TR-IT-0252

Improved input methods for CHATR

Sébastien Voyard

January 1998

ABSTRACT

In this report, I describe the work I did during the last six months. After a brief overview of the way CHATR works, especially in the field of prosody and multi-lingual input, I explain how we can make the speech sound more human with the use of a parser. Then I introduce interactive prosody prediction, and discuss the functionalities that should be present in CHATR to achieve this goal. Finally, I describe the work done with multi-lingual input, and show how texts containing more than one language can be processed.

Improved input methods for CHATR

Sébastien Voyard

ATR-ITL, January 1998

Acknowledgements

The information contained in this document is the result of a six month internship I spent in ATR. It has been a great opportunity for me, to discover a country I really wanted to know, and to improve my Japanese.

I would like to thank the people who helped we to come to Japan, especially Mr Guedj from INT, without whom all this would not have been possible.

I am also very grateful to everybody in ATR, who have been so kind.

Special thanks are due to Nick Campbell, who supervised my work and has been very helpful with all his advice. The whole department 2, shall not be forgotten. It has been a real pleasure to work here.

Thanks to ATR Foreign Staff Support Group, who were always ready to answer any question.

Contents

Chapter 1

What is CHATR?

CHATR is a speech synthesis system developed by department 2 of ITL (Interpreting Telecommunications Research Laboratories) at ATR (Advanced Telecommunications Research Institute). It is a multi-speaker and multi-language system. That is to say that the same text can be spoken with different voices (male, female and even child voice), and that different language can be processed (English, Japanese, German, Chinese, Korean). It differs from conventional speech synthesis in that it re-uses actual reading from a human speaker without recourse to signal processing.

1.1 Input

Specifically CHATR is not a text-to-speech system, though it does support a simple text input module. CHATR's main use in ATR will be within a speech translation system. To turn translated speech into plain text and then expect a text-to-speech system to recreate all the information that has just been thrown away is a waste of resources. The language generation system already has a labeled form of the utterance and it is this far richer form of the utterance that is intended as the input to the CHATR system - though many other types of input are possible.

Input to CHATR is in the form of an utterance created by the Utterance command. Several types of input may be specified at quite different levels, varying from raw text to a simple waveform. The current possibilities are following.

1.1.1 Text utterance

The Text utterance type is the most basic type of input, it consists of just raw text with no additional information.

```
(Utterance Text "You can pay for the hotel with a credit card.")
```

1.1.2 HLP utterance

The HLP utterance type is a high level "linguistic" structure. A sentence is represented by a tree, where each node brings information about syntax or speech type. The leaves of the tree are words. At the word level a series of features can be attached to each word to bring other information such as Focus.

```
(Utterance
        HLP
        (((CAT S) (IFT Statement))
         (((CAT NP) (LEX you)))
         (((CAT VP))
          (((CAT Aux) (LEX can)))
          (((CAT V) (LEX pay)))
          (((CAT PP))
           (((CAT Prep) (LEX for)))
           (((CAT NP))
            (((CAT Det) (LEX the)))
            (((CAT N) (LEX hotel)))))
          (((CAT PP))
           (((CAT Prep) (LEX with)))
           (((CAT NP))
            (((CAT Det) (LEX a)))
            (((CAT Adj) (LEX credit) (Focus +)))
            (((CAT N) (LEX card)))))))))
```

The attribute CAT determines the category of the node at each level.

- S stands for Sentence
- xP for Phrase
  - NP correspond to Noun Phrase
  - VP to Verbal Phrase
  - PP to Prepositional Phrase
- many features describe the word itself
  - Noun
  - Verb
  - Aux
  - Det
  - Prep

- Adj

The attribute IFT (Illocutionary Force Type) identifies the nature of the sentence. Currently, CHATR recognizes Statement, Question, YNQuestion and Interjection.

This kind of utterance is far more detailed and we can expect better output. However it is very time consuming and error-prone to type it in directly.

1.1.3 PhonoWord utterance

PhonoWord utterance uses a lower level representation. It is again a tree, but the depth is limited. There are only four levels: Discourse, Sentence, Clause and Phrase. You can add intonation information for each word in various formats including ToBI. In fact you can apply any feature on any word. Here is an example:

```
(Utterance
        PhonoWord
        (:D ()
            (:S ()
                (:C ()
                    (marianna (H*))
                    (made)
                    (the)
                    (marmalade (H*) (L-L%))))))
```

1.1.4 PhonoForm utterance

This format allows specification of prosodic phrases, words, syllables, intonation labels, segments, duration, power and pitch. In fact almost everything that CHATR itself might predict during synthesis of a text string. This form is typically used to represent full database information. Here is an example:

```
(Utterance
  PhonoForm
  (:D nil
    (:S ((PauseLength 65))
      (Word Attorney nil
       (Syl ax () (Phoneme ax 70 8.5100 ((187.0000 35))))
       (Syl t.er ((Stress 1) (Intones HiF0 H*))
        (Phoneme t 110 7.1200 ((242.0000 55)))
        (Phoneme er 80 8.7500 ((255.0000 40))))
       (Syl n.iy nil
```

```
      (Phoneme n 50 8.6700 ((233.0000 25)))
      (Phoneme iy 60 8.3400 ((193.0000 30)))))
    (Word General ((Break 1))
     (Syl d.jh.eh.n ((Stress 1) (Intones !H*))
      (Phoneme d 60 7.7300 ((173.0000 30)))
      (Phoneme jh 40 7.4100 ((226.0000 20)))
      (Phoneme eh 110 8.4300 ((205.0000 55)))
      (Phoneme n 30 8.2700 ((196.0000 15))))
        (Syl axr () (Phoneme axr 130 8.2600 ((158.0000 65))))
        (Syl el ((Intones L-H%))
      (Phoneme el 110 7.9000 ((180.0000 55))))))
  (:S nil
    (Word James nil
     (Syl d.jh.ey.m.z ((Stress 1) (Intones H*))
      (Phoneme d 70 7.0900 ((182.0000 35)))
      (Phoneme jh 50 7.0800 ((184.0000 25)))
      (Phoneme ey 150 8.2200 ((154.0000 75)))
      (Phoneme m 100 7.7600 ((143.0000 50)))
      (Phoneme z 30 6.7700 ((200.0000 15)))))
    (Word Shannon ((Break 1))
     (Syl sh.ae.n ((Stress 1) (Intones HiF0 H*))
      (Phoneme sh 90 6.9900 ((200.0000 45)))
      (Phoneme ae 150 8.3700 ((172.0000 75)))
      (Phoneme n 80 8.1000 ((144.0000 40))))
     (Syl ax.n ((Intones L-L%))
      (Phoneme ax 30 7.4400 ((104.0000 15)))
      (Phoneme n 50 7.1100 ((145.0000 25)))))))))
```

1.1.5 Phoneme utterance

The Phoneme input format is a simple list of words along with small integers identifying break points. These break indices go from 1 to 5, with 5 corresponding to a strong break like at the end of a sentence. With this kind of input you can type in any language, using the phoneme set of the speaker. See the following examples (English and Japanese).

```
(Utterance Phoneme
  (you can pay 3 for the hotel 3 with a credit card 5))

(Utterance Phoneme
  (sakujitsu'amega furima'shita 5 de'mo 1 kyo'uwa i'i te'Nkidesu 5))
```

1.1.6 other utterance types

Some other types of utterances exist such as Segment, SegF0, RFC, Syllable or Wave but they are of no relevance to this report. Some further information about them can be found in CHATR's manual [6].

1.2 Prosody

1.2.1 What is Prosody?

Prosody is beyond what we say; it's how we say it. As an example, take the sentence: "Could I have your phone number too?".

