Sci. J. Fac. Sci., Minufiya Univ., Vol. XVII (2003 : 333 - 360)

RULE-BASED ENGLISH TEXT TO PHONEME GENERATOR ONE PHASE APPROACH

Hany M. Harb, Ahmed T. Shehab El-Dean,

Abd El-Hameed M. Emarra, Osama Sheta

Received 6 / 8 / 2002

Abstract

This paper introduces English Text-To-Phoneme generator (ETTP) based on rules. The proposed generator consists of two components: phoneme generation rules and phoneme generation algorithm. Phoneme generation rules comprise a set of text to phoneme transformation rules that are based on the English pronunciation rules.

The phonemes output from the proposed generator are verified against the phonemes output from another premade tool. Applying the generated set of phonemes to a phoneme pronunciation tool, the sound is generated and its quality is compared to the qulaity of the sound generated using the MS Talk-It tool. The proposed phoneme generator is implemented using Visual C++ programming language.

يقدم هذا البحث مولدا للوحدات الصوتية (الفونيمات) للنص المكتوب باللغه الانجليزيه. و يعتمد هذا المولد على توليد الوحدات الصوتية في مرحلة واحدة باستخدام مجموعة من القواعد و خوارزم. و لقد ركز هذا البحث على صياغة و تطبيق قواعد لانتاج الوحدات الصوتية لكي يتم انتاج الموجات الصوتية المقابلة لها.

Introduction

As computer becomes more intelligent, speech synthesis becomes more important and many researches have payed great attention into this area **[1, 20, 21]**.

Speech processing can be classified into speech input to the computer, and speech output fr om the computer. **Speech input** to the computer means that the acoustic waveform is inputted to the computer. It involves two types of processing, speech recognition, and speech verification and identification. **Speech output** from the computer means that the acoustic waveform is outputted from the computer. Speech output from the computer is mainly called speech synthesis or **Text-To-Speech (TTS) [1, 6, 9, 13]**.

Speech synthesis is used as an external aid for some classes of disabled people. Also there are many other application of speech synthesis such as learning machines, talking books and toys, telecommunications services, language education, and guidance of operational procedures in production line **[1, 3, 9, 10, 19]**.

The general phases of English speech synthesis are text normalization, phoneme generation, and speech generation as shown in fig. 1 **[1, 5, 20]**.

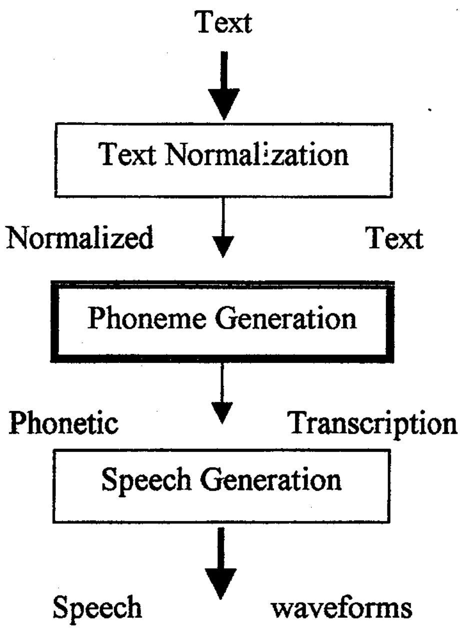

Figure 1: The general phases of **TTS** system.

Text normalization: the input text may contain a wide variety of abbreviations, symbols and numbers, so it is necessary to normalize the input text to produce a sequence of words instead of these abbreviations, symbols and numbers.

The following is an example of input and output of the text normalization phase.

Input: Dr. Harb has $1,234,567

Output: Doctor Harb has one million two hundred thirty four thousand five hundred sixty seven dollars.

Phoneme generation: it is the main phase in the speech synthesis process; it concerns with the transformation of normalized letters into the corresponding phonatic transcription. Speech synthesis methods can be classified according to the phoneme generation process into two strategies: dictionary_based synthesis, and rule_based synthesis. Dictionary_based synthesis depends on dictionary_based phoneme generation while rule_based synthesis depends on rule-based phoneme generation [**1, 6, 9, 11, 14, 16, 20**].

