LPC-based diphone synthesis for the PolyGlot text-to-speech system

Institute for Perception Research
P.O. Box 513 - 5600 MB Eindhoven

PK/pk 94/04
12.09.1994

Rapport no. 1003

LPC-based diphone synthesis for the PolyGlot text-to-speech system

Paul A.P. Kaufholz
Leo L.M. Vogten

LPC-based diphone synthesis for the PolyGlot text-to-speech system

Paul A.P. Kaufholz & Leo L.M. Vogten

September 1994

Contents:

0 Introduction

In the Netherlands two systems for speech synthesis are operational: the KUN allophone synthesis (Loman, Kerkhoff, Boves, 1989; Loman, Boves, 1993) and the IPO diphone synthesis (Elsendoorn, 't Hart, 1982; van Rijnsoever, 1988). Allophone synthesis is applied in the PolyGlot text-to-speech (TTS) system, which currently incorporates the most sophisticated linguistic analysis for Dutch, whereas diphone synthesis results in better speech than allophone synthesis with respect to acoustic signal processing.

In order to combine the strong aspects of both systems a project has been defined by SPIN-ASSP to integrate diphone synthesis into the PolyGlot text to speech system on a PC with a DSP board. The project has been worked on at IPO from july 1 until december 31 1993 by Rob Roddeman, using some existing software modules from Lex Elich, Nijmegen. After the end of the project the software has been completed and documented by Paul Kaufholz and Leo Vogten from IPO.

This report is two-folded. First, it describes the theoretical background and algorithms of LPC diphone synthesis within the context of literature on TTS conversion. Second, it is meant as software documentation of the diphone synthesis module in the PolyGlot TTS system. For both purposes functional diagrams are used to define and illustrate the different modules in the system. The syntax of the diagrams is a variant on the functional diagrams from the OMT (Object Modelling Technique) method by Rumbaugh et. al. (1991). A brief description is given in appendix D. It must be emphasized that the diphone synthesis module has not been designed using any software engineering method but is only documented post hoc using functional diagrams based on existing source code from different authors.

Sections 2, 3 and 4 end with a description of all relevant c-functions. All function descriptions follow the next 'blueprint'. The process name refers to the functional diagram. A process is implemented by the successive functions which can be found in the mentioned source file. The module name indicates what functional diagram the following processes refer to.

Module:

process: Description.

function1 [source-file]:
Description.

function2 [source-file]:
Description.

An index to all functions is supplied in appendix A.

1 From text to speech

Text to speech conversion can be modelled as a linear sequence of processes. In most current TTS systems the process can be divided in a linguistic analysis part and a speech generation part, each of which consists of other processes (e.g: van Leeuwen & te Lindert, 1993; Carlson & Granström, 1976). The linguistic analysis part determines a number of linguistic structures, the Phonemes and Prosody data store, which are used to generate the speech signal (figure 1). The major data store (see appendix D) consists of the following information streams: phoneme string, word accents, sentence accents and intonational phrasing.

The linguistic analysis part can roughly be divided in three modules. First, the input text is parsed into separate words and sentences. Second, a phonetic transcription, the word accent

and the syllabic structure is determined, either using a lexicon (Lammens, 1993), or by a morphologic analysis (Heemskerk, 1993; Nunn et.al., 1993) or by a set of grapheme to phoneme rules (Kerkhoff et. al., 1984). Third, sentence accents and intonational phrasing for each sentence are determined by a syntactic and prosodic analysis (Dirksen & Quene, 1992). The four mentioned minor information streams of Phonemes and Prosody are used in the speech generation part.

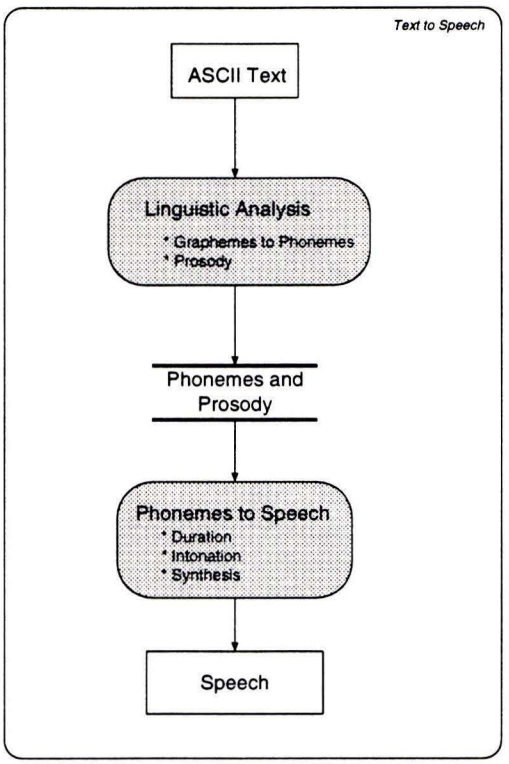

Figure 1. Text to Speech conversion. The major data store Phonemes and Prosody is determined by module Linguistic Analysis (not in this report). Module Phonemes to Speech converts the information from this data store to speech.

The speech generation part can be seen as a phonemes to speech system. The input text in this case consists of the phonetic transcription, together with the mentioned prosodic information. Figure 2 gives a schematic phonemes to speech system. The synthesis module calculates the final speech signal. Which synthesis method is used, e.g.: diphone- or allophone synthesis, is independent from the 'Duration', 'Intonation'- and 'Pitch' processes. The linguistic input has to be converted to a concrete duration for each phoneme in the phoneme string and in pitch values as a function of time. The duration can be determined in the 'Duration' process by means of a set of rules (van Coile, 1987). The process 'Intonation' determines the declination and the intonation patterns used for the word- and sentence accents (Terken, 1993). The output of Intonation is a specification of the intonation contour from which concrete pitch values as a function of time are calculated in process 'Pitch'.