The meaning can be changed completely depending on intonation used and which words are stressed. The following lines shows different possibilities, where the stressed words appear in bold.

1. Could **I** have your phone number too?
2. Could I have **your** phone number too?
3. Could I have your **phone** number too?
4. Could I have your **phone number** too?
5. Could I have your phone number **too**?

These five sentences each have a different meaning. The first one could mean that somebody already has your phone number and that I want it too, whereas in the second one, I already have other people's phone number, but I want yours particularly. In the third example, I could have already your fax number, but I also want the phone number. And so on.

So as you can see from this example, a single sentence can have many meaninBeyond skeleton parsing: producing a comprehensive large–scale general–English treebank with full grammatical analysis.gs. If you take this sentence without any context you will not be able to decide which one is the most appropriate.

Each time we stress a word while speaking the f_0 value rises. So to have a word stressed, you have to predict a higher f_0 value. But you must have also noticed that when a word is really important, you slow down when you are pronouncing it, whereas you speed up on words that do not bring much information such as articles and so on. Some may say that you could also speak louder; usually it is not for the same purpose and this possibility should not be thrown away.

1.2.2 Prediction within CHATR

From the information given as input, CHATR will predict the prosody. To each utterance is attached various streams concerning different kinds of informations and linked together (WordStream for words, SylStream for syllables or IntoneStream for intonation). The prosody prediction works with the IntoneStream, which get filled and enriched at many steps of the processing. CHATR currently uses the ToBI intonation system, and its variant in several languages (that is JToBI for Japanese and GToBI for German). For each syllable of the Sylstream, pitch accent(H*, L*, ...), phrase accents (H-, L-, ...) and boundaries tones (L%, ...) are predicted. See [2] for information about the ToBI system. Then according to these ToBI labels, the f_0 contour is predicted.

1.2.3 Using a parser

To make English sentences sound more human, the use of a parser could be of great utility. The work already done, concerning mainly pause prediction, leads to better results in prosody prediction.

The parser used for this study has been developed by department 3[3]. The input is an English sentence. The output is a very detailed tree which contains phrasal and lexical information at each node. Here is an example of the output of the parser:

```
[start [sprpd1 [sprime1 [sd1 [nbar1 [n1a Water_NN2PERSON n1a]
 nbar1] [vbar1 [v2 was_VBDZ [nbarq4 [nbar1 [n1a
 cascading_NN1DEVICE n1a] nbar1] [i1e [p1 down_IIDOWN [nbarq4
 [nbar4 [d1 the_AT d1] [n1a mountain_NN1PLACE n1a] nbar4]
 [i1e [p1 at_IIAT [nbarq4 [nbar4 [d1 a_AT1 d1] [n1a
 rate_NN1MEASURE n1a] nbar4] [i1e [p1 of_IIOF [nbar1 [n1a
 knots_NN2MONEY n1a] nbar1] p1] i1e] nbarq4] p1] i1e] nbarq4]
 p1] i1e] nbarq4] v2] vbar1] sd1] sprime1] ._. sprpd1] start]
```

The work that had been done by a previous INT student was to convert the output of the parser into a format understandable by CHATR, and extract information to make better prediction. This is done through the Skel_to_PhonWord program. This output is then obviously a PhonoWord utterance. The algorithm is described in Tony Hebert's report [5].

1.3 Overview

The following graph shows the different steps of the processing of an utterance.

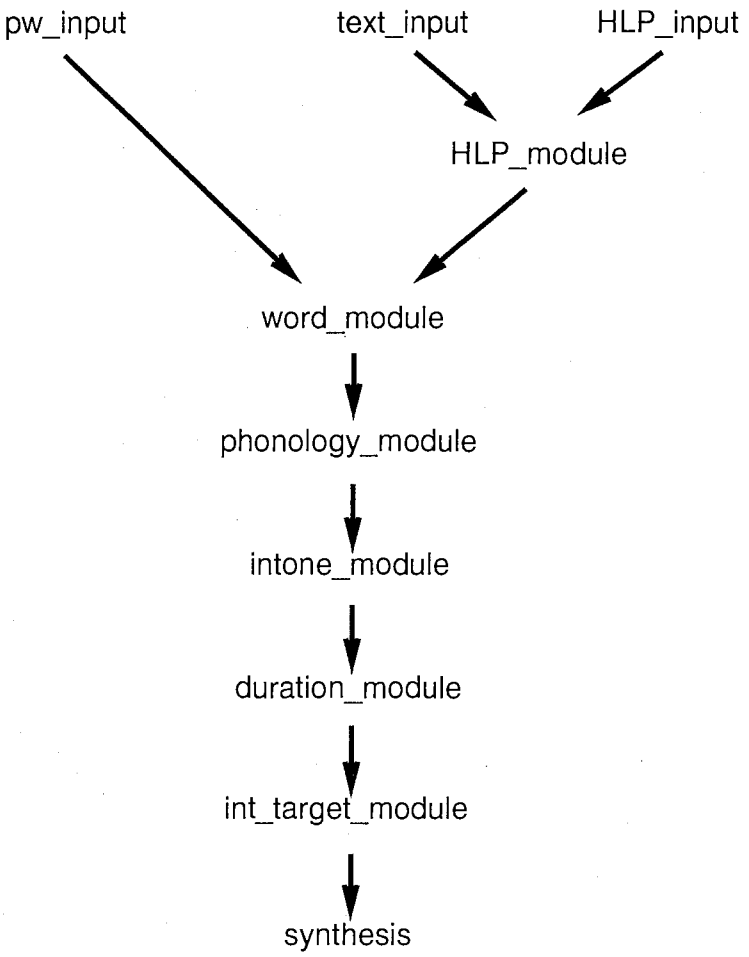
pw_input
text_input
HLP_input
HLP_module
word_module
phonology_module
intone_module
duration_module
int_target_module
synthesis

Figure 1.1:

Chapter 2

Multilingual Input

CHATR can speak many languages, using different speakers. But any voice should be able to speak any language. As far as the phoneme set are completely different, we cannot expect a Japanese speaker to be mistaken for an English one. The Japanese language consists of much fewer phonemes than the English one, so not every English sound is reproducible in Japanese. However, it will sound like a foreigner speaking in this language. Moreover, a single speaker should be able to speak a multi-lingual text, shifting the language each time it is necessary.

To give a simple example of an application, you will perhaps want to have your e-mail read, using CHATR, and what you receive may consist of English text or Japanese text or both.

2.1 Mtts

The way to deal with multi-lingual text is to use a CHATR text-to-speech module(tts) made for multi-lingual input: mtts. This was able to process either English or Japanese. This module considered that Kanji and Kana was Japanese text whereas Romaji was English. With only two such languages the difference between each is very simple to detemine. You can go on adding other languages like Chinese or Korean without any problem as long as the character set is different each time. But real problems come when you want to deal with several Romaji languages; you need some deeper search to determine which language is used. For example the use of *umlaut* which is a characteristic of German language, or the use of article like the or *das* could be part of the decision. However this point has to be discussed later. The main drawback of mtts is that every time a switch of language occurs, it corresponds to a switch of speaker. That is to say, English text was spoken by an English speaker and Japanese text by a Japanese speaker.

Mtts takes a file as input, so you cannot edit the Japanese text directly.

To be able to edit Japanese text as well as English, several functions have been implemented for mule (multi-lingual version of emacs). This basically writes to a file and calls mtts.

