Text-to-Speech Synthesis

Richard W. Sproat
Joseph P. Olive

This paper presents an overview of the problems that occur during text-to-speech (TTS) conversion and describes the particular solutions to these problems taken within the AT&T Bell Laboratories TTS system. In addition to discussing the linguistic and speech analysis issues that must be addressed in a high-quality TTS system, this paper also outlines the modular architecture of the AT&T Bell Laboratories TTS system and the advantages of its modular design.

Introduction

The AT&T Bell Laboratories TTS system, an automatic system for converting text into speech for English, is being extended to other languages. This paper describes the text analysis, linguistic, and speech analysis problems that arise during TTS conversion, as well as their solutions. Also described are the modular architecture of the system, shown in Figure 1, and the advantages of this design.

It is tempting to think of the problem of text-to-speech conversion as "speech recognition in reverse." Current speech recognition systems are generally deemed successful if they can convert speech input into the sequence of words that was uttered by the speaker, so one might imagine that a TTS synthesizer could take a body of text, convert the words one by one into speech (being careful to pronounce each correctly), and concatenate the results. However, if we consider what literate humans must do when they read a text aloud, it becomes clear that this model is far too simplistic. Pronouncing words correctly is only part of the problem faced by human readers. To sound as if they understand what they are reading, they must also emphasize (accent) some words and deemphasize others, "chunking" the sentence into meaningful phrases. They must pick an appropriate pitch contour and control certain aspects of their voice quality. They must also know that a *phoneme* (roughly, a speech sound) should be longer if it appears in some positions in the sentence than in others, because *segmental durations* are affected by various factors, including phrasal position.

What makes reading such a difficult task is that *all* writing systems systematically fail to specify many kinds of information that are important in speech. While the written form of a sentence (usually) completely specifies the words that are present, it will only partly specify the intonational phrases, typically by using some form of punctuation. It most often will not indicate which words to accent or deaccent, and will hardly ever give information on segmental duration, voice quality, or intonation. The orthographies of some languages—for example, Chinese, Japanese, and Thai—do not indicate word boundaries, so even these must be computed by the reader. Even in English, single *orthographic* words (for example, AT&T) can actually represent multiple words (that is, A T and T). Humans can perform these tasks because, besides knowing the grammar of their language, they usually understand the content of the text they are reading, and can appropriately manipulate various factors that affect it, including intonation and voice quality. The task of a TTS system is to mimic what people do when they read text. A machine, however, is hobbled because it "knows" the grammatical facts of the language only imperfectly, and generally "understands" nothing of what it is reading. TTS algorithms must do the best

they can, using, where possible, purely grammatical information to decide on such things as accentuation, phrasing, and intonation.

It is natural to divide the TTS task into two broad subproblems. The first of these is the conversion of text into some form of linguistic representation, which includes information on the phonemes to be produced, their duration, the locations of any phrase boundaries, and the pitch contour to be used. The second problem is to take this information and convert it into a speech waveform. Each main task naturally divides into further subtasks, which are described in the sections that follow.

Text Analysis and Linguistic Analysis

This section describes in some detail the process of converting text into a linguistic representation.

Text Preprocessing. Text enters the synthesizer in an electronically coded format, either ASCII or some other standard coding scheme, depending on the language. One of the first tasks facing a TTS system is dividing the input into reasonable chunks, the most obvious being the sentence. In some writing systems (for example, Chinese), a designated symbol marks the end of a declarative sentence and is used for nothing else; for those systems, end-of-sentence detection is not problematic. For English and other languages we are not so fortunate, because a period, besides being a sentence delimiter, is also used, for example, to mark abbreviations. Before concluding that a period marks the end of a sentence, the reader must eliminate other possibilities. In the English version of the TTS system, the text is divided, or *tokenized*, into words on the basis of white space and punctuation. The abbreviation expansion component of the text analysis module—also called the "front end"—is then invoked to check for common abbreviations that might allow one to eliminate one or more periods from further consideration. For example, if the preprocessor encounters the string *Mr.* in an appropriate context (for example, followed by a capitalized word), it will expand it as *mister* and remove the period. Of course, abbreviation expansion itself is not trivial, because many abbreviations are ambiguous. So, is *St.* to be expanded as *Street* or *Saint*? Such ambiguities are currently solved using a set of heuristics.[1] For *St.*, the system first checks to see if the abbreviation is followed by a capitalized word (that is, a potential name), in which case it is expanded as *Saint*. If it is preceded instead by a capitalized word, a number, or an alphanumeric—such as *49th*—it will be expanded as *Street*. Another problem that must be dealt with is the conversion of numbers into words: *232* should usually be expanded as *two hundred thirty two*, but if the same

