MULTILINGUAL SYLLABIFICATION USING WEIGHTED FINITE-STATE TRANSDUCERS

George Anton Kiraz and Bernd Möbius

Bell Labs – Lucent Technologies, Murray Hill, NJ 07974, USA

Just as vowels with consonants are the matter of syllables, so also syllables are the matter for the construction of nouns and verbs, and of the elements which are made out of them.

Antony the Rhetorician of Tagrit († *c.* 445?)
Knowledge of Rhetoric, Book Five, Canon One

ABSTRACT

This paper describes an approach to syllabification that has been incorporated into the English and German text-to-speech systems at Bell Labs. Implemented as a weighted finite-state transducer, the syllabifier is easily integrated – via mathematical composition – into the finite-state based text analysis component of the text-to-speech system. The weights are based on frequencies of onset, nucleus and coda types obtained from training data. While the training data is language-dependent, the formal approach is multilingual.

1. INTRODUCTION

In recent years finite-state transducers (FSTs) have become increasingly popular as a flexible and mathematically elegant computational model for the conversion and mapping between symbol strings, most notably in the domains of phonology and morphology. In this paper we illustrate the implementation in FST technology of a syllabifier in the framework of the Bell Labs multilingual text-to-speech (TTS) system [7].

Syllabification is an important component of any TTS system. In many languages the pronunciation of phonemes is a function of their location in the syllable relative to the syllable boundaries. Location in the syllable also has a strong effect on the duration of the phone, and is therefore a crucial piece of information for any model of segmental duration [9].

Syllabification can be achieved by writing a declarative grammar of possible locations of syllable boundaries in polysyllabic words. An extremely simplistic, constraint-based model of syllabification might state that each word in the utterance consists of one or more syllables of the structure C*VC*, i.e., of an obligatory syllable nucleus (V) optionally preceded or followed, or both, by any number of consonants (C). By assigning a higher cost to the last consonant of each syllable, we can enforce the syllable boundary to be placed as early as possible, thereby implementing the well-known maximal onset principle. The grammar would finally terminate every non-final syllable in the word by a syllable boundary symbol [8, p. 49f.].

A more realistic model of syllabification is presented in the remainder of this paper. In keeping with the multilingual design of the Bell Labs TTS system, our approach is applicable to any language. It relies, however, on the phonotactics and syllable structure of each particular language. For illustration purposes we concentrate here on English and German. Section 2 discusses aspects of syllable structure and phonotactics in these two languages.

The text analysis component of our TTS system consists of a multitude of modules which operate on different levels of linguistic description and analysis. This heterogeneous system has been implemented in a unified framework, viz. weighted finite-state transducer (WFST) technology. Despite the differences in the formalisms applied to different sub-tasks of text analysis, the linguistic descriptions can all be compiled into WFSTs.

Section 3 presents a finite-state model for syllabification. The syllabifier is implemented as a weighted finite-state transducer. The transducer is constructed by obtaining syllables as well as their structure and frequencies from a training database. Observed frequencies of onset, nucleus and coda types are converted into weights, which are then associated with the pertinent transitions between states in the transducer.

Illustrations of the properties and performance of the syllabifier are given in Section 4, with concluding remarks in Section 5.

2. SYLLABLE STRUCTURE

The phonotactics of English and German allow complex consonant clusters in both the onset and the coda of syllables. The maximum number of consonants in the onset is 3 in both languages. In German codas, clusters of up to 5 (or 6, if contractions such as *du schrumpfst's* [du: ʃrʊmpfsts] "you shrink it" are considered, too) consonants can be observed, whereas English allows up to four coda consonants. Thus, the maximum number of consecutive consonants across syllable boundaries is 9 in German and 7 in English.

Certain restrictions exist as to which consonants, or classes of consonants (see Table 1), can occur in which position within the onset or coda of a syllable. For instance, in both languages there are only a few possible onset clusters with three consonants, and no phones other than obstruents can occur *before* an obstruent in the onset. In codas, only combinations and alternations of voiceless dental stops and fricatives are possible in positions 2 through 4 in English, and 3 through 5 (or 6) in German, and after the first obstruent no phones other than obstruents can occur in the coda. Examples for the longest consonant clusters in English and German onsets and codas are given in Tables 2 and 3, respectively.