In **dictionary_based phoneme generation,** phonological knowledge, basically morphemes, is stored into a dictionary and the pronunciation is performed by inflectional, derivational, and compounding morphems taking into consideration the morphemic constituents and its effect on the phonetic transcription. Examples of this strategy are **MITTalk** and **AT&T TTS** systems **[1,20]**. In **MITTalk** 12000 morphemes are stored into a dictionary to cover 95% of the input words. In **AT&T TTS** system 43000 morphemes are stored into a dictionary .

In **rule_based phoneme generation**, phonological knowledge is transformed into a set of rules (sound/phoneme rules) and the special pronounced words are stored in an exceptions dictionary. The exceptions dictionary contains a limited number of words, for example in English 20000 words covers 70% of the input text. The quality of the generated sound depends on the quality of the generated phonemes used for sound synthesis. The quality of the generated phonemes in turn depends on the feature parameters and the phoneme generation rules. Phoneme generation rules must be based on the linguistic characteristics of natural speech so the construction of rules requires an extensive knowledge and a deep understanding of the speech production and perception process for a particular language. Also the rules must take into account all perceptually relevant acoustic changes that phonetic segments undergo in different context [**7, 8, 11, 12, 14, 17, 21**].

Speech generation: it is the final phase in the speech synthesis process and it concerns with the generation of the speech signals from the corresponding stream of phonemes and applying the generated waveforms to a loud speaker. Also speech generation phase takes speech characteristics such as prosody into consideration to ensure appropriate rhythm, tempo, accent, stress, and intonation [**20, 21**].

Recently, efforts are directed into designing sets of rules with a wide coverage of the input text and with hiqh quality of the output sound. Research is also focused on the representation of knowledge and data structures which support the text to sound transformation process **[4, 15, 20, 21]**.

This paper concerns with speech synthesis based on rules and focuses on the phoneme generation process as a key factor in the systhesis process. In section 2 a brief overview of some related work in the area of speech synthesis are introduced. In section 3 the proposed model of the phoneme generator is introduced, in section 4 some examples of using the proposed phoneme generator are given, and section 5 introduces a conclusion and future work.

Related Work

The real beginning of speech synthesis was done at the **Haskins laboratory** in **1947**; the human voice was being analyzed into a set of data and reconstructed at a later time. By **1960** many academic institutions around the world shared techniques of synthesizing human speech using minicomputers, storing only relatively small amounts of data of the order of 10000 binary digits for every second of speech. Experiments were made with speaking telephone directories and aircraft timetable. As early as **1964**, John Holemes et al., achieved some excellent synthesis results.

In **1971** a new digital technique was developed for speech synthesizing using computers. The method, known as Linear Prediction, was already known to other signal processing world, Atal and Hanauer showed how it could be applied to speech. There followed a surge of academic activity (e.g., Itakura **1972** and Markel, **1973**), and it was demonstrated that LPC provided a much needed algorithm technique of synthesis which was well suited to digital implementation.

The start of the speech revolution could be said that it has happened in **1977**. Around that time several consumer products started appearing on store shelves. The first product was a Talking Calculator designed by Telesensory Systems Inc., and this was followed shortly after by the Speak'n Spell from Texas Instruments. Both these products were based on **LPC**.

In **1980**, Richard Wiggins designed a low-cost linear-prediction synthesis chip to take advantage of the ability of linear prediction to represent critical spectral and temporal aspects of speech waveforms efficiently.

The Prose-2000 commercial text-to-speech system was first developed in conjunction with a reading machine for the blind project at Telesensory Systems by **James Bliss** and his associates (Groner et al., **1982**, Goldher and Lund, **1983**).

The **klattalk** system, by **Dennis Klatt** of **M.I.T.** text-to-speech system software was licensed to **Digital Equipment Corporation** as a basis for the commercial **DEC**talk text-to-speech system announced in **1983** [**6**].

In (**1985**, **1988**), a new **AT&T** Bell Laboratories text-to-speech system (Olive and Liberman, **1985**) uses the Olive (**1977**) diphone synthesis strategy in combination with a large morpheme dictionary (**Coker**, **1985**) and letter-to-sound rules (**Church**). The laboratory system was demonstrated at **1985** meeting of the Acoustical Society of America.