2.2 Conv_phoneme

Another program has been developed to deal with multi-lingual input. It is called conv_phoneme. I have been using 2 different versions of this program and I will discuss each of them, given that they have their own advantages and drawbacks. Basically, this program takes a multilingual text as input and produces an Utterance Phoneme which can be used by CHATR. The program takes the input string and parses it according to the language type to obtain several lists (one for each language). Then each list is converted to Romaji, using different programs. For instance kan2rom is used for Japanese, and Hcode for Korean. The output string is then reconstructed using the different lists. The output string uses the phoneme set of the selected language.

2.3 From a multilingal text to speech

In this section I will speak mainly about Japanese and English.

2.3.1 about kan2rom

kan2rom is currently used to convert a Japanese text (Kanji and Kana) into Romaji. The output of kan2rom is a list of words with break indices and information concerning the intonation.

For example, with the input:

- 今日は寒いですね。多分雪が降るでしょう。

  (Today is cold. It might snow)

the output is:

- kyo'uwa samu'idesune 5 ta'buN 1 yuki'ga fu'rudeshou 5

As you can see from this example, two other items are brought with the transcription. They are apostrophes and numbers.

Numbers indicates break indices, they show how much a word is linked to the next one. If no number appears the break index is considered to be 0, which means that the two words are strongly linked together. The greater the value, the less the words are linked. A break index of 5 corresponds to the end of a sentence.

Apostrophes are placed after the accented mora (syllable) of each "word".

Figure 2.1 shows the basic intonation contour of a Japanese phrase. On each word of the phrase the f_0, usually rises in the second mora ,and the apostrophe indicates the place where it goes down.

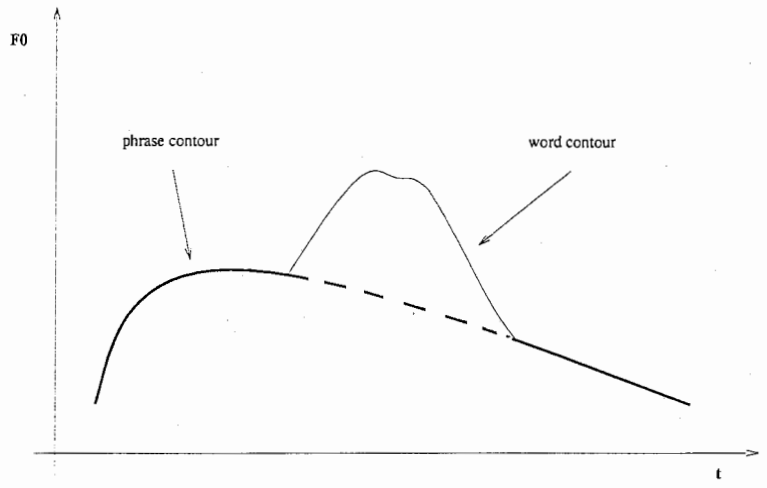

Figure 2.1: Japanese intonation contour

Note that what I called "word" here does not correspond necessarily to real lexical words. It describes a series of characters surrounded by to spaces.

2.3.2 phoneme_input

what it does

Basically, what the phoneme_input function does is to convert its input into another format which is the PhonoWord format. Break indices are replaced by boundaries:

- 1 and 2 give a phrase boundary (P_BOUND)
- 3 gives a clause boundary (C_BOUND)
- 4 and 5 give a sentence boundary (C_BOUND)

For example, the input:

```
(Utterance Phoneme( kyo'uwa samu'idesune 5 ta'buN 1 yuki'ga
fu'rudeshou 5 It's cold today 3 isn't it 5 Perhaps it will snow 5)
```

becomes

```
(:D
 nil
 (:S () (:C () (:P () ((kyo'uwa)) ((samu'idesune)))))
 (:S
  nil
  (:C
   nil
   (:P () ((ta'buN)))
   (:P () ((yuki'ga)) ((fu'rudeshou)))))
 (:S
  nil
  (:C () (:P () ((It's)) ((cold)) ((today))))
  (:C () (:P () ((isn't)) ((it)))))
 (:S
  nil
  (:C () (:P () ((Perhaps)) ((it)) ((will)) ((snow))))))
```

After the phoneme_input module, the word module is called as if the input has been PhonoWord.

2.3.3 word module

The next module called after phoneme input is the word module. This module calls the lexicon module, amongst others. The lexicon used is defined by the speaker currently selected. That is to say, if you are using a Japanese speaker the Japanese lexicon will be used.

This module gives us the lexical stress (which part of the word is accented). Each syllable is linked to a number, 1 if accented, 0 if not. For example the word "intermediate" has the following structure:

```
(intermediate
 (((i n) (1)) ((t @) (0)) ((m ii) (1)) ((d ii @ t) (0))))
```

As long as we are dealing with Multi-lingual text, the processing cannot be correct. For instance, English words will not be recognized by a Japanese lexicon, and the lexical stress field of each syllable will be set to 0. This will cause a lack in prosody prediction.

Chapter 3

My contribution

3.1 Further development on using a parser

After having understood the Skel_to_PhonoWord algorithm, I tried to make some improvements to it and then tried to use the parser more efficiently.

3.1.1 Pausing

The use of the parser led to better predictions of the pause, especially when the input was a very long sentence. However, even if results were better, using the Skel_to_PhonoWord algorithm, some problems remained. As a matter of fact, if a group of word was greater than six a pause was inserted a soon as possible, leading to strange breaks. On the other hand, there was no pause insertion with smaller groups. This event occurred because, when a group was too big, the algorithm decided to split it in two parts if possible. But it tried to cut it from the middle of the group.

The choice of 6 words limit to make groups and then insert pause, is arbitrary, but in my opinion, any choice will lead to strange breaks sometimes; the best we can do is find a limit to make the fewer mistake we can. Considering the second point, improvements were made by having the recursion search to split a big group applied on the whole group (not from its middle). And doing this we had fewer strange breaks. However, the algorithm is still not perfect.

3.1.2 From PhonoWord to HLP

When the PhonoWord format is used, we miss a part of the processing which is HLP processing. Of course the PhonoWord input used at this stage, contains more information than just text, but part of the prosody prediction is done inside the HLP module for text input. And the HLP module is not called for PhonoWord utterances (see Figure 1.1, page 7).

Moreover the output of the parser is very detailed, and it is not used as much as it could be. Therefore, it is perhaps better to use an another kind of input. The most appropriate at this step seemed to be HLP input. As a matter of fact, both formats have the same structure, they represent a tree. And all the information contained in the HLP tree, can be used by the HLP module to make better predictions.

Example of use

With the input

```
[start [sprpd1 [sprime1 [sd1 [nbar4 [d1 The_AT d1]
 [n4 [n1a price_NN1MONEY n1a] [n1a range_NN1CLASS n1a] n4]
 nbar4] [vbar1 [v2 is_VBZ [j11 smaller_JJRDEGREE [fc1
 than_CSN [nbarq13 [nbar9 [d1 any_DD d1] [p1 of_IIOF
 [nbar6 us_PPIO2 nbar6] p1] nbar9] [v1 expected_VVNMENTAL-ACT
 v1] nbarq13] fc1] j11] v2] vbar1] sd1] sprime1] ._. sprpd1]
 start]
```

we get the following HLP utterance

```
(Utterance HLP
           (((CAT S) (IFT Statement))
            (((CAT NP))
             (((CAT Det) (LEX The)))
             (((CAT NP))
              (((CAT Noun) (LEX price)))
              (((CAT Noun) (LEX range)))))
            (((CAT VP))
             (((CAT Verb) (LEX is)))
             (((CAT PP))
              (((CAT Adj) (LEX smaller)))
              (((CAT PP))
               (((CAT Word) (LEX than)))
               (((CAT NP))
                (((CAT NP))
                 (((CAT Det) (LEX any)))
                 (((CAT PP))
                  (((CAT Prep) (LEX of)))
                  (((CAT Noun) (LEX us)))))
                (((CAT Verb) (LEX expected)))))))))
```

3.1.3 Future work

Surely the output of the parser could be used more efficiently, considering the highly detailed output it gives. Currently what is recognized as an HLP Input does not cover all of the features used by the parser. Moreover it concerns only syntactic information, whereas the parser gives us much more.

