CHATR: a generic speech synthesis system

Alan W Black and Paul Taylor
ATR Interpreting Telecommunications Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, JAPAN
awb@itl.atr.co.jp or pault@cogsci.ed.ac.uk

Abstract

This paper describes a generic speech synthesis system called CHATR which is being developed at ATR. CHATR is designed in a modular way, module parameters and even which modules are actually used may be set and selected at run-time. Although some interdependencies exist between modules, CHATR offers a useful research tool in which functionally equivalent modules may be easily compared. It also acts as a simple system for those less interested in the internals of speech synthesis but just wish their computer to talk.
Topic: speech synthesis, generic systems.

Introduction

There are many requirements for a speech synthesis system, in addition to high quality natural sounding speech output, the system should be flexible and not simply be hard-wired. For example it should at least be the case that new words can be added to the lexicon. Other more general changes should also be possible e.g. specification of new intonational tunes, varying of output voices, choice of phoneme set to be used (e.g. if a different lexicon is to be used), and even the choice of language being spoken. A researcher requires access to internal structures, ability to mix and match techniques, graphical display of utterances and compatibility with other systems. But, those who are uninterested in the internals of speech synthesis, just want their computer to talk. To them the requirements of a synthesis system are different, although they still want a degree of control over synthesis, real-time production of speech, machine independence, and ease of use are the factors that are most important. As a well-engineered system CHATR meets these requirements.

Because ATR's main speech project is in the area of speech translation systems, input to CHATR can be much richer than simple plain text. During translation, utterances are represented in a rich structure including syntactic, semantic and speech act information. Unlike a conventional text-to-speech system which needs to reconstruct this information from raw text, CHATR can use this explicit information directly and hence produce more accurate synthesis. Although CHATR does also support text-to-speech, current development has concentrated on the use of labelled input rather than raw text.

CHATR is designed in a modular way so that functionally equivalent modules may exist within the system. Flow of control may be selected at run time, without recompilation. Within a speech synthesis research environment this is useful as it allows close comparison of components to identify differences. Thus equivalent modules may be tested within exactly the same environment interactively. For example, CHATR currently supports a number of different low-level (waveform) synthesizers. This process is quite independent of intonation or duration modules. CHATR's modularity allows synthesis of exactly same utterance through different waveform synthesizers.

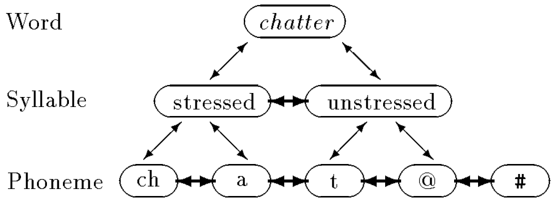
The next section discusses the internal representation of utterances within CHATR. Then the overall structure of the system is discussed with some typical modules described. Finally the current configuration of CHATR is described detailing its actual modules. Also some discussion is given about the shortcomings of the system and how we would like to see it improved.

Utterance representation

In conventional speech synthesis systems (such as MITalk [2]) a "pipeline" architecture is often used. Information is passed through a pipeline of modules. Each module defines what information is passed on to succeeding modules. But, if an earlier component does not pass on information which is later found to be needed down stream, all intermediate modules will need to be re-written to pass on that information. In contrast, CHATR uses a single "blackboard" representation for all aspects of an utterance. All modules have access to all parts. Although global, more than one utterance object may exist in the system at anytime.

There are effectively two types of module which act on utterance objects. Synthesis modules will typically modify the contents of an utterance based on its current content (and other parameters). Other modules also exist which are more general in nature, such as graphically displaying an utterance, saving its contents or playing the synthesized waveform.

An *utterance object* consists of a number of *streams*. Each *stream* consists of an ordered list of *cells*. The number of streams can easily be changed in CHATR and not all streams need exist in all utterances. Typical streams are: words, syllables, and phonemes. Relations may be set between stream cells and so, for example, it is possible to find which word a syllable is in. The following diagram shows a typical stream structure for part of an utterance object.



Each stream cell is linked to its preceding and succeeding cells. Cells contain all the appropriate information for that type of stream. For example our phoneme cell ultimately contains a name, phonetic features, a duration etc.

Note that although there will be hierarchical structure between streams this is not mandatory (e.g. the silence '#' phoneme above is not part of any syllable or word). For example in a treatment of intonation implemented within CHATR (based on [6]) the cells in the intonation stream are linked to syllables but no direct hierarchical relationship exists between intonation cells and phonemes.

The existing streams could even be ignored and others introduced if the current ones are inappropriate to some synthesis task. For example a different intonation model may require quite different streams from the Taylor model currently implemented. Streams must be defined at compile time but may be selected per utterance at synthesis time—that is, defining many different streams does not impinge on the size or efficiency of the utterance structures built.