Sonorants (nasals, liquids, and glides) can only occur adjacent to the syllable nucleus. This pattern is sometimes referred to as the sonority principle, which ranks phone classes according to their natural acoustic sonority, which in turn is a correlate of the de-

class	description	German phones	English phones
P	unvoiced stops	ʔ p t k	p t k
B	voiced stops	b d g	b d g
S	unvoiced fricatives	f s ʃ ç x h	f θ s ʃ h
Z	voiced fricatives	v z ʒ	v ð z ʒ
N	nasals	m n ŋ	m n ŋ
L	liquids, glides	l r R j	l r j w
V	vowels, diphthongs	i: y: e: ø: ɛ: u: o: a: aɪ aʊ ɔʏ ɪ ʏ ɛ œ ʊ ɔ a ə ɐ	i: eɪ u: oʊ aɪ aʊ ɔʏ ɪ ɛ æ ʊ ɔ ʌ a ɚ ə

Table 1: Classes of English and German phones.

Onsets		
class	clusters	examples
SPL	sp + l/r/j	split sprite spurious
	st + r/j	street studious
	sk + l/r/j/w	sclerosis script skewer squid
Codas		
class	clusters	examples
LPPS	lpts lkts	sculpts mulcts
LPSP	ltst	waltzed
LSSS	lfθs	twelfths
NPPS	mpts ŋkts	prompts adjuncts
NPSP	mpst ŋkst	glimpsed jinxed
PSPS	ksts	texts
PSSS	ksθs	sixths

Table 2: English allows up to 3 consonants in the onset and up to 4 consonants in the coda of a syllable. The longest consonant clusters are restricted to a small number of distinct types.

Onsets		
class	clusters	examples
SPL	ʃp + l/r	Splitter Spritze
	ʃt + r	Streit
	sk + l/r	Sklerose Skrupel
Codas		
class	clusters	examples
NPSSPS	mpfsts	schrumpfst's
NPSSP	mpfst	schrumpfst
	ntSst	plantschst
NPPSP	mptst	promptst
NSPSP	nftst	sanftst

Table 3: Similar to English, German allows up to 3 consonants in the onset of a syllable, but up to 6 consonants can occur in the coda. The longest consonant clusters are represented by only a small number of distinct types.

gree of constriction of the vocal tract. We have argued elsewhere [5] that this ranking is not entirely consistent with actual acoustic measurements and is certainly not a valid descriptor of syllable structure across languages.

The complexity of syllable onset and coda structures poses problems for a syllabification algorithm because, despite of the abovementioned restrictions, ambiguous and multiple alternative syllable boundary locations are usually observed in polysyllabic words.

Determining the syllable boundary is important because the pronunciation of most phonemes is a function of their position in the syllable. This is most evident in the case of phonologically voiced obstruents in German: the voicing opposition for stops and fricatives is neutralized in the syllable coda. For instance, the phonological minimal pair *Bund* "union" – *bunt* "colorful" is in fact homophonic: [bʊnt]. In English, voiceless stops are aspirated when they constitute the onset of a stressed syllable (e.g., [t] in *top*). They are not aspirated, however, if they are preceded in the onset by [s] (e.g., [t] in *stop*), followed in the onset by [l,r] (e.g., [t] in *stress*), or if they occur in the coda (e.g., [t] in *pot*).

3. THE FINITE-STATE MODEL

Finite-state techniques are not new in the field of phonology. Their usage to model phonological conditional changes has been suggested as early as 1972 by Johnson [2], and later—independently—by Kay and Kaplan [4]. In the past decade or so, finite-state transducers have been used ubiquitously in the domain of phonology—as well as in morphology—not only in modeling phonological (and morphological) regular rewrite rules, but also for all sorts of mappings between string descriptions. The mathematical features and elegance of these simple devices makes them attractive for such purposes and, as we show here, for modeling multilingual syllabification.

One of the attractive features of finite-state transducers is that they are closed under mathematical composition; that is, if transducer T_1 maps the string s_1 into the string s_2, and transducer T_2 maps the string s_2 into the string s_3, then the transducer T_3, which is constructed by taking the composition of the original transducers, maps the string s_1 into the string s_3. The composition of the two transducers is described by the expression

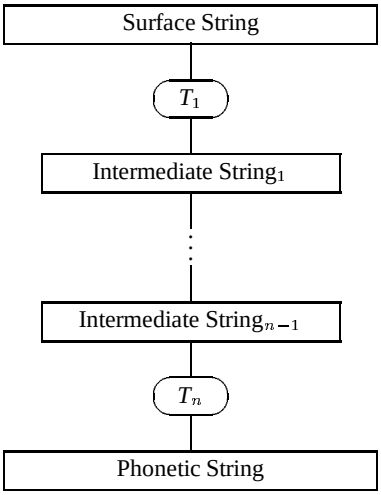


Figure 1: Cascade model: a series of transducers converts an orthographic input string into a phonetic output string by means of sequential composition. Additional transducers, e.g. for syllabification, can be inserted into the cascade.

