_word [52]
  0.0       1.29     0.00     2816     0.00     0.00  get_expanded__9_lex_word [112]
  0.0       1.29     0.00     2384     0.00     0.00  __tf13RunTimeStatus [800]
  0.0       1.29     0.00     2229     0.00     0.00  is_alphabet__Fc [113]
  0.0       1.29     0.00     1921     0.00     0.17  compress__8_segment [5]
  0.0       1.29     0.00     1841     0.00     0.00  t_splay [114]
  0.0       1.29     0.00     1531     0.00     0.00  _ungetc_unlocked [801]
  0.0       1.29     0.00     1531     0.00     0.00  ungetc [115]
  0.0       1.29     0.00     1368     0.00     0.00  get_word__FP4FILE [62]
  0.0       1.29     0.00     1021     0.00     0.17  __8_segment10mode_valueT1P4FILE13hea
ding_valueT4 [17]
  0.0       1.29     0.00      942     0.00     0.00  __13_phonemic_repP4FILE [802]
  0.0       1.29     0.00      942     0.00     0.00  phoneme2index__9_sentencePc [116]
  0.0       1.29     0.00      900     0.00     0.36  __13_phonemic_repP7_bundle [6]
  0.0       1.29     0.00      864     0.00     0.00  check_fvalue__Fi [117]
  0.0       1.29     0.00      864     0.00     0.00  set_feature__7_bundleii [118]
  0.0       1.29     0.00      864     0.00     0.17  set_feature__8_segmentii [23]
  0.0       1.29     0.00      596     0.00     0.00  __tcf_0 [803]
  0.0       1.29     0.00      547     0.00     0.00  string2feature_index__FPc [119]
  0.0       1.29     0.00      528     0.00     0.00  _sbrk [70]
  0.0       1.29     0.00      528     0.00     0.00  _sbrk_unlocked [804]
  0.0       1.29     0.00      386     0.00     0.00  add__t5Array2ZP14_pronunciationZ14_p
ronunciationP14_pronunciation [120]
  0.0       1.29     0.00      370     0.00     0.00  get_time__7_bundle [121]
  0.0       1.29     0.00      359     0.00     0.02  add__9_segmentsP8_segment [48]
  0.0       1.29     0.00      359     0.00     0.02  set_mode__8_segment10mode_value [49]
  0.0       1.29     0.00      340     0.00     0.00  __7_bundleP13_phonemic_rep [55]
  0.0       1.29     0.00      340     0.00     0.00  expand__8_segment [53]
  0.0       1.29     0.00      340     0.00     0.00  get_Header__7_bundle [122]
  0.0       1.29     0.00      340     0.00     0.00  get_STD_index__13_phonemic_rep [123]
  0.0       1.29     0.00      340     0.00     0.00  get_Value__7_bundle [124]
  0.0       1.29     0.00      340     0.00     0.00  get_bundle__8_segment [125]
  0.0       1.29     0.00      340     0.00     0.00  get_marker__13_phonemic_rep [126]
  0.0       1.29     0.00      340     0.00     0.00  get_prosody__7_bundle [127]
  0.0       1.29     0.00      340     0.00     0.00  get_rc__7_bundle [128]
  0.0       1.29     0.00      340     0.00     0.00  set_stress__7_bundle12marker_value [
129]
  0.0       1.29     0.00      340     0.00     0.00  transfer_features__7_bundleP7_bundle
 [54]
  0.0       1.29     0.00      330     0.00     0.00  strcat [130]
  0.0       1.29     0.00      322     0.00     0.00  get_line__FP4FILE [69]
  0.0       1.29     0.00      304     0.00     0.00  get_stress__7_bundle [131]
  0.0       1.29      0.00       278       0.00       0.63    __9_segmentsi10mode_valueT2P4FILE13h
eading_value [16]
  0.0       1.29      0.00       274       0.00       0.63    __14_pronunciationP4FILE [20]
  0.0       1.29      0.00       264       0.00       0.00    _morecore [71]
  0.0       1.29      0.00       256       0.00       0.67    __9_lex_wordP4FILE [19]
  0.0       1.29      0.00       256       0.00       0.00    __t5Array2ZP14_pronunciationZ14_pron
unciationi [805]
  0.0       1.29      0.00       256       0.00       0.00    add__t5Array2ZP9_lex_wordZ9_lex_word
P9_lex_word [132]
  0.0       1.29      0.00       256       0.00       0.00    set_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciationPc [74]
  0.0       1.29      0.00       241       0.00       0.00    file_get_item__FP4FILEPc [65]
  0.0       1.29      0.00       162       0.00       0.00    heading2string__F13heading_value [13
3]
  0.C       1.29      0.00       158       0.00       0.00    get_state__8_segment [134]
  0.0       1.29      0.00       136       0.00       0.00    _realbufend [806]
  0.0       1.29      0.00       121       0.00       0.00    atoi [135]
  0.0       1.29      0.00       112       0.00       0.00    __14_pronunciation [807]
  0.0       1.29      0.00       112       0.00       0.00    __9_segmentsi10mode_value [808]
