An Algorithm for High Accuracy Name Pronunciation by Parametric Speech Synthesizer

Tony Vitale*
Digital Equipment Corporation

Automatic and accurate pronunciation of personal names by parametric speech synthesizer has become a crucial limitation for applications within the telecommunications industry, since the technology is needed to provide new automated services such as reverse directory assistance (number to name).

Within text-to-speech technology, however, it was not possible to offer such functionality. This was due to the inability of a text-to-speech device optimized for a specific language (e.g., American English) to accurately pronounce names that originate from very different language families. That is, a telephone book from virtually any section of the country will contain names from scores of languages as diverse as English and Mandarin, French and Japanese, Irish and Polish. All such non–Anglo-Saxon names have traditionally been mispronounced by a speech synthesizer resulting in gross errors and unintelligible speech.

This paper describes how an algorithm for high accuracy name pronunciation was implemented in software based on a combination of cryptanalysis, statistics, and linguistics. The algorithm behind the utility is a two-stage procedure: (1) the decoding of the name to determine its etymological grouping; and (2) specific letter-to-sound rules (both segmental rules as well as stress-assignment rules) that provide the synthesizer parameters with sufficient additional information to accurately pronounce the name as would a typical speaker of American English. Default language and thresholds are settable parameters and are also described. While the complexity of the software is invisible to applications writers as well as users, this functionality now makes possible the automation of highly accurate name pronunciation by parametric speech synthesizer.

1. Background

There has been a great deal of interest recently in the generation of accurate phonetic equivalences for proper names. New and enhanced services in the telecommunications industry as well as the increasing interest in speech I/O for the workstation has renewed interest in applications such as the automation of name pronunciation by speech synthesizer in reverse directory assistance (number to name) applications (Karhan et al. 1986). In addition, speech recognition research can benefit by automatic lexicon construction to be ultimately used in such applications as directory assistance (name to number) and a variety of workstation applications (Cole et al. 1989).

* 30 Forbes Rd. (NRO5/14), Northboro, MA 01532 USA

The inaccuracy of name pronunciation by parametric speech synthesizer has been a problem often addressed in the literature (Church 1986; Golding and Rosenbloom 1991; Liu and Haas 1988; Macchi and Spiegel 1990; Spiegel 1985, 1990; Spiegel and Macchi 1990; Vitale 1987, 1989a, 1989b, and others). The difficulty stemmed from the fact that high-quality speech synthesizers were so optimized for a particular language (e.g., American English), that a non-English form such as an unassimilated or partially assimilated loanword would be processed according to English letter-to-sound rules only.[1] Since non-Anglo-Saxon personal names fall into the category of loanwords, the pronunciation of these forms ranged from slightly inaccurate to grossly unintelligible.

1.1 General Letter-to-Sound Rules

Letter-to-sound rules are a requirement in any text-to-speech architecture and take slightly different forms from system to system; however, they typically follow a standard linguistic format such as **x** → **y**/**z**, where **x** is some grapheme sequence, **y** some phoneme sequence, and **z** the environment, usually graphemic. The following is a typical example of a set of letter to sound rules:

$$\begin{array}{lll} C & \rightarrow & /s/ \quad /-\{E,I,Y\} \\ C & \rightarrow & /k/ \end{array}$$

This set would handle all such forms as CELLAR, CILIA, CY, CAT, COD, etc., but clearly not loanwords such as CELLO for exactly the same reasons that make the pronunciation of last names so difficult for a synthesizer having only English letter-to-sound rules. A number of letter-to-sound rule sets are in the public domain, (e.g., Hunnicutt 1976; Divay 1984, 1990). However, many rule sets that are currently in use in commercial speech synthesizers remain confidential. Venezky (1970) contains an extensive discussion of issues involving phoneme-grapheme correspondence.

The accuracy of pronunciation of normal text in high-quality speech synthesizers using exclusively or primarily letter-to-sound processing can now range as high as 95+%.[2] In tests we ran, however, this accuracy (without dictionary lookup), was degraded by as much as 30% or more when the corpus changed to high-frequency proper names. The degradation was even higher when the names were chosen at random and could be from any language group. Spiegel (1985) cites the average error rate for the pronunciation of names over four synthesizers as 28.7%, which was consistent with our results.

The reason for this degradation is due to the fact that the phonological intelligence of a speech synthesizer for a given language cannot discriminate among loanwords that are not contained in its memory (i.e., dictionary). In the case of names, these are really loanwords ranging from the commonly found Indo-European languages such as French, Italian, Polish, Spanish, German, Irish, etc. to the more "exotic" ones such as Japanese, Armenian, Chinese, Latvian, Arabic, Hungarian, and Vietnamese. Clearly, the pronunciation of these names from the many ethnic groups does not conform to the phonological pattern of English. For example, as pronounced by the average English speaker, most German names have syllable-initial stress, Japanese and Spanish names tend to have penultimate stress, and some French names have word-final stress.

1 That is, phonemic rules. Obviously, the phonetics output by a synthesizer would not be sufficient for multiple languages.

2 In an informal study, Klatt (personal communication) tested our rule set for English by replicating a study by Bill Huggins (Bolt, Beranek and Newman) using letter to sound rules *without dictionary* over 1678 complex polysyllabic forms. The algorithm tested (and the one used in this study) had an error rate of 5.1%. The error rate using a dictionary would be much lower.

Chinese names tend to be monosyllabic and consequently stress is a non-issue; in Italian names, stress may be penultimate or antepenultimate as is the case with Slavic languages and certain other groups.

But while stress patterns are relatively few in number, the letter-to-sound correspondences are extremely varied. For example, the orthographic sequence CH is pronounced [č] in English names e.g., CHILDERS, [š] in French names e.g., CHARPENTIER, and [k] in Italian names e.g. BRONCHETTI or the anglicized version of some German names e.g., BACH. This means that letter-to-sound must account for a potentially large number of diverse languages in order to output the correct phonetics.

Most researchers understand that in order to process the name accurately, at least two parameters must be known: (1) that the string is a name and thus needs to be processed by a special algorithm; and (2) that the string must be identified with a particular set of languages or language groups such that the specifics of the pronunciation (i.e., the letter-to-sound rules) can be formally described (Church 1986; Liu and Haas 1988; and others). While there has been some interest in attempting to identify a word as a name from random text, this present work assumes a database in which name fields are indexed as such (e.g., a machine-readable telephone directory) and no further mention of this will be made. This paper simply describes an implementation of this two-stage process, and details the first stage — the correct identification of a name as belonging to a certain language group. It should be stressed that there have been other attempts to implement similar algorithms, although few descriptions of such implementations are available.