Levels of input

CHATR offers input an many levels. At the most abstract it can accept linguistic descriptions of utterances from which it can generate prosodic phrasing and intonational tune through a rule driven process (described in [3]). Alternatively, input may explicitly include prosodic phrasing and intonational features specifying tune. This second level allows much more explicit control over phrasing and intonation. A third level allows even more degree of control specifying individual phonemes, durations and F_0 target values (or a slightly higher symbolic description of F_0). At the lowest level, waveforms themselves can be specified allowing CHATR to generate any arbitrary sound. These differing levels of input allow a user of CHATR to specify the form of an utterance in as much detail as is desired.

Multiple levels of input are useful in synthesis research. For example, naturally occurring durations and/or pitch may be explicitly specified in the input, allowing exact control over parts of the synthesis, thus emphasizing the other parts under investigation.

There is currently a fixed number of input types. Although new levels can easily be added, it would be better if an utterance may be specified at any level of precision in any stream and synthesis modules could be used to fill in missing parts. This has not yet been added to the system but some discussion of this is given below.

Overall structure

A command language based on Lisp is provided so the user can execute commands such as synthesis, play, set intonation statistics, define a phoneme set etc. The Lisp, although a full language, is designed merely for control rather than encoding speech synthesis algorithms. Lisp list structures are used to represent most of the ASCII data in the system (e.g. duration statistics, lexicons, phoneme set definitions etc.). This means that data can easily be changed and (re)loaded into the system. As all these files are s-expressions no new file i/o routines are required.

Flow of control, i.e. which modules are called, can also be specified in Lisp, thus, functionally equivalent modules may be selected between interactively at run time.

The system consists of one large executable which includes a number of different modules. Modules may be written in C or C++ (or in fact any other language if an interface to the stream and utterance structures is provided for that language). It may have been possible to write the whole CHATR system in Lisp (or Prolog or some other language designed for symbolic manipulation). This however was specifically decided against as in addition to the symbolic aspect of CHATR we also wish the signal processing aspects of speech synthesis to be efficient (and many such algorithms already exist in C). Although most Lisp systems support C interfaces they are typically non-standard and portability of the whole system was an important criterion.

Modules

A number of modules exist in the system but not all are used for the synthesis of all utterances. Utterance modules are those functions that are given a single utterance object as an argument. Typically they will access a number of streams and create (or modify) another stream. Selection of which modules get called is based on, the input type of the utterance, global options and the specified path.

Let us look at one typical module: the lexicon module. Our current lexicon module allows the construction and use of lexicons whose entries specify syllables, stress and phonemes for a given word (which is identified by a character string plus optional features). When the lexicon module is called the desired words are already set up in the word stream. The lexicon module looks up each word in the lexicon and creates the syllable and phoneme streams with the information found in the lexical entry (words not found can optionally be treated by letter-to-sound rules, ignored or cause synthesis to abort).

Some modules offer choices between functionally equivalent modules by simply setting global parameters. For example we have two modules which predict durations for phonemes. One is based on the Klatt duration rules in [2, Ch. 9], while the second is based on Campbell's work [4]. Selection between them is simply by a command of the form

```
chatr> (Parameter Duration_Method KLATT)
```

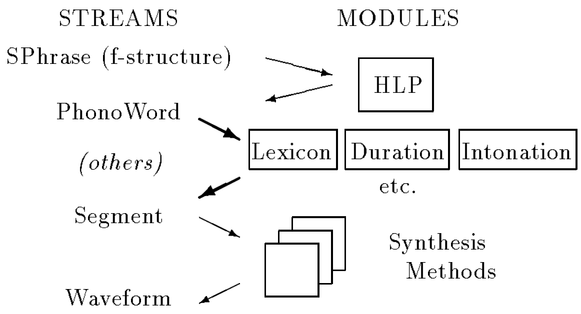
Another section where selection of equivalent modules is common is the low-level synthesis methods. We wish to allow comparison of different forms of waveform synthesis based on the same utterance. Currently, CHATR offers a number of synthesis methods: Klatt formant synthesis, LPC based diphone synthesis, and a number of concatenative synthesis methods (each with its own internal options to choose between different unit selection strategies). The same utterance (with the same segments, durations and intonation) can be resynthesized with a different waveform generation module allowing direct comparison between methods. New low-level synthesis methods can be easily added taking an utterance structure as a parameter and generating a waveform on return.

Certain other modules in CHATR are not directly part of the synthesis process. Audio output is provided for through a general module that plays the waveform stream of an utterance. We provide a number of mechanisms to do this. We wish CHATR to be independent of hardware so we offer support for *audio servers*. These are network transparent systems that allow access to audio hardware. In the same paradigm as X windows for graphics, audio servers offer a uniform access method for various audio devices such that waveforms (which internally described their encoding, byte order and sampling frequency) can be easily be played. We also offer command driven play routines to ensure CHATR will work on any machine with audio output.

Similarly a display module is offered that can graphically display an utterance's waveform, phonemes, words, syllables, intonation etc. Rather than incorporate a full graphical display mechanism in CHATR itself we offer interfaces to other systems with graphics capability. Currently we support two systems: Entropic's waves+ system and a free-software speech graphics package.