In **1992**, Bchenko and et al **[3]** described an application of text-to-speech for speech-impaired, deaf, and hard hearing people. The application is unusual because it requires real-time synthesis of unedited, spontaneously generated conversational texts transmitted via a telecommunications device for the deaf (**TTD**). They described a parser that they have implemented as a front end of a version of the Bell Laboratories text-to-speech synthesizer.

In **1995**, Hoory, R. and Chazan, D. [**11**] proposed a new text-to-speech synthesis techniques, for producing continuous, natural sounding speech of a specific speaker. The synthesis technique is based on selecting short speech frames from a phoneme-labeled speech database. The selection procedure involves minimization of a distortion criterion, by a dynamic programming algorithm.

In **1996**, R.W.P. Luk and R.I. Damper **[18]** introduced a trainable ("data-driven") technique for letter-to-phoneme conversion based on formal language theory. The spellings and pronunciations of English words are modeled as the productions of a stochastic grammar, inferred from example data in the form of a pronouncing dictionary. The terminal symbols of the grammar are letter-phoneme corresponding, and the rewrite (production) rules of the grammar specify how these are combined to form acceptable English word spellings and their pronunciations.

The Proposed Text-To-Phoneme Generator

The general model of the text to phoneme generator consists of three components: processing buffer, text-to-phoneme translation rules, and phoneme production algorithm as shown in fig. 2.

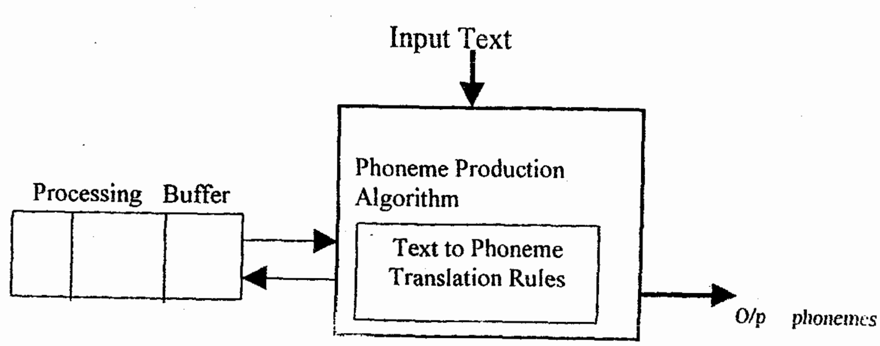

Figure 2: The general model of the English text to phoneme generator.

Processing Buffer

The processing buffer consistes of three parts: left context part, match context part, and right context part. Left context part represents the set of characters before the search (Match part), match context part represents the set of characters of the part which has been searched for, and right context part represents the set of characters after the search (Match part).

Text to Phoneme Translation Rults

Depending on the pronunciation rules of the English language, a set of text-to –phoneme translation rules are proposed. The proposed rules are expressed using the production rules formality [2]. The general form of the text to phoneme translation rules appears as the following:

$$L\_C\,T_1\,R\_C \quad [\mathbf{P_1}]$$
$$L\_C\,T_2\,R\_C \quad [\mathbf{P_2}]$$
$$\cdot \qquad \cdot$$
$$L\_C\,T_i\,R\_C \quad [\mathbf{P_n}]$$

where L_C is the left context, R_C is the right context.

Each T_i is a text pattern over the alphabet and $\mathbf{P_n}$ is the corresponding phoneme pattern.

Auxiliary Definitions are expressed by regular expressions as in the following:

Vowel	= A | E | I | O | U
consonant	= B| C | D | F | ....
+	= (vowel)$^+$
*	= (consonant)*
^	= consonant
%	= E | ER | ES | ED | ING | ELY
&	= E | I | Y
@	= T | S | R | D | L | Z | N | J | TH | CH | SH
!	= S | C | G | Z | X | J | CH | SH
ˇ	= Anything
^	= Nothing

Translation rules are classified into two types: general rules and exception rules. The general rules represent the character without considering the preceding or the following characters. The exception rules represent case sensitive phoneme production of characters depending on the left and right characters. The distripution of the proposed phoneme production rules over the alphabets is shown in fig.