1.2 Language Groups

For purposes of identification, sets of similar languages are more efficiently grouped together. However, the language groups used in this study may not always correspond to the set of language families familiar to most linguists. For example, while Japanese or Greek may be in groups by themselves, languages such as Spanish and Portuguese may be grouped together into a So. Romance group and this set may be different from, say, Italian, which may be grouped with Rumanian, or French, which may be grouped by itself. This is done to reduce the complexity of letter-to-sound (Section 4.1). However, the software is set up such that groupings can be moved around to accommodate different letter-to-sound rule sets. In addition, the number of groups is a variable parameter and could be modified as would the inclusion of any new rule sets in the letter-to-sound subsystem. Thus, for n language groups, the probability P of some language group L_i being the correct etymology is $P(L_i) = \frac{1}{n}$.

1.3 Etymology

Identification of a particular language group in the United States and many countries of Western Europe is not an easy task. According to the United States Social Security files (Smith 1969), there are approximately 1.5 million different last names in the United States, with about one-third of these being unique in that they occur only once in the register.[3] Furthermore, the etymologies of the names span the entire range of the world's languages, although the spread of these groupings is obviously related to geopolitical units and historical patterns of immigration and is different in the United States than it is, say, in Iceland, Ireland, or Italy.

3 Spiegel (1985) points this out. This is an excellent article that contains a number of useful statistics on personal names.

2. Role of the Dictionary

The first step in the process was the construction of a dictionary that contained both common and unusual names in their orthographic representation and phonetic equivalent. All sophisticated speech synthesizers today use a lexical database for dictionary lookup to process words that are, for one reason or another, exceptions to the rule. In generic synthesizers, these are typically functors that undergo vowel or stress reduction, partially assimilated or unassimilated loanwords that cannot be processed by language-specific letter-to-sound rules, abbreviations that are both generic and domain-specific, homographs that need to be distinguished phonetically, and selected proper nouns, such as geographical place names or company names.

In the case of proper surnames, however, dictionary lookups, while necessary, are of limited use. There are a number of reasons for this. First, while the most common names would have an extremely high hit rate (much like functors in a generic system), the curve quickly becomes asymptotic. Church (1986) has shown that while the most common 2,000 names can account for 46% of the Kansas City telephone book, it would take 40,000 entries to obtain a 93% accuracy rate. Furthermore, accuracy would decrease if one considers that geographic area has a profound influence on name grouping, and thus the figures for a large East or West Coast metropolitan area would certainly be significantly lower. It can be easily shown that the functional load of each name changes with the geographical location.[4] The name SCHMIDT, for example, is not in the list of the most frequent 2,000 names, yet it appears in the Social Security files as the most common name in Milwaukee (Spiegel 1985). Liu and Haas (1988) conducted a similar experiment that included 75 million households in the U.S. The first few thousand names account for 60% of the database, but the curve flattens out after 50,000 names and it would take 175,000 names in a dictionary to cover 88.7% of the population. This would mean that even with an extremely large dictionary (each entry of which would have to be phoneticized), there would still be an error rate of over 11%.

Even with these limitations, dictionary lookups are still quite important. Frequently occurring names, like functors, have a high functional load (above). Spiegel (1985) claims that if the most common 5,000 names are used in a dictionary for a population of 10 million people, even if letter-to-sound had an accuracy of only 75% (which is extremely low for a high-quality speech synthesizer), the error rate would be $< 2.5\%$. Most other researchers have also assumed a dictionary lookup as part of any procedure to increase the accuracy of name pronunciation. Therefore the general flow of text from the grapheme space to the phonetic realization must proceed first through a dictionary. Common last names such as SMITH, JOHNSON, WILLIAMS, BROWN, JONES, MILLER, DAVIS, WILSON, ANDERSON, TAYLOR, etc. and common names (both first and last names) from a variety of other languages should be included. The size of this dictionary is up to its creator. The dictionary used in this software contained about 4,000 lexical entries that were proper names.[5] There is, however, no reason to exclude

4 Functional load here is used in a slightly different sense than in linguistics. The functional load of a grapheme is its frequency of occurrence, in relation to other graphemes in the language, weighted equally, as measured over a sizable corpus of orthographic data.

5 In practice, the name dictionary could be contained within a larger dictionary that would be part of a genetic text-to-speech system. Moreover, the dictionary should be easily modifiable by an applications writer. Functions such as *add, remove, find, modify,* and the like can be used to maximize the effect of the dictionary, especially if some preliminary analysis has been done on population statistics. Experience has also shown that a programmer should be able to easily merge new word or name lists with a base dictionary and quickly examine a variety of statistics including the size in entries, bytes, or blocks as

very large dictionaries (e.g., > 50,000 words) although the choice of a search algorithm then becomes more important in real-time implementations.

When a dictionary lookup is used and a match occurs, the result is simply a translation from graphemes to phonemes, and the phoneme string (along with many other acoustic parameters picked up along the way) is output to the synthesizer.[6] When there is no match, (i.e., most cases), however, some algorithm is needed to increase pronunciation accuracy.

3. Identification Pass

It is assumed that certain textual elements are identified as names and are intentionally processed as such. This algorithm does not address the identification of proper names in random text, although there has been some activity in this area in recent years with the increased attention to voice prosthesis, remote access to electronic mail, and other applications. In database retrieval applications this is not usually a problem, since names fields in a database are typically marked by some field identifier. Similarly, the syntax of electronic mail message headers can often be used to mark a personal name.

The first stage in the identification procedure is the analysis of the sequence of graphemes that makes up the name, and its indexing as belonging to some language group. The concept of identification by orthographic trigram is by no means new and has been discussed in the literature (e.g., Church 1986; Liu and Haas 1988; and others). In our implementation, the identification is a complex procedure that includes filter rules for identification or elimination, graphemic (non-trigram) and morphological analysis, as well as trigram analysis. While this scheme may seem complex, it will run in real time, and thus the complexity is invisible to the user.

3.1 Filter Rules

It is well known in both linguistics and cryptanalysis that a text string from a language L_i will have unique sequence characteristics that distinguish L_i from all other languages in the set $\{L_i, L_j, \ldots L_n\}$. All alphabetic languages (as opposed to syllabaries or ideographic systems) have a quantifiable functional load of graphemes as well as phonemes, and this functional load will differ greatly from language to language. We have therefore created a set of rules that we call filter rules. Filter rules are rules that may positively identify a name or positively eliminate a name from further consideration. The use of nonoccurrence is not new but is refined to include a more elaborate filter mechanism for variable length grapheme sequences. When scanning the name to determine etymology, if the name cannot be positively identified, it is more efficient to eliminate some groups from consideration, thereby increasing the speed of the search (below).

There are some unique identification characteristics of grapheme strings from certain languages. In these cases, a grapheme G may help identify a string as being from L_i. For example, the grapheme E in English is well known as the most common letter, and has a functional load of 12.4% (Daly 1987). Scrabble and similar games are interesting indicators of this and mark functional load of graphemes by values of individual letters; the lower the value, the higher the functional load. Naturally, quantitative differences occur from language to language. While Z has an extremely low functional load in English, it is one of the most common letters in Polish. As an example of this

well as the average length of each field of an entry.