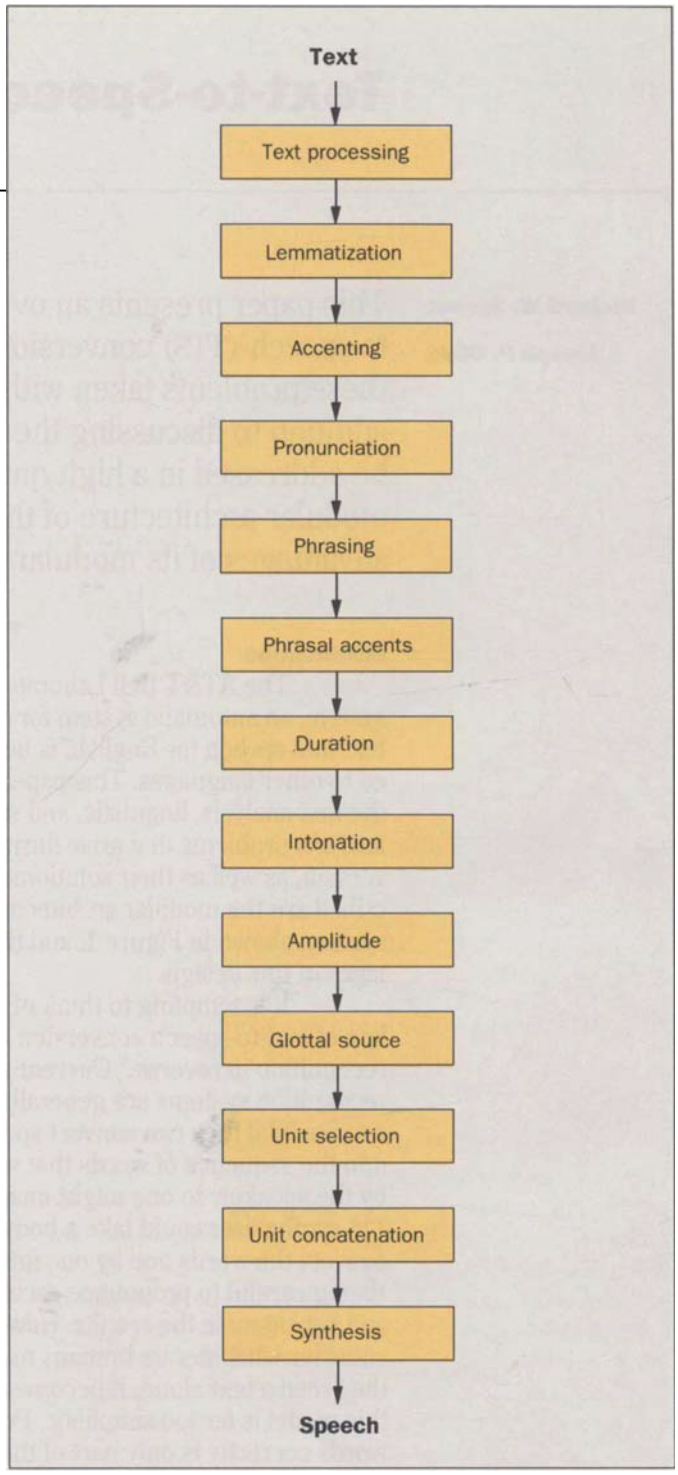

Figure 1. Modules of the English AT&T TTS system.

Panel 1. Abbreviations, Acronyms, and Terms

ASCII—American Standard Code for Information Interchange
F0—fundamental frequency
grapheme—a basic unit of a writing system
IPEX—interactive prosody editor running under X windows
LPC—linear predictive coding
phoneme—a speech sound
TTS—text-to-speech

sequence occurs as part of *232-3142*— a likely telephone number—it would normally be read as *two three two*.

Tokenization into words in the English system is mostly trivial. In many Asian languages, by contrast, where spaces are not used to delimit words, the process is not so simple. In a text-to-speech system for these types of languages, such as the Mandarin system being developed at AT&T, it is generally necessary to "reconstruct" the word boundary information that is missing in the orthographic representation of the input. A minimal requirement for this task is a large on-line dictionary. This is not enough on its own, however, because many words are not in the dictionary. These include personal names, foreign names in transliteration, and morphological derivatives, that is, derivatives of words in the dictionary that have been changed by the addition of a suffix or prefix. Special models must be built for these non-dictionary words.

After words have been tokenized and abbreviations and numbers have been expanded, the front end of the English AT&T TTS system assigns grammatical parts of speech using an algorithm developed by Kenneth Church.[2] The sequence *they can can cans* will be assigned the parts-of-speech sequence *PluralPronoun Modal Verb PluralNoun*. Church's algorithm is a probabilistic method that computes the most likely analysis of a sequence of words, maximizing the product of the *lexical probabilities* of the words in the sentence (the possible parts of speech of each word and their likelihoods) and the *trigram probabilities* (probabilities of triplets of parts of speech).

Accentuation. In languages like English, various words in a sentence are associated with *accents*, which are usually manifested as upward or downward movements of fundamental frequency, or *F0*. Usually, not every word in the sentence bears an accent, however, and the decision about which words should be accented and which ones should be unaccented is a problem that must be addressed during text analysis. The English version of the AT&T TTS system distinguishes three levels of prominence. Two are accented and unaccented, as just described, and the third is *cliticized*. Cliticized words are unaccented, but they also can be analyzed as having lost their word stress, so that they tend to be short in duration.