The first solution would be to increase the number of features recognized by CHATR and assign them a special processing. That is to say improve the HLP module.

A better solution would be to write a new module which takes the parser output directly as an input without any conversion, using as much information as possible. This way, we should be able to save time in processing. But most important, for further development, changes just have to be done in one single place, instead of at each stage of the different conversion.

However the main problem is perhaps that by now the parser is very slow. Therefore it cannot be used as is in CHATR. If this program could run real-time it could be of great use for CHATR.

3.2 Prosody

3.2.1 Interaction with the user

When the user wants to process such a sentence, he cannot expect CHATR to guess what meaning he wants. That's why the intervention of the user is required. In fact the user should be able to process a sentence as many times as he needs and make any necessary prosody adjustment between two processings. This should be realized through a loop. By that, I mean that the user processes the sentence, hears it, changes some parameters, does the processing again until he can hear what he really wants (or as close as possible).

The first step consists in marking words or groups of words to be "read" in a different way (concerning focus or speed). An editing loop should be available at this level.

The second step is to have another editor to refine the prosody. This time you should be able to directly modify f_0, having also another loop.

Graphical User Interface?

I tried to design a graphical interface to be more user-friendly, which could realize the loop described ahead. I called it CHATRface for CHATR interface. I have done this through a new module which is called directly from CHATR. It looks like figure 3.1

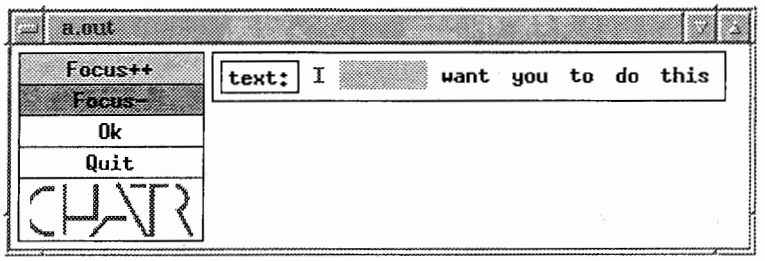

Figure 3.1: CHATRface

You just have to click on a word and then choose the action. The result of the previous processing is shown via different colors. Focusing uses the background color, and speed (speaking rate) uses foreground color. This module should work for any language written in Romaji. To be able to use Kanji and Kana it requires more work.

About the code

In this section I will try to briefly describe the code, to make it understandable, in case somebody wants to do further work on it. This program uses XWindow and Athena Xtoolkit. Although Athena is not very good looking, I choose it, because it was available on my machine, and because it can be used on every machine where XWindow is available. It uses also IPC (UNIX InterProcess Communication) to deal with communications between CHATR and the interface, which I decided should not be part of CHATR.

First of all, the program Chatrface should be running, before the function (Sebd) is called from CHATR. Once an Utterance has been synthesized, a call to (Sebd) will send the information to the program Chatrface waiting. Having this information, a window is created with function buttons(CommandWidget Class), and a button for each word(ToggleWidget Class). A word is selected by changing its state (it appears in reverse video). Then once a command is selected, the information is sent back to CHATR. In fact you can include any function in Chatrface, to be called from the "outside".

(too brief)

3.2.2 A new feature

We have seen in section 1.2.1 that a word can be stressed by raising the f_0 and/or slowing down on it. Another way could be to insert a pause before the stressed word. In fact, if you listen carefully, you can easily notice a short pause in speech just before saying an important word.

This kind of pause is inserted just before the word marked with the new feature (Focus +++) . It will correspond to a stronger stress. f_0 is raised as with the (Focus ++) feature. This works if the pause prediction is done by phrase break (which should be the default in my opinion). Perhaps you will have to set a parameter as follows.

```
(set pause_prediction 'by_phrase_break)
```

The pause length is defined in the Pause Stats parameter as follows.

```
(Stats Pause
  (
    (discourse  500)
    (sentence   350)
    (clause     200)
    (phrase     150)
    (focus      100) <- this is where the parameter is set
  )
 )
```

3.3 Multilingual

The first thing to be done was to make it possible to use conv_phoneme directly. For this purpose, an editor was needed, but this editor should be able to edit Kanji as well as Romaji. We could have written this editor, but it would have taken a great amount of time. Instead we chose to use mule, which combines the functions of a multi-lingual editor and the possibility to create your own functions. With only a few changes, I switched from mtts to conv_phoneme. If you call CHATR from emacs, the text will be entirely spoken by the chosen speaker, after calling conv_phoneme and CHATR.

3.3.1 New functionalities

What is described in section 3.2.1 should be available for every language. We add to include the same functionalities in this kind of input. I chose to use the same escape characters that are used in text input. Transferring the information from the Multi-lingual level to the phoneme level is difficult, so I began to included the escape characters in Phoneme Utterances.

Modification of phoneme_input

The conversion of Phoneme to PhonoWord only uses basic functions. The PhonoWord input can be much more detailed. So we can modify the Phoneme_input function to include new functionalities. In fact for each word you can include

ToBIlabels or features. So I made modifications to enable the same functionalities as for text input and some more. Now you can enter options before a "word" to be applied to it. Options are entered between two '/'and can be the following:

- + to insert a (Focus ++) feature
- - to insert a (Focus -) feature
- * to insert a (Focus +++) feature
- f to insert a (OPT Fast) feature
- s to insert a (OPT Slow) feature

two more are available although their feature have not been implemented yet

- h to insert a (Power +) feature
- l to insert a (Power -) feature

If the feature (Focus ++) is set on a word, f_0 values will be raised, with (Focus -) they will be decreased, and (Focus +++) has the same effect on f_0 but a pause is inserted before he word. (OPT Fast) and (OPT Slow) are used to regulate the speed. (Power +) and (Power -) should be used to raise or decrease power on the concerned word.

As an example:

```
(Utterance Phoneme(It's /+/ cold today 3
  /f/ isn't it 5 Perhaps it will snow 5))
```

will be converted into

```
(:D
 nil
 (:S
  nil
  (:C () (:P () ((It's)) ((cold (Focus ++))) ((today))))
  (:C () (:P () ((isn't (OPT Fast))) ((it)))))
 (:S
  nil
  (:C () (:P () ((Perhaps)) ((it)) ((will)) ((snow))))))
```

3.3.2 Remaining problems

As mentioned above, the transfer of information from Japanese text to Utterance Phoneme, is somehow difficult. This comes from the fact that lexical words are not usually separated by spaces, and we may want to stress a word which will be surrounded by other words with no space. The problem is to recognize the "word" in the series of characters.

At the present time, almost everything in CHATR has been designed for a monolingual input. Therefore, almost every module takes an Utterance and applies the same processing to the whole utterance regardless of what is inside. It is just done according to the parameters which were loaded when the speaker was loaded (for example: intonation method, lexicon).

From the two versions of conv_phoneme that I know, one leaves English text as it is whereas the other converts it to the phoneme set of the language selected. In the first case, when using a Japanese speaker it will sound like anything but english. It is spoken as if it was japanese Romaji."My name" will sound like "みなめ". In the second case, which is prefered, when the lexicon module is called, the lexical stress will be missing, as words will not be recognized.

To use other words the processing is language dependent until the target module (at least the duration module) is called.