$$T_3 = T_1 \circ T_2 \tag{1}$$

This feature allows, for example, a number of transducers to be applied in cascade form to an orthographic string (the input) to produce a phonetic string (the output) in a TTS system. Inserting additional transducers in the middle of this cascade is modular and fairly straightforward. Consider the cascade in Figure 1, which is made of n transducers. One can easily add an additional transducer between T_i and T_{i+1}, $1 \leq i < n$, to take care of syllabification; we shall designate the latter with T_σ. The only additional symbol that appears at the output of T_σ is the syllable boundary symbol, '$-$'. As long as this symbol is part of the alphabets of T_{i+1} through T_n, no other modification is required for the entire cascade.

In both English and German, morpheme boundaries frequently override default syllabification in compounded words. For this reason, the position in which T_σ is inserted in the cascade is crucial as the syllabifier needs to know about the morphological structure of its input. For instance, if the syllabifier is faced with the English phone sequence [naɪtreɪt], it will produce [naɪ–treɪt] by default. This would be the correct syllabification of "nitrate" but not of its homophone "nightrate" (the latter ought to be syllabified as [naɪt–reɪt]). This distinction is of relevance to TTS for at least two reasons: First, the acoustic properties of [r] differ depending upon the left context phone. The phonologically voiced consonant [r] is often initially or completely devoiced in the context of a preceding voiceless obstruent, such as [t] in this case, in the onset of the same syllable. The acoustic inventory of our TTS system includes two entries representing the diphone [r-eɪ], one of which is used only in the context mentioned above, and the other in all other contexts. The choice of the proper diphone depends on the existence, or absence, of a syllable boundary marker after [t]. Second, the duration assigned to [t] in our example depends on whether it is in the coda of the first or in the onset of the second syllable.

Another attractive feature of transducers is their bidirectionality. Inverting the transducers in Figure 1 produces a system that maps a phonetic string into an orthographic string. Hence, the inverse of T_σ, denoted by T_σ^{-1}, becomes a desyllabifier whose usefulness lies in the eye of the beholder!

The syllabifier described here is implemented as a weighted finite-state transducer; that is, each transition may be associated with a certain *weight* [6]. The construction of the transducer is based on training data with an additional mechanism for hand-tuning. The remainder of this section describes the procedure followed in constructing the syllabification transducer, T_σ.

3.1. Training From Data

The syllabification transducer is constructed from training data consisting of syllabified phonological words, which we shall call the *training database*. Various sources can be used to obtain the training database. For example, the current German syllabifier makes use of the *Celex Lexical Database* (release 2) [1], while the English syllabifier employs data from the Bell Labs pronunciation module, an in-house program, as well.

The procedure is as follows. First, a list of all the syllables in the training database is obtained. For example, the English phonological word *abductions* produces the three syllables [æb$-$dʌk$-\int$ənz]. This list is fed into a program that splits the syllables into plausible onsets, nuclei and codas. The above word produces the following onsets, nuclei and codas:

Onset	Nucleus	Coda
	æ	b
d	ʌ	k
$\int$	ə	nz

Second, sets of plausible onsets, nuclei and codas, with their frequencies of occurrence, are computed. This gives the statistics of each onset, nucleus and coda in the training data regardless of context. As a way of illustration, Table 4 gives the set of English nuclei found in the Celex database.

Each of the three sets (onsets, nuclei and codas) is then compiled into a weighted finite-state automaton by taking the disjunction of its members. The frequencies are converted into weights by taking their reciprocal as shown in the last column of Table 4. This results in three automata: A_o, A_n, and A_c, which accept onsets, nuclei and codas, respectively.