  0.0       1.29      0.00       112       0.00       0.00    change_subtitle__9_segments13heading
_valueT1 [136]
  0.0       1.29      0.00       112       0.00       1.38    construct_pronunciation__5_ruleP14_p
ronunciationiiiiP9_sentenceiiii [21]
  0.0       1.29      0.00       112       0.00       1.33    construct_segment__5_rulePP8_segment
i [22]
  0.0       1.29      0.00       112       0.00       0.00    set_title__9_segments13heading_value
 [137]
  0.0       1.29      0.00        98       0.00       0.00    get_NoOfData__t5Array2ZP5_wordZ5_wor
d [138]
  0.0       1.29      0.00        93       0.00       0.00    get_data__t5Array2ZP5_wordZ5_wordi [
139]
  0.0       1.29      0.00        79       0.00       0.01    __7_bundleP4FILE13heading_valueT2 [6
1]
  0.0       1.29      0.00        79       0.00       0.00    file_get_rorc__7_bundleP4FILE [68]
  0.0       1.29      0.00        69       0.00       0.00    nvmatch [140]
  0.0       1.29      0.00        67       0.00       0.00    .udiv [141]
  0.0       1.29      0.00        66       0.00       0.00    get_NoOfSegments__5_word [142]
  0.0       1.29      0.00        63       0.00       0.15    _fwrite_unlocked [45]
  0.0       1.29      0.00        63       0.00       0.00    fwrite [143]
  0.0       1.29      0.00        40       0.00       0.00    get_word__5_word [144]
  0.0       1.29      0.00        35       0.00       0.29    _xflsbuf [43]
  0.0       1.29      0.00        33       0.00       0.29    __flsbuf [46]
  0.0       1.29      0.00        30       0.00       0.00    get_time__8_segment [145]
  0.0       1.29      0.00        20       0.00       0.00    ___errno [809]
  0.0       1.29      0.00        20       0.00       0.00    _return_negone [810]
  0.0       1.29      0.00        20       0.00       0.00    thr_main [146]
  0.0       1.29      0.00        19       0.00       0.03    _fflush_u [63]
  0.0       1.29      0.00        13       0.00       0.00    __5_wordPci [80]
  0.0       1.29      0.00        13       0.00       0.00    add__5_treeP5_wordi [147]
  0.0       1.29      0.00        13       0.00       0.00    add__t5Array2ZP5_wordZ5_wordP5_word
[148]
  0.0       1.29      0.00        13       0.00       0.00    get_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciation [149]
  0.0       1.29      0.00        12       0.00       0.01    __filbuf [75]
  0.0       1.29      0.00        12       0.00       0.00    _read [811]
  0.0       1.29      0.00        11       0.00       0.00    _._8_lexicon [812]
  0.0       1.29      0.00        11       0.00       0.00    _._t5Array2ZP9_lex_wordZ9_lex_word [
813]
  0.0       1.29      0.00        11       0.00       0.00    __8_lexicon3nop [814]
  0.0       1.29      0.00        11       0.00       0.00    __t5Array2ZP5_wordZ5_wordi [815]
  0.0       1.29      0.00        11       0.00       0.00    __t5Array2ZP9_lex_wordZ9_lex_word3no
p [816]
  0.0       1.29      0.00        11       0.00      18.24    apply_rules__8_lexiconP6_rulesP9_sen
tencei [15]
  0.0       1.29      0.00        11       0.00       0.21    copy__8_lexicon [51]
  0.0       1.29      0.00        11      0.00     19.96  grow_aux__5_treeP6_rulesP8_lexiconi
[9]
  0.0       1.29      0.00         9      0.00      0.03  fflush [66]
  0.0       1.29      0.00         8      0.00      0.00  _findbuf [81]
  0.0       1.29      0.00         8      0.00      0.00  _ioctl [817]
  0.0       1.29      0.00         8      0.00      0.00  _isatty [818]
  0.0       1.29      0.00         8      0.00      0.00  _setbufend [819]
  0.0       1.29      0.00         7      0.00      0.00  _lseek64 [820]
  0.0       1.29      0.00         6      0.00      0.00  _cerror [821]
  0.0       1.29      0.00         6      0.00      0.00  _close [822]
  0.0       1.29      0.00         6      0.00      0.00  _endopen [823]
  0.0       1.29      0.00         6      0.00      0.00  _findiop [824]
  0.0       1.29      0.00         6      0.00      0.00  _fstat64 [825]
  0.0       1.29      0.00         6      0.00      0.00  _open [826]
  0.0       1.29      0.00         6      0.00      0.03  closefile__FP4FILE [72]
  0.0       1.29      0.00         6      0.00      0.03  fclose [73]
  0.0       1.29      0.00         6      0.00      0.00  fopen [150]
  0.0       1.29      0.00         6      0.00      0.00  readfile__FPc [151]
  0.0       1.29      0.00         5      0.00      0.00  getenv [83]
  0.0       1.29      0.00         4      0.00      0.00  .urem [152]