Future work

In my opinion, there should be a multi-lingual Input module which takes a multi-lingual text as an input and gives as output something closer to PhonoForm. With this format, you can specify either ToBI labels or the predicted f_0 contour directly. As soon as the method to make the target is the same for Japanese, English and German, i.e. by using linear regression based on ToBI labels. I think just having units and ToBI labels should be enough. With this kind of input the only thing to do is unit selection according to the targets.

Appendix A

Variable setting

- The pause_prediction_method must be set to "by_phrase_break"

  (set pause_prediction 'by_phrase_break)

- In order to use all the information contained in the HLP input, especially when using the Skel_to_HLP program, the HLP_phrase_strategy must be set to "Bachenko_Fitzpatrick

  (set HLP_phrase_strategy 'Bachenko_Fitzpatrick))

- It is important to check if the HLP_Patterns and HLP_Rules variables are non nil

- To use the features (Focus ++), (Focus -), (OPT Slow), (OPT Fast), you have to set the variable Tony_Params

- If you want to use the (Focus +++) feature you have to set the duration of the pause with the attribute focus of the Stats Pause

```
(Stats Pause
  (
    (discourse  500)
    (sentence   350)
    (clause     200)
    (phrase     150)
    (focus      100)
  )
 )
```

Appendix B

Skel_to_HLP

B.1 tree.h

th source code can be found under: x̃svoyard/src/lib

```
#ifndef _SEB_TREE_H
#define _SEB_TREE_H

typedef struct noeud * arbre;
typedef struct fils * pfils;
struct noeud {
  char LEX[256];
  char CAT[16];
  int nb_fils;/* peut-etre redondant */
  pfils branche;
  arbre pere;
  int phrlc;/* to add PhraseLevel :C */
};

struct fils {
  arbre objet;
  pfils suivant;
};

void affiche(arbre Stree);
void affiche2(arbre Stree,int st);
arbre newtree();
void freetree(arbre A);/* to free the whole tree */
void freenode(arbre A);/* to free the node only */
arbre newbranche(arbre A);/* l'arbre retourne sert juste pour le marker*/
void upnode(arbre *A);
```

```
void trslate(char ent[]);
void mvbranche(arbre source, arbre dest);
void cleantree(arbre A);
void edit_sentence(arbre Stree);
void addbreak(arbre A);

#endif
```

B.2 tree.c

Source is under: x̃svoyard/src/lib

```
#include <stdio.h>
#include "tree.h"

void affiche(arbre Stree)
{
pfils courant;
  if (Stree->branche==NULL)
    {
/*       printf("%s \n",Stree->LEX); */
      trslate(Stree->CAT);
      printf("(((CAT %s) (LEX %s)))\n",Stree->CAT,Stree->LEX);
    }
  else
    {

      if(Stree->nb_fils>1)
        {
switch((Stree->CAT)[0])
  {
             case 'n':
               printf("(((CAT NP)");
               break;
             case 'v': case 'o':
               printf("(((CAT VP)");
               break;
             case 's':
               printf("(((CAT S)");
               break;
             default:
               printf("(((CAT PP)");
               break;
```

```
            }
if(Stree->phrlc) printf(" (PhraseLevel :C)");
printf(")\n ");
        }
      courant=Stree->branche;
      while (courant!=NULL)
        {
          affiche(courant->objet);
          courant=courant->suivant;
        }
      if(Stree->nb_fils>1) printf(")");
    }
}

void affiche2(arbre Stree,int st)
{
pfils courant;
printf("(Utterance HLP\n(((CAT S) (IFT ");
switch(st) {
 case 1: printf("Statement))\n");
   break;
 case 2: printf("Question))\n");
   break;
 case 3: printf("Interjection))\n");
   break;
 default: printf ("Statement))\n");
   break;
 }
affiche(Stree);
printf("))\n");
}

arbre newtree()
{
arbre aux=(arbre)malloc(sizeof(struct noeud));
aux->branche=NULL;
aux->pere=NULL;
aux->nb_fils=0;
aux->phrlc=0;
return aux;
}
void freetree(arbre A)
{
```

```
pfils courant,aux;
  if (A->branche==NULL)
    free(A);
  else
    {
      courant=A->branche;
      while (courant!=NULL)
        {
          freetree(courant->objet);/*tree was missing*/
          courant=courant->suivant;
        }
      courant=A->branche;
      while (courant!=NULL)
        {
          aux=courant;
          courant=courant->suivant;
          free(aux);
        }
      free(A->LEX);free(A->CAT);/* not sure */
      free(A);
    }
}
void freenode(arbre A)
{
free(A);
}

arbre newbranche(arbre A) {
pfils aux;
if (A->nb_fils==0) {
  A->branche=(pfils)malloc(sizeof(struct fils));
  A->nb_fils=1;
  (A->branche)->objet=newtree();
  (A->branche)->suivant=NULL;
/* peut-etre d'autres initialisations */
  (A->branche)->objet->pere=A;
  return (A->branche)->objet;
}
else {
  (A->nb_fils)++;
  aux=A->branche;
  while (aux->suivant!=NULL) aux=aux->suivant;
  aux->suivant=(pfils)malloc(sizeof(struct fils));
```

```
  (aux->suivant)->objet=newtree();
  (aux->suivant)->suivant=NULL;
  aux->suivant->objet->pere=A;
  return (aux->suivant)->objet;}
}

void upnode(arbre *A) {
if ((*A)->pere!=NULL) *A=(*A)->pere;
else perror("upnode: already at the top");
}

void trslate(char ent[])
{
char sor[16];
if (ent[0]=='n')
  strcpy(sor,"Noun");
else if (ent[0]=='j')
  strcpy(sor,"Adj");
else if (ent[0]=='d')
  strcpy(sor,"Det");
else if ((ent[0]=='o')||(ent[0]=='v'))
  strcpy(sor,"Verb");
else if ((strcmp(ent,"p")==0)||(strcmp(ent,"pr")==0)
||(strcmp(ent,"pzero")==0))
  strcpy(sor,"Prep");
else if ((strcmp(ent,"r")==0)||(strcmp(ent,"rqzero")==0)
||(strcmp(ent,"rzero")==0))
  strcpy(sor,"Adv");
else strcpy(sor,"Word");/* a modifier :( */
strcpy(ent, sor);
}

void mvbranche(arbre source, arbre dest)
{
pfils aux;
if (dest->nb_fils==0) {
  dest->branche=(pfils)malloc(sizeof(struct fils));
  dest->nb_fils=1;
  (dest->branche)->objet=source;
  (dest->branche)->suivant=NULL;
}
else {
  (dest->nb_fils)++;
```

```
  aux=dest->branche;
  while (aux->suivant!=NULL) aux=aux->suivant;
  aux->suivant=(pfils)malloc(sizeof(struct fils));
  (aux->suivant)->objet=source;
  (aux->suivant)->suivant=NULL;
}
/* !!!enlever source a son pere!!! */
if (source->pere->branche->objet==source){
  source->pere->branche=source->pere->branche->suivant;
  /*manque un free*/
}
else {
  aux=source->pere->branche;
  while (aux->suivant->objet!=source) aux=aux->suivant;
  aux->suivant=aux->suivant->suivant;
  /*manque un free*/
}
(source->pere->nb_fils)--;
source->pere=dest;
/* peut-etre d'autres initialisations */

}

void cleantree(arbre A)
{
pfils courant,aux,aux2;

  if (A->branche==NULL)
    {
   /*nothing to do at the moment*/
    }
  else
    {
      if((A->nb_fils==1)&&(A->pere!=NULL))
        {
          aux=A->pere->branche;
          while(aux->objet!=A) aux=aux->suivant;
          aux->objet=A->branche->objet;/*remove non-useful nodes*/
          aux->objet->pere=A->pere;
          aux->objet->phrlc=((A->phrlc)||(aux->objet->phrlc));/*transfert au fil
          cleantree(A->branche->objet);
          free(A->branche);/* was missing */
          freenode(A);
```

```
        }
      else
        {
          courant=A->branche;
          if ((A->CAT[0]=='s')&&(A->pere!=NULL))
            {

              strcpy(A->CAT,"p");
            }
          while (courant!=NULL)
            {
              cleantree(courant->objet);
              courant=courant->suivant;
            }
        }
    }
}

void edit_sentence(arbre Stree)
{
pfils courant;
  if (Stree->branche==NULL)
    {
      printf("%s ",Stree->LEX);
    }
  else
    {
      courant=Stree->branche;
      while (courant!=NULL)
        {
          edit_sentence(courant->objet);
          courant=courant->suivant;
        }
    }
}
```