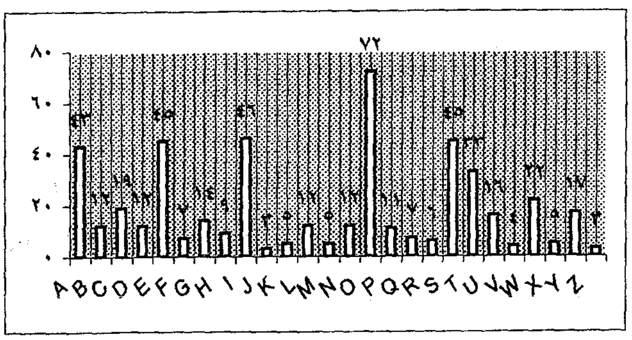

Figure 3: The distripution of the phoneme production rules over the alphabets.

The general rule for each character is tested as the last rule while the exception rules are tested first. The following are examples of the phoneme production rules and the full documentation are shown in appendex A:

The general translation rule for the charactei **A**:

˘ **A**˘ › · AE /

Examples of the exception translation rules for the character **A**:

˘ **A**^ › / EY

^ **ARE**^ › / AA-r /

^ **AS**^ › / AE-s /

^ **AT**^ › / AE-t /

Text-to-Phoneme Generation Algorithm

Read the word into the Match_Context Part

While (not empty Match_Context) **Do**

Search for the Match_Context in the rules of the **1st letter**.

If found

Produce the corresponding phonemes

Add the output phonemes to the phoneme list

Shift Left the content of the Match_Context into Left_Context.

Shift Left the content of the Right_Context into Match_Context

Else

If **empty** Right_Context

Shift Right the Match_Context **except 1st letter** into the Right_Context.

Else

Shift Left the **1st letter** of the Right_Context into Match_Context

End IF

End IF

Loop

Produce the corresponding phoneme list

Stop

Implementation

The proposed model of the text to phoneme generator was implemented using Virtual C++ compiler under Microsoft Windows. The input text is assigned to the input_text_Var and the output phonemes are assigned to the output_phoneme_Var. The processing buffers consists of three variables: Left_Part_Var, Match_Part_Var, and Right_Part_Var. Text-to-phoneme translation rules are represented using two dimensional array with the structure of (n,4) where n is number of translation rules and 4 represents the left context, match context, right context, and the output phoneme cells.

Examples

Example 1: Speech › / s-p-IY-CH /

Read the word into the **M**atch_Context Part

	speech	

Search for the **Match**_Context in the rules of the **1st letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

	s	peech

Search for the **Match**_Context in the rules of the **1st letter**.

Found

˘ S˘ › ′s ′

Produce the corresponding phoneme: / *s* /
Add the output phonemes to the phoneme list
Phoneme list: {*s*}
Shift_Left the content of the **Match**_Context into **L**eft_Context.
Shift_Left the content of the **R**ight_Context into **Match**_Context

S	Peech	

Search for the **Match**_Context in the rules of the **1st letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

S	P	eech

Search for the Match_Context in the rules of the 1st **letter**.

Found

˘ P˘ › /*p*/

Produce the corresponding phoneme: / *p* /
Add the output phonemes to the phoneme list

Phoneme list: {*s*, *p*}

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

Sp	Eech	

Search for the Match_Context in the rules of the 1st **letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

Sp	E	ech

Search for the Match_Context in the rules of the 1st **letter**.

Found

^ E^ › / *IY* /

Produce the corresponding phoneme: / *IY* /

Add the output phonemes to the phoneme list

Phoneme list: {*s*, *p*, *IY*}

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

Spe	Ech	

Search for the Match_Context in the rules of the 1st **letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

Spe	E	ch

Search for the Match_Context in the rules of the **1st letter**.

Found

+*E^ › */ Silent /*

Produce the corresponding phoneme: */ Silent /*

Add the output phonemes to the phoneme list

Phoneme list: {*s, p, IY, Silent*}

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

Spee	Ch	

Search for the Match_Context in the rules of the **1st letter**.