  0.0       1.29      0.00         2      0.00      0.63  __5_ruleP4FILE [57]
  0.0       1.29      0.00         2      0.00      0.63  __9_sentenceP4FILE [58]
  0.0       1.29      0.00         2      0.00      0.03  _flushlbf [76]
  0.0       1.29      0.00         2      0.00      0.00  _rw_rdlock [827]
  0.0       1.29      0.00         2      0.00      0.00  _time [828]
  0.0       1.29      0.00         2      0.00      0.00  add__t5Array2ZP5_ruleZ5_ruleP5_rule
[153]
  0.0       1.29      0.00         2      0.00      0.00  atexit [84]
  0.0       1.29      0.00         2      0.00      0.00  close__5_rule [154]
  0.0       1.29      0.00         2      0.00      0.00  output_aux__5_tree11destinationP4FIL
EPci11output_mode [155]
  0.0       1.29      0.00         2      0.00      0.00  rw_unlock [156]
  0.0       1.29      0.00         1      0.00      0.00  __5_treeP9_sentence [829]
  0.0       1.29      0.00         1      0.00      1.32  __6_rulesP4FILE [56]
  0.0       1.29      0.00         1      0.00    171.86  __8_lexiconP4FILE [18]
  0.0       1.29      0.00         1      0.00      0.00  __t5Array2ZP5_ruleZ5_rulei [830]
  0.0       1.29      0.00         1      0.00      0.00  __t5Array2ZP9_lex_wordZ9_lex_wordi [
831]
  0.0       1.29      0.00         1      0.00      0.00  _exithandle [89]
  0.0       1.29      0.00         1      0.00      0.00  _profil [832]
  0.0       1.29      0.00         1      0.00      0.00  _wrtchk [90]
  0.0       1.29      0.00         1      0.00      0.00  difftime [157]
  0.0       1.29      0.00         1      0.00      0.00  exit [88]
  0.0       1.29      0.00         1      0.00    219.58  grow__5_treeP6_rulesP8_lexicon [10]
  0.0       1.29      0.00         1      0.00    219.58  grow_aux2__5_treeP6_rulesP8_lexiconi
 [11]
  0.0       1.29      0.00         1      0.00    394.14  main [3]
  0.0       1.29      0.00         1      0.00      0.00  output__5_tree11destinationP4FILE [1
58]
```

```
************************************************************
*       NEW MATCHER RESULTS FOR:                           *
*                                                          *
*       "The baby can go to him in the bus today."         *
*                                                          *
************************************************************

#pragma ident   "@(#)gprof.flat.blurb   1.8     93/06/07 SMI"


flat profile:

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.
```

```
granularity: each sample hit covers 4 byte(s) for 2.00% of 0.50 seconds

  %   cumulative   self              self     total           
 time   seconds   seconds    calls  ms/call  ms/call  name    
 64.0       0.32     0.32                             internal_mcount [1]
  8.0       0.36     0.04   271264     0.00     0.00  get_feature__7_bundlei [13]
  4.0       0.38     0.02    59039     0.00     0.00  get_feature__13_phonemic_repi     <c
ycle 1> [16]
  4.0       0.40     0.02                             _brk_unlocked [18]
  2.0       0.41     0.01   330227     0.00     0.00  get_feature__8_segmenti   <cycle 1>
[12]
  2.0       0.42     0.01    75585     0.00     0.00  get_data__9_segmentsi [30]
  2.0       0.43     0.01    49745     0.00     0.00  _malloc_unlocked [37]
  2.0       0.44     0.01    10609     0.00     0.00  copy__8_segment [32]
  2.0       0.45     0.01     6895     0.00     0.01  feature_match__5_ruleP8_segmentT1 [1
1]
  2.0       0.46     0.01     3387     0.00     0.00  strcpy [33]
  2.0       0.47     0.01      112     0.09     0.09  get_bundle__8_segment [34]
  2.0       0.48     0.01      112     0.09     0.09  tree2seg__FP6_VowelP7_bundleP8_segme
nt [29]
  2.0       0.49     0.01       69     0.14     0.14  __11_VocalFoldsii [40]
  2.0       0.50     0.01        6     1.67     1.67  _open [39]
  0.0       0.50     0.00   103957     0.00     0.00  _return_zero [866]
  0.0       0.50     0.00    51977     0.00     0.00  _mutex_lock [867]
  0.0       0.50     0.00    51976     0.00     0.00  mutex_unlock [70]
  0.0       0.50     0.00    49388     0.00     0.00  cleanfree [71]
  0.0       0.50     0.00    49224     0.00     0.00  malloc [31]
  0.0       0.50     0.00    34484     0.00     0.00  _smalloc [53]
  0.0       0.50     0.00    29057     0.00     0.00  get_NoOfSegments__9_segments [72]
  0.0       0.50     0.00    28452     0.00     0.00  strcmp [73]
  0.0       0.50     0.00    19371     0.00     0.00  strsame__FPcT0 [74]