6 A dictionary entry in currently-used generic text-to-speech algorithms is really nothing more than a complex context-free letter-to-sound rule.

metric, if we take 1+ occurrences of the letter K in a name over the total number of unique names in a corpus, in Japanese, the frequency of this letter is 40.1%, in German it is only 18.9% and in Italian, the letter does not occur. Since the distribution of letters in proper names will differ from that of the general lexicon, statistics on letter frequency in names should be compiled independently but could be used for determining probabilities. Similarly, the orthographic length of a name, like the length of a common noun, could, in a more elaborate scheme, be also used as a factor in determining probabilities. In the dictionary used in this study, both names and non-names together had an average length of slightly under 7.5 graphemes. This coincides with the findings of Daly (1987), in which normal words had an average length of 7.35 graphemes.[7] Neither of these were used as factors in determining probabilities in this implementation.

Sequences of graphemes are much more useful in determining the identification of a language group. Sequences of 2 or more letters including larger morphological elements within a name may be considered characteristic of a language group although each of these may also effectively exclude a set of other language groups. For example, sequences such as CZ, PF, SH, EE (or longer ones) unambiguously define certain language groups. A trivial example of this would be the sequence #MC (where # is a word-boundary), which unambiguously identifies the word as Irish resulting in a probability of 1 for the identification of the corresponding language group.

However, even if a sequence cannot identify a language group unambiguously, the filter rules often eliminate one or more groups from consideration, thereby drastically altering the statistical chances of an incorrect guess. As might be expected, the longer the (legal) sequence in L_i, the less likely it is to occur in another language group. In some languages, either alphabetic ones or those that are transliterated into alphabetic systems (e.g., Japanese), certain letters do not occur. For example, the letter L does not occur in Japanese, X does not occur in Polish, J does not occur in Italian, and so on. The occurrence of any of these graphemes in a name string would then immediately eliminate that language from consideration. Thus, if m language groups have been eliminated, the probability of some language group L_i being the correct etymology is now $P(L_i) = \frac{1}{n-m}$. Analysis has shown that the filter rules eliminate an average of 54% of all possible language groups.[8]

Filter rules, therefore, consist of (a) identification rules and (b) elimination rules. Identification rules match a grapheme sequence against an identical sequence in the name. A match is a positive identification and the filter routines stop. Elimination rules also match a grapheme sequence against an identical grapheme sequence in the name. A match eliminates that language group from consideration. There are a number of different ways these rules could be applied. One of the more efficient methods is to create a hash table of grapheme strings and search for substrings for identification and elimination at the same time. Whichever way a compiler for these rules is written, it is clear that the routines stop after a positive identification occurs. The benefit of using filter rules prior to the trigram analysis (below) is one of speed.

One minor problem that had to be examined was the fact that many names have been anglicized from their original form, resulting in varied and disparate pronuncia-

7 The average length of an English word is 3.98 letters when the words are weighted by frequency of appearance (Daly 1987) clearly due to the shorter length of commonly occurring forms such as function words. While no similar statistics have been compiled for names, it is doubtful whether the discrepancy in length between weighted and unweighted would be as large.

8 The ISO-Latin character set (or an equivalent) could also be utilized in situations where proper names can be written with special symbols (e.g., ü, ø, é and others), since these orthographic symbols could be used to eliminate or positively identify language groups.

tions (not to mention some rather strange spellings, including graphemes that do not exist in the source language). For example, a Polish name such as ALEXANDROWICZ contains the grapheme X, although X does not occur in Polish (i.e. KS $\rightarrow$ X). The orthographic sequence SCI (= [š]) in Italian is occasionally anglicized as SH even through the sequence SH does not occur in the language. Therefore, the elimination rules have to be carefully tailored to take such phenomena into consideration. Sequences that positively identify a language must also be carefully screened for the same reason. Names like O'SHINSKI are not uncommon.[9] In this case, whether the name is considered Irish or Polish may not matter in terms of the phonemic output, but there are cases where it would make enough of a difference to cause intelligibility problems in the final output.

3.2 Trigram Analysis[10]

The job of the filter is to positively identify a language or to effectively eliminate one or more groups within the set of possible language groups when positive identification is not possible. Elimination obviously reduces the complexity of the task of the remaining analysis of the input name. Assuming that no language group is positively identified as the language group of origin by the filter, some further analysis is needed. This further analysis is performed by a trigram analyzer, which receives the input name string and a vector of uneliminated language groups. The trigram analyzer parses the string into trigrams. If word boundary symbols are included as part of the string, then the number of trigrams in the string will always be equal to the number of elements (graphemes). Thus, the name SMITH $\Longrightarrow$ #SMITH# will contain five trigrams: #SM, SMI, MIT, ITH, and TH#.

A trigram table is a four-dimensional array of trigram elements and language group. This array contains numbers that are probabilities (generated from a large reference corpus of names labeled as belonging to a particular language group) that the trigram is a member of that language group. Probabilities are taken only to four decimal places, although there is no empirical reason for this.

3.2.1 Creation of Trigram Databases. The creation of a trigram database would be an extensive and time-consuming task if it were to be done manually. Nevertheless, it was initially necessary to hand-label a large list of names with language group tags associated with each name. Fortunately, this was expedited with country-specific personnel lists from a large company.[11] Once these lists were completed, computational analysis was performed on the list, decomposing each name into grapheme sequences of varying lengths, including trigrams, and searching for recurring morphological elements as well. This analysis, in turn, created a set of tables (language-specific n-grams, trigrams, etc.), which was then used for further analysis. The language identifier itself can be utilized as a tool to pre-filter a new database in order to refine the probability table. This is illustrated in Figure 1. The name, language group tag, and statistics from the language identifier are received as input. This analysis block takes this information and outputs the name and language group tag to a master language file and produces rules to a filter rule-set. In this way, the database of the system is expanded as new

9 Murray Spiegel (personal communication) has pointed out that there are 79 households in the U.S. that have this name.

10 For our purposes here, trigram will be used synonymously with the term trigraph. Trigram analysis is by no means new and has been discussed often in the literature (e.g., Church 1986).

11 Although these had to be carefully verified because of the increasing numbers of expatriates living and working in any given country.