B.3 SKEL_to_HLP.c

Source is under: xsvoyard/src It is an adaptation of the SKEL_to_PhonoWord program written by Tony Hebert

{/home/as65/xsvoyard/src/SKEL_to_HLP.c}

Appendix C

CHATRface

The source code is under: ~xsvoyard/XW/chatrface

C.1 chatrface.h

```
#include <stdio.h>

#include <X11/StringDefs.h>
#include <X11/Intrinsic.h>

#include <X11/Xaw/Label.h>
#include <X11/Xaw/Command.h>
#include <X11/Xaw/Paned.h>
#include <X11/Xaw/Box.h>
#include <X11/Xaw/Toggle.h>
#include <X11/Xaw/Scrollbar.h>
#include <X11/Xaw/Viewport.h>
#include <X11/Shell.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

/*----------Widgets----------*/

Widget pressW;
Widget press2W;
Widget press3W;
Widget Focus2W;
Widget toplevel;
```

```
Widget boxW,box2W,skrollW;
Widget panedW;
Widget titreW;
Widget quitW,OkW,confW,ctitrW,cboxW,yesW,noW;

int msgqid,nb,i;

/*----------Callbacks----------*/
void
PressCB( Widget widget, XtPointer client_data, XtPointer call_data );
void
FocusCB( Widget widget, XtPointer client_data, XtPointer call_data );
void
QuitCB( Widget widget, XtPointer client_data, XtPointer call_data );
void
yesCB( Widget widget, XtPointer client_data, XtPointer call_data );
void
noCB( Widget widget, XtPointer client_data, XtPointer call_data );
void
OkCB( Widget widget, XtPointer client_data, XtPointer call_data );
```

C.2 callbacks.c

```
#include "chatrface.h"


/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
void
PressCB( Widget widget, XtPointer client_data, XtPointer call_data )
{
  printf( "Ok\n" );
}

/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
void
FocusCB( Widget widget, XtPointer client_data, XtPointer call_data )
{
struct msgbuf{
  long mtype;
  char mtext[128];
```

```
};
struct msgbuf msg;

msg.mtype=2;
sprintf(msg.mtext,"%s %s",(char*)client_data,(char*)XawToggleGetCurrent(press2W)

    if (msgsnd(msgqid, &msg,128,0)==-1)
      {
        puts("serv: erreur sur msgsnd");
        exit(1);
      }
  printf( "Focus added on %s\n",(char*)XawToggleGetCurrent(press2W) );
}

/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
void
QuitCB( Widget widget, XtPointer client_data, XtPointer call_data )
{
XtPopup( confW, XtGrabNone );
}
/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
void
yesCB( Widget widget, XtPointer client_data, XtPointer call_data )
{
struct msgbuf{
  long mtype;
  char mtext[128];
};
struct msgbuf msg;

msg.mtype=2;
strcpy(msg.mtext,"@Quit");
    if (msgsnd(msgqid, &msg,128,0)==-1)
      {
        puts("serv: erreur sur msgsnd");
        exit(1);
      }
  exit( 0 );
}
/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
```

```
void
noCB( Widget widget, XtPointer client_data, XtPointer call_data )
{
XtPopdown( confW );
}
/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
void
OkCB( Widget widget, XtPointer client_data, XtPointer call_data )
{
struct msgbuf{
  long mtype;
  char mtext[128];
};
struct msgbuf msg;

msg.mtype=2;
strcpy(msg.mtext,"@Say");
    if (msgsnd(msgqid, &msg,128,0)==-1)
      {
        puts("serv: erreur sur msgsnd");
        exit(1);
      }
}
```

C.3 main.c

```
#include <stdio.h>
#include "chatrface.h"

void logo(Widget);
/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
main( int argc, char **argv )
{
struct msgbuf{
  long mtype;
  char mtext[128];
};
struct msgbuf msg;
struct msqid_ds *buf;
/*-------------------------------*/
```

```
  Arg       arg[8];
  char       text[256];
  char       flags[16];
/*---------init queue------------*/
msgqid=msgget(27,IPC_CREAT|0777);
if (msgqid==-1)
  {
    puts("serv: erreur sur msgget");
    exit(1);
  }
/*-----------------------------*/

  toplevel= XtInitialize( argv[0], "Chatrface", NULL, 0, &argc, argv );

  boxW= XtCreateManagedWidget( "boite", boxWidgetClass, toplevel, NULL, 0 );
  XtSetArg( arg[0], XtNorientation, 0 );
  XtSetValues( boxW, arg, 1 );

/*-----------------------------------------------------------------
                              Action box
-----------------------------------------------------------------*/

  panedW= XtCreateManagedWidget( "panneau", panedWidgetClass, boxW, NULL, 0 );

  pressW= XtCreateManagedWidget( "Focus++", commandWidgetClass, panedW, NULL, 0
  XtAddCallback( pressW, XtNcallback, FocusCB,"Focus ++" );
  XtSetArg( arg[0], XtNbackground, 15 );
  XtSetValues( pressW, arg, 1 );

  Focus2W= XtCreateManagedWidget( "Focus-", commandWidgetClass, panedW, NULL, 0
  XtAddCallback( Focus2W, XtNcallback, FocusCB,"Focus -" );
  XtSetArg( arg[0], XtNbackground, 14 );
  XtSetValues( Focus2W, arg, 1 );

  OkW= XtCreateManagedWidget( "Ok", commandWidgetClass, panedW, NULL, 0 );
  XtAddCallback( OkW, XtNcallback, OkCB, NULL );


  quitW= XtCreateManagedWidget( "Quit", commandWidgetClass, panedW, NULL, 0 );
  XtAddCallback( quitW, XtNcallback, QuitCB, NULL );

  titreW= XtCreateManagedWidget( "action", labelWidgetClass, panedW, NULL, 0 );
  XtSetArg( arg[0], XtNforeground, 13 );
```

```
  XtSetValues( titreW, arg, 1 );
  logo(titreW);
/*-----------------------------------------------------------------
-----------------------------------------------------------------*/
confW=XtCreatePopupShell( "cnf",topLevelShellWidgetClass, toplevel, NULL,
cboxW= XtCreateManagedWidget( "conf", boxWidgetClass, confW, NULL, 0 );
ctitrW= XtCreateManagedWidget( "Do you really want to quit?", labelWidgetC
yesW= XtCreateManagedWidget( "Yes", commandWidgetClass, cboxW, NULL, 0 );
XtAddCallback( yesW, XtNcallback, yesCB, NULL );
noW= XtCreateManagedWidget( "No", commandWidgetClass, cboxW, NULL, 0 );
XtAddCallback( noW, XtNcallback, noCB, NULL );

skrollW= XtCreateManagedWidget( "asc", viewportWidgetClass, boxW, NULL, 0
/*   XtSetArg( arg[0], XtNallowHoriz, True ); */
/*   XtSetValues( skrollW, arg, 1 ); */

  box2W= XtCreateManagedWidget( "boite2", boxWidgetClass, skrollW, NULL, C
  XtSetArg( arg[0], XtNorientation, 0 );
  XtSetValues( box2W, arg, 1 );


/* while((nb=msgrcv(msgqid,&msg, 128, 1,IPC_NOWAIT))==-1); */
nb=msgrcv(msgqid,&msg, 128, 1,0777);
  press2W= XtCreateManagedWidget( "text:", toggleWidgetClass, box2W, NULL,
  XtAddCallback( press2W, XtNcallback, PressCB, NULL );
  while(msg.mtext[0]!='@'){
      XtSetArg( arg[0], XtNbackground, 0 );
      XtSetArg( arg[1], XtNforeground, 1 );
  for(i=1;msg.mtext[i]!='/';i++){
    /*tests*/
    switch(msg.mtext[i]) {
    case '+':
      XtSetArg( arg[0], XtNbackground, 15 );
      break;
    case '-':
      XtSetArg( arg[0], XtNbackground, 14 );
      break;
    case 's':
      XtSetArg( arg[1], XtNforeground, 13 );
      break;
    case 'f':
      XtSetArg( arg[1], XtNforeground, 12 );
      break;
```

```
      }
  }
  sscanf(msg.mtext+i,"/%s",text);
  press3W= XtCreateManagedWidget( text, toggleWidgetClass, box2W, NULL, 0 );
  XtAddCallback( press3W, XtNcallback, PressCB, NULL );
  XtSetArg( arg[2], XtNradioGroup, press2W );
  XtSetArg( arg[3], XtNborderWidth, 0 );
  XtSetValues( press3W, arg, 4 );

/*   while((nb=msgrcv(msgqid,&msg, 128, 1,IPC_NOWAIT))==-1); */
nb=msgrcv(msgqid,&msg, 128, 1,0777);

}

  XtRealizeWidget( toplevel );
  XtMainLoop();
}

void logo(Widget W) {
/*Affichage de la bitmap*/
  Arg           arg[1];
  Pixmap        Bm;
  Display       *d=XtDisplay(W);
  Window        w=DefaultRootWindow(d);
  unsigned int  width,height;
  int           x,y;

  XReadBitmapFile(d,w,"chatr_logo.xbm",&width,&height,&Bm,&x,&y);

  XtSetArg( arg[0], XtNbitmap, Bm );
  XtSetValues( W, arg, 1 );
}
```