  0.0       0.50     0.00    19347     0.00     0.00  get_symbol__7_bundle [75]
  0.0       0.50     0.00    19347     0.00     0.00  get_symbol__8_segment [76]
  0.0       0.50     0.00    18431     0.00     0.00  exact_match__Fii [77]
  0.0       0.50     0.00    16537     0.00     0.00  realfree [78]
  0.0       0.50     0.00    11895     0.00     0.00  get_data__t5Array2ZP9_lex_wordZ9_lex
_wordi [79]
  0.0       0.50     0.00    11663     0.00     0.00  check_STD__Fv [80]
  0.0       0.50     0.00    10909     0.00     0.00  getGlide__6_Vowel [81]
  0.0       0.50     0.00    10721     0.00     0.00  __8_segment10mode_value [868]
  0.0       0.50     0.00    10609     0.00     0.00  __13_phonemic_repi12marker_value [86
9]
  0.0       0.50     0.00    10609     0.00     0.00  copy__13_phonemic_rep [82]
  0.0       0.50     0.00     8448     0.00     0.00  get_NoOfData__t5Array2ZP5_ruleZ5_rul
e [83]
  0.0       0.50     0.00     7314     0.00     0.00  getConsonant__6_Glide [84]
  0.0       0.50     0.00     6110     0.00     0.00  strcopy__FPc [41]
  0.0       0.50     0.00     6028     0.00     0.02  construct_pronunciation__5_ruleP14_p
ronunciationP9_sentencei [7]
  0.0       0.50     0.00     6028     0.00     0.00  get_data__t5Array2ZP5_ruleZ5_rulei [
85]
  0.0       0.50     0.00     5928     0.00     0.00  get_NoOfData__t5Array2ZP14_pronuncia
tionZ14_pronunciation [86]
  0.0       0.50     0.00     3992     0.00     0.00  getLingual__10_Consonant [87]
  0.0       0.50     0.00     3455     0.00     0.00  derive_main_stress__13_phonemic_rep
<cycle 1> [47]
  0.0       0.50     0.00     3455     0.00     0.00  derive_reduced__13_phonemic_rep   <c
ycle 1> [48]
  0.0       0.50     0.00     3455     0.00     0.01  matched__5_ruleP14_pronunciationiiii
 [14]
  0.0       0.50     0.00     3440     0.00     0.01  matched__5_ruleP9_sentenceiiii [15]
  0.0       0.50     0.00     3150     0.00     0.00  strlen [88]
  0.0       0.50     0.00     3138     0.00     0.00  get_data__t5Array2ZP14_pronunciation
```

```
Z14_pronunciationi [89]
  0.0       0.50     0.00      3125     0.00      0.00  matching__5_treeP14_pronunciationP9_
sentencei [25]
  0.0       0.50     0.00      3022     0.00      0.00  bundle_match__FP8_segmentT0 [27]
  0.0       0.50     0.00      3014     0.00      0.00  __14_pronunciation3nop [870]
  0.0       0.50     0.00      3014     0.00      0.00  __9_segments3nop [871]
  0.0       0.50     0.00      3014     0.00      0.00  copy__14_pronunciation [21]
  0.0       0.50     0.00      2827     0.00      0.00  get_NoOfData__t5Array2ZP9_lex_wordZ9
_lex_word [90]
  0.0       0.50     0.00      2816     0.00      0.00  __9_lex_word3nop [872]
  0.0       0.50     0.00      2816     0.00      0.00  __t5Array2ZP14_pronunciationZ14_pron
unciation [873]
  0.0       0.50     0.00      2816     0.00      0.03  apply_rules__9_lex_wordP6_rulesP9_se
ntencei [8]
  0.0       0.50     0.00      2816     0.00      0.01  copy__9_lex_word [20]
  0.0       0.50     0.00      2816     0.00      0.00  get_expanded__9_lex_word [91]
  0.0       0.50     0.00      2312     0.00      0.00  getBlade__8_Lingual [92]
  0.0       0.50     0.00      2284     0.00      0.00  getPharyngeal__6_Glide [93]
  0.0       0.50     0.00      2229     0.00      0.00  is_alphabet__Fc [94]
  0.0       0.50     0.00      1768     0.00      0.00  _free_unlocked [874]
  0.0       0.50     0.00      1768     0.00      0.00  free [95]
  0.0       0.50     0.00      1680     0.00      0.00  getBody__8_Lingual [96]
  0.0       0.50     0.00      1531     0.00      0.00  _ungetc_unlocked [875]
  0.0       0.50     0.00      1531     0.00      0.00  ungetc [97]
  0.0       0.50     0.00      1356     0.00      0.00  check_fvalue__Fi [98]
  0.0       0.50     0.00      1356     0.00      0.00  set_feature__7_bundleii [99]
  0.0       0.50     0.00      1334     0.00      0.00  get_word__FP4FILE [50]
  0.0       0.50     0.00      1164     0.00      0.00  getGlottis__11_Pharyngeal [100]
  0.0       0.50     0.00      1156     0.00      0.00  getVclFolds__6_Vowel [101]
  0.0       0.50     0.00      1133     0.00      0.00  compress__8_segment [102]
  0.0       0.50     0.00      1120     0.00      0.00  getPharynx__11_Pharyngeal [103]
  0.0       0.50     0.00      1021     0.00      0.00  __8_segment10mode_valueT1P4FILE13hea
ding_valueT4 [45]
  0.0       0.50     0.00       976     0.00      0.00  _sbrk [876]