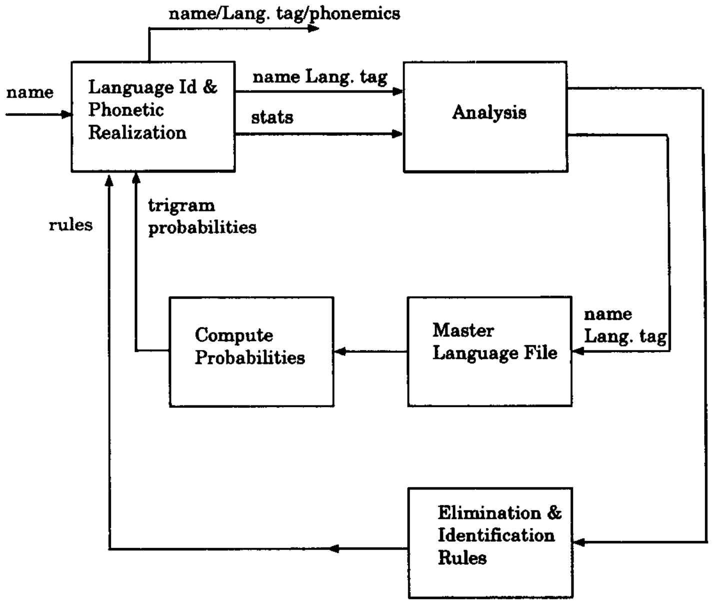

Figure 1

input names are processed so that new names can be more accurately recognized. The filter rule store provides the filter rules to the filter module for identification or elimination.

3.2.2 Trigram Array and Statistical Analysis. The final trigram table itself then has four dimensions: one for each grapheme of the trigram and one for the language group. The trigram probabilities are sent to the language group blocks, phonetic realization block, and to the trigram analysis, which produces a vector of probabilities that the grapheme string belongs to the various language groups.

The master file contains all grapheme strings and their language group tag. The trigram probabilities are arranged in a data structure designed for ease of searching a given input trigram. For example, if we use an n-deep three-dimensional matrix where n is the number of language groups, then trigram probabilities can be computed from the master file using the following algorithm:

```
compute total number of occurrences of each trigram for
all language groups L (1-n)
for all grapheme strings S in L
          for all trigrams T in S
               if (count [T][L] = 0)
                    uniq [L] + = 1
               count [T][L] + = 1

for all possible trigrams T in master
     sum = 0
     for all language groups L
          sum + = count [T][L]/uniq[L]
     for all language groups L
          if sum > 0, prob[T][L]=count [T][L]/uniq[L]/sum
          else prob[T][L]=0.0;
```

Table 1
Sample matrix of probabilities.

Trigram	L_i	L_j	...	L_n
#VI	.0679	.4659	...	.2093
VIT	.0263	.4145	...	.0000
ITA	.0490	.7851	...	.0564
TAL	.1013	.4422	...	.2384
ALE	.0867	.2602	...	.2892
LE#	.1884	.3181	...	.0688
AV.	.0866	.4477	...	.1437

In any case, the result of the trigram analysis is a vector of probabilities for a given trigraph over the number of language groups. Table 1 shows an example of what the probability matrix would look like for the name string VITALE.

In the matrix shown in Table 1, L is a language group, and n is the number of language groups not eliminated by the filter rules. The probability that the grapheme string #VITALE# belongs to a particular language group is actually produced as a vector of probabilities from the total probability line. In this case, the trigram #VI has a probability of .0679 of being from language group L_i .4659 of being from the language group L_j and only .2093 of being from the language group L_n. The average of the probability table entries identifies L_j as being the most probable language group. In this case, L_j was Italian.

The probability of a trigram being a member of a particular language group can be derived by a number of different methods. For example, one could use a standard Bayesian formula that would derive the probability of a language group, given a trigraph T, as $P(L_i|T)$ where

$$P(L_i)|T) = \frac{P(T|L_i)P(L_i)}{\sum_k P(T|L_k)P(L_k)}$$

Furthermore, where x is the number of times the token T occurred in the language group L_i and y is the number of uniquely occurring tokens in the language group L_i, always, where n is the number of language groups (nonoverlapping). Therefore,

$$P(L_i|T) = \frac{P(T|L_i)}{\sum_{k=1} \frac{P(P|L_k)}{n}} = \frac{P(T|L_i)}{\sum_{k=1} P(T|L_k)}$$

While this is not the most mathematically optimal or elegant method (since averaging tends to favor a noneven distribution of trigram probabilities) and is certainly a simplistic method of performing such calculations, it works reasonably well and is computationally inexpensive. It should be noted, however, that multiplying probabilities, calculating and adding log probabilities, or even averaging the two highest probabilities, may all work, but each of these approaches assumes that trigrams are independent of one another. It is beyond the scope of this paper to discuss the elegance of one mathematical solution over another but it would be interesting to examine other options, such as higher order conditional probabilities, e.g.,

$$P(L_i|T_1, T_2, T_3) = \frac{P(T_1|T_2, T_3, L_i)P(T_2|T_3, L_i)P(T_3|L_i)P(L_i)}{P(T_1, T_2, T_3)}$$

although these would clearly be computationally quite expensive.

Table 2
Name pronunciation statistics.

Name	Identified Language	Highest Probability
Partington	English	.4527
Bischeltsrieder	German	1.000
Villalobos	Spanish	.4377
Kuchenreuther	German	.6973
O'Banion	Irish	1.000
Zecchitella	Italian	1.000
Pederson	English	.3258
Hashiguchi	Japanese	1.000
Machiorlatti	Italian	1.000
Andruszkiewicz	Polish	1.000
Fujishima	Japanese	1.000
Macutkiewicz	Polish	.6153
Fauquembergue	French	.4619
Zwischenberger	German	1.000
Youngblood	English	.8685
Laracuente	Italian	.2675
Laframboise	French	.3778
McAllister	Irish	1.000
Abbruzzese	Italian	.5113
Rodriguez	Spanish	.6262
Yanagisako	Japanese	.7074
Migneault	French	1.000
Znamierowski	Polish	1.000
Shaughnessy	Irish	.6239

Table 2 is an example of the output of the language group identification module. The table consists of twenty-four proper names randomly but equally selected from the eight separate language groups.[12] Twenty-three out of twenty-four were correctly identified. The only error is on the name LARACUENTE, which is the lowest score and is identified as Italian instead of Spanish.

Note also that .2675 is the lowest score in the list. In practice, this would not have presented a problem, since the letter-to-sound rules for language groups such as Italian and Spanish are very similar (e.g., the stress pattern would be penultimate, etc.) and thus the phonetic realization would be almost identical. When pronunciation is included in the evaluation, the scores would be slightly higher in certain cases, since an incorrect identification does not always result in an incorrect pronunciation.

3.3 Thresholding

Since the output of the etymology analyzer is a vector of probabilities and only the highest score is chosen (i.e., a best guess), a number of different situations can arise regarding the total spread among the numbers, the difference in spread between any two numbers, or the spread between some number and 0 (i.e. an absolute comparison). For this reason, and to make use of this information, thresholding has been applied.

Essentially, thresholding allows for analysis to be made over the vector of probabilities such that statistical information can be used to help determine the confidence level for the language group with the highest score (i.e., the best guess). Two types of threshold criteria have been applied: absolute and relative.