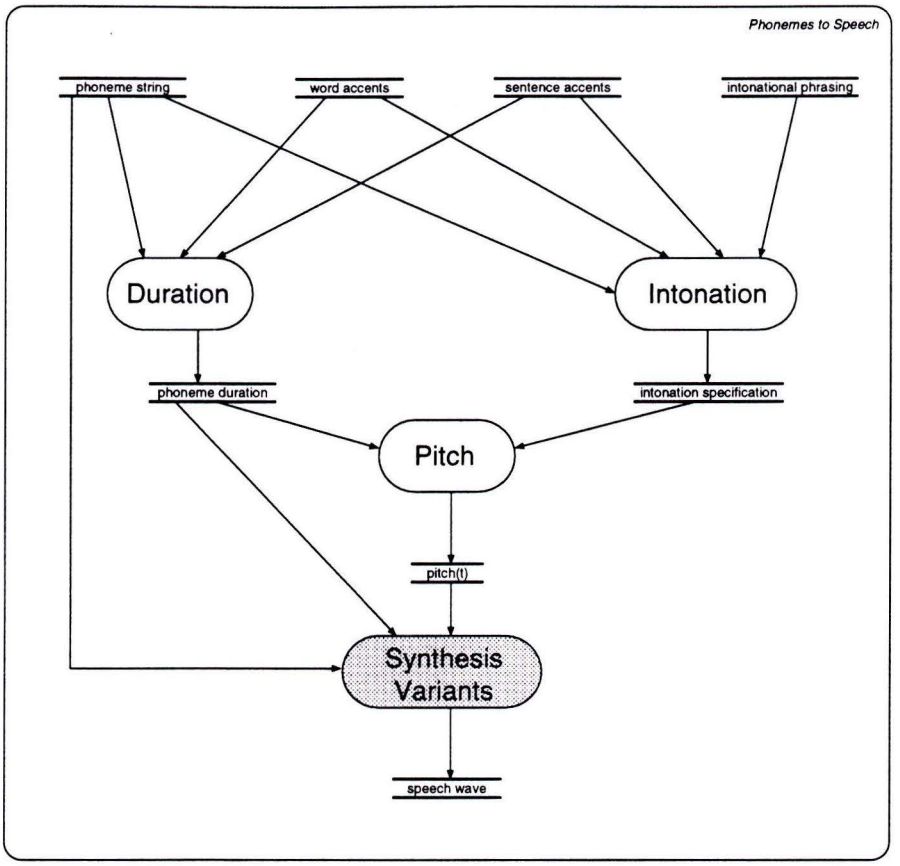

Figure 2. Phonemes to Speech conversion. From the linguistic input streams the phoneme duration, pitch as a function of time and the intonation specification are determined by the processes Duration, Pitch and Intonation. These streams are the input to the module Synthesis Variants where different synthesis methods can be chosen.

2 Synthesis in the PolyGlot TTS system

The PolyGlot TTS system covers the whole sequence from text to speech for Dutch and can, in principle, also be divided in a linguistic analysis and a speech generation part. It contains an advanced linguistic analysis which presently gives the best analytic results for a Dutch TTS system. The synthesis is based on allophone synthesis (Loman, Boves, 1993). However, diphone synthesis often results in better speech quality. Therefore, the LPC-based IPO diphone synthesis has been implemented into the PolyGlot TTS system as an alternative to the allophone synthesis. The diphone synthesis module is also implemented in a stand-alone TTS program, 'dsn', where the linguistic analysis, the duration and intonation modules are limited, using only a small set of simple rules. The program allows a direct comparison between the LPC-diphone synthesis and allophone synthesis in a small TTS-system. For more details, see section 4.

2.1 Synthesis variants and pre-processing

In figure 3 the synthesis is shown. The relevant input consists of the phoneme string, the phoneme durations and the pitch as a function of time. In the current system both diphone and allophone synthesis can be chosen. The output of both methods is a digital speech signal which can be sent to the DA-converter and/or to a speech output file. For the DA-conversion (DAC) and the time consuming calculations in the synthesis part in the PolyGlot system a Loughborough DSP board is used.

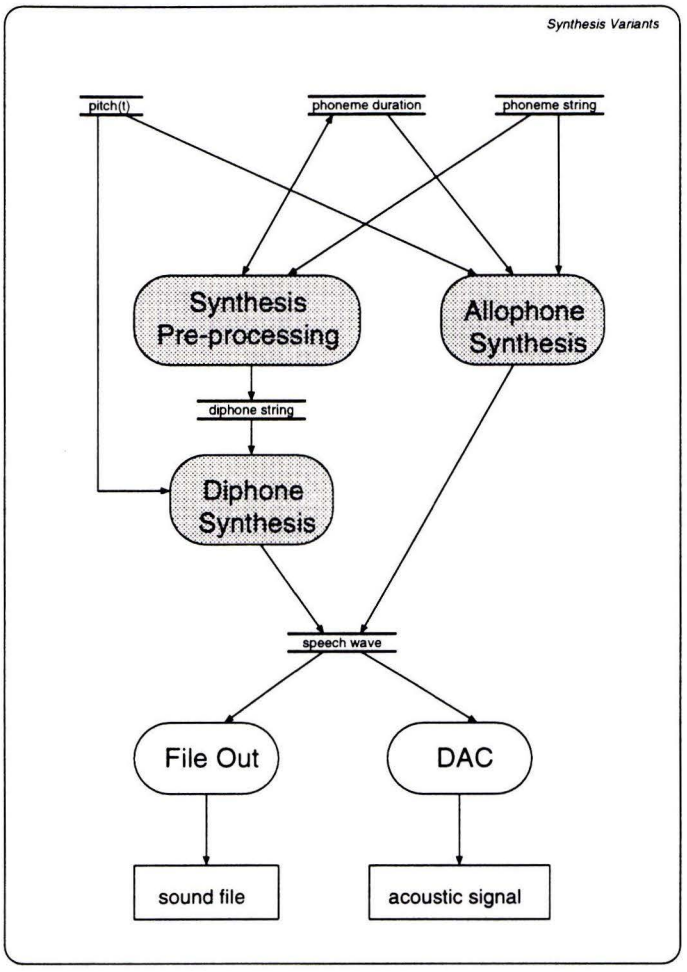

Figure 3. Synthesis Variants. The three input streams can serve both the diphone and the allophone synthesis modules (Allophone Synthesis not in this report). Before the Diphone Synthesis module is called some modifications to the input streams take place in module Synthesis Pre-processing. The final speech wave can be written to a speech file or directly transformed to an acoustic signal by the DA-converter.

The phoneme string, duration- and intonation information as a result of the linguistic analysis in the PolyGlot system are not suitable for the diphone synthesis module as such. Figure 4 illustrates which adjustments have to be made to the input streams. The phonetic transcription contains prosodic markers, all with a duration of 0 seconds, which are irrelevant for the synthesis. Also silences at the start and at the end of the text, necessary for the diphone synthesis, are missing. The process 'prepare' takes care of the addition of the silences, the deletion of the prosodic markers and the deletion of zero phoneme durations.

The phoneme string is transcribed into KUN phoneme symbols as used in the PolyGlot system, but must be converted to the IPO phoneme symbols (appendix B). Combinations of two KUN phonemes, each with its own duration, are sometimes translated to a single IPO symbol. In this case the phoneme duration of the new IPO phoneme has to be adjusted to the sum of the two KUN phoneme durations. Next, the IPO phoneme string can be converted to a diphone string. Now, all information streams are suitable for the diphone synthesis module. The functions for the different processes of figure 4 are described in section 2.2.

