RECHERCHE

Le Projet Euler

Vers une synthèse de parole générique et multilingue

Michel Bagein* — Thierry Dutoit* — Nawfal Tounsi — Fabrice Malfrère** — Alain Ruelle*** — Dominique Wynsberghe***

**Laboratoire de Traitement du Signal et Théorie des Circuits*
Faculté Polytechnique de Mons
Avenue Copernic, B-7000 Mons, Belgium
{bagein, dutoit, wynsberg}@tcts.fpms.ac.be

***Babel Technologies*
33 Boulevard Dolez, B-7000 Mons, Belgium
{tounsi, malfrere}@babeltech.com

**** MULTITEL-ASBL*
Avenue Copernic, B-7000 Mons, Belgium
ruelle@multitel.be

RÉSUMÉ. Euler est à la fois un logiciel et un projet de collaboration. Le logiciel Euler constitue un synthétiseur vocal à partir du texte, mis gratuitement à la disposition des chercheurs dans le cadre du projet du même nom. Les sources de son noyau et de certains de ses modules sont fournies et libres d'utilisation. Le projet Euler a été conçu pour permettre aux développeurs de systèmes TTS de collaborer à la mise au point de systèmes de synthèse collectifs, chacun apportant ses modules et ses bases de données propres tout en bénéficiant des modules et bases de données fournis par d'autres. La simple réalisation d'un tel logiciel, son organisation modulaire, sa documentation et l'organisation de sa mise à disposition constituent un travail conséquent. Nous en exposons ici les résultats actuels : la structure générale du logiciel et les modules et bases de données disponibles à ce jour pour le français, l'arabe, l'anglais américain, l'anglais insulaire, l'espagnol, le néerlandais et le turc. Nous examinons également les moyens disponibles pour l'extension d'Euler à d'autres langues.

ABSTRACT. Euler is a software and a project. Euler-the-software implements a text-to-speech synthesizer, made freely available for research purposes in the framework of Euler-the-project. The source code of its kernel and of some of its modules are provided under the GNU

public license. The project was set-up with a view to making it possible for TTS developers to collaborate to the design of collective TTS systems, by sharing modules and/or databases.

The simple design of such a software, its modular architecture, its documentation, and its organization for public release is a problem in itself. This paper summarizes its current state: the general architecture of the software, as well as modules and databases available for French, Arabic, American English, British English, Spanish, Dutch, and Turkish. We also examine possible means for extending Euler to other languages.

MOTS-CLÉS : *synthèse vocale, multilingue, multi-plateforme, généricité, modularité, multi-niveaux.*

KEY WORDS: *speech synthesis, multilingual, multi-platform, genericity, modularity, multi-level.*

1 Introduction

Depuis une quinzaine d'années, les laboratoires de recherche publics et privés (universités et opérateurs de télécommunications) ont investi des ressources considérables pour la mise au point de synthétiseurs de parole (TTS : Text-To-Speech) multilingues. Dans la plupart des cas, ces travaux de recherche non coordonnés ont généré des incompatibilités inter-systèmes dues à un manque évident d'unification, et ce, malgré les outils et bases de données connus et publiquement disponibles pour l'élaboration de systèmes TTS. Il en résulte que bon nombre de synthétiseurs consistent en une implémentation, spécifique à un laboratoire, de principes de base très similaires. De plus, hormis quelques systèmes tels [CAR76, SPR98, BLA97], la plupart des synthétiseurs multilingues ne sont en fait que des collections de systèmes monolingues, développés individuellement dans un laboratoire natif de la langue concernée.

Non seulement cette situation a un impact négatif sur les possibilités d'extensions de ces systèmes vers d'autres langues, dialectes, accents, voix et styles de parole, mais elle complique également leur intégration dans des produits finis. Enfin, le manque d'harmonisation dans la conception des systèmes TTS rend leur comparaison qualitative, module par module, très difficile à réaliser, ce qui restreint le déploiement de perfectionnements.

En réponse à cette situation, des outils et des bases de données pour le développement de systèmes TTS multilingues ont été récemment, et de façon indépendante, mis à disposition par quelques universités et centres de recherche européens. Parmi eux, la Faculté Polytechnique de Mons (FPMs) a contribué au développement de synthétiseurs multilingues «phonèmes vers parole» sous la forme du projet Internet MBROLA [DUT96].

La FPMs a décidé d'étendre ce projet au développement d'un système TTS multilingue sous la forme du projet Euler. Il s'agit d'un projet de recherche et de développement qui vise à intégrer progressivement les résultats d'autres projets de recherche tant en synthèse de parole qu'en traitement du langage naturel. L'objectif principal de ce projet est de réunir, grâce à une collaboration internationale, un ensemble de ressources homogènes pour la réalisation d'un synthétiseur de parole multilingue libre de droits pour utilisation non commerciale dans le plus grand nombre de langues et dialectes possibles, et ceci pour Windows, Linux, et Macintosh.

Après un bref rappel des divers traitements réalisés par un système de synthèse à partir du texte (section 2), nous détaillons dans cet article l'architecture logicielle du système Euler (section 3) et en examinons les composants disponibles à ce jour pour le français (section 4). Actuellement, la langue arabe est complètement réalisée et les systèmes pour l'anglais américain, l'anglais insulaire, l'espagnol, le néerlandais et le turc existent actuellement sous forme embryonnaire (sans prosodie). Nous examinons également les moyens disponibles pour l'extension d'Euler à d'autres langues (section 5).

2 Architecture générale d'un système de synthèse TTS

La Figure 1donne le diagramme fonctionnel d'un synthétiseur TTS.
On y retrouve le bloc de traitement du langage naturel, capable de produire la transcription phonétique du texte à lire, et d'y associer une prosodie aussi naturelle que possible. Ce bloc génère des informations symboliques. Ces informations sont transmises au bloc de traitement du signal, homologue fonctionnel de l'appareil phonatoire, qui transforme cette information symbolique en signal acoustique de parole.