12 Randomly selected from names over 7 graphemes in length to increase complexity somewhat.

3.3.1 Absolute Thresholding. Absolute thresholding can apply when the highest probability determined by the trigram analyzer is less than a predetermined threshold that is variable or can be set programmatically. This would mean that the trigram analyzer could not determine, from among the language groups, a single language group with a specified degree of confidence. For example, if empirical evidence (i.e., over a given corpus) suggests that $P < n$ (where P is the highest probability and n is some number predetermined to be too low for an adequate confidence level), then some other action should be taken. n should be set by analysis of data. While this "other action" is variable, one approach would be to choose a default language that may or may not be the same as the language group identified by the highest probability. Evidence suggests that typically it is not.

As an example, if the absolute threshold were set at $P < .1000$ and the highest score were .0882 for some language L_i, then the default language is chosen whether or not this is the same as L_i. There may be circumstances where the accuracy might be able to be tuned by adjusting the absolute threshold.[13] However, this parameter should be construed more as a partial filter which, if set to some reasonable value, will filter out only scores showing a very low confidence level, and thus it would rarely affect the result.

3.3.2 Relative Thresholding. Another type of thresholding scheme that was implemented is a relative thresholding. In this case, Δ spans a number of probabilities provided that the distance between the highest score and the default language is $< n$. Therefore, if P_j was the probability assigned to the default language group, no matter where this occurred relative to the best guess P_i, if $\Delta(P_i, P_j) < n$, the default language is chosen. (Typically, n is a smaller number than it was for absolute thresholding.) This is, of course, empirical and should be judged according to an analysis of the database used. It is our impression that if the default language group falls within the Δ, the algorithm should force a choice of the default language.

It should be noted, however, that there are other ways in which relative thresholding could have been implemented, e.g., when the distance in probabilities between the language group identified as having the highest probability and that identified as having the second highest probability is $< n$, where again n is some number determined by analysis of the data. Thus where P_i is the highest probability and P_j the second highest, then, if $\Delta(P_i, P_j), < n$, the default language is chosen. The problem with this approach is that it would result in two close scores (i.e. between similar languages), forcing a default to a third and possibly structurally dissimilar language. For example, a name for which the scores for Italian or Spanish fell within the Δ might then be forced into the default language, say English. This is clearly not optimal for a generic use of the algorithm, although it might be useful under certain application-specific circumstances.

3.4 Default Language

To solve practical application problems of name pronunciation, it was necessary to define a default language group. The concept of using a default language proves to be useful for several reasons: (1) it is consistent with the philosophy that where

13 Such *a priori* probabilities for thresholding can either be adjusted once early in the application or may even be biased by a running average based on the population that used the application within some particular time frame.

mistakes are made, they will reflect human errors in pronunciation;[14] (2) the software underlying this algorithm is designed to be used in speech communities anywhere in the world; and (3) the default language could be adjusted for communities in the U.S. or elsewhere where one language group predominates.

If P_i or $\Delta(P_i, P_j)$ falls within some range, it signals, for whatever reason, a low confidence level. Humans, when faced with a decision in these circumstances, often opt for the "familiar," in this case, some predefined default pronunciation. This would almost always be the language of the speech community in which the application is running. In other words, if the confidence level is measured as low via some thresholding mechanism, then a conservative approach would be to default to some "safe" language group, even if this might result in an error in the correct pronunciation (see fn. 14).

Secondly, whether an application is running in Berlin, Paris, Dublin, or Milan, the default language setting could be changed to reflect the predominant language group. In Germany, for example, in cases where threshold confidence level scores are too low for the language group identifier, the default (presumably German) would reflect a reasonable guess.

The default language parameter could also be used in other cases where the predominant linguistic base of an area is known to be different from that of the wider speech community. In telecommunications applications, for example, telephone number prefixes are unambiguous indicators of geographical areas, some of which are relatively homogeneous in ethnic makeup. In cases like these, an application might make use of an automatic default language change for calls pertaining to these areas. Thus, in a Hispanic neighborhood, the default would be to Spanish. This could also be used for homographic first and last names in an elaboration of this system such that ambiguities like JULIO, JESUS, and the like could be resolved. The default language setting is certainly the most important of any of the settable parameters, since it determines the base language that is used in all cases of low confidence.

4. Letter-to-Sound Rules

Much of the discussion of this paper has been devoted to an explanation of the identification of the etymology of the name. While this is certainly the more difficult problem to solve, there is a great deal more that needs to be done to arrive at some reasonable approximation of a phonetic realization. The identifier merely takes an orthographic sequence and adds a tag that marks it as a member of a particular language group. The output remains a graphemic sequence. It is the tag, however, that forces the name through one of a special set of letter-to-sound rules optimized for the languages of that particular group. Therefore, the sole difference between a name run through the identifier and a word from generic English text is that the name is tagged as a special case and undergoes different letter-to-sound rules.

4.1 Optimization of Rule Sets

The letter-to-sound module is a knowledge-rich complex subsystem that takes a grapheme input and converts it into its appropriate phonemic equivalent. In normal letter-to-sound systems that apply to generic text (see Klatt 1987 p. 767ff), an orthographic sequence is, according to some rule set, converted into symbols that typically correspond

14 As Spiegel (1985) has pointed out, "the guiding principal behind all work should be that the synthesis rules should make errors that are similar to human mistakes."

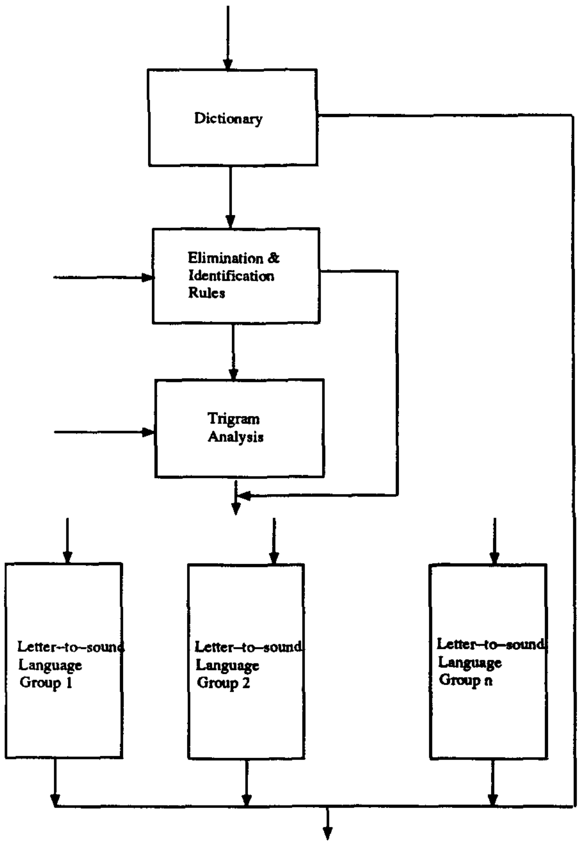
Dictionary
Elimination & Identification Rules
Trigram Analysis
Letter-to-sound Language Group 1
Letter-to-sound Language Group 2
Letter-to-sound Language Group n

Figure 2

to segmental phonemes and stress patterns for that form. In this case, however, the task becomes much more complex because of the many language groups involved (Section 1.1).

