Computer Speech and Language (2002) **16**, 533–550
doi: 10.1016/S0885-2308(02)00023-2
Available online at http://www.idealibrary.com on IDEAL®

Efficient integrated response generation from multiple targets using weighted finite state transducers

Ivan Bulyko[†] **and Mari Ostendorf**

Electrical Engineering Department, University of Washington, Seattle, WA 98195, U.S.A

Abstract

In this paper, we describe how language generation and speech synthesis for spoken dialog systems can be efficiently integrated under a weighted finite state transducer architecture. Taking advantage of this efficiency, we show that introducing flexible targets in generation leads to more natural sounding synthesis. Specifically, we allow multiple wordings of the response and multiple prosodic realizations of the different wordings. The choice of wording and prosodic structure are then jointly optimized with unit selection for waveform generation in speech synthesis. Results of perceptual experiments show that by integrating the steps of language generation and speech synthesis, we are able to achieve improved naturalness of synthetic speech compared to the sequential implementation.

1. Introduction

Improvements in automatic speech recognition (ASR) have led to many new deployments of speech-enabled computer interfaces, particularly in telephone-based applications. In such systems, the quality of speech output impacts user acceptance, and many commercial human–computer dialog systems are constrained in order to be able to make use of pre-recorded voice prompts. While applications with very limited capabilities can make use of pre-recorded speech, many applications require more dynamic response generation which requires speech synthesis. The simple solution is to use a general text-to-speech synthesis system, but higher quality synthesized speech can be obtained using a system tailored for the specific domain. Further, the fact that the language is generated automatically provides an opportunity to pass more information to the waveform generation module than would be available in unrestricted text-to-speech synthesis. One example that has often been cited is the annotation of automatically generated syntactic and semantic structure as well as dialog context for purposes of improved prosody prediction. While this is important, our focus is on a

[†]E-mail: bulyko@ssli.ee.washington.edu (I. Bulyko).

0885-2308/02/$35

different opportunity that takes advantage of advances in concatenative speech synthesis: that is, to provide flexibility to the synthesizer in terms of possible responses.

The key idea is that there is more than one acceptable response at any point in a dialog. In particular, we take advantage of two main areas of flexibility: choice of wording and prosodic realization of an utterance. Instead of passing a single text string (or prosodically annotated text string) to a synthesizer, we pass an annotated network. Thus, the search for wording, prosody prediction and speech units is optimized jointly. In other words, instead of predicting a specific word sequence and prosodic realization first and then searching for units to match that target, our approach effectively makes a "soft" decision about the target words and prosody and evaluates alternative realizations of a given utterance.

Of course, since providing more flexibility increases the search space, then it also increases the computational cost and potentially the implementation complexity. Thus, a key to making this approach practical is implementation using weighted finite-state transducers (WFSTs). Each step of network expansion is then followed by minimization, and both can be implemented using a general purpose toolbox.

In the paper, we propose a general architecture based on: (i) representing a response in terms of an annotated word network rather than a single word sequence, and (ii) using a symbolic representation of prosodic structure for annotation. We also describe a specific implementation using a template-based language generator and a variable-length unit selection synthesis system. The paper is organized as follows. We will begin in Section 2 with a review of previous work in the area of language generation and speech synthesis in dialog systems. The architecture of our system will be described in Section 3, followed by details of how generation and synthesis can be integrated using WFSTs in Section 4. Experiments are described in Section 5, and we conclude by summarizing the key advances and laying out directions for future work in Section 6.

2. Background

This section provides an overview of recent work related to language generation and speech synthesis in dialog systems. We will start by describing different approaches to tying generation and synthesis within dialog systems (Section 2.1), followed by a summary of the key differences in our approach in Section 2.2. Then, in Section 2.3 we review recent developments in limited domain synthesis that we build on in the work.

2.1. Tying language generation and speech synthesis in dialog systems

The simplistic view of response generation treats natural language generation and speech synthesis separately. Decisions about text planning and sentence realization are made ignoring the capabilities of speech synthesis. Decisions about prosodic structure in speech synthesis are based on simplistic (error prone) text analysis, ignoring any syntactic information that might be available from the process of language generation. Clearly this strict separation is inefficient, and researchers have long considered the idea of more tight integration of these modules (Young & Fallside, 1979), often referred to as concept-to-speech (CTS) synthesis. In recent years, the increasing feasibility of human–computer dialog systems due to improved ASR performance has prompted the need for better response generation. As a result, there is growing research activity in the general area of CTS, particularly for limited domain applications.

An important motivation for integrating language generation and speech synthesis is improved prosody prediction. The basic idea is to augment the text output from a generator with additional information that is a by-product of the generation process. The key research problem is to determine what types of information – including semantic, syntactic and/or discourse information – that actually leads to improved synthesized speech quality. Several studies have been conducted (Davis & Hirschberg, 1988; Prevost & Steedman, 1994; Steedman, 1996; Pan & McKeown, 1997; Pan & McKeown, 2000; Hitzeman, Black, Taylor, Mellish, & Oberlander, 1998), though there is still much to be learned about what linguistic factors are most important for prosody prediction. An important aspect of these studies is that they leverage generators with more sophisticated linguistic internal representations. However, few such generators are used in speech dialog systems; most are hand-crafted template-based generators.

Template-based language generation (Seneff & Polifroni, 2000; Pellom, Ward, & Pradhan, 2000) involves mapping a semantic frame in a given dialog act directly into a template utterance assigned to this dialog act and then filling in required values from the semantic frame. The sequence of words in the template is predetermined by the system developer. This approach has the advantage of implementational simplicity, greater control for developer, and predictable output quality. It has the disadvantage that there is little reuse of tools from one application to another, and there is little (if any) explicit representation of linguistic structure that might be useful for prosody prediction.

Seneff and Polifroni (2000) overcome the limitation of communicating with prosody prediction by "tuning" the generator for improved synthesis performance. Instead of using plain text, the generator communicates with the synthesizer via a special mark-up language that can include additional information, such as prosodic characteristics of the template, and synthesis "shortcuts" which allow the developer to bypass the search when desired and explicitly represent waveform segments. Although this approach requires greater effort from the system developer, it has a potential for producing higher quality speech output.