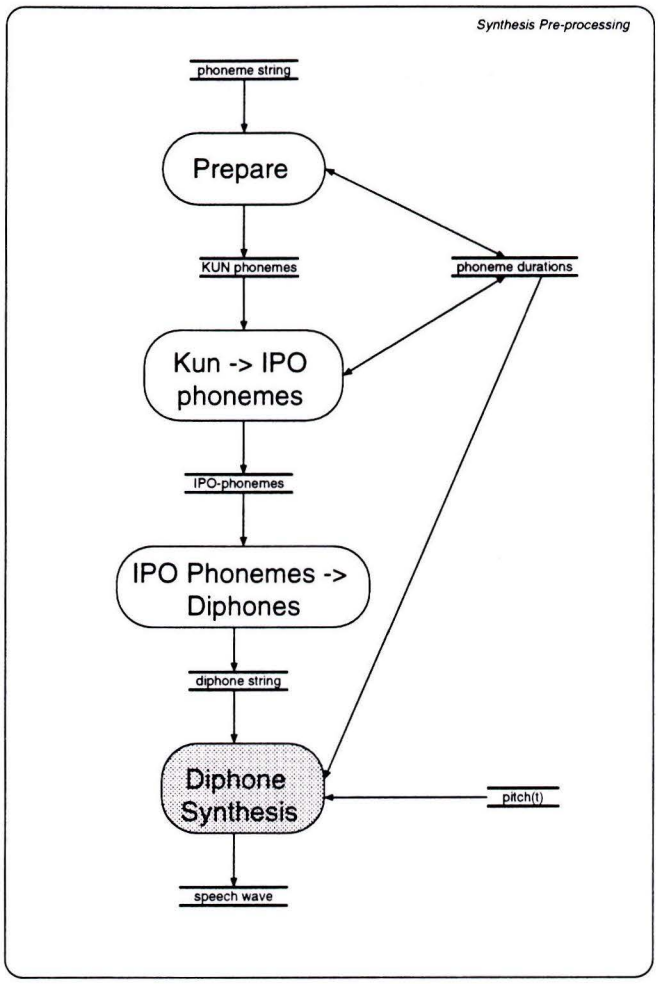

Figure 4. Synthesis Pre-processing. The phoneme string is adjusted and translated by procedures Prepare, KUN -> IPO phonemes and IPO Phonemes to Diphones to a diphone string which can be used in the Diphone Synthesis module. The phoneme durations are updated accordingly.

2.2 Functions for synthesis

Synthesis Variants:

inputs: Five input streams are used, three major data stores and two options, defined and determines in the PolyGlot main program must be accessible to the diphone synthesis module. For each input stream the interface to the main program is given:

> Options 'synthesis type' and 'output type' are set in *main* [dutch.c]
> The major data stores, phoneme string, duration and pitches, are retrieved in *synthesize_contour* [parsyn.c]

File Out: Calculated sound samples are saved into a speech data file of the so called Wendy format. The raw data in 2 byte integers is preceded by a header of 128 4 byte integers with at position 125 the sample rate and at position 126 the number of 2 byte samples.

init_phonlevel [pho_modu.c]:
Opens and initializes the speech output file.

close_phonlevel [pho_modu.c]:

Updates and closes the speech output file.

synth_diph [dip.c]:
Writes speech samples of one diphone to the file.

DAC: Converts sound data to an acoustic speech signal. A Loughborough Digital Signal Processor board DSP32C is used to perform this task. Existing DSP software (putsamp.o and intr.o) is used for the low-level operations. The interface from the PC host computer to the DSP is done in the source module 'dsyn.c'. It has to be mentioned that the main program for the LPC diphone synthesis is also performed on the DSP-board using the source in module 'dip.c' (see section 3).

Synthesis Pre-processing:

Prepare: Removes prosodic markers in the phoneme string and adjusts the duration accordingly.

ds_alf [ds_alf.c]:
The main program for diphone synthesis. For this process it adjusts the phoneme durations.

prepare_start / prepare_stop [ds_alf.c]:
Opens/closes the diphone data-base (see section 3) and initialises/closes the DSP board.

rem_accents [ds_alf.c]:
Removes prosodic markers.

KUN -> IPO phonemes: Translation of KUN-formatted phoneme string to IPO format by rules.

make_ipo_fon [ds_alf.c]:
Rules for translation implemented in source code. This function could be replaced by rules implemented according to a rule formalism and compiled by a rule compiler.

IPO phonemes -> diphones: Translation of IPO formatted phoneme string to an IPO diphone string.

make_diphones [ds_alf.c]:
Translation module. Since this is a linear straightforward procedure no rule set is used.

Diphone Synthesis:

read_records [ds_alf.c]:
Retrieves diphones from the diphone database and adjusts length and pitch for each frame. Adjusted diphones are synthesized (see section 3).

3 LPC diphone synthesis

Each diphone of the LPC diphone database consists of parametrized frames, each of which is the result of a 10th order LPC-analysis where one frame represents a quasi-stationary speech segment of 10 ms obtained using an analysis window of 25 ms. Each frame is described by 13 parameters: the pitch, the voiced/unvoiced information, the gain and five pairs of filter parameters, defining five spectral resonances (van Rijnsoever, 1988).

For the synthesis of a diphone all diphone parameters must be completely specified. The prepared diphones are held in a diphone data base from which the main parameters for each diphone can be retrieved. Next, the parameter setting must match the intended phoneme duration and pitch. Finally, the gain of two subsequent diphones is interpolated (figure 6). Because some procedures need information about the following diphone, always two diphones are available during the adjustments: the current diphone and the next diphone. Figure 5 illustrates the terminology used in this section.

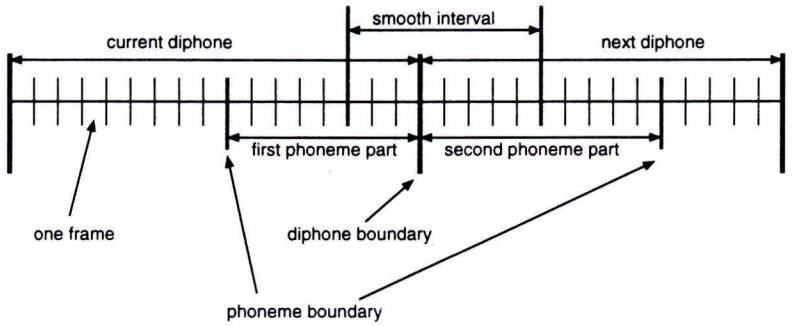

Figure 5. An example diphone concatenation. One diphone consists of the second part of a phoneme followed by the first part of the next phoneme. The phoneme and diphone boundaries are indicated. The smooth interval covers the frames relevant for gain smoothing from one diphone to the other.

3.1 The diphone data base

A diphone is a short speech segment containing the transition between one phoneme and another, e.g. for vowels starting in the quasi stationary part of the first phoneme and ending at the quasi stationary part of the second phoneme. Using diphones as building blocks instead of allophones overcomes the problem of specifying the complicated and perceptually important transitions between two phonemes. Concatenation of diphones implies the easier task of interpolating two quasi stationary parts of the same phoneme. Theoretically there exists one diphone for each combination of two phonemes. For Dutch this would be 1600 diphones for 40 phonemes. However, in practice the number of diphones is smaller since some combinations of phonemes never occur.

Each diphone consists of a number of frames preceded by a header containing general information about the frames. Each frame contains the necessary filter parameters.