A good first step in assigning accents is to determine accents based on broad lexical categories or parts of speech. Content words—nouns, verbs, adjectives and, perhaps, adverbs—tend, in general, to be accented; function words, including auxiliary verbs and prepositions, tend to be deaccented; and short function words tend to be cliticized. Accenting, however, has a wider function than merely communicating lexical category distinctions between words. In English, complex noun phrases—roughly, a noun preceded by one or more modifiers[3]—are an important set of constructions whose accent is difficult to predict. In a "neutral" context, some noun phrases are accented on the final word (*Madison* **Avenue**), some on the penultimate (**Wall** *Street*, *kitchen* **towel** *rack*), and some on an even earlier word (**sump** *pump factory*). The assignment of accent depends on syntactic and complex lexical and semantic factors. So, while compounds consisting of two nouns often deaccent the second member, cases where the first noun denotes a material out of which the object denoted by the second noun is made (*iron* **bar**) frequently accent the second noun.

Accenting is not only sensitive to syntactic structure and semantics, but also to properties of the discourse.[4] One straightforward effect is *contrast*, as in the example, "I didn't ask for **cherry** *pie*, I asked for **apple** *pie*." For most speakers, the "discourse neutral" accenting of pie terms would place the main accent on *pie*. In this example, however, the clear intention is to contrast the ingredients in the pies, and *pie* is thus deaccented to emphasize the contrast between *cherry* and *apple*. While human-like accenting capabilities are possible in many cases, there are still some intractable problems. For example, just as a speaker would often deaccent a word that had been previously mentioned, so would one often deaccent a word if a supercategory of that word had been mentioned: *My son wants a* **Labrador**, *but I'm allergic to dogs*. Handling such cases in any general way is beyond the capabilities of current TTS systems.

Word Pronunciation. The next stage of analysis involves computing pronunciations for the words in the input, given the orthographic representation of those words. The simplest approach is to have a set of "letter-to-sound" rules that map sequences of graphemes into sequences of phonemes, along with possible diacritic information, such as stress placement. This approach is best suited to languages where there is a simple relation

between the way a word is spelled and how it is pronounced. Languages such as Spanish and Finnish fall into this category. However, languages like English manifestly do not, so a highly accurate word pronunciation module must contain a dictionary of pronunciations that, at the least, records words whose pronunciation cannot be predicted based on general rules. However, having a dictionary that is merely a list of words presents us with familiar problems of coverage: many text words occur that are not to be found in the dictionary, including morphological derivatives from known words, or previously unseen personal names. For a word that is a morphological derivative, standard techniques of morphological decomposition can be used. The pronunciation of the whole can then be computed from the (presumably known) pronunciation of the morphological parts, applying appropriate phonological rules of the language. Additional mechanisms may be necessary for novel personal names.

One such method used in the English version of the AT&T TTS system involves computing the pronunciation of a new name by analogy with the pronunciation of a similar name. For instance, if we have the name *Califano* in our dictionary, then we can derive the pronunciation of a hypothetical name *Balifano* by noting that both names share the "suffix" *alifano*. The pronunciation of *Balifano* can then be computed by removing the initial *phoneme* /k/ of *Califano*, and replacing it with the phoneme /b/. A further method, used as a fallback option if morphology and analogical reasoning fail, is to attempt to deduce the language of origin of the name—using statistics on the distribution of letter trigrams in names from various languages—and then apply a set of language-specific pronunciation rules.

Some word forms are inherently ambiguous in pronunciation. For these, a word pronunciation module, such as the one described here, can only return a *set* of possible pronunciations, from which one must be selected. To pick a hard example, consider that the string *bass* is most likely to be pronounced /bæs/ in a "fishy" context like "*He was fishing for bass*," but /beɪs/ in a musical context like "*He plays bass*." Without doing a deep semantic analysis of the text, we can nonetheless accurately guess which kind of context we have by considering the words that accompany *bass* in the sentence. Words such as *fish* or *boat* are clues to the "fishy" context, whereas words such as *play* or *orchestra* indicate a musical context. Homographs are handled accurately in the English TTS system by a statistical method developed by David Yarowsky. This method uses both local (n-gram) and wider-context information to predict the most likely pronunciation.

Intonational Phrasing. When reading a long sentence, speakers will typically break the sentence up into several phrases, each of which can be said to "stand alone" as an intonational unit. If punctuation is used liberally enough to leave only a few words between the commas, semicolons, or periods, then a reasonable guess at an appropriate phrasing would be simply to break the sentence at the punctuation marks (though this is not always appropriate). The real problem emerges when long stretches occur without punctuation. In these cases, human readers would normally break the string of words into phrases, determining by context where to place these breaks. For the TTS system, the simplest approach is to have a list of words, typically function words, that are likely indicators of good places to break. One has to use some caution, however. While a particular function word like *and* may coincide with a plausible phrase break in some cases (*He got out of the car and walked towards the house*), in other examples it might coincide with a particularly *poor* place to break (*I was forced to sit through a dog and pony show that lasted most of Wednesday afternoon*).