Recently, stochastic methods and related data-driven learning techniques have been introduced into generation systems. An n-gram language model trained on transcribed human–human speech has been used to generate candidate responses that are subsequently filtered to eliminate grammatical errors (Oh & Rudnicky, 2000), and alternatively to score candidate responses for "quality" (Langekilde & Knight, 1998; Galey, Fosler-Lussier, & Potamianos, 2001). Ratnaparkhi (2000) proposed a statistical learning approach to produce natural language text directly from a semantic representation that could lead to a more useful means of scoring candidate sentences. Work by Bangalore, Rambow, and Walker (2001) offers the potential to introduce scores at the sentence planning level. These developments are important for the work reported here, because they involve generation of multiple responses associated with a score, which our work can take advantage of. However, because many current dialog systems are based on template generators, we chose to initially work with the template paradigm where wording alternatives are unweighted.

Improvements in speech synthesis for response generation have mainly involved designing unit-selection concatenative systems that are tailored to a given domain, taking advantage of the fact that a specific application is limited in scope. It has been a common practice in developing limited-domain synthesis systems to record a large number of domain-specific prompts. High quality synthetic speech is achieved by concatenating whole phrases and words with subword units for infrequent or new words (Yi & Glass, 1998; Donovan, Franz, Sorensen, & Roukos, 1999; Black & Lenzo,

2000). This requires having a speech corpus that sufficiently covers the generator's vocabulary and provides recordings of frequently used words and phrases in the appropriate prosodic context.

2.2. Extensions of the current work

While most work on integrating generation and synthesis has looked at prosody prediction, the focus of this work is primarily on *increased flexibility,* specifically the use of multiple alternatives at early stages and joint optimization of choices about wording, prosodic structure and segment realization. Language generators up to now have been constrained to produce a single surface representation for each given concept, though in many cases they actually produce multiple possible sentence realizations. Ignoring reasonable alternative wordings limits the possibilities for the synthesizer to achieve the best quality. Natural languages allow a variety of word sequences to represent the same concept. Provided with alternative wordings, a speech synthesizer can choose which one to produce, based on the quality of synthetic speech. Similarly, we can take advantage of the fact that it is often acceptable to use more than one prosodic rendition of an utterance – even in a specific discourse context (Ross & Ostendorf, 1996). Speech synthesis can benefit from this prosodic variability by choosing the prosodic structure that results in the most natural speech output, taking into account both prosodic structure and concatenative synthesis distortion effects.

Second, we look at the problem of more tightly coupled prosody prediction specifically for the case of template-based generators. Little attention has been paid to prosody for limited-domain synthesis tied to such generators, with the assumption that prosody is accounted for in the phrase-size units. However, as complexity of the domain increases, it becomes impossible to store all responses as phrases, and prosodic structure must be automatically predicted to achieve natural speech responses. Since syntactic structure is not explicitly represented in the template generator, we learn the mapping from different templates to a network of possible prosodic realizations by collecting and optionally reducing the set of prosodic patterns from different spoken versions of the template (i.e., with simple word variations). The method relies on a symbolic representation of prosody that the data is labeled with. The use of symbolic prosodic labels in unit selection has also been explored in (Wightman, Syrdal, Stemmer, Conkie, & Beutnagel, 2000; Taylor, 2000), but for unrestricted text-to-speech synthesis. In addition to the template prosody module, we also include a generic prosody prediction module to handle cases that are not covered in the synthesis database.

Introducing flexibility means that there is a bigger search space that makes it possible to find a better overall response, but a bigger search space has increased computational costs. Thus, a key to the success of this approach is efficient implementation of the search process, which we solve using the WFST framework. We build on a large body of work, which is summarized next. The key difference with respect to other work in our implementation is the inclusion of multiple levels of the response generation process in this framework.

2.3. Limited domain speech synthesis and WFSTs

Recently, a growing amount of attention in speech synthesis research has been drawn toward unit selection methods, based on using dynamic programming to search for

speech segments in a database that minimize some cost function (Hunt & Black, 1996; Hon, Acero, Huang, Liu, & Plumpe, 1998; Beutnagel, Conkie, Schroeter, Stylianou, & Syrdal, 1998; Donovan & Woodland, 1999). The cost function is designed to quantify distortion introduced when selected units are modified and concatenated. Typically, there are two components to the unit selection cost function: the *target cost*, which is an estimate of distortion that the database unit will be subject to when modified to match the target, and the *concatenation cost*, which is an estimate of the distortion associated with concatenating units that were not originally spoken in sequence. Target and concatenation costs have mostly focused on segmental distortion, and have included linguistically motivated distances based on phonetic categories (Yi & Glass, 1998) and/or spectral distances (Beutnagel *et al.*, 1998; Coorman, Fackrell, Rutten, & Van Coile, 2000). However, acoustic parameters such as fundamental frequency ($F0$) and duration are also often used in computing the target cost (Hon *et al.*, 1998).

Because the implementation of unit selection involves a minimum cost search, it is well suited to using WFST technology. Many areas of language and speech processing have adopted the WFST formalism (Roche & Shabes, 1997; Sproat, 1998; Mohri, Pereira, & Riley, 2002), because it supports a complete representation of regular relations and provides efficient mechanisms for performing various operations on them. Relations are represented by the states and arcs specified in the WFST topology, where the arcs carry input and output labels and may have weights (costs) assigned to them. Given these weights, the application of the WFST entails finding the best (minimum cost) path through the network. Transducers can be cascaded by means of composition; their functionalities can be combined by the union operation; and they can be reduced in size with standard minimization techniques.

The unit selection database can be efficiently represented in the form of a WFST, as was suggested by Hunt and Black (1996) and implemented by Yi, Glass, and Hetherington (2000), Bulyko and Ostendorf (2001b). The basic idea is to have units represented in the transducer by states, and the arcs carry out the transduction from input target labels into unit indices with weights derived from the combination of target and concatenation costs. Yi *et al.* (2000) use phones as the fundamental synthesis unit. In order to constrain the number of links in the WFST, they introduce a series of intermediate layers of states, where all possible unit transitions are mapped into more general classes based on phonetic categories, and transition costs between each pair of classes are computed. The system also incorporates word-to-phoneme conversion in a WFST module. This module is then composed with the unit selection WFST, allowing conversion of input words directly into a sequence of database units. In (Bulyko & Ostendorf, 2001b), half-phone units are used and the space of unit boundary frames is vector quantized to avoid precomputing concatenation costs between all possible pairs of units. A vector quantization concatenation network is constructed as a fully interconnected graph, where the states represent codebook entries and concatenation costs are assigned to the arcs.