When the tag (above) is attached to some name thereby identifying it as belonging to some language group L_i, the orthographic sequence is funneled through a special set of letter-to-sound rules for L_i, and similarly for $L_j, \ldots, L_n$. Figure 2 is a block diagram of the entire flow of the procedure with the letter-to-sound modules (slightly oversimplified) occurring after identification has been completed.

While the concept of separate rule sets is a valid one, in practice this would be an unnecessarily complex system since it is obvious that there would be a great deal of redundancy and overlapping of rules from one rule set to the other. For example, the simple rule (K $\Longrightarrow$ /k/ might be valid for many of the language groups. This would fail to capture the generalization that this rule can be shared by a subset of the total number of language groups, would therefore waste computing resources, and consequently is architecturally suboptimal.

As a result of this need for rule-sharing, a slightly different strategy can be devised. Using a complex rule-set, a single rule can contain from 1 to n language identification tags, n again being equal to the total number of language groups. In practice, there are rarely more than four tags on a single rule. However, this does reduce the computational complexity and redundancy of having separate rule sets.

5. Some Further Issues

5.1 First Names

Historically, surnames in many parts of the world are simply extensions of the first name to distinguish different owners of the same first name. As a consequence of this, first names have a higher frequency of occurrence than surnames since the list of first names is a smaller set for most languages.[15] Therefore, many of these are appropriately included in the lexicon or dictionary. In addition, the same first name may span a wide variety of languages. Common European first names are found in disparate regions of the world, due to extralinguistic factors such as the geography of former colonial powers. Put another way, more people in anglophone countries have the first name JOHN than have the last name SMITH.[16] For this reason, the most common first names in each language group are included in the dictionary. However, in the event that there is no dictionary match on first names, they should still be processed, like the surname, by the language identification module (below).

Processing the first name can be done independently or, in an elaboration of the algorithm, may be done in conjunction with the surname. For example, if the name YUKIO KOBAYASHI were processed and one name was found by a dictionary lookup, that identification could assist in the identification of the other. If the probability of both names were low but the best guess showed the same language group identified for both, this could also be used as a factor in the determination of the correct etymology. In this scheme, caution must be used especially with female names in many countries with a heterogeneous ethnic makeup (e.g. the United States) since a surname (taken from a husband) and female first name will often be from different language groups. This, of course, can be overcome by marking such names in the lexicon as female. However, it was found that such additional loading is not necessary since the etymology of names can be accurately ascertained without utilizing information outside of the name itself.

5.2 Hyphenated Surnames

Hyphenated names would be processed as if they were two separate names. Again, this is because of the potential confusion that could occur where part of the hyphenated name is a married name and the other a birth name. Thus, surnames such as ARROYO-PETERSEN, MAHONEY-RIZZO, KILBURY-MEISNER, and the like should be treated as if the hyphen were a language separator. Furthermore, in such combinations, the sequence is often unpredictable. Some cultures adopt the convention (for women) of birth name-married name (e.g., English, Polish), while others have the opposite order of married name-birth name (e.g., German).[17]

5.3 Homographs

A classic problem that faces letter-to-sound algorithms is the existence of homographs. These are words that are spelled the same but have two different pronunciations, usually signaling a difference in form class. English contains, in its general vocabulary,

15 Certain languages, however, have a more open-ended system than those of the familiar Indo-European languages. In Chinese languages, for example, individuals have a wide variety of names which derive from the general lexicon. A female name might be **WANG MEI HWA** where **WANG** is the surname, **MEI** 'beautiful' and **HWA** 'flower'. Her sister might be **WANG MEI YEH** 'beautiful leaves.'

16 Liu and Haas (1988) point out that the name **SMITH** occurs 676,080 times for a database of 75 million households in the U.S. representing 1% of the total.

17 In certain Slavic languages, names function as adjectives and are marked for gender. In Polish, for example, the common name **KOWALSKA** is the female counterpart of the male name **KOWALSKI**.

over 300 of these, some of which are high frequency forms. Examples of this are DELIBERATE, ARITHMETIC, REFUSE, PRODUCE, COORDINATE, SOW, BASS, and many others. Modern-day text-to-speech systems do not yet process these automatically. An even more difficult problem faces a proper name pronunciation algorithm. Whereas one could, in principle, devise a method for generic text using form class or part of speech to distinguish pairs such as those just mentioned, it is not always possible to predict which variant will occur in a proper name.[18]

5.4 Free Variation and Cross-Language Variation

5.4.1 Homographic Variation and Cross-Language Variation. Different pronunciations may occur in either free variation or in cross-language variation, with the two occasionally overlapping. Free variation occurs when the same last name occurs with two or more different pronunciations but these are from the same language group. One person with the name BERNSTEIN may pronounce it [*bŕ̩nsta^{y}n*], whereas a second person may pronounce it [*bŕ̩nsti^{y}n*]. This is true free variation where typically one pronunciation represents something closer to the source language. In a slightly different type of variation, one name may indeed be from a different language group or else the alternation reflects a radical anglicization of the name, e.g., PACE may be pronounced either [*pés*] or [*páči*]. Stress patterns also vary greatly depending upon the degree of assimilation into English. For example, Slavic names show a great deal of variation from penultimate stress (which is the canonical stress pattern) to antepenultimate (the assimilated pattern). E.g., ANDRUSZKIEWICZ could be [andrušk^{y}évič] or [andrúšk^{y}evič]. Often, cues such as orthography may give a hint as to the degree of assimilation. If the native Slavic phoneme [v] is written orthographically as **W**, for example, it has a higher chance of retaining the source (i.e. penultimate) stress pattern than if the same phoneme were written orthographically as **V**.

Cross-language variation causes greater problems in this regard, since names are listed in the dictionary in only one way. However, this algorithm is not language-specific but can be used anywhere in the world. While the entry could be modified, no procedure that would allow for different lexical entries for the same spelling has been implemented. In any case, names like ROBERT could be [*rábr̩t*] or [*robér*] depending upon whether the name is English or French. Similarly, names like JULIO, PETER, WALTER, GILES, BERNARD, GUY, RICHARD, JAN, CLAUDE, and hundreds of others have different segmental forms or stress patterns depending upon where they originate from and whether the name has been assimilated into English. One way in which this could be handled is by marking these in a dictionary and then using the "loading" strategy of last name etymology as discussed above (Section 5.1). Another method is simply to adopt one variant as the default with the others also listed as homographs and marked for language type with the corresponding phonetic equivalents. Since the default language is a settable parameter (Section 3.4), multiple phonetic entries could, in principle, be used.