Approaches to intonational phrasing have been proposed, including methods that depend on syntactic parsers with varying degrees of sophistication. The method used in the AT&T system[5] uses a decision tree model that is trained on a corpus of text annotated with prosodic phrase-boundary information. The current phrasing system predicts only one level of intra-sentential phrasing: a sentence (a *major phrase*, or in standard intonational terminology, an *utterance*) is chunked into one or more *minor phrases* (in intonational terminology, an *intonational phrase*). The computation of finer grained divisions into *intermediate phrases* is a topic for future work.

Segmental Durations. After the TTS system has computed which phonemes are to be produced, it must decide how long to make each one. What duration to assign to a phonemic segment depends on many factors, including the identity of the segment in question, the stress of the syllable of which the segment is a part, whether that syllable bears an accent, and the position of

the segment in the phrase. Various approaches have been taken to modeling segmental durations in TTS systems. A method used in previous versions of the AT&T TTS system involves *duration rules*, which are rules of the form "if the segment is *X* and it is in phrase-final position, then lengthen *X* by *n* milliseconds."[6] In rule-based systems of this kind, it is not unusual for the duration of a given segment to be rewritten several times as the conditions for the application of various rules are considered. The rule-based approach can be formalized explicitly in terms of the second approach—*duration models*—which are mathematical expressions that prescribe how to use the various conditioning factors to compute the duration of a segment. The successive application of the rules can, in effect, be "compiled" into a single mathematical expression that implements the combined effect of the rules. As argued by Jan van Santen, all extant duration models can be viewed as instances of a more general *sum-of-products* model, in which the duration of a segment is predicted by summing over the products of scaled factors, where each factor is some computable property of the segment.

Rather than deciding, a priori, on a particular sums-of-products model (or set of such models) within the space of all possible models, AT&T researchers handle segmental duration by using exploratory data analysis to arrive at models whose predictions show a good fit to durations found in a corpus of labeled speech (see Reference 7). (Ideally, this corpus should be designed to account for most factors and their combinations that are likely to be relevant to duration. The design of such a corpus is not a trivial task.) In general, some specific duration models may be better suited than others to different sets of conditions. For example, in the English duration system, consonants that occur between vowels are associated with a different sums-of-products model than are consonants that occur in clusters. In the actual implementation of segmental duration predictions, a decision tree is used to determine, on the basis of contextual factors appropriate to the segment at hand, what particular sums-of-products model to use. This model is then used to compute the duration of the segment.

Intonation. Intonation contours for the English AT&T TTS system are computed by an algorithm described more fully in Reference 8. As input, the intonation module needs the following information, which is provided by previous modules:

- Which syllables in the utterance are to be accented, as computed by the accenting module (computing which words are to be accented) and the word pronunciation module (computing which syllable in the word has primary stress and, therefore, ultimately bears the accent).
- Which *types* of accent (e.g., *low* or *high)* are to be used, as well as the types of initial and final *boundary tones* and *phrase accents* (accents associated with the phrase as a whole). This information is computed in a phrasal accentuation module not described here.
- The durations of all segments in the utterance, as computed by the segmental duration module.

During the first stage in the computation of an intonation contour, a tone-timing function sets up nominal times for each accent in the sentence. Separate routines are invoked for initial boundary tones, final boundary tones, pitch accents, and phrase accents. Roughly, initial boundary tones are aligned with the silence that is placed at the beginning of each minor phrase, whereas final boundary tones are aligned with the final vowel of each minor phrase. Phrase accents are aligned after the final word accent of the minor phrase, if there is one, or at the end of the first vowel of the first word, or at the end of the first phoneme. Finally, accents on words are aligned with their associated syllables using a complex set of contextual factors.

These nominal accent times are then converted into actual F0/time pairs using another function. F0 values are computed according to the prominence of the accent (either determined automatically, or else definable by the user), and various phrasal parameters from the intonation model, as well as the particular type of accent involved. These phrasal parameters include values for the *topline*, *baseline*, and *reference line*. Basically, the topline sets a (soft) upper limit for a speaker's *pitch range*; the baseline sets a (soft) lower limit; and the reference line represents an F0 value intermediate between the topline and the baseline. This F0 value is used as a reference point from which to compute F0 values for high tones, which generally occur between the reference and topline, and low tones, which generally occur between the reference and baseline. The F0/time pairs generated can be viewed using the interactive prosody editor running under X windows (IPEX), described later in this paper.

Finally, an F0 contour is produced by interpolating the computed F0/time pairs, and smoothing via con-

volution with a rectangular window. As with the F0/time values, the generated contour can be viewed in IPEX.

Speech Synthesis

Once the text has been transformed into phonemes, and their associated durations and a fundamental frequency contour have been computed, the system is ready to compute the speech parameters for synthesis. The AT&T TTS system uses a concatenative approach. Parametrized short speech segments of natural speech are connected to form a representation of the synthetic speech. The majority of the natural speech segments are merely transitions between pairs of phonemes. However, because of the large contextual variation of some phonemes, segments consisting of three or more phoneme elements are often necessary. Such elements consist of the transition from the first phoneme to the second, and a transition from the penultimate phoneme to the last, but the intermediate phonemes are stored completely. Approximately 2,900 different speech elements—also called *dyads*—also exist in the present acoustic inventory for English. These elements contain all the legal phoneme combinations for English.