In his work, we used words as the fundamental units in the database, hence we needed to compute few concatenation points and pruning of candidate units was unnecessary. States representing units were directly connected with weighted arcs; i.e., no additional network for concatenation costs was needed. Extension to the more general case that includes subword units is straightforward, but requires a more sophisticated unit database design, such as those described above. Our system architecture does not impose constraints on the internal structure of the individual WFST components and can support a wide variety of unit database designs.

3. Response generation components

Our response generation system is part of a mixed-initiative human–computer dialog system that is designed to help people solve travel planning problems, where users can get information about flights, hotels and rental cars. The complete dialog system is based on the University of Colorado (CU) Communicator system (Pellom *et al.*, 2000), with changes only to the response generation components. The CU system uses a client-server architecture, developed in the DARPA Communicator program with the specific goals of promoting resource sharing and plug-and-play capability across multiple sites (Seneff *et al.*, 1998). Under this architecture the system is composed of a number of servers that interact with each other through the Hub. The behavior of the Hub is controlled by a simple script. In the paper, we will focus on the aspects involving communication between the language generation and speech synthesis servers.

3.1. Overall architecture

There may be several modules involved in response generation, from dialog management and planning to waveform generation. However, because our system currently makes only very limited use of user preferences, the planning process is relatively simplistic and is handled by the dialog manager. Thus, there are three main modules involved in this work: sentence realization, prosody prediction and waveform generation. (These are implemented as two servers in the current system, with prosody prediction and waveform generation combined, but this is a choice of convenience rather than necessity.) As illustrated in Figure 1, the modules communicate by passing annotated graphs or lattices that get progressively expanded until the final search process determines the best path through the network. Since the choice of response may have an affect on the future dialog strategy, the synthesizer needs to inform the dialog manager and possibly other modules about the final output. (For example, the knowledge of the specific words in the generated response may be useful for detecting future error corrections by the user, as observed by Kirchhoff (2001)). Thus, the final outputs are the synthesized waveform for playback to the user and the best path through the network.

The lattices passed from the generator to the synthesizer contain arcs that are annotated with an index and optionally attributes, as well as weights associated with a

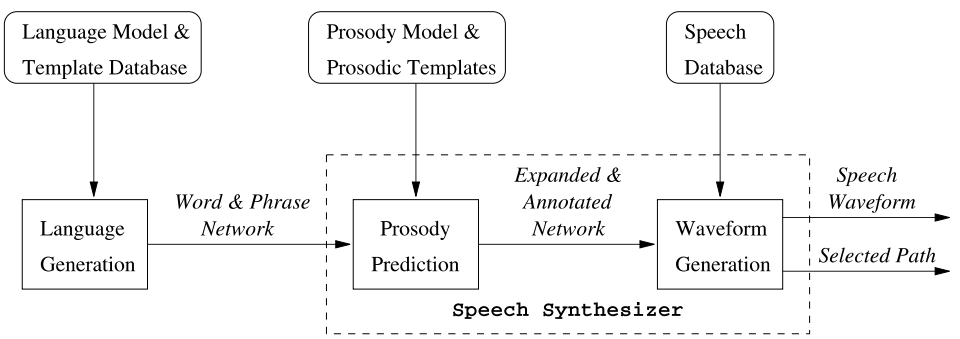

Figure 1. Communication strategy between the language generator and speech synthesizer.

cost for that particular arc, if any. At each stage, the lattice may be expanded and/or further annotated. The representation in terms of a weighted network fits nicely into the WFST framework, as will be described further in Section 4. Each module has access to a database for interpreting and further annotating the arc indices and attributes, and each adds new weights in any further expansion of the network.

The flexibility of the system is improved by having the generator and prosody prediction module output more than one candidate response. This gives more freedom to the synthesizer for selecting an utterance that will sound most natural. Generation of multiple candidate responses also frees the developer from the burden of redesigning the synthesizer database every time there is a minor modification to the generator, since the dynamic search over many alternatives makes it possible to automatically optimize the system response given a sufficiently rich speech database and inventory of templates. Moreover, it achieves a tight integration without sacrificing overall modularity of the system since the generator and the synthesizer are not internally coupled.

3.2. Language generation

The architecture described above can support any generation mechanism that can provide weighted alternatives, including all stochastic and hybrid approaches described in Section 2.1. Here, we build on a system that uses a template generator, which has a set of pre-determined responses for different dialog states. Depending on the response, there are empty slots that are filled in with values from the semantic frame provided by the understanding module. Example responses include:

Will you return to CITY-ORIG from CITY-DEST?
I have you going from CITY-ORIG to CITY-DEST. Is that correct?
Do you have an airline preference?
What time do you want to leave CITY-ORIG?

While the template generator is relatively limited and most current work seeks to go beyond this framework, it does have the advantage of simplifying the implementation of multilingual systems.

For the template generator, the lattices passed from the generator to the synthesizer contain arcs that correspond to words, phrases or templates, so the annotation on an arc could be a specific word, a phrase index, or a template index with attached words or values to fill the associated slots. A message from the generator may also include some dialog state information that may be relevant for prosody prediction.

The baseline generator provides a single text string to the synthesizer. Our modifications involved introducing multiple alternatives for a subset of the prompts and changing the interface from a text string to weighted, indexed lattices. An example of the alternatives for one prompt is illustrated in Figure 2. We modified seven prompts, and the resulting networks characterized 2.7 sentence alternatives on average.

The framework provides for the possibility of weighting the different text alternatives. In a template generator, which is hand-crafted, it is not clear how to specify the weights, so we simply gave all alternatives equal (zero) costs. With a stochastic generator, one could use grammar probabilities or n-gram scores to determine weights, as suggested in Section 2.1.

Will you	return	to	CITY_ORIG	from	CITY_DEST?
Will you	return	from	CITY_DEST	to	CITY_ORIG?
Would you like to	return	to	CITY_ORIG	from	CITY_DEST?
Would you like to	return	from	CITY_DEST	to	CITY_ORIG?
Do you want to	return	to	CITY_ORIG	from	CITY_DEST?
Do you want to	return	from	CITY_DEST	to	CITY_ORIG?