Found

CH › */ CH /*

Produce the corresponding phoneme: */ CH /*

Add the output phonemes to the phoneme list

Phoneme list: {*s, p, IY, Silent, CH*}

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

Speech		

Empty Match_Context

Produce Phoneme list: ***{s, p, IY, Silent, CH}***

Stop

Example 2: synthesis › / s-IH-n-TH-EH-s-IH-s /

Read the word into the Match_Context Part

	Synthesis	

Search for the Match_Context in the rules of the **1st letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

	S	ynthesis

Search for the Match_Context in the rules of the **1st letter**.

Found

' S' › /s/

Produce the corresponding phoneme: / *s* /

Add the output phonemes to the phoneme list

Phoneme list: {*s* }

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

S	ynthesis	

Search for the Match_Context in the rules of the **1st letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

S	Y	nthesis

Search for the Match_Context in the rules of the **1st letter**.

Found

˘ Y˘ › */IH/*

Produce the corresponding phoneme: ***/IY/***

Add the output phonemes to the phoneme list

Phoneme list: {*s, IH*}

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

Sy	nthesis	

Search for the Match_Context in the rules of the **1st letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

Sy	N	Thesis

Search for the Match_Context in the rules of the **1st letter**.

Found

˘ N˘ › */n/*

Produce the corresponding phoneme: ***/n/***

Add the output phonemes to the phoneme list

Phoneme list: {*s, IH, n*}

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

Syn	Thesis	

Search for the **Match_Context** in the rules of the **1st letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

Syn	T	Hesis

Search for the **Match_Context** in the rules of the **1st letter**.

Not found and not empty Right_Context

Shift_Left the **1st letter** of the Right_Context into Match_Context

Syn	Th	esis

Search for the **Match_Context** in the rules of the **1st letter**.

Found

ˇ TH˘ › */ TH /*

Produce the corresponding phoneme: */ TH /*

Add the output phonemes to the phoneme list

Phoneme list: {*s, IH, n, TH* }

Shift_Left the content of the Match_Context into Left_Context.

Shift_Left the content of the Right_Context into Match_Context

Synth	Esis	

Search for the **Match_Context** in the rules of the **1st letter**.

Not found and empty Right_Context

Shift_Right the Match_Context **except 1st letter** into the Right_Context.

Synth	E	Sis

Search for the Match_Context in the rules of the 1st **letter**.

Found

˘ E˘ › */EH/*

Produce the corresponding phoneme: */ EH /*
Add the output phonemes to the phoneme list

Phoneme list: {*s, IH, n, TH, EH* }

Shift Left the content of the Match_Context into Left_Context.

Shift Left the content of the Right_Context into Match_Context

Synthe	Sis	

Search for the Match_Context in the rules of the 1st **letter**.

Found

˘ SIS˘ › */s-IH-s/*

Produce the corresponding phoneme: */ s-IH-s /*

Add the output phonemes to the phoneme list

Phoneme list: {*s, IH, n, TH, EH, s-IH-s* }

Shift Left the content of the Match_Context into Left_Context.

Shift Left the content of the Right_Context into Match_Context

Synthesis		

Empty Match_Context
Produce Phoneme list: {*s, IH, n, TH, EH, s-IH-s* }

Stop

Conclusion And Future Work

This paper has introduced an English text to phoneme generator based on rules. The proposed phoneme generator is composed of two components: a set of phoneme generation rules and phoneme generation algorithm. The set of phoneme generation rules is represented using production rules formalism and is implemented using array data structures. The output phonemes of a set of words are applied to a phoneme pronunciation tool, and the generated sound was acceptable when compared with the generated sound using MS Talk-It tool.

In the future work English Text to Phoneme (***ETTP***) will take into consideration the semantics of the sentences not only the forms or the set of character to phoneme generation rules. Also sound characteristics such as prosody will be taken into consideration. Also the model of the proposed rule-based English text to phoneme generator can be adapted and implemented to support Arabic text to phoneme generation.

References

[1] Abd Al-Hameed M. Emara, "*Speech Synthesis*", M. Sc. Dissertation, Computers & Systems Eng., Faculty of Eng., Al-Azhar Univ., 1999.

[2] Alfred Aho, Ravi Sethi, & Jeffrey Ulmann, "***Compilers: Principles, Techniques, and Tools***", Addison-Wesley Pub. Co., 1985.