More formally, let $\mathcal{O}, \mathcal{N}$ and $\mathcal{C}$ be the sets of onsets, nuclei and codas, respectively; each element in the set is a pair (C, F), where C is the constituent in question (i.e., onset, nucleus or coda) and F is its frequency in the training database. Then we construct the three automata A_o, A_n, and A_c from the sets as follows:

$$A_o = \bigcup_{(o,f)\in\mathcal{O}} o\langle 1/f \rangle \tag{2}$$

Nucleus	f	$1/f \times 10^{-3}$
ə	62500	0.016
ɔ	12048	0.083
ɑ	5076	0.197
eɪ	21739	0.046
i:	30303	0.033
aɪ	15873	0.063
oʊ	13698	0.073
ɝ	22727	0.044
u:	10752	0.093
aʊ	4878	0.205
ɔʏ	1672	0.598
ʌ	15151	0.066
æ	25000	0.040
ɛ	26315	0.038
ɛɝ	7	142.857
ɪ	71428	0.014
ɪɝ	1	1000.000
a	13157	0.076
ʊ	3154	0.317

Table 4: The set of English syllable nuclei found in the Celex database, with number of observations (f). Weights for the transitions between states in the nucleus automaton are obtained by taking the reciprocal of the frequency of each nucleus type.

$$A_n = \bigcup_{(n,f)\in\mathcal{N}} n\langle 1/f\rangle \quad (3)$$

$$A_c = \bigcup_{(c,f)\in\mathcal{C}} c\langle 1/f\rangle \quad (4)$$

A weight given in angle brackets is associated with the preceding symbol in an expression.

3.2. The Phonotactic Automaton

Having A_o, A_n, and A_c at hand, one is ready to construct an automaton that enforces phonotactic constraints. This is a language-dependent step. For both English and German, syllabic phonotactics is described by the extended regular expression

$$A_{ph} = Opt(A_o)\, A_n\, Opt(A_c) \quad (5)$$

where Opt is the optional operator defined as

$$Opt(A) = A \cup \epsilon \quad (6)$$

and ϵ denotes the empty string. In other words, eq. 5 accepts an optional onset from A_o, followed by an obligatory nucleus from A_n, followed by an optional coda from A_c. The above automaton accepts one syllable at a time.

The syllabification automaton, denoted by A_σ, needs to accept a sequence of syllables, each—except for the last—followed by a boundary marker '$-$'. This is achieved by the expression

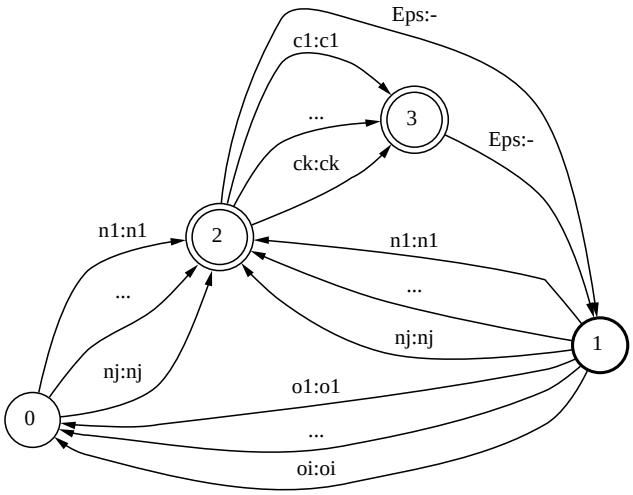


Figure 2: The syllabification transducer, depicting i onsets (shown as o1, $\cdots$, oi), j nuclei (shown as n1, $\cdots$, nj) and k codas (shown as c1, $\cdots$, ck). State 1 (in bold) is the initial state, double circles denote final states; 'Eps' stands for ϵ. The automaton maps each symbol on the transitions between two states onto themselves (e.g., o1:o1), and inserts the syllable boundary marker '$-$' after each non-final syllable, by mapping ϵ:$-$.

$$A_\sigma = A_{ph}(-A_{ph})^* \quad (7)$$

That is, a syllable from A_{ph} followed by zero or more occurrences of a) the boundary marker '$-$', and b) a syllable from A_{ph}.

3.3. The Syllabification Transducer

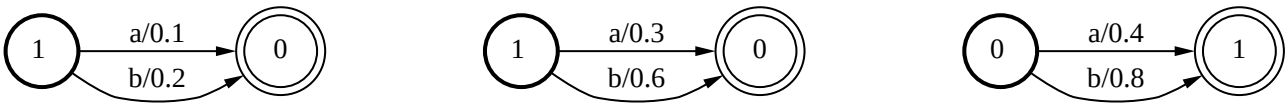
The automaton A_σ *accepts* a sequence of one or more syllables, each—but the last—followed by the syllable boundary marker '$-$'. We need to transform this into a transducer which *inserts* a '$-$' after each—but the last—syllable.