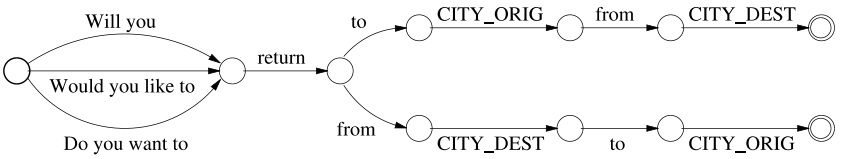

Figure 2. An example of a word and phrase lattice representing six alternative wordings of a prompt.

3.3. Prosody expansion

In addition to generating multiple candidate templates our system supports alternative prosodic targets for each word string. While it is certainly not the case that any prosodic realization will be acceptable, there is some degree of variability that is perceived as natural and appropriate for the context. Anecdotally, it is easy to notice differences in story readings or dramatic performances. More formal evidence is provided by Ross and Ostendorf (1996), where it was found that multiple readings of the same text (news stories) resulted in more than one symbolic representation for the prosodic structure of most utterances. In our prosodically labeled synthesis database, we find the same phenomenon. For example, the utterance "Will you return from Boston to Seattle?" had two prosodic realizations as illustrated in Figure 3. This variety does not appear to be correlated with location in the dialog because most utterances were not read in the context of a complete dialog, so presumably the different prosodic renditions are used by the speaker for variety. Style-related variety is characterized in a non-deterministic algorithm for prosody prediction proposed by Cahn (1998). Here, we allow statistical variety, but the joint search means that the overall algorithm is deterministic.

As we are building a constrained domain synthesizer, we can expect much of the input to resemble the utterances recorded when collecting data. Therefore, we want our model to precisely describe prosodic structures represented frequently in the training data in terms of so-called prosodic "templates". The "templates" are not

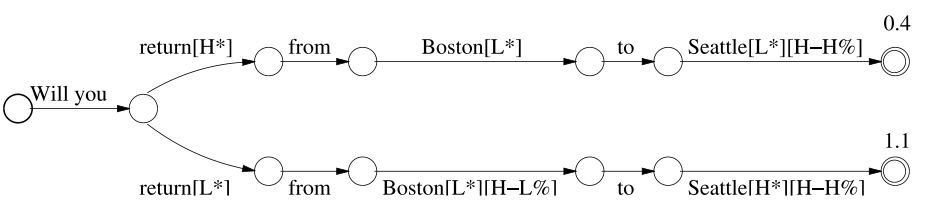

Figure 3. Representing prosodic variability in the form of a network of prosodic templates. In brackets are ToBI labels for accent and boundary tones.

stored speech waveforms, but rather a symbolic sequence of words and word "slots" that are associated with symbolic prosodic labels describing pitch accents and phrase boundary markers. Thus, the templates may or may not correspond to actual phrases in the synthesis database, depending on the particular words chosen. The candidate prosody templates are weighted according to their relative frequency in the training data, using negative log probability as the "cost" of choosing a particular template. The usefulness of multiple prosody templates was demonstrated in our previous work (Bulyko & Ostendorf, 2001a).

It is conceivable that none of the prosodic templates has a precise match in the unit database, either because of the particular choice of words for filling a slot or because there has been some change to the generator. In this situation, prosody is predicted for each word using a generic prosody prediction module.[1] The new predictor is effectively used as a back-off mechanism, though it differs somewhat from the "back-off prosody prediction" suggested by Taylor (2000) in that the implementation is integrated (because of the WFST implementation) rather than a two stage process.

The "generic" prosody prediction component uses decision trees, following the method proposed in (Wang & Hirschberg, 1992), Hirschberg (1993) and used in many studies since then. The features predicted include major prosodic phrase breaks and associated boundary tones and three types of pitch accents–the same labels as used in the prosody templates. The features used in prosody prediction were extracted at the word level, i.e., for each word in the corpus we produced one vector of features describing the word's part-of-speech, a function word indicator, and features describing the word's position in the syntactic tree and within the prosodic phrase. Given that our corpus is characteristic of the travel planning domain, we also used two features from the generator indicating if the word is a city name or a number. The predictor was trained and evaluated on prosodically labeled synthesis data, as described in Section 5. We also designed predictors on a standard Radio News corpus (Ostendorf, Price, & Shuttuck-Hufnagel, 1995), confirming that the accuracy was similar to previously reported results (Ross & Ostendorf, 1996) on the same corpus. In our application, unlike most prosody prediction work for speech synthesis, the predicted values are not used directly. At the leaf of the decision tree is a distribution of possible alternatives for a given context. All cases with non-zero probability are allowed as possible arcs, weighted according to their negative log probability.

3.4. Waveform generation

The waveform generation module is a constrained-domain concatenative synthesis system that uses a unit selection framework to incorporate variable size units, from single words to phrases. The module takes as input the prosodically annotated network, expands it into target phoneme sequences (annotated with lexical stress and prosodic markers), and then searches for the lowest cost unit sequence from a synthesis database that corresponds to a path in the network. The search linearly combines the standard target and concatenation costs used in unit selection with the other costs included in the network from previous modules.

In the system implemented for the work, the minimal size units are words, so target costs are zero for a matching word and infinite otherwise. The concatenation cost

[1]This idea was proposed but not implemented in Bulyko and Ostendorf (2001a).

between units U_i and U_j is the maximum distance[2] between overlapping frames of features: $\max(d_1, d_2)$, where d_1 is the distance between the last frame in unit U_i and the last frame in unit U_{j-1} which precedes unit U_j (as the database was naturally recorded) and similarly, d_2 is computed between the first frame in unit U_j and the first frame in unit U_{i+1} which follows unit U_i. This approach is more robust than computing a distance between two consecutive frames, because it does not imply continuity at join points. Motivated by work described by Wouters and Macon (2001), we used a line spectral frequencies (LSF) representation of frames at unit boundaries. Squared differences between individual LSF vector components were weighted by the inverse of the distance between the adjacent spectral lines. We also included $F0$ and energy for computing the concatenation costs, using the same max framework and normalizing the contribution of these terms so that the average was similar to the LSF average.

Because this is a constrained-domain system, it is possible to get relatively good quality speech without any waveform modification, and our current implementation does not include it. However, there are some cases for which it would be useful to add waveform modification and it is planned for future work.

4. Integrating generation and synthesis with WFSTs

A WFST architecture provides a framework for a flexible and efficient implementation of the response generation components. The flexibility of WFSTs accommodates the use of variable size units and different forms of prosody and text generation. The computational efficiency of WFST composition and finding the best path allows real-time synthesis, particularly for constrained domain applications. This section gives details about our approach to integrating language generation and speech synthesis under a common framework of WFSTs.