Example synthesis

As stated above CHATR offers many levels of synthesis but here we will discuss one particular configuration. One of the uses CHATR is put to is in a speech translation system. The translation part of the system generates syntactic and semantic trees (represented as feature structures) of the utterance to be spoken. This is used as the input to one of CHATR's input modes. The following diagram sketches the information flow



Specifically the input specifies speech act information and topic/focus information. A rule driven system translates this input to a lower level form. Prosodic phrasing is generated from syntactic structure and special features in the input. Intonation tune is generated based on speech act and topic markers. The result, held in the *PhonoWord stream* is then passed to lower levels. Words are looked up finding their syllable structure and default pronunciation. An intonation module generates F_0 target points based on the generated intonation features (and speaker specific intonation parameters). A duration module generates phoneme durations based on phoneme context and intonational features. All this low-level information is brought together in the *segment stream*. Depending on selection, one low-level synthesis module is then called to generate a waveform based on the information in the segment stream.

Using parameter setting to select the form of synthesis required means that CHATR can easily be used for multi-speaker synthesis, and also we hope for multi-language synthesis.

Implementation

CHATR is written in a mixture of ANSI C and C++. The core architecture is written in C, but perhaps C++ would be more suitable as the core objects (utterances and streams) fit well into the object-oriented paradigm. Other modules are written in C or C++ for reasons related to history as well as appropriateness. The Lisp command system is in fact a small Scheme interpreter written specially for CHATR. An interactive command line interface offers command line editing, history and completion for commands, their arguments, variables and file names. This interface makes the system significantly easier to use, though CHATR may also be used in batch mode.

The system runs on a number of different architectures (including those with different byte order) including Sun SPARCs, HPs, DECstations and 386BSD. It should port to any Unix system with an ANSI C and C++ compiler.

Discussion

One enhancement to the system currently being discussed is a much more formal definition of modules. A module has prerequisites and provides some result. It should be possible to explicitly declare these so that a module will only be invoked when the necessary prerequisites are met.

A much clearer way of dumping an utterance in a form which can be easily reloaded is also required. As we wish to allow CHATR to interface with other existing speech synthesis systems this may involve communication with a completely separate program. Being able to dump the full utterance structure and convert it to some alternative form for another program to operate on and then convert it back, reload and continue is something that would make CHATR much more useful in co-operating with other synthesis programs.

Complete freedom of development can sometimes be too general. Although as a system, CHATR does not restrict how modules interact, if we are to be able to compare similar sub-systems it is necessary that those sub-systems act on the same data. Hence most of our low-level synthesis methods actually work from the information in the segment stream. That is they take exactly the same input. Even when existing synthesizers are integrated into CHATR we encourage use of the segment stream as a common intermediate stage between high-level synthesis and low-level waveform generation.

Other laboratories are also aware of the problems of multiple synthesizers and require a common environment for their development. COMPOST [1] is one such system. Unlike CHATR it introduces a new language for high-level synthesis specification but like CHATR it offers a choice of low-level synthesizers that can be selected for each utterance.

CHATR's currently implemented features include: a well defined architecture, multiple types of input, choice of waveform synthesis methods, parameterized intonation features, two duration modules, abstract phoneme sets, a text-to-speech module, graphical displays and an utterance object inspector. Current expansion includes improving unit selection for concatenative synthesis and integrating ν-TALK ([5]) a Japanese non-uniform unit concatenative synthesis system, making CHATR into a multi-language synthesis system.

Thus CHATR may be used at many levels. First, simply as a black box speech synthesizer. Simple control of voice is possible and the text-to-speech component is adequate for many purposes. At a deeper level, CHATR can be used interactively, allowing experimentation with intonational features and rules, resynthsizing existing utterances with modified durations and pitch, building new unit databases, all without modification of C sources. At the deepest level CHATR may be used to develop new synthesis algorithms: unit selection strategies, new intonation modules etc, may easily be added, building cleanly on the existing architecture. In summary CHATR goes a fair way to meet our original criteria.

Acknowledgements: The authors wish to acknowledge the help and comments by Nick Campbell and Norio Higuchi and the members of Department 2.

References

[1] M. Alissali and G. Bailly. COMPOST: a client-server model for applications using text-to-speech systems. In *Proceedings of EUROSPEECH '93*, volume 3, pp 2095–2098, 1993.

[2] J. Allen, M. Hunnicut, and K. Klatt. *Text-to-speech: The MITalk system.* Cambridge University Press, Cambridge, UK., 1987.

[3] A. W. Black and P. Taylor. A framework for generating prosody from high level linguistic descriptions. In *Proceedings of the Acoustics Society of Japan*, pp 239–240, 3–8–5, Spring, 1994.

[4] N. Campbell. Syllable-based segmental duration. In G. Bailly and C. Benoit, eds, *Talking Machines*, pp 211–225. North-Holland, 1992.

[5] Y. Sagisaka, N. Kaiki, N. Iwahashi, and K. Mimura. ATR – ν-TALK speech synthesis

system. In *ICSLP 92*, volume 1, pp 483–486, 1992.

[6] P. Taylor. *A Phonetic Model of English Intonation*. PhD thesis, Edinburgh University, 1992.