This is simply achieved by computing the identity transducer for A_{ph} and replacing '$-$' in eq. 7 with a mapping 'ϵ:$-$'. In other words, the syllabification transducer is

$$T_\sigma = Id(A_{ph})\left((\epsilon{:}-)Id(A_{ph})\right)^* \quad (8)$$

The Id operator produces the identity transducer of its argument [3]. That is, each symbol a on a transition in the argument becomes a mapping a:a. The transducer T_σ is depicted in Figure 2.

As mentioned above, the only language-dependent expression is A_{ph} in eq. 5. Other languages may use different expressions. For example, in languages where the onset is obligatory, as in Arabic and Syriac (considering the glottal stop [ʔ] as a consonant), one omits the optional operator applied to A_o in eq. 5.



(a) A_1 (b) A_2 (c) $A_1 \cap A_2$

Figure 3: Intersection of weighted automata. States shown in bold are initial states, double circles indicate final states. The intersection of two automata (a) and (b) produces a third automaton (c) that accepts the strings that both of the original two machines accept. In weighted automata, the weights of common paths in the original machines are summed up.

3.4. Hand-Tuning

While the procedure of constructing the syllabification transducer from training data is automatic, some *post hoc* hand-tuning may still be required. For example, Table 4 includes two nucleus types ([ɛɔ˞] and [ɪɔ˞]) whose numbers of observations are extremely low, indicating an artifact induced in the training process or possibly erroneous entries in the database. The weights derived from the frequencies of these nucleus types have to be manually corrected.

Hand-tuning becomes even more crucial when dealing with exotic languages where training data is either scarce or entirely not extant. In the latter case, the syllabifier is constructed solely by means of hand-tuning.

The mathematical elegance of weighted transducers makes hand-tuning not a difficult task. Say the nucleus [ɪɔ˞] is associated with the weight 1000×10^{-3}, but based on some observation needs to be hand-tuned to the value of 0.014×10^{-3} (the value of its monophthong counterpart [ɪ] in Table 4).

To achieve this, a partial duplicate set of $\mathcal{N}$ can be created, which contains only the nuclei that are to be fine-tuned, and where each nucleus is paired with the adjustment in weight that needs to be performed (e.g., for [ɪɔ˞], $(0.014 - 1000) \times 10^{-3}$). Let this new set be called $\mathcal{N}'$. Then, the automaton for the hand-tuned nuclei, A'_n, becomes

$$A'_n = \left(\bigcup_{(n',f) \in \mathcal{N}'} n'\langle 1/f \rangle \right) \bigcup \left(\bigcup_{(n,f) \in \mathcal{N} - \mathcal{N}'} n\langle 0 \rangle \right) \quad (9)$$

The first component in eq. 9 is identical to the expression A_n in eq. 3, but computes the weights for the hand-tuned nuclei instead. To complete this set, it is unioned with a disjunction of the remaining non-hand tuned nuclei (i.e., $\mathcal{N} - \mathcal{N}'$), where each element in $\mathcal{N} - \mathcal{N}'$ is given a weight of zero.

Now, the new automaton which incorporates both A_n and A'_n, denoted by A_{nuclei}, is simply the intersection of the two automata,

$$A_{nuclei} = A_n \cap A'_n \quad (10)$$

How does it come to pass that intersection produces the desired result? This lies in the definition of intersecting weighted automata. When no weights are used, the intersection of two automata produces a third automaton that accepts the strings that both of the original two machines accept. In the weighted version the same applies, with the addition that the weights of common paths in the original machines are *added* in the result. This is illustrated in Figure 3.

Similarly, one computes expressions for A_{onsets} and A_{codas}. These are incorporated in eq. 5 to form the new expression

$$A_{ph} = Opt(A_{onsets})\, A_{nuclei}\, Opt(A_{codas}) \quad (11)$$

In turn, this expression is incorporated in eq. 8, which builds T_σ.

It is crucial that the hand-tuning is done *before* T_σ is constructed, for the simple reason that eq. 10 (as well as the expressions for A_{onsets} and A_{codas}) make use of intersection, an operation under which accepting automata are closed, but transducers are not.

4. ILLUSTRATIONS

After having given a formal account of our approach in the previous section, we provide a few concrete examples that illustrate certain aspects and properties of the finite-state syllabifier.