It should be clear that a word network can be represented as a finite-state automaton, which is a particular class of finite-state transducers. The process of building prosody prediction and unit selection WFSTs used for speech synthesis will be described in Sections 4.1 and 4.2, respectively. Then Section 4.3 will explain how the steps of template generation, prosody prediction and unit selection are tightly coupled by composing the corresponding WFSTs.

4.1. Prosody prediction

The overall modular structure of our finite-state prosody model is summarized in Figure 4, including two types of WFST models: *template prosody* which describes specific prosodic structures in the training data, and *general prosody* prediction which predicts prosody for unseen cases. Both prosody prediction models are generated at two levels: utterance and phrase. At each level the prosody prediction WFST may itself be produced by composing individual transducers, as in the case of accent and tone prediction at the phrase level. Then, the resulting models at each level (i.e., utterance level and phrase level) are composed to form the overall prosody prediction WFST. The order of terms in the composition corresponds to the order of steps during prosody prediction, i.e., utterance level prosody is generated first and then used as a predictor

[2] The idea of using the maximum (vs. the average) was borrowed from Coorman *et al.* (2000), but the specific distances used here differ somewhat.

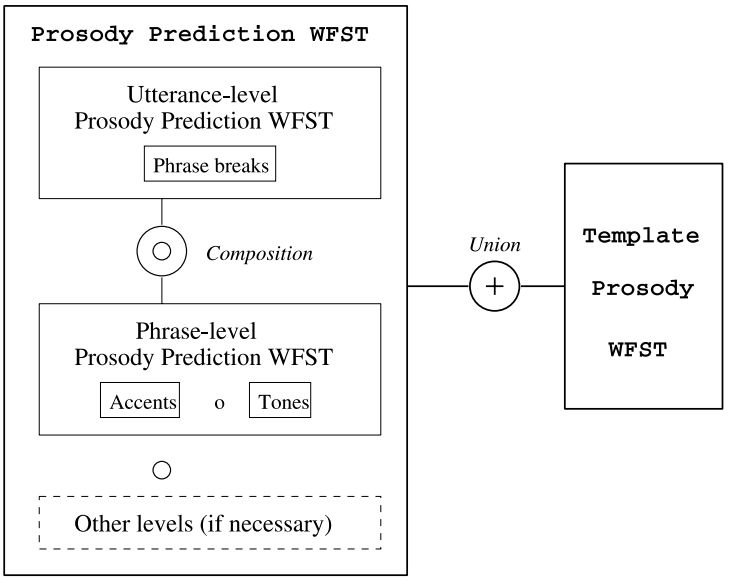

Figure 4. Modular structure of our prosody model, where $\oplus$ indicates union and $\circ$ indicates composition operations on WFSTs.

for the phrase level prosody. Finally, the template and prosody prediction WFSTs can be combined into a single transducer by means of the union operation. The modular structure allows other levels of prosodic structures (such as word or paragraph levels) to be easily added if desired.

The general prosody prediction module is based on decision trees, which can be efficiently compiled into weighted finite-state transducers. A simple decision tree can be represented by a WFST with just two states (a start and an end state) and the number of arcs equal to the number of leaves in the tree times the number of different values that the output variable can take (as illustrated in Figure 5). The costs in the resulting WFST should reflect the conditional probability distributions that can be computed at each leaf when training the tree. As suggested by Sproat (1996, 1998), we used $c(x) = -\log(p(x \mid leaf))$ as the cost function in our experiments.

For cases where there is more than one prosody "template", no prosodic target will be given zero cost. In this case, it may happen that the template with the higher prosody

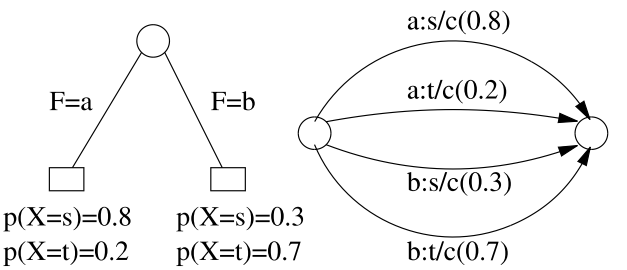

Figure 5. A simple decision tree and its WFST representation, where F is a prediction variable and $\{s, t\}$ are the possible class labels.

cost has zero concatenation cost (recorded in the database), while the template with the lower prosody cost has some non-zero concatenation cost. In most cases, we want the total cost to favor the version that is a complete phrase in the database, so the prosody costs are scaled so that the concatenation costs dominate the total cost in such situations. In addition, the costs in the template prosody transducer should also be scaled so that they are (on average) lower than the cost of the general decision tree-based prosody prediction, since the templates (when applicable) can presumably model prosody with greater accuracy than the decision tree. Costs for likely prosodic sequences from the general model were scaled so that on average they were close to the average concatenation cost.

4.2. Unit selection

The units in the synthesis database can be treated as states in the finite state transducer with the state transition cost given by the concatenation cost. The units can be of arbitrary size. It is, however, important to match the unit inventory to the output of the prosody prediction module in order to satisfy necessary conditions for composing the prosody prediction and the unit selection WFSTs, i.e., the set of output strings from the prosody prediction WFST must be acceptable as input to the unit selection transducer. In the case of limited domain synthesis, many of the responses that a text generator produces are likely to contain phrases that were recorded during data collection.