Header:

* name of the diphone
* default frame duration in samples
* duration of the first phoneme part in samples per frame
* duration of the second phoneme part in samples per frame
* number of frames of the first phoneme part
* total number of frames
* frame number where gain smoothing ends
* frame number where gain smoothing starts

Frame:

* pitch
* gain
* voiced / unvoiced
* pre-emphasis (fixed, -0.9)

* 5 pairs of filter parameters

The diphones are synthesized one by one. Before synthesizing, several parameters have to be adjusted, but first, the diphone has to be retrieved from the diphone data-base. In this implementation for the PolyGlot system all diphones are stored in one binary file. For more information on the precise format, see section 6. For fast access, the diphones are indexed by a hash function.

3.2 Diphone parameter adjustments

After the information about the current and the next diphone is read from the diphone database, the information on duration and pitch has to be mapped to diphone parameters and the gains of two adjacent diphones have to be smoothed (figure 6).

Phoneme duration to frame duration

The minor data store duration contains a duration in milli seconds for each phoneme. However, the original text to be synthesized is expressed as a string of diphones. Every phoneme is constructed by the second phoneme part of the current diphone and the first phoneme part of the next diphone. Each phoneme part consists of a number of frames each of equal length. The duration is distributed over the frames of equal length of both phoneme parts. Thus, the duration is mapped to the number of samples per frame, which is the same for both phoneme parts.

The number of samples per frame (S), is determined by the total number of frames F and the given duration of a phoneme D [ms], where F is the sum of the number of frames of both phoneme parts, F_r and F_l. For a given total duration of D ms for a phoneme, the number of samples per frame for both phoneme parts is:

$$S = \frac{D \cdot f_s}{F} \qquad \text{(EQ 1)}$$

where f_s is the sample frequency. This value is stored in the header of the corresponding diphones. For the first and the last phoneme, F_r and F_l are zero respectively.

Pitch

The pitch values as a function of time, pitch(t), have to be mapped to the pitch parameters of the individual frames. During one LPC-frame the pitch is defined to be constant. Now, for a particular frame the pitch parameter is pitch(t_s), where t_s is the starting time of this frame. Since the frame duration of every frame has been set, the exact starting time t_s can be calculated.

All frames are synthesised one after the other, therefore a time index is used to keep track of the total time of the already synthesised frames resulting in the starting time of the current frame. Because all frames within one diphone part are of equal length the increment of the time index is constant during one phoneme part of a diphone and can be calculated in advance. Hence, the increment value of the time index is updated only at the start of a new phoneme part.

Gain smoothing

A diphone is cut from a longer segment of real speech. Different diphones stem from different speech segments. Therefore, the corresponding phoneme parts of two adjacent diphones also stem from different speech segments. Hence, in general there occur differences between the amplitude and spectral shape of the two quasi-stationary parts that have to be concatenated. Gain smoothing of the frames near the diphone boundaries compensates for different amplitudes. No compensation for spectral difference as a result of different filter parameters is

implemented in the PolyGlot system. If the difference between the parameter sets is significant, the imposed difference in spectral energy can still result in a perceptually different amplitude after gain smoothing. An interpolation of the formant tracks, parametrized in terms of the filter parameters, would be necessary.

For gain smoothing in the PolyGlot system the start frame of the first diphone and the end frame of the second diphone are linearly interpolated. These frames are retrieved from the diphone's header information. The gain values of the frame prior to the start frame and the one after the end frame are used as edges. A simple linear function can be retrieved to find the gain values of the intermediate frames. The original gain values are not used.

Gain smoothing is only desirable for voiced phonemes. For unvoiced phoneme parts the start frame or end frame in the diphone's header is simply set to frame number last+1 or 0 to prevent gain smoothing.

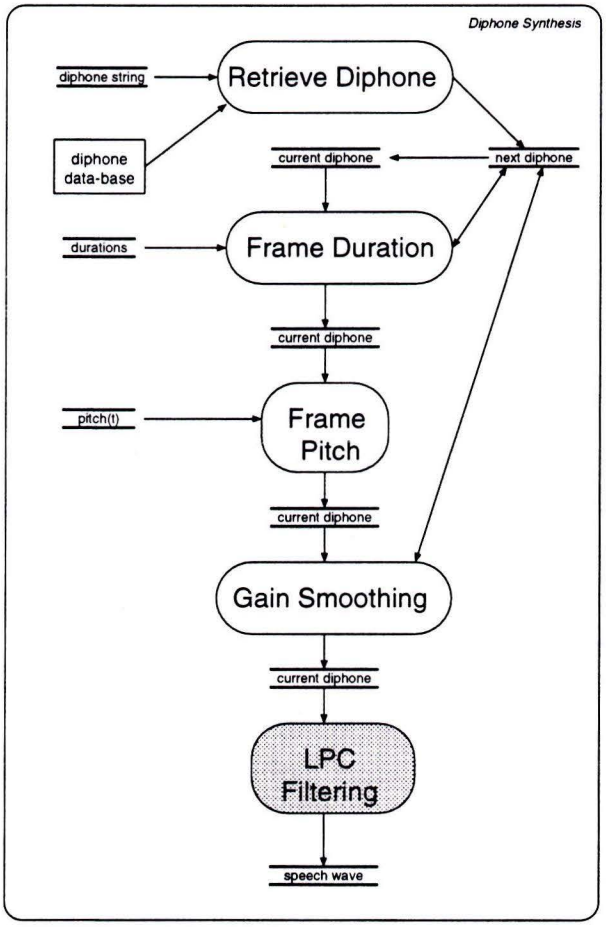

Figure 6. Diphone Synthesis. The diphones in the diphone string are retrieved one by one from the diphone data-base. In each frame duration, pitch and gain are adjusted by the procedures Frame Duration, Frame Pitch and Gain Smoothing. Then module LPC Filtering calculates a speech wave for a diphone.

3.3 LPC filtering

The implemented LPC-resynthesis is based on the source-filter model (Markel & Gray, 1976). Five second order filters are cascaded, each representing a formant filter specified by one so-called p, q parameter pair. The cascade is preceded by a first order de-emphasis filter that compensates the pre-emphasis from the radiation at the lips resulting in a spectral tilt of +6 dB/

octave, specified by a constant parameter p1=0.9 (Vogten, 1983). The filter is realized for ease as a second order filter with a q parameter of zero (figure 7). Pitch, voiced/unvoiced information and gain determine the source signal.

The synthesis model differs from the one used for the allophone synthesis in the PolyGlot TTS system. In allophone synthesis the formant tracks must be calculated in detail, including features like burst onsets in consonants and nasality, whereas in diphone synthesis these effects are implicit. The synthesis model used for allophone synthesis (Boves et. al., 1987), which is a variant of the Klatt synthesizer (Klatt, 1980), is therefore more complex and includes anti resonances to model, for example, nasality.