5.5 Loanword Assimilation and Segmental Modification

It is obvious that the segmental phonetic output of letter-to-sound rules is restricted to the inventory of allophones of the synthesizer. That is, an English-based speech synthesizer should not be expected to make a French uvular [R] or a German velar fricative. Moreover, it would be counterproductive to even attempt to approximate

18 Distinguishing non-name homographs automatically could, in principle, be done with a front-end parser, which would provide the syntactic information necessary to choose one pronunciation or the other.

these sounds, since the listener would not be expecting this and intelligibility of the name would certainly decrease. For this same reason, even the segmental phonemes of the source language can distract the listener's attention and reduce intelligibility. For example, while the name CARBONE must have penultimate stress to be considered correct, it would be inappropriate to phonemicize the final orthographic vowel as a mid front /e/, as would be the case in the target language (i.e., Italian). This is because the assimilation of this and other similar names into English has raised the vowel phonemically to /i/ (and diphthongized it phonetically to [i^{y}]. Thus, while CARBONE would be /karbóne/ in Italian, it would be /karbóni/ in anglophone countries such as the United States.

Furthermore, the different degrees of assimilation can be formally described in an elaboration of the algorithm presented here. For example, assuming allophones as produced by a speech synthesizer are optimized to American English, one could apply a number of rules to the name CARBONE as follows:

1. Orthography → **(CARBONE)**
2. Phonemicization → /karbóne/
3. Assimilation Rule 1 → /karbóni/
4. Assimilation Rule 2 (optional) → /karbón/

Thus, we can place different degrees of assimilation on the target language (i.e. English) using formal (ordered) rules. This would allow a generic synthesizer's letter to sound (and allophonic) rules the ability to change degree of assimilation in much the same way that we will eventually be able to specify shifts in style and register within text-to-speech systems.

6. Testing and Evaluation

6.1 Performance

The performance goal of the software developed around this algorithm was real-time processing. We benchmarked the performance on a Digital Equipment Corporation Vax 8800 running VMS V5.1. A total of 34,337 names were processed in 25 minutes and 27 seconds, or equivalently 22.65 names per second. After some code optimization and streamlining of the filter rules, we later ran similar tests using the same databases on an 33 MHz PC running MS-DOS V5.0. While these tests were run on the identification portion only, we were able to process several thousand names per second. Large commercial applications will have similar compute power, and thus real-time processing is not a problem. It should be noted that many applications do not require real-time processing since the processed name and address can be simply stored in a separate field in the database. The routines can thus be used to create a database of phonemicized names by preprocessing the name, storing the phonemic equivalent of the name in some field, and sending that field to the synthesizer at some later time.

6.2 Pronunciation Accuracy

A number of different tests were conducted for accuracy of pronunciation. Accuracy here was measured in terms of the level of segmental and suprasegmental (i.e., stress placement) output determined by a linguist to be reasonable behavior. A more elaborate (and possibly more practical) criterion for accuracy might include the transcription (by a linguist) of a number of pronunciation tokens provided by nonlinguists.

Table 3
Names vs. generic lexical items.

Lexical type	% Error before	% Error after	Dictionary
All function words	0	0	+
2000 common surnames	31.9	0	+
Complex poly test	5.1	5.1	–
828 Single L-group surnames	92	6.3	–

Our reasoning was that the output should minimally model human behavior. However, because the algorithm contains more linguistic information than is known by the average person, the software has the potential to be more accurate than a person (i.e., make fewer gross pronunciation errors). Testing of human vs. computer pronunciation of names from a test database is being conducted independently at the present time within the artificial intelligence community (Golding and Rosenbloom 1991) as well as within the telephone industry.

One of the problems we faced is a definition of what constitutes correctness. Very often, more than one pronunciation is acceptable and many readers of this paper have had their own names pronounced differently by other individuals. Even professional linguists faced with names such as MOUDRY, FUCHS, SOUTO, D'ANGELO, BADKE, DUJMUCH, SMYTHE, and others cannot say definitively whether one pronunciation is correct or not (Hochberg et al. 1990). For the purposes of our evaluation, in cases like these, a pronunciation was accepted if linguists felt that the segmental phonemic output and stress placement were reasonable. Again, another and possibly more realistic approach might be to phonemicize a set of names from the pronunciation of a group of individuals who are not owners of the name. These pronunciations could then be phonemicized by a linguist and correctness could then be evaluated by a simple matching of the majority pronunciation. In any case, both [fyuks] and [fuč] were considered correct for FUCHS but [fúčiz] and [f^ks] were not; [sma^{y}Θ] and [smIΘ] for SMYTHE but not [smÍΘi]; [di^{y}ǽnǰelo] and [dǽnǰelo] for D'ANGELO but not [dænǰélo] and so on. Similarly, because of homographic variation (Section 5.4.1) and the attempt to make errors replicate what humans might say, we would accept certain pronunciations for names that we knew came from two very different sources as long as one were reasonable. For example, [pés] would be an acceptable pronunciation for PACE even if the first name were Antonio. In fact, often one cannot say definitively that one pronunciation or the other is the one used without asking the person who owns the name. Again, with the loading strategy factoring in first name (above), one might increase the probability of a reasonable pronunciation.

Testing was done with several databases that were not used to compile the trigrams. Some degradation was expected when using a new (test) database. However, as shown in Figure 1, after testing, a new database could be merged with the reference database, and new and more complete trigram statistics calculated.

Table 3 shows the error rate with and without a dictionary over different subsets of a corpus. The dictionary covered all functors and the 2000 most common surnames. The complex polysyllabic test (see fn. 3) is simply a benchmark for the generic letter to sound rules without use of a dictionary. The last line of the table suggests the improvement possible in name pronunciation (in this case, Japanese names were used). Note the degradation in performance (without the name pronunciation software) from common names to Japanese names.

Table 4
Database tests — no dictionary.

Database	% Error no ident.	% Error with ident.
Reference – set 1	18%	8%
Reference – set 2	24%	8%
Test – hardcopy	32%	15%
Test – softcopy	22%	12%

Table 5
First, last, and street names.

Word type	No dictionary	Dictionary
Last names	12%	7%
Street names	24%	7%
First names	23%	1%

In a second test, we had a subject randomly choose two sets of 100 names from our reference database and two further sets of 100 names from each of two telephone books. One telephone book was hardcopy from a large region in the East and the second was an on-line directory from a large region in the mid-West. In the case of the hardcopy listings, the data were put on line to be analyzed. The softcopy was edited to remove unwanted material.[19] We included the softcopy database to minimize any bias, conscious or otherwise, that the subject may have had and these names were chosen with a simple program that pulled out the required number of names from the name field. In spite of the fact that trigrams tend to be repeated over a database (above), we nevertheless expected some degradation going to new test lists, as the data in Table 4 illustrate. The error rate was calculated with and without the identification algorithm on four databases of 100 names each using *no dictionary lookup*.