  0.0       0.50     0.00       976     0.00      0.00  _sbrk_unlocked [877]
  0.0       0.50     0.00       942     0.00      0.00  __13_phonemic_repP4FILE [878]
  0.0       0.50     0.00       942     0.00      0.00  phoneme2index__9_sentencePc [104]
  0.0       0.50     0.00       636     0.00      0.00  getLips__10_Consonant [105]
  0.0       0.50     0.00       632     0.00      0.00  getSftP__6_Glide [106]
  0.0       0.50     0.00       538     0.00      0.00  __6_Vowel [879]
  0.0       0.50     0.00       513     0.00      0.00  string2feature_index__FPc [107]
  0.0       0.50     0.00       488     0.00      0.00  _morecore [880]
  0.0       0.50     0.00       481     0.00      0.00  setConsonant__6_GlideP10_Consonant [
108]
  0.0       0.50     0.00       448     0.00      0.00  getSonorant__10_Consonant [109]
  0.0       0.50     0.00       441     0.00      0.00  setSftP__6_GlideP12_Soft_Palate [110
]
  0.0       0.50     0.00       441     0.00      0.00  setVcl__6_VowelP11_VocalFolds [111]
  0.0       0.50     0.00       412     0.00      0.00  getRound__5_Lips [112]
  0.0       0.50     0.00       405     0.00      0.00  __11_Pharyngeal [881]
  0.0       0.50     0.00       405     0.00      0.00  __6_Glide [882]
  0.0       0.50     0.00       405     0.00      0.00  __8_Lingual [883]
  0.0       0.50     0.00       405     0.00      0.00  setBlade__8_LingualP6_Blade [113]
  0.0       0.50     0.00       405     0.00      0.00  setBody__8_LingualP5_Body [114]
  0.0       0.50     0.00       405     0.00      0.00  setGlide__6_VowelP6_Glide [115]
  0.0       0.50     0.00       405     0.00      0.00  setGlottis__11_PharyngealP8_Glottis
[116]
  0.0       0.50     0.00       405     0.00      0.00  setLingual__10_ConsonantP8_Lingual [
117]
  0.0       0.50     0.00       405     0.00      0.00  setLips__10_ConsonantP5_Lips [118]
  0.0       0.50     0.00       405     0.00      0.00  setPharyngeal__6_GlideP11_Pharyngeal
 [119]
  0.0       0.50     0.00       405     0.00      0.00  setPharynx__11_PharyngealP8_Pharynx
[120]
  0.0       0.50     0.00       386     0.00      0.00  add__t5Array2ZP14_pronunciationZ14_p
```

```
ronunciationP14_pronunciation [121]
  0.0        0.50     0.00       380      0.00       0.00   getConstr__8_Glottis [122]
  0.0        0.50     0.00       372      0.00       0.00   getAnt__6_Blade [123]
  0.0        0.50     0.00       372      0.00       0.00   getDistr__6_Blade [124]
  0.0        0.50     0.00       372      0.00       0.00   getNasal__12_Soft_Palate [125]
  0.0        0.50     0.00       359      0.00       0.00   add__9_segmentsP8_segment [126]
  0.0        0.50     0.00       359      0.00       0.00   set_mode__8_segment10mode_value [127
]
  0.0        0.50     0.00       336      0.00       0.00   __10_Consonant [884]
  0.0        0.50     0.00       336      0.00       0.00   __11_VocalFolds [885]
  0.0        0.50     0.00       336      0.00       0.00   __12_Soft_Palate [886]
  0.0        0.50     0.00       336      0.00       0.00   __5_Body [887]
  0.0        0.50     0.00       336      0.00       0.00   __5_Lips [888]
  0.0        0.50     0.00       336      0.00       0.00   __6_Blade [889]
  0.0        0.50     0.00       336      0.00       0.00   __8_Glottis [890]
  0.0        0.50     0.00       336      0.00       0.00   __8_Pharynx [891]
  0.0        0.50     0.00       336      0.00       0.00   getAtr__8_Pharynx [128]
  0.0        0.50     0.00       336      0.00       0.00   getBack__5_Body [129]
  0.0        0.50     0.00       336      0.00       0.00   getContinuant__10_Consonant [130]
  0.0        0.50     0.00       336      0.00       0.00   getCtr__8_Pharynx [131]
  0.0        0.50     0.00       336      0.00       0.00   getDominant__10_Consonant [132]
  0.0        0.50     0.00       336      0.00       0.00   getDominant__6_Glide [133]
  0.0        0.50     0.00       336      0.00       0.00   getHigh__5_Body [134]
  0.0        0.50     0.00       336      0.00       0.00   getLat__6_Blade [135]
  0.0        0.50     0.00       336      0.00       0.00   getLow__5_Body [136]
  0.0        0.50     0.00       336      0.00       0.00   getRhotic__6_Blade [137]
  0.0        0.50     0.00       336      0.00       0.00   getSlack__11_VocalFolds [138]
  0.0        0.50     0.00       336      0.00       0.00   getSpread__8_Glottis [139]
  0.0        0.50     0.00       336      0.00       0.00   getStiff__11_VocalFolds [140]