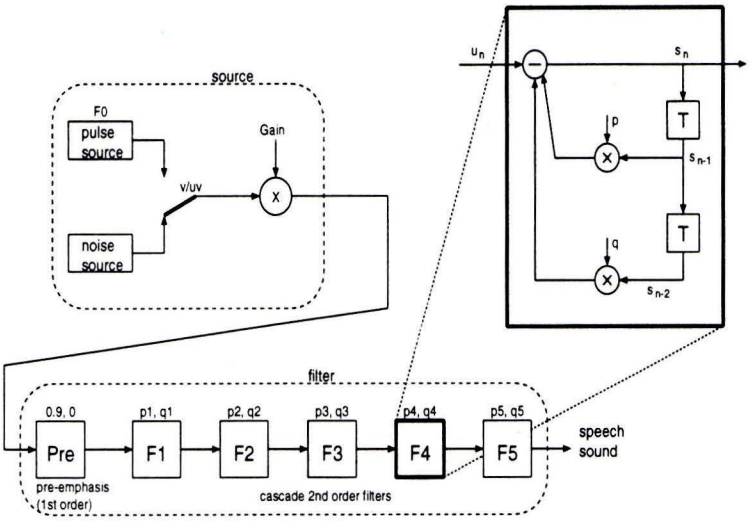

Figure 7. Source-filter model of LPC-synthesis. The source determines the excitation to the filter sequence, which is different for voiced and unvoiced frames. The filter sequence, consisting of a 1st order de-emphasis filter followed by five 2nd order filters, determines the final sample values. The inset represents one 2nd order filter containing two internal states, s_{n-1} and s_{n-2}, and is specified by a p, q parameter pair.

After all parameters of the individual frames are adjusted, the speech wave can be calculated. The input to this sequence is a pulse determined by the procedure 'Determine Excitation'. The resulting output is one speech sample. Internally every formant filter has two internal state values and one filter output, which is shown in the inset.

The filters are reset to zero only at the start of a synthesis procedure. When a new diphone is synthesized, the filter coefficients, or p, q parameters, of the first frame are passed to the filters. The procedure 'Calculate Samples' determines the next input to the first filter and calculates the next output sample by a one step propagation of the input through the cascade of filters, until all samples for the current frame are calculated. Next, a new frame is going to be synthesized and the 'Calculate Samples' procedure orders 'Update Filters' to pass the filter coefficients of the next frame to the filters, where after the sequence of sample calculation just continues (figure 8).

For every sample the input to the filters is determined by 'Determine Excitation'. The excitation is different for a voiced or an unvoiced frame. For voiced frames mono-pulse excitations are used, that is, a Dirac-pulse with an amplitude corresponding to the gain of the particular frame is used as an excitation at the start of every pitch period within the frame. Within a pitch period the remaining input to the filters is zero. Unvoiced frames are excited by a continuous stream of gaussian random noise pulses, which stem from the noise generator 'noise'. The current

synthesis is 'frame synchronous', which means that the filter parameters are updated every frame, independant of the excitation pulses.

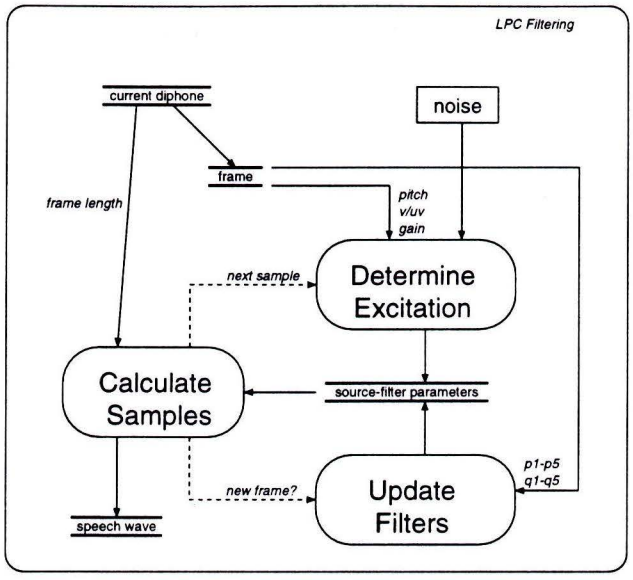

Figure 8. fig 8: LPC Filtering. Calculate Samples calculates the speech wave frame by frame. The filter parameters are updated at the start of every frame. Determine Excitation presents an input to the synthesis filter, which is mono-pulse excitation for voiced frames and a sequence of gaussian random pulses, that stem from 'noise', for unvoiced frames.

An alternative could be a 'pitch synchronous' synthesis where the filter parameters are updated every pitch period with interpolated filter parameters, obtained by the interpolation of the filter parameters of two adjacent frames proportional to the distance of the pitch period to the frame boundaries. Pitch synchronous synthesis might give slightly better results, especially in cases where the framelenght is relatively long compared with the pitch period.

In the frame synchronous synthesis a pitch period can cross a frame boundary. Calculation of samples continues until the end of the pitch period, then the parameters of the next frame determine the excitation of the following pitch period. However, at the immediate start of a new frame the filter parameters are passed to the filters for both voiced and unvoiced frames. One could choose to finish the current pitch period with the parameter setting of the first frame, however, in practice no significant difference will be perceived.

3.4 Functions for LPC diphone synthesis

Diphone Synthesis:

Retrieve Diphone: Retrieves a selected diphone from the diphone data base.

dhash [ds_alf.c]:
Calculates the hash value for a specific diphone, indicating the starting position for diphone search in the diphone data-base.

look_up [ds_alf.c]:
Positions the file pointer in the diphone data base file according to the hash value, searches the correct diphone from this position and retrieves its parameters.

Frame Duration: Mapping of phoneme duration to frame duration.

fil_dur [ds_alf.c]:

The frame duration is determined according to the phoneme duration. The duration of a phoneme is distributed over the corresponding frames of the second and first phoneme parts which are of equal length.

Frame Pitch: Mapping of the pitch stream to individual frame pitch parameters.

adjust_pitch [ds_alf.c]:

The corresponding pitch in the input array for the pitch parameter of every frame in a diphone is determined by a pointer that keeps track of the starting time of every frame.

Gain Smoothing: Smoothing of the gain parameters of frames near a diphone border to correct for amplitude differences.

synth_diph [ds_alf.c]:

Here, the smoothing algorithm, see smooth, is called. From this function the completed diphone is passed to the LPC filtering module.

smooth [ds_alf.c]:

Linear smoothing of gain for two phoneme parts. The start and end frame numbers for smoothing are retrieved from the diphone's headers.

LPC Filtering:

Calculate Samples: Main procedure for calculation of the speech wave for one diphone. Includes filtering, based on the source-filter model, and initialization of filters.

synth_diphone [dip.c]:

Main loop for synthesizing a diphone. Checks for update, determines new excitation.

formants [dip.c]:

Calculates each sample value based on a cascade of a first order de-emphasis and five 2nd order formant filters.

init_filters [dip.c]:

Initializes the internal states of the filter cascade to zero. Uses *init_hfilters* and *init_lfilters*. This function is called by an external procedure at the beginning of a synthesize sequence, hence, before the first diphone is synthesized.

Determine Excitation: Determines excitation to the filter sequence. For voiced frames this is one pulse every pitch period, for unvoiced frames a gaussian random pulse sequence.

synth_diphone [dip.c]:

Type and timing of excitations are determined here. For unvoiced frames the random pulses are retrieved from the function *noise*.

noise [dip.c]:

Fetches a gaussian random noise sample by random choosing one out of the 1024 values in the noise array prepared in *prepare_noise*.