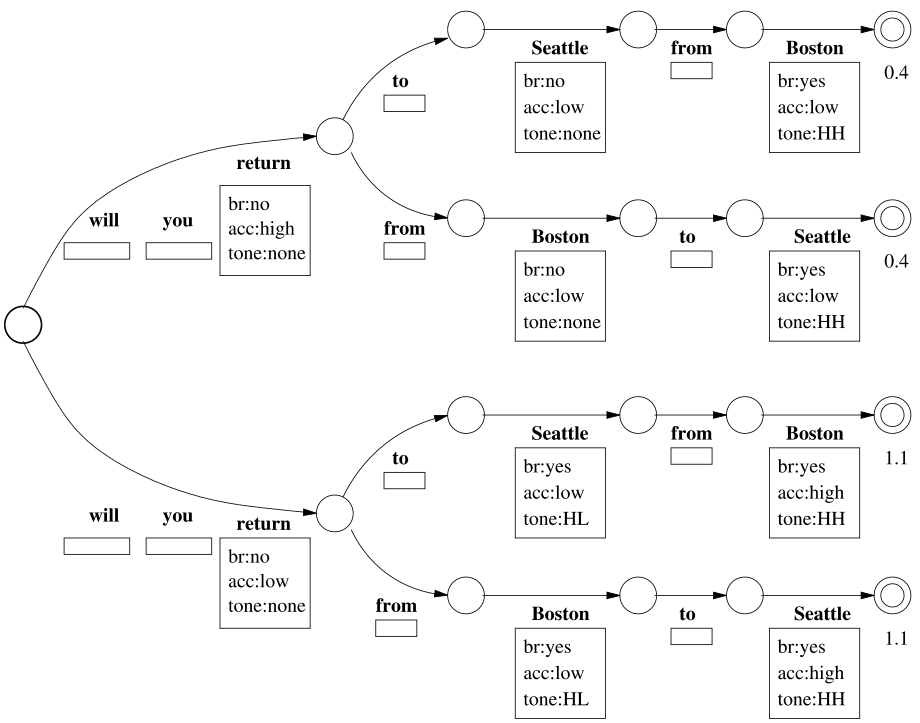

Figure 6. An example of a finite-state network of targets containing multiple word sequences and various prosodic specifications. Values in boxes describe prosody, i.e breaks, accents and tones. Empty boxes correspond to an absence of any prosodic labels.

Both words and phrases are indexed directly and treated as units in the database in the implementation here. For instance in Figure 6, a phrase "*will you return*" is mapped into a single arc, and hence as a whole would have an instance in the unit database. Subword elements of these words and phrases can also be used as units when it is necessary to be able to synthesize new words.

4.3. WFST composition

WFSTs provide a convenient representation of the different types of targets used in response generation: text, prosodic structure, and speech units. As explained earlier, the language generator outputs a network of templates to the synthesizer. Each of these templates is then mapped into one or more prosodic specifications producing a network of targets. For example, as illustrated in Figure 2, the system response can either be "*Will you return to Seattle from Boston?*" or "*Will you return from Boston to Seattle?*" and each of these word sequences can have various prosodic realizations as shown in Figure 3. At run time, the word and phrase lattice is expanded by composing it with the network of prosodic templates and the resulting target, shown in Figure 6, is then combined with the decision-tree-based word-level prosody prediction module by the union operation. The resulting network is composed with the unit selection WFST and the best path through the final transducer determines the output sequence of units. The composition operation is better than a sequential application of the two transducers because it allows for a wider search space by not making a hard decision when predicting prosody. Prior to performing the union and composition operations we minimize the transducers for efficiency.

Each alternative path through the network is weighted according to its relative frequency in the training data. The weights (negative log probabilities) are linearly combined in the composition step and assigned to the exit states, represented by double circles in Figure 6. The relative scaling of costs across these two WFSTs can be tuned according to a given task. We used informal listening tests in order to obtain relative scaling factors for each type of cost. However, perceptual experiments may be useful to better optimize the scaling factors.

The modular WFST architecture makes it easy to add new components to the synthesizer. For example, one can design a letter-to-sound WFST that models different pronunciations or simply store a short list of alternatives in a dictionary (actually, a separate WFST), which would expand the space of candidate units even further, provided that the unit selection WFST supports subword units.

5. Experiments

This section provides details on our experiments. First, in Section 5.1 we introduce the corpora that we used. Then, Section 5.2 describes the prosody prediction results. Finally, Section 5.3 covers our perceptual experiment demonstrating the benefits of synthesizing from multiple targets.

5.1. Corpora

We implemented the joint prosody prediction and unit selection as the synthesis component of a travel planning system developed at the University of Colorado (Pellom *et al.*, 2000). The corpus (approximately 2 hours of speech) contains recordings

of system responses spoken in isolation by one female speaker. The utterances were automatically segmented with subsequent hand-correction applied to the word boundaries. A trained linguist annotated a subset of the corpus (412 utterances) with ToBI prosodic labels (Silverman et al., 1992). To alleviate the data sparsity and to lessen the effects of labeling inconsistency we have converted the ToBI labels into a simplified representation, where pitch accents were compressed into three categories: high (H^*, $L+H^*$), downstepped ($!H^*$, $L+!H^*$, $H+!H^*$), and low (L^*, L^*+H). Four possible types of boundary tones were used (L–L%, L–H%, H–L%, H–H%), but only major prosodic boundaries (break index 4) were annotated. The corpus was also automatically labeled with part-of-speech tags (Ratnaparkhi, 1996) and syntactic structure (Collins, 1996). In concept-to-speech generation, you would not typically need a parser or a tagger since the generator would provide that information, but the generator used in this implementation was not designed with this interface in mind. Information available from the generator included semantic classes (city, number).

5.2. Prosody prediction

The entire data set (2752 words) was randomly split into training and test sets. Three quarters of the data was used for training the decision trees, and the rest was used for testing. Prediction results for the symbolic prosodic labels described earlier and a reduced class of accented vs. unaccented words are summarized in Table I. Chance performance was computed by always predicting the most frequent label; all predictors did significantly better than chance.

By examining the questions used in the trees we found that use of features describing a word's position in the prosodic phrase did not improve performance on the test set. Syntactic features were especially useful for predicting phrase breaks, while the part-of-speech labels, function word indicator, city name and number flags substantially benefited accent and tone prediction. We also found that break labels were somewhat useful for predicting accents, and accents as a feature, in turn, improved tone prediction.

While we were able to achieve high accuracy with breaks and boundary tones, accent tone prediction was not very accurate. This is consistent with the findings of Ross and Ostendorf (1996) and the view that there is more variability allowed in accent tone type. Even if we compress all types of accent labels into one category ($\pm$ presence of a pitch accent, labeled "accents-binary" in the table), the accuracy is low (74.2%) compared to the 80–85% accuracy reported for decision tree labeling of binary accents on other corpora (Hirschberg, 1993; Ross & Ostendorf, 1996). However, we verified that accuracy of the decision trees on a Radio News corpus is similar to other results on that

TABLE I. Prosody prediction results, where "chance" performance is computed by always predicting the most frequent label

Type of prosody and prosodic labels	Prediction accuracy (%)	Chance performance (%)
Breaks (none, major break)	92.2	81.0
Accents (none, high, low, downstepped)	59.4	46.6
Accents-binary (none, accent)	74.2	53.4
Boundary tones (LL, LH, HL, HH)	86.4	69.9

corpus (Ross & Ostendorf, 1996). The low accuracy on accent tones provides further motivation for allowing multiple targets rather than a single one.