The rules for choosing the appropriate elements of the acoustic inventory and connecting them in a suitable way work in two stages. The first stage converts the phonemic representation into a representation that contains the inventory elements to be used. The second stage retrieves, connects, and interpolates the parameters. Before describing these two tasks, however, this paper presents some background on the theory of speech production, on which speech synthesis depends.

Parametric Representation of Speech. The concatenative approach to speech synthesis requires that speech samples be stored in some parametric representation that will be suitable for connecting the segments and independently changing the signal's characteristics of loudness, pitch, and spectrum. One method for changing the characteristics of natural speech is to analyze the speech in terms of a source/filter model, as shown in Figure 2.

This model of speech synthesis has a variety of independent input controls. At the left side of Figure 2, two possible source generators are shown: a noise generator and a simple pulse generator. The noise generator has no controlling input. The pulse generator, however, is controlled by the F0 parameter, which specifies the distance between any two pulses, thereby controlling the F0 of the periodic source. A switch controlled by a voicing flag selects these inputs. It is also possible to have a mixer to control the relative contribution of the noise and pulse source, and to insert a glottal pulse with additional controls for the shape of the glottal source in place of the simple pulse generator. To the right of the switch, a multiplier multiplies the source using an amplitude parameter, which controls the loudness of the system. The signal from the multiplier is fed into a filter controlled by the filter coefficients, which are varied slowly to shape the speech spectrum.

The source/filter model can be used to replicate naturally spoken speech when the parameters are obtained by analyzing natural speech. Speech can be parametrized by an amplitude control, voiced/voiceless flag, F0, and filter coefficients at a small interval (about 5 to 15 milliseconds [ms]). The loudness control is simply the power of the speech during the time frame of the analysis. Pitch extraction algorithms determine whether the speech is voiced or not, as well as its fundamental frequency. The filter parameters can be determined using various analysis techniques. Using parameters obtained from the analysis, the source filter model can reproduce the analyzed speech. However, these parameters can also be varied independently to change the speech. The ability to alter the analysis parameters is crucial to a concatenative approach, where the spectral parameters have to be smoothed and interpolated whenever two elements from different utterances are connected, or when the duration of the speech has to be altered. During synthesis, the fundamental frequency of the original speech is completely discarded and replaced by a rule-generated F0, as this paper described earlier.

In this case, linear predictive coding (LPC) was chosen as the parametric representation. However, AT&T researchers are experimenting with various other representations, such as formants, waveform interpolation, and articulatory parameters. In addition, a more complex source than just a simple pulse is used; the source model permits control of the spectral balance and degree of aspiration in the voiced portion of the speech.

Converting Phonemes to Acoustic Inventory Elements. The process of translating a phonemic text to a list of acoustic inventory elements consists of a set of rewriting rules that depend on the nature of the acoustic inventory. Generally, the process tries to match the longest possible

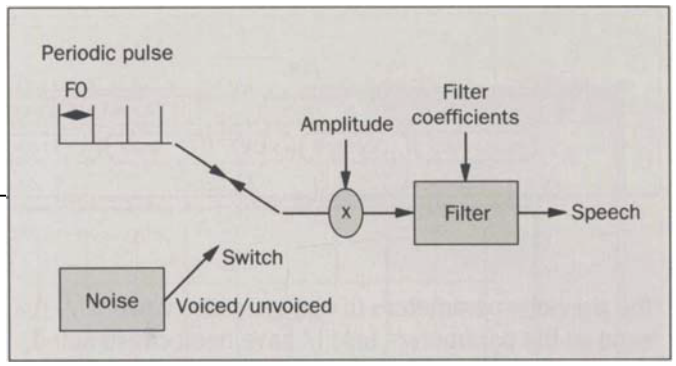

Figure 2. The source-filter model for speech synthesis has a variety of independent input controls. Shown at left are two possible source generators: a noise generator and a simple pulse generator. While the noise generator has no controlling input, the pulse generator is controlled by the F0 parameter, which specifies the distance between any two pulses, thus controlling the fundamental frequency of the periodic source. A switch controlled by a voicing flag selects these inputs. To the right of the switch, a multiplier multiplies the source using an amplitude parameter. The signal from the multiplier is fed into a filter controlled by the filter coefficients, which are varied slowly to shape the speech spectrum.

input phoneme string to an element in the inventory. To explain how the process works, the acoustic inventory must be described first. Many of the elements in the acoustic inventory consist of transitions between two phonemes. However, not all pairs of phonemes are stored in the inventory. For example, the transitions between fricatives and stops, or between two fricatives, are not stored. Also many highly coarticulated phonemes are stored within polyphonic units. In addition, an inventory element consisting of their steady state exists for all fricatives; the element is labeled by a single phoneme.