Update Filters: Resets the filter parameters with the p, q values of the current frame. To keep a continuous signal the internal states and outputs of the individual filters are not reset.

parcoe_cascade [dip.c]:

Reads the p, q parameters of the next frame to the corresponding filter parameters.

Noise: Supplies gaussian random noise samples.

prepare_noise [dip.c]:
Prepares an array of 1024 gaussian random noise samples.

gran [dip.c]:
Returns one gaussian random noise sample.

4 The stand-alone system

The stand-alone program, 'dsn', a small TTS system based on LPC diphone synthesis is meant as a research tool. The core synthesis module can be tested in isolation, instead of being a part of a large and complex TTS system. Its main goal is not to generate high quality speech but to test or compare the synthesis modules. For this purpose several options have been added to the program. Users can switch off duration and intonation modules to generate a sole diphone concatenation, place accents in the input string themselves or use diphone names as input instead of KUN or IPO phoneme symbols. However, by default the systems incorporates a complete but simple TTS system. The linguistic analysis resulting in the essential input streams, phoneme string, phoneme durations and pitch(t), is realized by a set of simple rules. The phoneme durations are calculated by another set of rules, while the intonation, consisting only of a declination, is realised in c-code.

The various program options will be explained in appendix C. Other differences and a brief description of the rule compiler is given below.

4.1 Differences between the PolyGlot module and the stand-alone system

The stand-alone system, 'dsn', differs from the diphone synthesis module of the PolyGlot system in the functionality they support. The PolyGlot module only synthesises the input information streams whereas the stand-alone system also takes care of the invocation of rule sets to achieve the necessary information, and of the interpretation of program options.

The differences are consequently found in the Synthesis Pre-processing module. That is, the two entry functions, *ds_alf()* in case of the PolyGlot module and *main()* in case of 'dsn', differ and some new functions are added to 'dsn'.

First, the command-line arguments are read and interpreted by the function *arguments()*. Second, the linguistic analysis by rules has to be invoked. This is done by the interface function *convertor()*. This function calls three rule sets depending on its argument: the grapheme to phoneme (G->P) conversion, a small rule set to adjust the result of the G-> P conversion, and the transformation of KUN phoneme to IPO phoneme symbols. The latter one could also be used in the PolyGlot module to replace the conversion in c-code. The result of a rule set is always returned in a text array. The rule sets are compiled to c-code by a special rule compiler. More about the rule compiler can be read in section 4.2. Third, simple code in the PolyGlot module's entry function *ds_alf()* is replaced here by separate functions. Function *adjust_dur()* adds two silences to the phoneme string and *read_fr_dura()* adjusts the durations after the addition of silences. Fourth, duration and intonation are determined. The phoneme duration is determined by another rule set, whereas the intonation, and finally the pitch as a function of time, is calculated in c-code. The entry functions to the procedures are *dura_into_rules()* and *pitch_rules()* respectively. The intonation is restricted to the calculation of only a declination.

4.2 Rules and compilation

In the stand-alone system four rule sets are used. Three are invoked by the function convertor and one, the duration rule set, by *dura_into_rules()*. A rule set is specified in the format described by Kerkhoff & Wester (1987). A rule set is compiled by the rule compiler 'flowc.exe'. This program takes a file *<name>.dat* as input and generates two c-functions depending on the type of rules: *fone_<name>()* in case of phonological substitution rules (e.g. grapheme to phoneme conversion), and *fono_<name>()* in case of parametrization rules (e.g. duration rules). Each rule set can make use of feature tables which are compiled to c-modules, used by 'flowc.exe' to generate the final output functions, by a feature compiler 'featc.exe'. The program 'featc.exe' takes two kinds of feature tables as input: one contains character features that are common among rule sets, and the other contains character features and optionally default parameter values for each character.

An overview is given of the rule modules and feature tables the different rule sets depend on:

Rule set:	**Rule module:**	**Feature table:**
G->P conversion	*grafon.dat*	*gfeat.dat*
	grafo2.dat	*grafeat.dat*
KUN->IPO phonemes	*ipofono.dat*	
dura_into_rules	*duur.dat*	*ffeat.dat*
		fonfeat.dat

4.3 Extra functions for the stand-alone system

Text to Speech:

Linguistic Analysis

convertor(..,1) [grafon.dat] and *convertor(..,2)* [grafo2.dat]:
Interface functions for grapheme to phoneme conversion and an adjustment to the result by rules respectively.

Phonemes to Speech:

Duration

dura_into_rules [g-f_int.c]:
Interface function for the determination of phoneme durations by rules.

Intonation & Pitch

pitch_rules [g-f_int.c; Uses source-modules: ipo.c and streams.c]:
Interface function for the determination of the intonation contour and finally the pitch as a function of time, implemented in c-code.

Synthesis Pre-processing:

Prepare

arguments [ds_ipo.c]:
Reads and interprets the command line options to the stand-alone program.

adjust_dur [ds_ipo.c]:
Adds two silence phonemes to the beginning and end of the phoneme string.

read_fr_dura [ds_ipo.c]:
Adjusts phoneme durations because of the added silences.

KUN -> IPO phonemes

convertor(..,3) : [g-f_int.c]:
Interface function for the conversion of KUN phoneme symbols to IPO phoneme symbols by rules.

5 Conversion of VAX diphone format to PC format

For the current system two diphone sets have been prepared for the PC: the SPIN-ASSP diphones of speaker PB and the IPO diphone set of speaker HZ. The phoneme durations have been standardised for both diphone sets and both sets only contain diphones from accented (nonsense)words. Diphones from unaccented words (speaker PB) are not included because they do not contribute to a more natural sounding synthesis (Drullman, Collier, 1993). The diphones are sampled at 10 kHz and have been LPC-coded in 10 ms frames of 5 pairs of 2nd order filter parameters (p, q parameters) and a fixed de-emphasis of 0.9. For further information on the file format see Allain (1990).

A diphone database on the VAX is a so-called key-indexed file where fast access to the individual diphones is performed by means of a key, in our case the name of the diphone, e.g. 'A1S1'. Because this file format is VAX specific and does not exist on a PC, a hash table is used to find the correct diphone in the diphone database. The hash table is held in the file 'difonen.lst'. A hash table points to a starting location in the diphone database 'difonen.bin' which is not necessarily unique for one diphone. From this point the diphone database is searched for the correct diphone. In practice the address points mostly directly to the wanted diphone.

Several tools take care of the transformation of the VAX diphone database to the hash table indexed PC database:

V2P (VAX) [source: V2P.PAS]: Create key-indexed file DIFONEN.BIN.
vax2pc [source: vax2pc.c]: Transform VAX format diphone database 'vaxdif.bin' to the PC format database 'difonen.bin'.
hashdip [source: hashdip.c]: Generate hash table 'difonen.lst' for diphone database 'difonen.bin'.
dump_dip [source: dump_dip.c]: Display the parameters of individual diphones in detail. Takes as arguments 'difonen.bin', 'difonen.lst' and an ASCII file with a list of diphone names to be displayed.