5.3. Synthesizing from multiple targets

We constructed prosodic templates (as described in Section 4.1) for several types of target sentences common to the text generator, each comprised of a sequence of the compressed ToBI labels. We limited our focus to several types of sentences containing city names in various prosodic contexts. Through informal interactions with the dialog system we found that these types of utterances often had incorrect prosody and could potentially benefit from better prosody prediction.

Twenty target sentences were synthesized by three different methods, outlined in Table II. Version A made no use of prosody, and hence, the synthesis was driven entirely by the concatenation costs. In version B, each template had only one word sequence represented by a single zero-cost prosodic target; all other prosodic alternatives were weighted according to the word-level prosody prediction performed by the decision trees as described in the previous section. Finally, in version C, we made use of alternative word sequences as well as multiple (weighted) prosodic targets. The target sentences were chosen so that they do not match any single continuously recorded utterance in the database in its entirety.

We conducted a perceptual experiment, where subjects ranked versions A, B, and C (relatively) based on their naturalness. There were 16 subjects, all native speakers of American English, and each ranked all three versions of 20 responses. The subjects were allowed to play the three versions of each response any number of times in any order. Scores of equal ranks were allowed in case a subject cannot perceive the difference between given samples (some are identical) or considers them equally natural. The order of sentences and of the three different versions for each was randomized. Some of the subjects were speech researchers but all were naive with respect to the system implementation and corpus. The rankings submitted by one subject were excluded from the final results because they exhibited a much larger variance than that of other subjects. Excluding this subject, there were 300 trials in total. However, in several cases waveforms were identical in versions A and B, so there were in effect only 194 trials for the A–B comparison.

The results of our perceptual experiments are illustrated in Table III. A pairwise comparison of the three versions shows that: (1) on average, version B was rated more natural than version A, indicating that even simple prosody controls can be helpful; and (2) version C was preferred by the listeners more often than version B, which points out the benefit of using multiple word sequences and prosodic targets in the template. Version C was rated the best of all three versions in 202 trials (or 67.3% of the time), and was at least as good as either A or B in 223 trials (or 74.3% of the time).

TABLE II. Differences in synthesis methods used to produce the three versions evaluated in the perceptual experiment

Version	Word sequences in the template	Prosodic targets in the template	Word-level prosody prediction
A	Single	None	No
B	Single	Single	Yes
C	Multiple	Multiple	Yes

TABLE III. Perceptual experiment results

A vs. B				B vs. C			
Wins		Ties	Significance test, P	Wins		Ties	Significance test, P
A	B			B	C		
59	90	45	0.0122	70	212	18	4.60×10^{-17}
30.4%	46.4%	23.2%		23.3%	70.7%	6.0%	

Shown are pairwise comparisons: A vs. B and B vs. C. Significance of the results is measured by the probability P of sample medians not being significantly different.

These results support the main conclusion in Bulyko and Ostendorf (2001a), which was that it is useful to allow prosodic alternatives in the unit selection search. However, the experiment design here is somewhat different than our previous work. First of all, more subjects participated in this experiment (sixteen as opposed to five), and most of the subjects this time were not speech researchers. In our current experiment, we used hand-corrected word boundaries, which improved the overall speech quality for all synthesized versions. Perhaps most importantly, we used a general prosody prediction module as a back-off mechanism which may have led to the improved performance of version B over version A, which was not observed for the case of a single prosodic prediction in the experiment by Bulyko and Ostendorf (2001a).

It is interesting to note that several subjects reported that their scoring was influenced by the alternative word sequences that version C made use of. Sometimes they strongly preferred one wording over the other hence biasing their scores toward one word sequence. The latter aspect emphasizes the importance of being able to weigh alternative word sequences according to user preferences.

6. Discussion

In summary, we have demonstrated that by expanding the space of candidate responses in a dialog system we can achieve higher quality speech output. Specifically, alternative word sequences in addition to multiple prosodic targets were used to diversify the output of the language generator, taking advantage of the natural (allowable) variability in spoken language. Instead of specifying a single desired utterance, we make a "soft" decision about the word sequence and the prosodic target and let the synthesizer select the utterance that will sound most natural.

The WFST architecture offers a convenient representation of multiple targets for all components in the system. The standard operations of union, composition and finding the best path on WFSTs allow efficient ways of combining the network of targets, the template and prosody prediction WFSTs, and the unit database at run time, offering the real-time performance needed for interactive dialog applications. The flexibility of WFSTs facilitates the use of variable size units and various forms of prosody prediction, hence making it possible to synthesize out-of-domain utterances.

Generation of multiple candidate responses allows the system to automatically optimize the output for the best quality, when provided with a rich inventory of templates. This takes the burden of tuning the language generator to the speech synthesizer away from the system developer, providing for better plug-and-play capability of the individual modules.

The WFST architecture can easily be extended to include other factors, such as subword units and multiple pronunciations in the unit selection modules as well as other types of prosodic information. In addition, it would be of interest to explore the use of joint search algorithms with stochastic generators, which can provide more wording choices than explored here, as well as a non-uniform cost associated with each. Finally, it may also be useful to include user preferences as well as the dialog dynamics in assigning weights to the different wording alternatives.

We thank Bryan Pellom and the Center for Spoken Language Research at the University of Colorado for providing the speech corpus, and Lesley Carmichael for prosodic labeling of the corpus. This material is based upon work supported by the National Science Foundation under Grant No. (IIS-9528990). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Bangalore, S., Rambow, O. & Walker, M. (2001). Natural language generation in dialog systems. In *Proceedings of Human Language Technology Conference*, pp. 67–73.

Beutnagel, M., Conkie, A., Schroeter, J., Stylianou, Y. & Syrdal, A. (1998). The AT&T Next-Gen TTS system. In *Joint Meeting of ASA, EAA, and DAGA*, pp. 18–24.

Black, A. & Lenzo, K. (2000). Limited domain synthesis. *Proceedings of ICSLP*, 2, pp. 411–414.

Bulyko, I. & Ostendorf, M. (2001a). Joint prosody prediction and unit selection for concatenative speech synthesis. *Proceedings of ICASSP*, 2, pp. 781–784.