C.4 sebvtst.c

Source code is under: xsvoyard/src/chatr-0.93/display

```
#include <stdio.h>
#include "list.h"
#include "interface.h"  /* for print functions */
#include "table.h"
#include "word.h"
```

```
#include "syllable.h"
#include "intonation.h"

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

void sebvtst(Utterance utt)
{
struct msgbuf{
  long mtype;
  char mtext[128];
};

  Stream w;
  List syls,s;
  Stream intone;

  /*for ipc*/
  int msgqid,nb;
  struct msgbuf msg1,msg2;
  char f1[64],f2[64],f3[64];
  msgqid=msgget(27, IPC_ALLOC);
  if (msgqid==-1)
    {
      P_Message("erreur sur msgget");
      exit(1);
    }
  msg1.mtype=1;

  for (w=utt_stream("Word",utt); w!=SNIL; w=SC_next(w))
    {
      P_Message("Word %s:",SC(w,Word)->text);
      P_Message("%s",pprint(SC(w,Word)->features));
     /*---------------------------*/
      strcpy(msg1.mtext,"/");
      if (hlp_has_feat(w,"Focus","++")){
        strcat(msg1.mtext,"+");
      }
      if (hlp_has_feat(w,"Focus","-")){
        strcat(msg1.mtext,"-");
      }
      if (hlp_has_feat(w,"OPT","Slow")){
```

```
        strcat(msg1.mtext,"s");
      }
      if (hlp_has_feat(w,"OPT","Fast")){
        strcat(msg1.mtext,"f");
      }

      strcat(msg1.mtext,"/");
      strcat(msg1.mtext,SC(w,Word)->text);

/*       else strcpy(msg1.mtext,"0"); */

      /* send message to server */
      if (msgsnd(msgqid, &msg1,128, 0777)==-1)
        {
          puts("erreur sur msgsnd");
          exit(1);
        }
      /*----------------------------*/
   }
      /* send message to server */
      strcpy(msg1.mtext,"@end");
      if (msgsnd(msgqid, &msg1,128, 0777)==-1)
        {
          puts("erreur sur msgsnd");
          exit(1);
        }
      /*----------------------------*/

/* now waiting for information from user */
while(1){
/* while ((nb=msgrcv(msgqid,&msg2, 128, 2, IPC_NOWAIT))==-1); */
nb=msgrcv(msgqid,&msg2, 128, 2, 0777);
if (msg2.mtext[0]!='@'){
  P_Message("Request: %s",msg2.mtext);
  sscanf(msg2.mtext,"%s %s %s",f1,f2,f3);
  P_Message("%s : %s : %s",f1,f2,f3);
simple_input(utt);

  for (w=utt_stream("Word",utt); w!=SNIL; w=SC_next(w))
    {
      /*     P_Message("%s",SC(w,Word)->text); */
      if (strcmp(SC(w,Word)->text,f3)==0) break;
    }
```

```
  if (w==SNIL) exit(-1);
  hlp_delete_feat(w,f1,"-");
  hlp_add_feat(w,"Focus","++");
/*   hlp_module(utt); */
/*   P_Message("OK %s",pprint(SC(w,Word)->features)); */
}
else if (msg2.mtext[1]!='Q') {
/*  P_Message("input"); */
/*  simple_input(utt); */
 P_Message("hlp");
  hlp_module(utt);
 P_Message("word");
  word_module(utt);

 P_Message("phono");
phonology_module(utt);
 P_Message("intone");
 intone_module(utt);
 P_Message("durat");
duration_module(utt);
 P_Message("int targ");
 int_target_module(utt);
 P_Message("synth");
synthesis(utt);
 P_Message("say");
  play(GETWAVE(utt));
for (w=utt_stream("Word",utt); w!=SNIL; w=SC_next(w))
    {
      P_Message("Word %s:",SC(w,Word)->text);
      P_Message("%s",pprint(SC(w,Word)->features));
}
}
else {
  break;
}
}
}
```

C.5 utt_modules.c

```
struct LE_um_name_func com2umfunc[] =
{
```

```
...
{"Sebd",sebvtst,NIL,NIL,
     "essai d'affichage par seb."},
...
}
```