  0.0        0.50     0.00       336      0.00       0.00   getStrident__10_Consonant [141]
  0.0        0.50     0.00       336      0.00       0.00   makeTree__FP6_Vowel [142]
  0.0        0.50     0.00       336      0.00       0.00   setContinuant__10_Consonanti [143]
  0.0        0.50     0.00       336      0.00       0.00   setSonorant__10_Consonanti [144]
  0.0        0.50     0.00       330      0.00       0.00   strcat [145]
  0.0        0.50     0.00       322      0.00       0.00   get_line__FP4FILE [56]
  0.0        0.50     0.00       278      0.00       0.04   __9_segmentsi10mode_valueT2P4FILE13h
eading_value [24]
  0.0        0.50     0.00       274      0.00       0.04   __14_pronunciationP4FILE [26]
  0.0        0.50     0.00       267      0.00       0.00   t_delete [146]
  0.0        0.50     0.00       258      0.00       0.00   setDominant__10_Consonanti [147]
  0.0        0.50     0.00       256      0.00       0.05   __9_lex_wordP4FILE [23]
  0.0        0.50     0.00       256      0.00       0.00   __t5Array2ZP14_pronunciationZ14_pron
unciationi [892]
  0.0        0.50     0.00       256      0.00       0.00   add__t5Array2ZP9_lex_wordZ9_lex_word
P9_lex_word [148]
  0.0        0.50     0.00       256      0.00       0.00   set_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciationPc [49]
  0.0        0.50     0.00       241      0.00       0.00   file_get_item__FP4FILEPc [46]
  0.0        0.50     0.00       231      0.00       0.00   setDominant__6_Glidei [149]
  0.0        0.50     0.00       224      0.00       0.00   copyTree__FP6_VowelT0 [150]
  0.0        0.50     0.00       224      0.00       0.00   get_treeData__9_segmentsi [151]
  0.0        0.50     0.00       224      0.00       0.00   setAnt__6_Bladei [152]
  0.0        0.50     0.00       224      0.00       0.00   setAtr__8_Pharynxi [153]
  0.0        0.50     0.00       224      0.00       0.00   setBack__5_Bodyi [154]
  0.0        0.50     0.00       224      0.00       0.00   setConstr__8_Glottisi [155]
  0.0        0.50     0.00       224      0.00       0.00   setCtr__8_Pharynxi [156]
  0.0        0.50     0.00       224      0.00       0.00   setDistr__6_Bladei [157]
  0.0        0.50     0.00       224      0.00       0.00   setHigh__5_Bodyi [158]
  0.0        0.50     0.00       224      0.00       0.00   setLat__6_Bladei [159]
  0.0        0.50     0.00       224      0.00       0.00   setLow__5_Bodyi [160]
  0.0        0.50     0.00       224      0.00       0.00   setNasal__12_Soft_Palatei [161]
  0.0        0.50     0.00       224      0.00       0.00   setRhotic__6_Bladei [162]
  0.0        0.50     0.00       224      0.00       0.00   setRound__5_Lipsi [163]
  0.0        0.50     0.00       224      0.00       0.00   setSlack__11_VocalFoldsi [164]
  0.0        0.50     0.00       224      0.00       0.00   setSpread__8_Glottisi [165]
  0.0       0.50     0.00      224     0.00     0.00  setStiff__11_VocalFoldsi [166]
  0.0       0.50     0.00      224     0.00     0.00  setStrident__10_Consonanti [167]
  0.0       0.50     0.00      162     0.00     0.00  heading2string__F13heading_value [16
8]
  0.0       0.50     0.00      158     0.00     0.00  get_state__8_segment [169]
  0.0       0.50     0.00      148     0.00     0.00  get_cv__5_rule [170]
  0.0       0.50     0.00      148     0.00     0.00  makeChange__FP6_VowelT0i [171]
  0.0       0.50     0.00      141     0.00     0.00  _realbufend [893]
  0.0       0.50     0.00      112     0.00     0.00  _._10_Consonant [894]
  0.0       0.50     0.00      112     0.00     0.00  _._11_Pharyngeal [895]
  0.0       0.50     0.00      112     0.00     0.00  _._11_VocalFolds [896]
  0.0       0.50     0.00      112     0.00     0.00  _._12_Soft_Palate [897]
  0.0       0.50     0.00      112     0.00     0.00  _._5_Body [898]
  0.0       0.50     0.00      112     0.00     0.00  _._5_Lips [899]
  0.0       0.50     0.00      112     0.00     0.00  _._6_Blade [900]
  0.0       0.50     0.00      112     0.00     0.00  _._6_Glide [901]
  0.0       0.50     0.00      112     0.00     0.00  _._6_Vowel [902]
  0.0       0.50     0.00      112     0.00     0.00  _._8_Glottis [903]
  0.0       0.50     0.00      112     0.00     0.00  _._8_Lingual [904]
  0.0       0.50     0.00      112     0.00     0.00  _._8_Pharynx [905]
  0.0       0.50     0.00      112     0.00     0.00  __14_pronunciation [906]