A complete conversion scenario is as follows:

At the VAX:

- Make an ASCII file called 'DIFONEN.LIST' containing a list of all diphones.
- Define a logical VAXIND, pointing to the VAX-indexed file in which the diphones are stored.
- Run the Pascal-programme 'V2P.PAS' creating 'DIFONEN.BIN'.

At the PC:

- Get the VAX-file 'DIFONEN.BIN' to the PC by ftp (binary mode) and rename it to 'vaxdif.bin'.
- Run the programme 'vax2pc', converting 'vaxdif.bin' into the PC diphone database 'difonen.bin'.

- Run the programme 'hashdip' generating hash table file 'difonen.lst'.
- Copy and eventually rename both files 'difonen.bin' and 'difonen.lst' to the diphone database directory (e.g. DDB).
- Eventually inspect individual diphones using dump_dip.

The header of the VAX file contains information which is irrelevant for the synthesis, for example the speaker's sex. This information is not copied to the PC format for memory saving reasons. Consequently the structure of one diphone, that is, the header plus a sequence of frames, is different for both formats. One diphone is implemented as an array of shorts. At the PC the first 18 positions are used for the header, and are followed by blocks of 20 positions each specifying one frame. Each parameter is located at a fixed position in the array which is a different position in the VAX format. The exact meaning of each position at the PC side is given below with the corresponding position in the VAX format in brackets. The profile of one line is: dip[*position*]=*example value* (*VAX position*) *Meaning.*

Header:

dip[0]	Not used	
dip[1]=4	(none)	Number of symbols for diphone identification
dip[2]=65	(none)	symbol for identification, e.g. 'A'
dip[2]=49	(none)	symbol for identification, e.g. '1'
dip[2]=83	(none)	symbol for identification, e.g. 'S'
dip[2]=49	(none)	symbol for identification, e.g. '1'
...		maximum 8 characters for identification
dip[10]=1	(none)	0: do synthesize, 1: do not synthesize (not used)
dip[11]=100	(111)	default frame duration in samples
dip[12]=80	(none)	duration of the first phoneme in samples per frame
dip[13]=120	(none)	duration of the second phoneme in samples per frame
dip[14]=6	(122)	frame number indicating start of second phoneme
dip[15]=15	(35)	total number of frames
dip[16]=14	(121)	frame number where smooth interval starts
dip[17]=3	(123)	frame number where smooth interval ends

Frame: (contents and order identical to VAX format)

dip[18]		negative pitch period in samples
dip[19]		gain factor
dip[20]		RMS (not used)
dip[21]		voiced/unvoiced (20 means voiced/ 200 means unvoiced)
dip[22]		-14744 (-0.9 * 16383 represents pre-emphasis value, fixed at -0.9)
dip[23]	Not used	
dip[24]-dip[33]		5 second order p, q filter parameter pairs
dip[34]-dip[37]		not used

Some remarks have to be made about header parameters 12 - 15. Since some parameters indicate a frame number, and not a number of frames, it is important to know the start number. It is decided that the counting of frames starts at 1. Since dip[14] indicates the frame number where the second phoneme starts and taking into account the fact that each phoneme part consists of at least one frame, the following condition is always valid:

$$2 <= dip[14] <= dip[15]$$

In the procedure Frame Duration the values dip[12] and dip[13] are calculated such that the total duration in the corresponding phoneme parts of two adjacent diphones, equals the calculated duration of the corresponding phoneme. That is:

$$dip[12]*dip[14] + dip[13]*(dip[15]-dip[14]) = \text{phoneme duration}$$

References

Allain, P. (1990). LVS Programmer's Manual. IPO internal report, no. 107.

Boves L., Kerkhoff J., Loman H. (1987) "A new synthesis model for an allophone based text to speech system", Proc. Europ. Conf. on Speech Technol. Vol 2, p385-388.

Boves, L. (1991). Considerations in the design of a multi-lingual text-to-speech system. Journal of Phonetics 19, pp. 25-36.

Carlson, R. & Granström, B. (1976). A text-to-speech system based entirely on rules. In Proceedings of ICASSP '76, pp. 686-688. Philadelphia, April '76.

Coile, van B.M.J. (1987). A model of phoneme durations based on the analysis of a read Dutch text. In Proceedings of the European Conference on Speech Technology '87, Vol 2, pp. 233-236. Edinburgh, September '87.

Coile, van B.M.J. (1989). The depes development system for text-to-speech synthesis. In Proceedings of ICASSP, May 1989.

Drullman, R & Collier, R. (1991). On the combined use of accented and unaccented diphones in speech synthesis. Journal of the Acoustical Society of America, 90, 1766-1775.

Elsendoorn, B.A.G. & 't Hart, J. (1982). Exploring the possibilities of speech synthesis with Dutch diphones. IPO Annual Progress Report 17, pp. 63-65.

Heemskerk, J.S.M. (1993). MORPA, a morpheme lexicon based MORphological PArser. In Analysis and Synthesis of Speech, Strategic research towards high-quality text-to-speech generation (van Heuven, V.S. & Pols, L., eds). Mouton de Gruyter, Berlin.

Kerkhoff, J., Wester, J. & Boves, L. (1984). A compiler for implementing the linguistic phase of a text-to-speech conversion system. In Linguistics in the Netherlands (Bennis, H. & Van der Kloecke, W.U.S., eds), pp. 111-117. Foris, Dordrecht.

Kerkhoff, J. & Wester, J. (1987). FONPARS1 User Manual; Part 1: Rule Format. Internal publication Institute of Phonetics, Nijmegen University.

Klatt, D.H. (1980). Software for a cascade/parallel formant synthesizer. Journal of the Acoustical Society of America, 67(3), March 1980.

Lammens, J.M. (1993). Multiple paths to the lexicon. In Analysis and Synthesis of Speech, Strategic research towards high-quality text-to-speech generation (van Heuven, V.S. & Pols, L., eds). Mouton de Gruyter, Berlin.

Loman H., Kerkhoff J., Boves L. (1989). A working environment for speech synthesis by rule. In AFN-proceedings 13, p51-63, KUN (Nijmegen).

Loman, H. & Boves, L. (1993). Development of allophone synthesis for Dutch. In Analysis and Synthesis of Speech, Strategic research towards high-quality text-to-speech generation (van Heuven, V.S. & Pols, L., eds). Mouton de Gruyter, Berlin.

Markel, J.D. & Gray, A.H. (1976). Linear Prediction of Speech. Springer, Berlin.

Nunn, Anneke M., van Heuven, Vincent J. (1993). MORPHON: Lexicon-based text-to-phoneme conversion and phonological rules. In Analysis and Synthesis of Speech, Strategic research towards high-quality text-to-speech generation (van Heuven, V.S. & Pols, L., eds). Mouton de Gruyter, Berlin.

Rijnsoever, van P.A. (1988). A multilingual text-to-speech system. IPO Annual Progress Report 23, 1988.