Appendix D

Multi-lingual input

D.1 pw_input.c

```
...
static int is_stressed(List *a); /* seb */
List prosodic_options(List *a); /* seb */
...
void phoneme_input(Utterance utt)
{
    /* Build a pphrase and word stream based on this romaji input     */
    /* The input form is a simple list of words with small integers   */
    /* identifying break points -- this input is as from the KDD prog */
    /* This does this by building the same Lisp structure that would  */
    /* have been written if this were a PhonoWord input               */
    List input,p,phr_c,phr_p,phr_s,phr_d,intrm;
    int blvl = NO_BOUND;
    int focus =0; /* seb */
    input = UTTERANCE(utt);

    if (list_is_atomic(input) == FALSE)
    {
        P_Error("Input form for Phoneme should be simple list of words/bre
        list_error(On_Error_Tag);
    }
    phr_p = phr_c = phr_s = phr_d  = NIL;
    for (p=input; blvl != D_BOUND; p = cdr(p))
    {
/*      P_Message("w %s",pprint(prosodic_options(&p)));*/
      intrm=prosodic_options(&p);
/*      focus=is_stressed(&p);*/
```

```
        blvl = nuu_blvl(p);
        if (blvl == NO_BOUND)
        {
        /* seb ->
          if (focus) intrm = cons(cons(mkatom("H*"),NIL),NIL);
          else intrm = NIL;
         <- seb */
          phr_p = cons(cons(cons(list_copy_tree(car(p)),list_copy_tree(intrm)),N
        }
        if (blvl >= P_BOUND)
        {
            if (phr_p != NIL)
            {
                phr_p = cons(mkatom(":P"),cons(NIL,list_reverse(phr_p)));
                phr_c = cons(phr_p,phr_c);
                phr_p = NIL;
            }
        }
        if (blvl >= C_BOUND)
        {
            if (phr_c != NIL)
            {
                phr_c = cons(mkatom(":C"),cons(NIL,list_reverse(phr_c)));
                phr_s = cons(phr_c,phr_s);
                phr_c = NIL;
            }
        }
        if (blvl >= S_BOUND)
        {
            if (phr_s != NIL)
            {
                phr_s = cons(mkatom(":S"),cons(NIL,list_reverse(phr_s)));
                phr_d = cons(phr_s,phr_d);
                phr_s = NIL;
            }
        }
        if (blvl == D_BOUND)
            phr_d = cons(mkatom(":D"),cons(NIL,list_reverse(phr_d)));
    }

    utt_set_stream("Word",SNIL,utt);
    utt_set_stream("Pphrase",build_phrase_tree(phr_d,utt),utt);
P_Message("%s",pprint(phr_d));
```

```
    list_free_tree(phr_d);

}

...

static int is_stressed(List *a)/* seb */
{
if (streq("/s/",STRVAL(car(*a))))
  {
    *a=cdr(*a);
    return 1;
  }
else return 0;
}

List prosodic_options(List *a)/* seb */
{ /* this function extract options from the phoneme input
     options are applied to the following "word"
     format is /([+-fs])+/
      + add a (Focus ++) feature
      -  "    (Focus -)    "
      f  "    (OPT Fast)   "
      s  "    (OPT Slow)   "
     more options could be added such as
      l and h to change the Power
      I included the concerning code but as the features
      are not defined yet it has to be changed later!!! */
List p_opt;
int i;
if ((*a!=NIL)&&(STRVAL(car(*a))[0]=='/'))
  {
    p_opt=NIL;
    for(i=1;STRVAL(car(*a))[i]!='/';i++)
      {
        switch(STRVAL(car(*a))[i])
          {
          case '+':
            p_opt=cons(cons(mkatom("Focus"),cons(mkatom("++"),NIL)),p_opt)
            break;
          case '*':
            p_opt=cons(cons(mkatom("Focus"),cons(mkatom("+++"),NIL)),p_opt
            break;
```

```
          case '-':
            p_opt=cons(cons(mkatom("Focus"),cons(mkatom("-"),NIL)),p_opt);
            break;
          case 's':
            p_opt=cons(cons(mkatom("OPT"),cons(mkatom("Slow"),NIL)),p_opt);
            break;
          case 'f':
            p_opt=cons(cons(mkatom("OPT"),cons(mkatom("Fast"),NIL)),p_opt);
            break;
/*        case 'h':
            p_opt=cons(cons(mkatom("Power"),cons(mkatom("+"),NIL)),p_opt);
            break;
          case 'l':
            p_opt=cons(cons(mkatom("Power"),cons(mkatom("-"),NIL)),p_opt);
            break;*/
          default:
            break;
          }
      }
    *a=cdr(*a);
    return list_copy_tree(p_opt);
  }
else return NIL;
}
```

Appendix E

Focus +++

E.1 phrase_int.c

```
void insert_phrase_pause(Utterance utt)
{
    Stream s_syl;
    for (s_syl = SYLSTREAM(utt); s_syl != SNIL; s_syl = SC_next(s_syl))
    {
        if ((SC(s_syl,Syl)->ph_final == P_BOUND) &&
            (grammar.pause_stat.p_dur > 0))
            insert_pause(s_syl, utt);
        else if ((SC(s_syl,Syl)->ph_final == C_BOUND) &&
                 (grammar.pause_stat.c_dur > 0))
            insert_pause(s_syl, utt);
        else if ((SC(s_syl,Syl)->ph_final == S_BOUND) &&
                 (grammar.pause_stat.s_dur > 0))
            insert_pause(s_syl, utt);
        else if ((SC(s_syl,Syl)->ph_final == D_BOUND) &&
                 (grammar.pause_stat.d_dur > 0))
            insert_pause(s_syl, utt);
        /* modification by seb for Focus +++ */
        else if ((hlp_has_feat(Rword1(SC_next(s_syl)),"Focus","+++")) && !
            insert_pause(s_syl, utt);
        /* else don't add a pause */
    }
}
```

E.2 grammar.c

```
static int pause_list_parse(List data)
```

```
{
    int num_entries, i;
    List l_cell, a_cell;
    char *tmpstr;

    num_entries = list_length(data);
    P_Debug("list length = %d\n", num_entries);

    for (i = 0, l_cell = data; l_cell != NIL; l_cell = cdr(l_cell), ++i)
    {
        a_cell = car(l_cell);
        tmpstr = print_cc(a_cell);
        P_Debug("entry is %s\n", tmpstr);
        xfree(tmpstr);

        if (list_length(a_cell) != 2)
        {
            P_Error("Wrong number of fields in file %s at %dth entry\n", i);
            P_Error("Expected 2, got %d\n", list_length(a_cell));
            return (-1);
        }

        if (streq(nth_val(1, a_cell), "discourse"))
            grammar.pause_stat.d_dur = atoi(nth_val(2, a_cell));

        else if (streq(nth_val(1, a_cell), "sentence"))
            grammar.pause_stat.s_dur = atoi(nth_val(2, a_cell));

        else if (streq(nth_val(1, a_cell), "clause"))
            grammar.pause_stat.c_dur = atoi(nth_val(2, a_cell));

        else if (streq(nth_val(1, a_cell), "phrase"))
            grammar.pause_stat.p_dur = atoi(nth_val(2, a_cell));

        else if (streq(nth_val(1, a_cell), "focus"))/* seb */
            grammar.pause_stat.f_dur = atoi(nth_val(2, a_cell));/* seb */

        else
        {
            P_Error("Unknown Pause type: %s\n", nth_val(1, a_cell));
            return(-1);
        }
    }
```

```
    return(0);

}
```

E.3 grammar.h

```
\struct Pause_Stat {
    int d_dur;
    int s_dur;
    int c_dur;
    int p_dur;
    int f_dur;/* seb */
};
```

Bibliography

[1] M. Beckman and G. Ayers. Guidelines to ToBI Labelling. Version 2.0, February 1994.

[2] M.E Beckman and G.M Ayers. The ToBI Handbook. Technical report, Ohio-State University, Colombus, U.S.A., 1993.

[3] E. Black, S. Eubank, H. Kashioka, R. Garside, G. Leech, and Magerman. D. Beyond skeleton parsing: producing a comprehensive large–scale general–english treebank with full grammatical analysis. In *Proceedings of the 16th Annual Conference on Computational Languistics*, 1996.

[4] Nick Campbell. Tones and break indices (tobi). *The journal of the Acoustical Society of Japan*, 53(3):223, 1997.

[5] Tony Hébert. Prosody within CHATR. Technical report, ATR Interpreting Telecommunications Research Laboratories, 1997.

[6] Martyn Weeks. *CHATR - a generic speech synthesis system.* ATR Interpreting Telecommunications Research Laboratories, http://www.itl.atr.co.jp/~ mweeks.

Index