The elements are retrieved from the inventory as shown in the English example "We are in the test." The input to the dyad-selection module contains the phoneme string

/* w i a r ɪ n ð ə t ɛ s t */,

expanded to include the silence element '*' at the beginning and end. Starting from the left, the algorithm will find the elements /*-w/ and /w-i/ in the dyad table. For the next element, the table contains dyads for both /i-a/ and /i-a-r/. Because the longer three-phoneme element matches the input string, it is the one that is chosen. Next, the algorithm searches for the longest string beginning with the phoneme /a/. If /a-r-ɪ/ were found (overlapping with /i-a-r/), it would be used; but in fact the only such element matching the input is /a-r/, which is already covered by /i-a-r/, and is therefore discarded. The algorithm continues from left to right in this fashion, finding the longest matches against the input. For the sample sentence, the selection will result in the sequence of elements:

/*/ /*-w/ /w-i/ /i-a-r/ /r-ɪ/ /ɪ-n/ /n-ð-ə-t/ /*/
/t-ɛ/ /ɛ-s/ /s/ /s-*/ /*/ /t-*/ /*/.

Some phonetic classes, such as stops, require special treatment. Stops contain two distinct sections, a closure and a burst. The inventory element /t-ɛ/ consists of the burst portion of the /t/ and the transition from /t/ to /ɛ/. Because an unvoiced closure is simply a silence, the silence element /*/ is inserted before the /t-ɛ/ element. Fricatives, such as /s/, also require special treatment. The elements /ɛ-s/ and /s-*/ contain transitions from /ɛ/ to /s/ and from /s/ to silence. To get a high-quality fricative, a steady-state element (/s/ in this case) is inserted between the elements /ɛ-s/ and /s-*/. There is no /s-t/ element in the table, so the algorithm inserts a silence of zero duration between the two phonemes. In general, whenever a desired /a-b/ element—where /a/ and /b/ are any phonemes—does not exist, the rewrite rules substitute the pair /a-*/ and /*-b/ for the missing element. Thus, the /s-t/ element is configured from the two elements /s-*/ and /*-t/. However, because the stop /t/ begins with a silent interval to simulate the closure, /*/ is used instead of /*-t/.

Converting Parameters to Acoustic Inventory Elements. The next component retrieves the desired acoustic elements from the acoustic inventory table—using the element name as an index—and prepares them for synthesis. This process depends on three types of information: the inventory element description; the durations associated with each phoneme; and the fundamental frequency values for every 5 ms, as generated by the intonation module described earlier. After the elements are retrieved, they are connected by an appropriate interpolation procedure, adjusting the duration of each phoneme, and an F0 value is imposed on each period of voiced speech.

When the parameters of a given phoneme have been computed, the algorithm saves the unused parameters already retrieved from the previous acoustic element and connects them to the parameters obtained from reading the next element. Thus, in our previous example, after the phoneme /w/ has been synthesized from the acoustic elements /*-w/ and /w-i/, the parameters remaining are used for the opening transition of the vowel /i/. Next, the element /i-a-r/ is read, and the parameters for the closing transition of the /i/ are used with

the previous parameters to construct the vowel /i/. As soon as the parameters for /i/ have been constructed, the vowel /a/ is available in its entirety and can be used for synthesis. After the vowel is synthesized, the algorithm still retains the parameters for the oncoming transition of the /r/.

Setting the parameters for a phoneme requires adjusting them to fit the duration of the phoneme previously computed by the duration module. In the case of phonemes retrieved from the middle of a multiphone inventory element (as the /a/ in the /i-a-r/ element), the parameters are linearly interpolated so that the duration of the phoneme is correct. If the phoneme contains parameters from two different acoustic inventory elements, the algorithm uses strategies to connect the parameters and adjust the durations of the phonemes. Basically, the interpolations can be grouped into two types: one type connects the parameters and adjusts the duration with added parameters between the incoming and outgoing transitions by interpolating between two adjacent endpoints. The other type connects the two segments (opening and closing transitions) directly and adjusts the duration of the phoneme by interpolating the parameters of the entire phoneme.

After the phoneme parameters have been computed and the duration of the phoneme has been adjusted, the F0 data is included in the parametric representation and a waveform is calculated for the phoneme. The waveform is computed as it would be for an ordinary LPC synthesis scheme.

TTS Architecture

The English version of AT&T TTS (see Figure 1) contains 13 modules, each of which is responsible for part of the problem of converting text into speech. These modules are:

- Text processing;
- Lemmatization of word forms;
- Accentuation;
- Word pronunciation;
- Intonational phrase boundary assignment;
- Phrase accent assignment, including placement of boundary tones;
- Segmental duration;
- Intonation;
- Amplitude;
- Glottal source;
- Dyad selection;
- Dyad concatenation; and
- Synthesis.

The dyad-concatenation module outputs a stream of LPC parameters and source-state information for the waveform synthesizer; the synthesis module reads the LPC and source-state stream. Except for these, each module in the pipeline communicates with the next using a uniform set of data structures. (The individual programs can also be compiled into a single executable program.) Information is passed along on a sentence-by-sentence basis, and consists of a set of arrays of linguistic structures. Each array is preceded by the following information:

- The structure's type, *T*: for example, word, syllable, phrase, phoneme, dyad, etc.;
- *N*, the number of structures of type *T* for this sentence; and
- *S*, the size of an individual structure of type *T*.