Terken, J.M.B. (1993). Synthesizing natural sounding intonation for Dutch: rules and perceptual evaluation. Com-

puter Speech and Language, 7, 27-48.
Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. (1991). Object-Oriented Modeling and Design. Prentice-Hall International, Inc., New York.
van Leeuwen, H.C. & te Lindert, E. (1993). Speech Maker: a flexible and general framework for text-to-speech synthesis, and its application to Dutch. Computer Speech and Language 7, 149-167.
Vogten, L.L.M. (1983). Analyse, zuinige codering en resynthese van spraakgeluid. Ph.D. thesis, Eindhoven University of Technology.

Appendix A: Index to functions

Appendix B: KUN, IPO and SAM-PA phoneme notations

Dutch graphemes	KUN	IPO	SAM-PA
mAt	A	A	A
bEdoel	&	C	@
pUt	U	CC	Y
lEs	E	E	E
pIt	I	I	I
rOt	O	O	O
rOEt (!)	y	U	u
fUUt (!)	u	Y	y
lIEp	i	II	i
mAAt	a	AA	a:
lEEs	e	EE	e:
kEUs	@	OE	2:
rOOd	o	OO	o:
kOUd	AU	AU	Au
rEIs	EI	EI	Ei
mUIs	UI	UI	9y
crEme	E:	EH	E:
zOne	O:	OH	O:
frEUle	U:	UH	9:
Pas	p	P	p
Tas	t	T	t
Kas	k	K	k
Bas	b	B	b
Das	d	D	d
Goal	G	G	g
Lang	l	L	l
Rang	r	R	r
Jan	j	J	j
Wang	w	W	w
Fok	f	F	f
Sok	s	S	s
SJaak	S	SJ	S
Gok	x	X	x
Veer	v	V	v
Zeer	z	Z	z
bagaGe	Z	ZJ	Z
Hang	h	H	h
Meer	m	M	m
Neer	n	N	n

baNG	N	Q	N
silence	,	SI	.
glottal stop	c	GS	?

No SAM-PA phonemes:

poTJe	C	TJ
fraNje	!	NN

Only KUN phonemes:

iNgaan	~
aaNwas	*
vraGen	g

Appendix C: A short user's guide

The stand-alone system as well as the PolyGlot system with diphone synthesis in combination with real-time speech output using the DSP board, works well on a PC-486, 33 MHz machine or higher.

The stand-alone system, ds_ipo.exe

The program `ds_ipo.exe` is started as follows:

```
ds_ipo <difonen.bin> <difonen.lst> [options]
```

`<difonen.bin>` and `<difonen.lst>` point at the diphone database (with extension .bin) and the diphone index list (with extension (lst).These arguments can be omitted by using a dos-key, e.g. the one defined in `doskey.bat`. The program comes up with the prompt:

```
ds ->
```

Now, Dutch text can be typed or text can be read from file by redirection:

```
dsn < tekstfile.
```

The program can be quit by entering a period (.).

Command-line options:

-? prints a help page to the screen.

-i No intonation rules, default: on. Intonation rules add a simple declination to the speech. Option -i results in monotonous speech of 100 Hz.

-c Concatenation of diphones. By default duration rules are applied. This option is useful to synthesize without duration- and intonation rules. In practice, option -c is always combined with option -i. Using both options no duration rules are applied but diphones are concatenated with a default duration of 10 ms per frame. If intonation rules are used a default duration of 40 ms per phoneme is used.

-k input of KUN phonemes instead of text. (accents: see -a)

-p input of IPO phonemes instead of text. (no accents)

-d input of diphones instead of text.

-a accents can be added to the input. This option **only** works in combination with KUN phonemes input (-k). Accents:

sentence accent: ‘

word accent: ’

-f Output to file. The generated speech signal is written to the file speech.raw (raw data, 2 bytes per sample, little endian binary file). All calculations now take place at the PC, providing the possibility to use the program without a DSP-board.

The PolyGlot system

Two options have been added to the functionality of the system:

-i Use diphone synthesis

-r Write the diphone synthesis result to file `speechdi`

The first option replaces the allophone synthesis by diphone synthesis. The second option pro-

vides the possibility to write the result to file. Using this option the result is only written to file, no speech is generated by the DSP board in parallel.

The file speechdi has a header of 128 long integers (512 bytes) containing the sample rate at position 125 (byte 500) and the number of 2 byte samples at position 126 (byte 504). This is the same format as the allophone speech output file TS1.

Appendix D: Functional diagrams

The functional diagrams used in this report follow the syntax of the OMT method according to Rumbaugh, 1991, except for some minor modifications. The most essential part of the syntax is described in this appendix (figure 9).

A process changes data. In principle, each functional diagram that is a part of the whole system is module. A module consists of other modules, specified in a separate diagram, and, non-dividable processes. The name of a module is displayed in the top right corner of the diagram. Since all lower level modules finally will end up in processes the exact separation between a module and a process is rather arbitrary to the designer.

Three different data stores are used. A major data store contains more than one independent data streams. A minor data store contains at most one data stream and eventually one value parameter. A parameter contains exactly one value.

A data store can be accessed, updated or both. A data store that merely delivers data to the system is called an actor. A data store only receiving data is called a terminator.

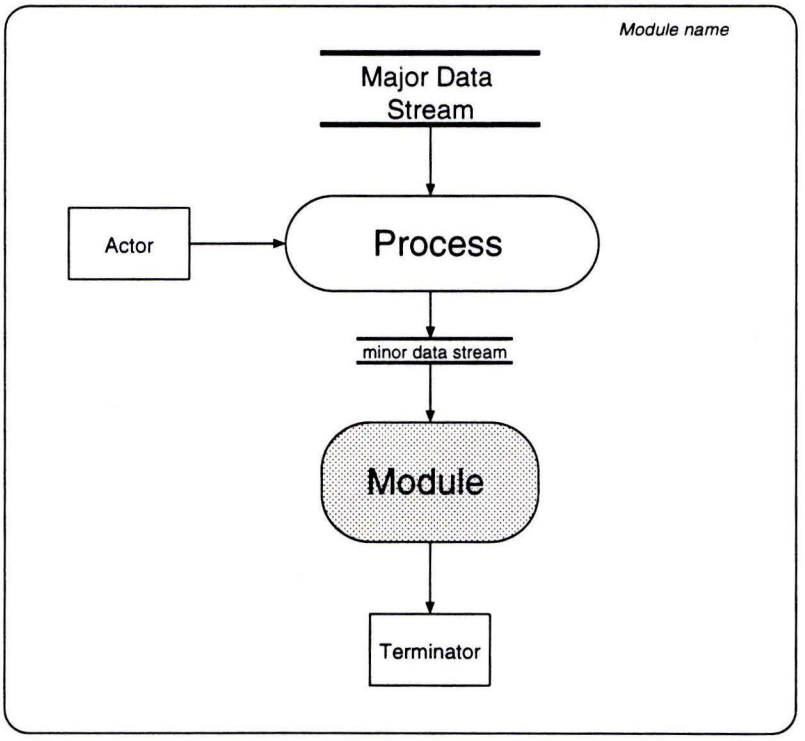

Figure 9. : Syntax of functional diagrams.