  0.0       0.50     0.00      112     0.00     0.00  __7_bundle [907]
  0.0       0.50     0.00      112     0.00     0.00  __9_segmentsi10mode_value [908]
  0.0       0.50     0.00      112     0.00     0.00  change_subtitle__9_segments13heading
_valueT1 [172]
  0.0       0.50     0.00      112     0.00     0.19  construct_pronunciation__5_ruleP14_p
ronunciationiiiiP9_sentenceiiiii [17]
  0.0       0.50     0.00      112     0.00     0.00  getDominant__6_Vowel [173]
  0.0       0.50     0.00      112     0.00     0.00  get_RULErc__7_bundle [174]
  0.0       0.50     0.00      112     0.00     0.00  get_STD_index__13_phonemic_rep [175]
  0.0       0.50     0.00      112     0.00     0.00  get_phonemic__8_segment [176]
  0.0       0.50     0.00      112     0.00     0.00  get_sc_l__5_rule [177]
  0.0       0.50     0.00      112     0.00     0.00  get_sc_r__5_rule [178]
  0.0       0.50     0.00      112     0.00     0.00  set_data__8_segmentP7_bundle [179]
  0.0       0.50     0.00      112     0.00     0.00  set_title__9_segments13heading_value
 [180]
  0.0       0.50     0.00       98     0.00     0.00  get_NoOfData__t5Array2ZP5_wordZ5_wor
d [181]
  0.0       0.50     0.00       95     0.00     0.00  t_splay [182]
  0.0       0.50     0.00       93     0.00     0.00  get_data__t5Array2ZP5_wordZ5_wordi [
183]
  0.0       0.50     0.00       79     0.00     0.01  __7_bundleP4FILE13heading_valueT2 [4
4]
  0.0       0.50     0.00       70     0.00     0.00  .udiv [184]
  0.0       0.50     0.00       69     0.00     0.00  __10_Consonantiii [909]
  0.0       0.50     0.00       69     0.00     0.00  __12_Soft_Palatei [910]
  0.0       0.50     0.00       69     0.00     0.00  __5_Bodyiii [911]
  0.0       0.50     0.00       69     0.00     0.00  __5_Lipsi [912]
  0.0       0.50     0.00       69     0.00     0.00  __6_Bladeiiii [913]
  0.0       0.50     0.00       69     0.00     0.00  __8_Glottisii [914]
  0.0       0.50     0.00       69     0.00     0.00  __8_Pharynxii [915]
  0.0       0.50     0.00       69     0.00     0.00  determineDominant__FP8_segment [58]
  0.0       0.50     0.00       69     0.00     0.00  file_get_rorc__7_bundleP4FILE [57]
  0.0       0.50     0.00       69     0.00     0.15  makeTree__FP6_VowelP8_segment [28]
  0.0       0.50     0.00       69     0.00     0.00  nvmatch [185]
  0.0       0.50     0.00       69     0.00     0.00  setDominant__FiP6_Vowel [186]
  0.0       0.50     0.00       66     0.00     0.00  _fwrite_unlocked [66]
  0.0       0.50     0.00       66     0.00     0.00  fwrite [187]
  0.0       0.50     0.00       66     0.00     0.00  get_NoOfSegments__5_word [188]
  0.0       0.50     0.00       40     0.00     0.00  get_word__5_word [189]
  0.0       0.50     0.00       36     0.00     0.00  _write [916]
  0.0       0.50     0.00       36     0.00     0.00  _xflsbuf [917]
  0.0       0.50     0.00       34     0.00     0.00  __flsbuf [918]
  0.0       0.50     0.00       30     0.00     0.00  get_time__7_bundle [190]
  0.0       0.50     0.00       30     0.00     0.00  get_time__8_segment [191]
```

```
   0.0        0.50     0.00          28       0.00       0.00   setDominant__6_Voweli [192]
   0.0        0.50     0.00          20       0.00       0.00   ___errno [919]
   0.0        0.50     0.00          20       0.00       0.00   _return_negone [920]
   0.0        0.50     0.00          20       0.00       0.00   thr_main [193]
   0.0        0.50     0.00          19       0.00       0.00   _fflush_u [921]
   0.0        0.50     0.00          13       0.00       0.00   __5_wordPci [59]
   0.0        0.50     0.00          13       0.00       0.00   add__5_treeP5_wordi [194]
   0.0        0.50     0.00          13       0.00       0.00   add__t5Array2ZP5_wordZ5_wordP5_word
[195]
   0.0        0.50     0.00          13       0.00       0.00   get_Label__t5Array2ZP14_pronunciatio
nZ14_pronunciation [196]
   0.0        0.50     0.00          12       0.00       0.00   __filbuf [62]
   0.0        0.50     0.00          12       0.00       0.00   _read [922]
   0.0        0.50     0.00          11       0.00       0.00   _._8_lexicon [923]
   0.0        0.50     0.00          11       0.00       0.00   _._t5Array2ZP9_lex_wordZ9_lex_word [
924]
   0.0        0.50     0.00          11       0.00       0.00   __8_lexicon3nop [925]
   0.0        0.50     0.00          11       0.00       0.00   __t5Array2ZP5_wordZ5_wordi [926]