Bulyko, I. & Ostendorf, M. (2001b). Unit selection for speech synthesis using splicing costs with weighted finite state transducers. *Proceedings of Eurospeech*, 2, pp. 987–990.

Cahn, J. (1998). A computational memory and processing model for prosody. *Proceedings of ICSLP*, 3, pp. 575–578.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. *Proceedings of the 34th Annual Meeting of the ACL*.

Coorman, G., Fackrell, J., Rutten, P. & Van Coile, B. (2000). Segment selection in the L&H Realspeak laboratory TTS system. *Proceedings of ICSLP*, 2, pp. 395–398.

Davis, R. J. & Hirschberg, J. (1988). Assigning intonational features in synthesized spoken directions. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pp. 187–193.

Donovan, R. & Woodland, P. (1999). A hidden Markov-model-based trainable speech synthesizer. *Computer Speech and Language* **13(3)**, 223–241.

Donovan, R., Franz, M., Sorensen, J. & Roukos, S. (1999). Phrase splicing and variable substitution using the IBM trainable speech synthesis system. *Proceedings of ICASSP*.

Galey, M., Fosler-Lussier, E. & Potamianos, A. (2001). Hybrid natural language generation for spoken dialog systems. *Proceedings of Eurospeech*, 3, pp. 1735–1738.

Hitzeman, J., Black, A. W., Taylor, P., Mellish, C. & Oberlander, J. (1998). On the use of automatically generated discourse-level information in a concept-to-speech synthesis system. *Proceedings of ICSLP*, 6, pp. 2763–2766.

Hirschberg, J. (1993). Pitch accent in context: Predicting prominence from text. *Artificial Intelligence* **63**, 305–340.

Hon, H., Acero, A., Huang, X., Liu, J. & Plumpe, M. (1998). Automatic generation of synthesis units for trainable text-to-speech systems. *Proceedings of ICASSP*, pp. 293–296.

Hunt, A. & Black, A. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. *Proceedings of ICASSP*, 1, pp. 373–376.

Kirchhoff, K. (2001). A comparison of classification techniques for the automatic detection of error corrections in human–computer dialogues. *Proceedings of NAACL Workshop on Adaptation in Dialogue Systems*, pp. 33–40.

Langekilde, I. & Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. *Proceedings of COLING-ACL*, pp. 704–710.

Mohri, M., Pereira, F. & Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech and Language* **16**, 69–88.

Oh, A. H. & Rudnicky, A. (2000). Stochastic language generation for spoken dialogue systems. *Proceedings of ANLP/NAACL 2000 Workshop on Conversational Systems*, pp. 27–32.

Ostendorf, M., Price, P. & Shuttuck-Hufnagel, S. (1995). The Boston University Radio News Corpus. BU Technical report ECS-95-001.
Pan, S. & McKeown, K. (1997). Integrating language generation with speech synthesis in a concept to speech system. *Proceedings of ACL/EACL Concept to Speech workshop*, pp. 23–28.
Pan, S. & McKeown, K. (2000). Prosody modeling in concept-to-speech generation: methodological issues. *Philosophical Transactions of the Royal Society* **358**, 1419–1431.
Pellom, B., Ward, W. & Pradhan, S. (2000). The CU Communicator: an architecture for dialogue systems. *Proceedings of ICSLP*, 2, pp. 723–726.
Prevost, S. & Steedman, M. (1994). Specifying intonation from context for speech synthesis. *Speech Communication* **15**, 139–153.
Ratnaparkhi, A. (1996). A Maximum Entropy Part-Of-Speech Tagger. *Proceedings of the Empirical Methods in Natural Language Processing Conference*.
Ratnaparkhi, A. (2000). Trainable methods for surface natural language generation. *Proceedings of the1st Meeting of the North American Chapter of the Association of Computational Linguistics*, pp. 194–201.
Finite-state language processing (Roche, E., and Y. Shabes, eds.) (1997). MIT Press Cambridge, MA.
Ross, K. & Ostendorf, M. (1996). Predicting abstract prosodic labels for speech synthesis. *Computer Speech and Language* **10**, 155–185.
Seneff, S. & Polifroni, J. (2000). Formal and natural language generation in the Mercury conversational system. *Proceedings of ICSLP*, 2, pp. 767–770.
Seneff, S., Hurley, E., Lau, R., Pao, C., Schmidt, P. & Zue, V. (1998). Galaxy-II: a reference architecture for conversational system development. *Proceedings of ICSLP*, pp. 931–934.
Silverman, K., Beckman, M., Pitrelli, J. & Ostendorf, M., et al. (1992). ToBI: a standard for labeling English prosody. *Proceedings of ICSLP*, pp. 867–870.
Sproat, R. & Riley, M. (1996). Compilation of weighted finite-state transducers from decision trees. *Proceedings of the34th Annual Meeting of the Association for Computational Linguistics*, pp. 215–222.
Multilingual text-to-speech synthesis (Sproat, R., ed.) (1998). Kluwer, Dordrecht.
Steedman, M. (1996). Representing discourse information for spoken dialogue generation. In *International Conference on Spoken Language Processing*, *Proceedings of International Symposium on Spoken Dialogue*, pp. 89–92.
Taylor, P. (2000). Concept-to-Speech synthesis by phonological structure matching. *Philosophical Transactions of the Royal Society Series A* **356(1769)**, 1403–1416.
Wang, M. & Hirschberg, J. (1992). Automatic classification of intonational phrase boundaries. *Computer Speech and Language* **6**, 175–196.
Wightman, C., Syrdal, A., Stemmer, G., Conkie, A. & Beutnagel, M. (2000). Perceptually based automatic prosody labeling and prosodically enriched unit selection improve concatenative speech synthesis. *Proceedings of ICSLP*, 2, pp. 71–74.
Wouters, J. & Macon, M. (2001). Control of spectral dynamics in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing* **9(1)**, 30–38.
Yi, J. & Glass, J. (1998). Natural-sounding speech synthesis using variable-length units. *Proceedings of ICSLP*, pp. 1167–1170.
Yi, J., Glass, J. & Hetherington, L. (2000). A flexible scalable finite-state transducer architecture for corpus-based concatenative speech synthesis. *Proceedings of ICSLP*, 3, pp. 322–325.
Young, S. & Fallside, F. (1979). Speech synthesis from concept: a method for speech output from information systems. *Journal of the Acoustical Society of America* **66(3)**, 685–695.

(Received 26 September 2001 and accepted for publication 24 April 2002)