Each header is followed by an *N* x *S* byte array consisting of *N* structures of type *T*.

Every module in the pipeline reads in the structures a sentence at a time using a library function, performs some processing on the input, and then writes out the structures (again using a library function) for the next module. The flow of information from module to module is basically monotonic, that is, a module typically just adds information. For example, the accent module adds accent structures to the data stream.

This architecture has a number of advantages over a more monolithic approach. The first advantage is a standard observation: If the system's modules share a common interface, then it is easy for several researchers to work on different modules independently of one another, as long as they agree on the input/output behavior of each module and the manner in which the modules should relate to one another. In this regard, the architecture has already proven fruitful, allowing new modules to be easily added to our English system. These include work on accentuation and segmental duration, as well as adding or modifying some modules with wider applicability, such as the waveform synthesizer module.

Second, the pipeline design makes it simple to stop processing at any point in the pipeline. During the

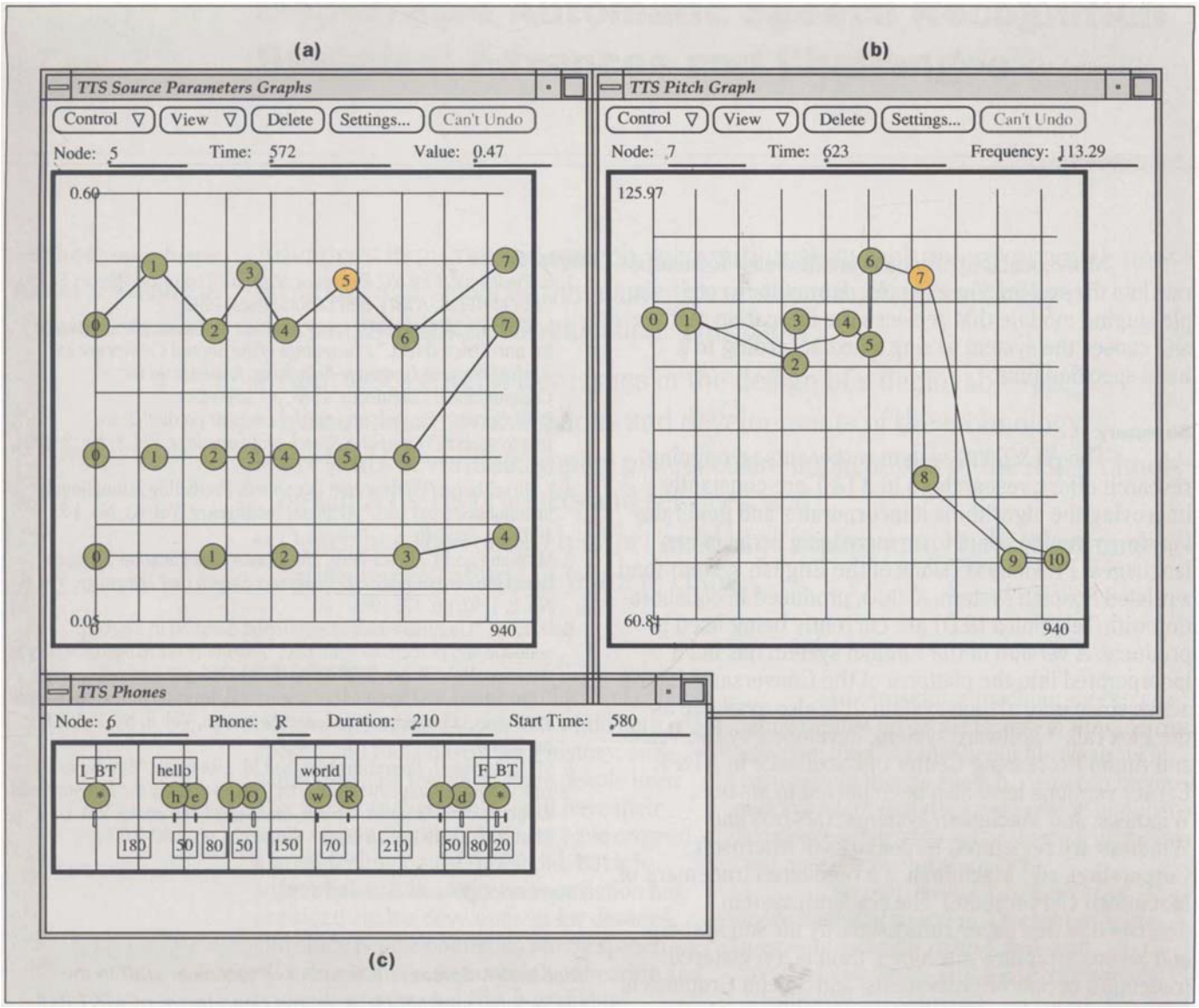

Figure 3. A partial IPEX display for the input sentence, "*Hello, world.*" Depicted are the (a) *glottal source* parameters, consisting of (from top to bottom) *glottal open quotient*, *spectral tilt*, *aspiration noise*; (b) *F0*; and (c) *phonemes and durations*. The circles in the top two panels represent computed target values for the contours in question, and these can be interactively manipulated. For example, the highlighted node labeled "7" in (b) represents one of the targets associated with the H accent assigned to *world* (as computed by the intonation module). This target can be raised or lowered in pitch, or moved earlier or later in time as desired. Target nodes can also be added or deleted, resulting in modifications of the curves. The circles in (c) represent individual phonemes; these can be selected and changed, or moved to decrease or increase their segmental duration.