[3] Bachenko, J. Daugherty, J. Fitzpatrick, E.,"***A Parser for Real-Time Speech Synthesis of Conversational Texts***", Third Conference on Applied Natural Language Processing, pp. 25-32, 1992.

[4] Cervantes, "***Knowledge Representation for Speech Processing***", From Neural Networks to Artificial Intelligence Proceeding of an U. S. Mexico Seminar, pp. 359-370, 1993.

[5] Christophe d'Alessandro, & Jean-Sylvain Lienard, "***Synthetic Speech Generation***", Survey of the State of the Art in Human Language Technology, National Science Foundation (NSF), European Commission, 1996.

[6] D. H. Klatt, "***Review of Text-To-Speech Conversion for English***", JASA, 82, 3, pp. 737-793, Sept. 1987.

[7] F. Charpentier, "***Application of an Optimization Technique to the inversion of an Articulatory Speech Production Model***", IEEE ICASSP, pp. 1984-1987, 1982.

[8] G. Bailly & A. Tran, "***Compost: A Rule-Compiler for Speech Synthesis***", Euro Speech, Paris, 1989.

[9] Gorden E. Pelton, "***Voice Processing***", McGraw-Hill, Inc. New York, 1993.

[10] H. Witten, "***Making Computer Talk: An Introduction to Speech Synthesis***", Prentice-Hall, Englewood Gliffs, 1986.

[11] Hoory, R., Chazan, D., "***Speech Synthesis for a Specific Speaker Based on a Labeled Speech Database***", Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3, pp. 146-148, 1995.

[12] K. Ross and M. Ostendorf, "***Prediction of abstract Prosodic Labels for Speech synthesis***", Academic press, 1996.

[13] Kathleen R. McKeown, & Johanna D. Moore, "***Spoken Language Generation***", Survey of the State of the Art in Human Language Technology, National Science Foundation (NSF), European Commission, 1996.

[14] L. Lee, C. Tsang, and M. Yaung, "***The Synthesis Rule In a Chinese Text-To-Speech System***", IEEE Trans. ASSP, Vol. 37, pp. 1309-1320, Sept. 1989.

[15] P. Cosi, "***A Graph-Oriented Approach to the Grapheme-to-Phoneme Transcription Italian Writer Text***", Proc. Edinburgh, pp. 273-276, 1987.

[16] Pual C. Bagshaw, "***Phonemic Transcription By analogy in Text-To-speech Synthesis: Novel word pronunciation and Lexicon Compression***", Academic Press, 1998.

[17] Paul Taylor and Alan W. Black, "***Assigning Phrase Breaks from Part-of-Speech Sequences***", Academic Press, 1998.

[18] R. W. P. Luk and R. I. Damper, "***Stochastic Phonographic Transduction for English***", Academic Press, Computer Speech and Language, 1996.

[19] S. Hertz, "***From Text to Speech With SRS***", J. Acoust. Soc. Amer., Vol. 4, pp. 1155-1170, 1982.

[20] Thierry Dutoit, "***A Short Introduction to Text-to-Speech Synthesis***", TCTS Lab, December 1999.

[21] Yoshinori Sagisaka, "***Spoken Output Technologies: Overview***", Survey of the State of the Art in Human Language Technology, National Science Foundation (NSF), European Commission, 1996.

Appendix A: Examples of the phoneme production rules.

Character Set	Phoneme Set
ˇ Aˇ	AE
ˇ A^	EY
^ ARE^	AA-r
^ AND^	AE-n-d
^ AS^	AE-s
^ AT^	AE-t
^ AN^	AE-n
^ Ar*o*	AX-r
ˇ AR+	EH-r
^AS+	EY-s
ˇ AWˇ	AO
*ANYˇ	EH-n-IY
^ AL+	AE-l
ˇ AGAINˇ	AX-g-EH-n
ˇ ABOUTˇ	AX-b-AW-t
ˇ A^%	EY
ˇ Al*f*	AE
^ ARR ˇ	AX-r
ˇ ARRˇ	AE-r
*AR ^	AA-r
ˇ AR ^	ER
ˇ ARˇ	AA-r
ˇ AIRˇ	EH-r
ˇ AIˇ	EY
ˇ AYˇ	EY
ˇ AUˇ	AO
ˇ ALKˇ	AO-k
ˇ AL^	AO-l
ˇ ABLEˇ	AX-b-AX-l
ˇ A*vo*	EY

Character to phoneme production rules-cont.