Because of the high functional load of dictionary entries (see Section 2), scores were expected to be considerably higher when the dictionary lookup module was included. We tested this hypothesis and found that when the dictionary was included in the softcopy test-database analysis (above), the error rate was reduced from 12% to 7%. Other tests also indicated that the use of a dictionary cuts the error rate approximately in half.

Due to the fact that many applications written around this software will require the accurate pronunciations of first name and street name as well as last name, we decided to examine the accuracy for each of these categories as well. The anticipation was that the accuracy rate for first names (using a dictionary) would be slightly higher than that of last names and that the accuracy rate for street names would be slightly lower. This is because of the higher frequency of occurrence of first names (above) as well as the fact that the pronunciation of street names tends to be extremely variable and, like place names, has been observed to vary between local and non-local population groups. Table 5 indicates the error rates of the first name and street name tests compared with the last name tests mentioned above, run over the test database with and without a dictionary.

19 Telephone listings typically contain a variety of information including, *inter alia*, the telephone number and street number, demarcation of upper case (e.g., MC*ADOO) special symbols for unlisted numbers and so on.

Note that both street names and first names have much lower accuracy than last names without the use of a dictionary. First names, like functors and irregular verb forms, exhibit unusual behavior in terms of the canonical segmental phonology of the language, e.g., THOMAS, where the first segment is /t/ rather than /Θ/, MICHAEL, where orthographic CH, is /k/ and not /č/, and so on. In the case of street names, many are the same as place names (OTTAWA BLVD), first names (JOYCE ST.), or last names (EISENHOWER AVE). In any event, note that the use of the dictionary with these name fields is crucial to the success of the algorithm, much more so than in the case of surnames. In fact, the non–Anglo-Saxon surnames (LUELLA, LEONARDO, etc.) are handled quite adequately without use of a dictionary lookup. In the case of first names, the error rate is extremely low since the vast majority of these would be found in the dictionary.

Naturally, the final and most crucial test of accuracy is the overall intelligibility of the name, that is, whether an individual on the receiving end of a telephone line (with its reduced bandwidth) can hear, repeat, and correctly transcribe (in normal orthography) a person's name and address. These tests and others remain for future research. We set out simply to attempt to improve pronunciation accuracy of proper names by creating a more intelligent front-end processor and a more complex letter-to-sound rule set that would take into account the variability of the text to be processed. Tests indicate that an algorithm can be successfully implemented to significantly increase accuracy of name pronunciation. This helps make possible applications in which proper names are output intelligibly using a speech synthesizer, as well as text-processing functions such as the construction of a name dictionary for automatic speech recognition. The algorithm has, in fact, been implemented for speech synthesis and is currently being used in a commercially available product within the telecommunications industry.

Acknowledgments
I would like to thank Patrick Gili, Dave Conroy, Bob Curtis, and Tom Levergood of Digital Equipment Corporation for their many ideas and their programming expertise, which allowed for the development of tools that helped determine the initial feasibility of the algorithm and later helped improve accuracy of the identification pass; and Murray Spiegel of Bell Communications Research for numerous discussions and a profitable exchange of ideas. I also thank Janine Haley of CONTEL and Bob Weide of Carnegie Mellon University who shared databases which allowed me to test and refine the identification portion of the algorithm.

References

Church, K. W. (1986). "Stress assignment in letter to sound rules for speech synthesis." In *Proceedings, IEEE International Conference on Acoustics Speech and Signal Processing 4*, pp. 2423–2426.

Cole, R. A.; Inouye, J. W. T.; Muthasamy, Y. K.; and Gopalakrishnan, M. (1989). "Language identification with neural networks: A feasibility study." In *Proceedings, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing.*

Daly, N. A. (1987). *Recognition of Words from Their Spellings: Integration of Multiple Knowledge Sources.* M.Sc. Thesis, Massachusetts Institute of Technology.

Divay, Michel (1990). "A written text processing expert system for text to phoneme conversion." In *Proceedings, International Conference on Spoken Language Processing (ICSLP).* Kobe, Japan. pp. 853–856.

Divay, Michel (1984). "De l'écrit vers l'oral ou contribution a l'étude des traitements des textes écrits en vue de leur prononciation sur synthétiseur de parole." Thèse d'Etat, Université de Rennes.

Golding, A. R.; and Rosenbloom, P. S. (1991). "A comparison of anapron with seven other name pronunciation systems." Knowledge Systems Laboratory Report No. KSL 91-26. Stanford University.

Golding, A. R.; and Rosenbloom, P. S. (1989). "Combining analytical and similarity-based CBR." In *Proceedings, 2nd Case-Based Reasoning Workshop.* Pensacola, FL.

Hochberg, J.; Mniszewski, S. M.; Calleja, T.; and Papcun, G. J. (1990). "What's in a Name?: Last Names as a Computational Problem." Unpublished paper, Los Alamos National Laboratory, Los Alamos, NM.

Hunnicutt, S. (1976). "Phonological rules for a text-to-speech system." *American Journal of Computational Linguistics Microfiche 57.*

Karhan, C.; Hardzinski, M.; Holinka, V.; and Viets, M. (1986). "Text-to-speech synthesis for pronouncing names and addresses in a telecommunications service: designing the user interface." In *Proceedings, Voice I/O Systems Applications Conference '85.* pp. 51–57.

Klatt, D. H. (1987). "Review of text-to-speech conversion for English." *Journal of the Acoustical Society of America* 82/3. pp. 737–793.

Liu, F. C.; and Haas, L. J. (1988). "Synthetic speech technology for enhancement of voice-store-and-forward systems." In *Proceedings, American Voice Input/Output Society.*

Macchi, M.; and Spiegel, M. (1990). "Using a demisyllable inventory to synthesize names." In *Proceedings, Speech Tech '90.* pp. 208–212.

Smith, E. C. (1969). *American Surnames.* Rednor, PA: Chilton.

Spiegel, M. (1990). "Speech synthesis for network applications." In *Proceedings, Speech Tech '90.* pp. 347–355.

Spiegel, M. (1985). "Pronouncing surnames automatically." In *Proceedings, American Voice Input/Output Society.* pp. 109–132.

Spiegel, M.; and Macchi, M. (1990). "Synthesis of names by a demisyllable-based speech synthesizer." *Journal of the American Voice Input/Output Society* 7, pp. 1–10.

Venezky, R. L. (1970). *The Structure of English Orthography.* The Hague: Mouton.

Vitale, A. J. (1989a). "Application-driven technology: automated customer name and address." In *Proceedings, American Voice Input/Output Society.* Newport Beach, CA.

Vitale, A. J. (1989b). "The automation of name and address output as a utility for the telecommunications industry." In *Proceedings, National Communications Forum* 43(**2**), pp. 1109–1113.

Vitale, A. J. (1987). "Engineering speech systems to meet market needs." In *Proceedings, Speech Tech '87.* pp. 149–151.