   0.0        0.50     0.00          11       0.00       0.00   __t5Array2ZP9_lex_wordZ9_lex_word3no
p [927]
   0.0        0.50     0.00          11       0.00       8.95   apply_rules__8_lexiconP6_rulesP9_sen
tencei [9]
   0.0        0.50     0.00          11       0.00       1.74   copy__8_lexicon [19]
   0.0        0.50     0.00          11       0.00      11.72   grow_aux__5_treeP6_rulesP8_lexiconi
[4]
   0.0        0.50     0.00          10       0.00       0.00   file_get_RULErorc__7_bundleP4FILE [6
0]
   0.0        0.50     0.00           9       0.00       0.00   fflush [197]
   0.0        0.50     0.00           8       0.00       0.00   _findbuf [61]
   0.0        0.50     0.00           8       0.00       0.00   _ioctl [928]
   0.0        0.50     0.00           8       0.00       0.00   _isatty [929]
   0.0        0.50     0.00           8       0.00       0.00   _setbufend [930]
   0.0        0.50     0.00           7       0.00       0.00   _lseek64 [931]
   0.0        0.50     0.00           6       0.00       0.00   _cerror [932]
   0.0        0.50     0.00           6       0.00       0.00   _close [933]
   0.0        0.50     0.00           6       0.00       1.67   _endopen [38]
   0.0        0.50     0.00           6       0.00       0.00   _findiop [934]
   0.0        0.50     0.00           6       0.00       0.00   _fstat64 [935]
   0.0        0.50     0.00           6       0.00       0.00   closefile__FP4FILE [198]
   0.0        0.50     0.00           6       0.00       0.00   fclose [199]
   0.0        0.50     0.00           6       0.00       1.67   fopen [35]
   0.0        0.50     0.00           6       0.00       1.67   readfile__FPc [36]
   0.0        0.50     0.00           5       0.00       0.00   getenv [200]
   0.0        0.50     0.00           4       0.00       0.00   .urem [201]
   0.0        0.50     0.00           2       0.00       0.04   __5_ruleP4FILE [54]
   0.0        0.50     0.00           2       0.00       0.04   __9_sentenceP4FILE [55]
   0.0        0.50     0.00           2       0.00       0.00   _flushlbf [936]
   0.0        0.50     0.00           2       0.00       0.00   _rw_rdlock [937]
   0.0        0.50     0.00           2       0.00       0.00   _rw_unlock [938]
   0.0        0.50     0.00           2       0.00       0.00   _time [939]
   0.0        0.50     0.00           2       0.00       0.00   add__t5Array2ZP5_ruleZ5_ruleP5_rule
[202]
   0.0        0.50     0.00           2       0.00       0.00   atexit [203]
   0.0        0.50     0.00           2       0.00       0.00   atoi [204]
   0.0        0.50     0.00           2       0.00       0.00   close__5_rule [205]
   0.0        0.50     0.00           2       0.00       0.06   output_aux__5_tree11destinationP4FIL
EPci11output_mode [52]
   0.0        0.50     0.00           1       0.00       0.00   __5_treeP9_sentence [940]
   0.0        0.50     0.00           1       0.00       3.42   __6_rulesP4FILE [43]
   0.0        0.50     0.00           1       0.00      11.69   __8_lexiconP4FILE [22]
   0.0        0.50     0.00           1       0.00       0.00   __t5Array2ZP5_ruleZ5_rulei [941]
   0.0        0.50     0.00           1       0.00       0.00   __t5Array2ZP9_lex_wordZ9_lex_wordi [
942]
   0.0        0.50     0.00           1       0.00       0.00   _exithandle [943]
   0.0        0.50     0.00           1       0.00       0.00   _memcpy [944]
```

```
  0.0      0.50    0.00        1     0.00      0.00  _profil [945]
  0.0      0.50    0.00        1     0.00      0.00  _wrtchk [67]
  0.0      0.50    0.00        1     0.00      0.00  difftime [206]
  0.0      0.50    0.00        1     0.00      0.00  exit [207]
  0.0      0.50    0.00        1     0.00    128.96  grow__5_treeP6_rulesP8_lexicon [5]
  0.0      0.50    0.00        1     0.00    128.96  grow_aux2__5_treeP6_rulesP8_lexiconi
 [6]
  0.0      0.50    0.00        1     0.00    150.97  main [2]
  0.0      0.50    0.00        1     0.00      0.12  output__5_tree11destinationP4FILE [5
1]
```

THESIS PROCESSING SLIP

FIXED FIELD: ill. ______ name ______

index ______ biblio ______

▶ COPIES: [✓ Archives] Aero Dewey [✓ Eng] Hum

Lindgren Music Rotch Science

TITLE VARIES: ▶☐ ______

NAME VARIES: ▶☑ Theatre

IMPRINT: (COPYRIGHT) ______

▶ COLLATION: 104 l

▶ ADD: DEGREE: ______ ▶ DEPT.: ______

SUPERVISORS: ______

NOTES:

cat'r: date:

page:

▶ [illegible]

▶ DEPT: E.P.

▶ YEAR: 1979 ▶ DEGREE: [illegible]

▶ NAME: MALDONADO [illegible]