Character Set	Phoneme Set
ˇ A*a*	*Silent*
ˇ Bˇ	b
^ B^	b-IY
^ BE ^ +	b-IH
ˇ BEINGˇ	b-IY-IH-NG
^ BOTH^	b-OW-TH
^ BY ^	b-AY
^ BUT^	b-AH-t
^ BEEN ^	b-IH-n
^ BUS +	b-IH-z
ˇ BUILˇ	b-IH-l
*b*Bˇ	Silent
ˇ Cˇ	*K*
^ C^	s-IY
^ CH ^	K
^*e*CHˇ	K
ˇ CHA*r*+	k-EH
CHˇ	CH
*s*CI+	s-AY
ˇ CI*a*	SH
CI*o*	SH
ˇ CI*en*	SH
ˇ C*&*	s
ˇ CKˇ	k
ˇ CC*&*	k-s
ˇ CCˇ	k
ˇ Dˇ	d
^ D^	d-IY
^ DO^	d-UW

Character to phoneme production rules-cont.

Character Set	Phoneme Set
^ DOESˇ	d-AH-z
ˇ DOWˇ	d-AW
ˇ DUC&	d-UW-s
*d*Dˇ	Silent
ˇ Eˇ	EH
^ E^	IY
+*E^	Silent
+ED^	d
+*E*d*	Silent
+*ERS^	ER-z
@EWˇ	UW
ˇ EEˇ	IY
ˇ EARNˇ	ER-n
ˇ EUˇ	y-UW
ˇ Fˇ	f
^ F^	EH-f
^ FOR^	f-AX-r
ˇ FE*male*	f-IY
ˇ F*f*	Silent
ˇ Gˇ	g
^ G^	j-IY
ˇ GIVˇ	g-IH-v
ˇ GGˇ	g
*i*G*m*	Silent
ˇ G &	j
^ GNˇ	n
+GHˇ	Silent
ˇ Hˇ	Silent
^ H^	EY-CH
^ HAVˇ	HH-AE-v

Character to phoneme production rules-cont.

Character Set	Phoneme Set
^ HAD^	HH-AE-d
^ HAS^	HH-AE-z
^ HEREˇ	HH-IY-r
^ HOURˇ	AW-ER
ˇ Iˇ	IH
^ IN ^	IH-n
^ IS^	IH-z
^ IF^	IH-f
^ IS^	IH-z
^ I ^	AY
ˇ I^	AY
ˇ IR+	AY- r
ˇ IZ%	AY- z
ˇ IS%	AY- z
+*IC^	IH-k
ˇ IQUEˇ	IY-k
ˇ Jˇ	J
^ J ^	j-EY
*j*Jˇ	Silent
ˇ Kˇ	k
^ Kˇ	k-EY
^ K*n*	Silent
*k*Kˇ	Silent
ˇ Mˇ	M
^ M^	EH-m
*m*Mˇ	Silent
ˇ Pˇ	P
^ P^	p-IY
ˇ PHˇ	F
ˇ PIT*ch*	p-IH

Character to phoneme production rules-cont.

Character Set	Phoneme Set
ˇ P*p*	Silent
^ P*n*	Silent
^ P*s*	Silent
ˇ Rˇ	R
^ R^	AA-r
ˇ R*r*	Silent
ˇ Sˇ	S
^ S^	EH-s
ˇ SHˇ	SH
+SUR+	ZH-ER
ˇ SUR+	SH-ER
+SU+	ZH-UW
+S+	Z
ˇ S*s*	Silent
+SMˇ	z-AE-m
ˇ Uˇ	y-UW
^ U^	y-UW
^ UN*i*	y-UW-n
^ UNˇ	AH-n
@UR+	UH-r
ˇ UR+	y-UH-r
ˇ URˇ	ER
*U*Uˇ	Silent
ˇ Xˇ	/k-s
ˇ XC&	k-s
*X*Xˇ	Silent
^ Xˇ	Z
^ X^	/ EH-k-s
ˇ Zˇ	Z
*Z*Zˇ	Silent
^ Z^	z-IY