The first example is a case in German where only one legal syllabification is possible in a cluster of four consecutive consonants. The noun *Künstler* [kʏnstlɐ] "artist" is correctly syllabified as [kʏnst–lɐ] by our syllabifier. Any other syllabification would require onsets ([nstl, stl, tl]) or codas ([nstl]) that are not in $\mathcal{O}$ or $\mathcal{C}$, the sets of plausible onsets and codas collected from the training database.

The second example illustrates the case of a polysyllabic German word where more than one syllabification is plausible. The noun *Fenster* [fɛnstɐ] "window" can be syllabified as [fɛn–stɐ] $\langle 75 \rangle$, [fɛns–tɐ] $\langle 74 \rangle$, or [fɛnst–ɐ] $\langle 87 \rangle$ (with weights shown in angle

brackets). Obviously, [fɛns–tɐ], with the smallest weight, is the most probable solution, reflecting the observed frequencies of the relevant onsets and codas in the training database. It is indeed the correct syllabification: the second best solution, which puts [s] in the onset of the second syllable, would result in [s] being incorrectly pronounced as [ʃ].

The third example comes from English and demonstrates another aspect of the syllabifier at hand. Standard dictionaries, such as *The American Heritage* and *Webster's Third*, provide syllabification that is influenced by the morphological structure of words; it is common in such dictionaries to split prefixes and suffixes from stems. For instance, both dictionaries syllabify the word *glamour* as [glæm–ɚ], whereas the more plausible syllabification in speech is [glæ–mɚ]. The output of our syllabifier produces the latter and hence is more faithful to spoken syllables. Since TTS produces spoken language, albeit synthetic, syllabification ought to represent the properties of spoken utterances, rather than morphological structure.

5. CONCLUSION

This paper demonstrated the design and implementation of a syllabifier as a weighted finite-state transducer. The syllabifier has been integrated into the finite-state based text analysis component of the Bell Labs English and German text-to-speech systems. The transducer was constructed by obtaining syllables as well as their internal structures and frequencies of occurrence from a lexical database. Weights on the transitions between states of the transducer were derived directly from the frequencies of onset, nucleus and coda types in the database. The weights reflect the plausibility of onset, nucleus and coda types, and thus play a significant role in obtaining the correct syllabification, especially in the case of consonant clusters in languages such as English and German, which offer ambiguous syllable boundary locations.

While the procedure of constructing the syllabification transducer from training data is automatic, manual or interactive fine-tuning is straightforward, if so required, due to the mathematical properties of weighted automata.

Syllabification is an important component of many language and speech applications, especially TTS systems. Syllable boundary location is a crucial piece of information for several components of a TTS system: for instance, the pronunciation of a phoneme as well as its duration depends upon the location of the phoneme in the syllable.

In keeping with the multilingual design of the Bell Labs TTS system, our approach is applicable to any language. It relies, however, on the phonotactics and syllable structure of each particular language. We discussed the syllable structure and phonotactics of English and German, and illustrated properties and performance of the syllabifier using examples from these two languages.

6. REFERENCES

1. Celex. The CELEX lexical database—Release 2. CD-ROM, 1995. Centre for Lexical Information, Max Planck Institute for Psycholinguistics, Nijmegen.
2. Johnson, C. *Formal Aspects of Phonological Description*. Mouton, 1972.
3. Kaplan, R., and Kay, M. Regular models of phonological rule systems. *Computational Linguistics 20*, 3 1994, 331–378.
4. Kay, M., and Kaplan, R. Word recognition. [This paper was never published. The core ideas are published in Kaplan and Kay (1994)], 1983.
5. Möbius, B. Word and syllable models for German text-to-speech synthesis. In *Proceedings of the Third ESCA Workshop on Speech Synthesis* (Jenolan Caves, Australia, 1998), ESCA.
6. Mohri, M., Pereira, F., and Riley, M. A rational design for a weighted finite-state transducer library. In *Automata Implementation*, D. Wood and S. Yu, Eds., Lecture Notes in Computer Science 1436. Springer, 1998, pp. 144–158.
7. Sproat, R., Ed. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Kluwer, Dordrecht, 1998.
8. Sproat, R., Möbius, B., Maeda, K., and Tzoukermann, E. Multilingual text analysis. In Sproat [7], ch. 3, pp. 31–87.
9. van Santen, J. P. H., Shih, C., Möbius, B., Tzoukermann, E., and Tanenblatt, M. Multilingual duration modeling. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)* (Rhodos, Greece, 1997), vol. 5, ESCA, pp. 2651–2654.