design of a segmental duration module, for example, a user might want to test the system on a large corpus of text and sample the segmental durations that are computed. For that task, a user would not normally care about what happens in modules subsequent to duration, such as intonation, amplitude, or dyad selection. In the AT&T TTS architecture, all modules, up to and including the duration module, can be run, and then the pipeline can be terminated with a program that prints out information about segments and their durations. Similarly, programs that modify TTS parameters in various ways can be easily inserted into the pipeline. One particularly useful program is the graphical interface tool called IPEX. IPEX fits into the pipeline after the glottal source module, and allows interactive manipulation of segments, segmental durations, pitch contours, three glottal source parameters, and amplitude contours. Figure 3 shows an example of some IPEX display panels.

More jocular applications are also easy to incorporate into the system. For example, one author wrote a simple singing module that replaces the intonation module and causes the system to sing a text according to a hand-specified tune.

Summary

The AT&T TTS system represents an ongoing research effort; researchers in AT&T are constantly improving the algorithms it incorporates and generalizing the system to adapt to an increasing array of new languages. Previous versions of the English system (and a related Spanish system, AMIGO, produced in collaboration with Telefónica I&D) are currently being used in products. A version of the English system has been incorporated into the platform of the Conversant® interactive voice information system. It is also available as the FlexTalk™ software system, developed by the Voice and Audio Processing Center of Excellence in AT&T. Earlier versions have also been ported to MS-DOS, Windows, and Macintosh systems. (MS-DOS and Windows are registered trademarks of Microsoft Corporation, and Macintosh is a registered trademark of Macintosh Corporation.) The research system described in this paper runs primarily on Sun systems and Silicon Graphics machines. (Sun is a registered trademark of Sun Microsystems, and Silicon Graphics is a registered trademark of Silicon Graphics Inc.)

Acknowledgments

Many people have helped develop the current version of the AT&T TTS system. In addition to those who have worked on various modules of the English version—and who have been cited in the text—several other people have already helped with the implementation. Of these, we particularly thank Mark Beutnagel, Luis Oliveira, James Rowley, David Talkin, and Michael Tanenblatt, who also wrote IPEX. The work on IPEX also benefited from advice from Doug Blewett, on whose *xtent* system IPEX is based. Michael Riley provided much useful discussion about software design issues. Jan van Santen helped with the section on segmental duration. Beutnagel and Tanenblatt were primarily responsible for the MS-DOS and Macintosh ports.

References

1. P. Pavlovcik, *FREND Reference Manual*, Technical Report 55430-91-0711-01TM, AT&T Bell Laboratories, 1991.
2. K. Church, "A stochastic parts program and noun phrase parser for unrestricted text," *Proceedings of the Second Conference on Applied Natural Language Processing*, Association for Computational Linguistics, 1988, pp. 136-143.
3. R. W. Sproat, "English noun-phrase accent prediction for text-to-speech," *Computer Speech and Language*, Vol. 8, No. 2, 1994, pp. 79-94.
4. J. Hirschberg, "Pitch accent in context: Predicting intonational prominence from text," *Artificial Intelligence*, Vol. 63, No. 1-2, 1993, pp. 305-340.
5. M. Wang and J. Hirschberg, "Automatic classification of intonational phrase boundaries," *Computer Speech and Language*, Vol. 6, No. 2, 1992, pp. 175-196.
6. D. Klatt, "Linguistic uses of segmental duration in English: acoustic and perceptual evidence," *Journal of the Acoustic Society of America*, Vol. 59, No. 3, 1976, pp. 1209-1221.
7. J. van Santen, "Assignment of segmental duration in text-to-speech synthesis," *Computer Speech and Language*, Vol. 8, No. 2, 1994, pp. 95-128.
8. M. Anderson, J. Pierrehumbert, and M. Liberman, "Synthesis by rule of English intonation patterns," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, March 19-21, 1984, San Diego, California, pp. 2.8.1-2.8.4.

(Manuscript approved December 1994)

Richard W. Sproat *is a member of technical staff in the Linguistics Research Department of AT&T Bell Laboratories in Murray Hill, New Jersey. He is responsible for text-to-speech synthesis, text analysis, and natural language processing. Mr. Sproat joined AT&T in 1985 after receiving a Ph.D.in linguistics from the Massachusetts Institute of Technology, Cambridge.*

Joseph P. Olive *is a supervisor in the Linguistics Research Department of AT&T Bell Laboratories in Murray Hill, New Jersey. He is responsible for text-to-speech synthesis. Mr. Olive received a B.S., M.S., and Ph.D. in physics, and an M.A. in music, from the University of Chicago in Illinois. He joined AT&T in 1969.*