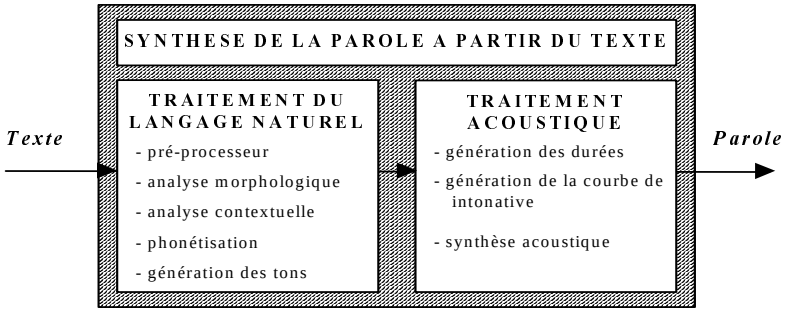

Figure 1. *Diagramme fonctionnel d'un système de synthèse TTS*

2.1 Traitement du langage naturel

Le module de traitement du langage naturel est généralement composé de trois modules principaux qui gèrent respectivement l'analyse morpho-syntaxique, la phonétisation automatique et la génération des tons.

2.1.1 Analyse morpho-syntaxique

Ce module gère l'extraction d'informations morphologiques, et cherche à mettre en évidence la nature des mots et leur organisation syntaxique. Il est composé de :
— Un module de pré-traitement (ou pré-processeur), qui joue le rôle d'interface entre le texte et la structure de données interne gérée par le synthétiseur. Ce module identifie toutes les séquences de caractères qui risquent de poser un problème de prononciation. Il doit pouvoir transcrire les abréviations, les nombres, détecter la fin des phrases et les acronymes.
— Un analyseur morphologique, dont la tâche consiste à déduire toutes formes de base de chaque mot pris individuellement, en fonction de sa graphie.
— Un analyseur contextuel, qui considère les mots et leurs formes de base associées, dans leurs contextes, ce qui permet de déduire la ou les natures possibles de chacun des mots ainsi que leurs fonctions grammaticales dans la phrase analysée.
— Enfin, un analyseur syntaxique-prosodique, qui s'appuie sur les résultats précédents afin d'établir un découpage du texte en groupes de mots, ce qui permettra d'y associer une prosodie.

2.1.2 Phonétisation

Le module de phonétisation automatique rassemble toutes les informations des modules précédents et produit la transcription phonétique du texte.
Au-delà de la phonétisation des mots isolés, il faut aussi tenir compte de phénomènes d'élisions et de liaisons phonétiques.

2.1.3 Génération des tons

Un générateur de tons s'appuie sur toutes les données produites par les modules précédents pour attribuer des tons aux syllabes. Ces tons sont des informations symboliques

2.2 Traitement acoustique

2.2.1 Synthèse de la prosodie

Le but de la génération de la prosodie est d'associer un rythme et une intonation à la phrase, à partir des groupes prosodiques déterminée en amont et des tons assignés aux syllabes. Ces données acoustiques sont produite par un générateur de durées et par un générateur de mélodie qui donne une courbe intonative décrivant l'évolution de la fréquence fondamentale des phonèmes.

2.2.2 Synthèse acoustique

Le bloc de traitement du signal numérique transforme les informations provenant du bloc de traitement du langage en un signal acoustique de parole. Cette opération peut être effectuée de plusieurs façons (synthèse par règles, synthèse articulatoire, par concaténation de diphones ou d'unités de parole non-uniformes). L'approche par concaténation est de loin la plus utilisée de nos jours [BOI00].

3 Architecture d'Euler

3.1 Les objectifs à atteindre

Le projet Euler a été conçu pour permettre aux développeurs de systèmes TTS de collaborer à la mise au point de systèmes de synthèse collectifs, chacun apportant ses modules et ses bases de données propres tout en bénéficiant des modules et bases de données des autres. Cela implique que le logiciel Euler doit être :
— **Public**. Dans la mesure du possible, les sources sont publiques sous licence GNU, ceci afin d'éviter une situation où un développeur de modules ou d'applications se retrouverait irrémédiablement soumis au bon vouloir des initiateurs du projet. De même, tous les développeurs sont vivement encouragés à fournir les sources de leurs ressources sous licence GNU GPL lorsque c'est possible. Néanmoins, le projet Euler offre aussi la possibilité de ne fournir au projet qu'une version protégée d'un module ou d'une base de données. Une licence Euler a été mise au point à cet effet, calquée sur la licence MBROLA et spécifiant qu'un

module fourni sous licence spécifique Euler (par opposition à la licence GNU GPL) ne peut être utilisé qu'à des fins non militaires et non commerciales.

— **Ouvert**. L'ouverture du projet Euler est une conséquence de la disponibilité de la plupart de ses sources et bases de données. L'extension du projet peut être effectuée à différents niveaux (qui seront mieux détaillés à la Section 5):

- En affinant, ajoutant ou remplaçant les bases de données existantes (exemple : pour la construction d'une nouvelle voix).
- En implémentant ou en créant de nouveau algorithmes (exemple : pour l'intégration de synthétiseurs à base d'unités non uniformes).
- En créant de nouveaux modules (exemple : analyse grammaticale par règles, pré-processeur de pages HTML, etc.)
- En étendant les fonctionnalités mêmes du logiciel Euler (exemple : couplage de la synthèse à des visages parlants).

— **Générique**. Dans la mesure du possible, le projet Euler utilise des ressources génériques. Les moteurs (voir 3.2) implémentent des technologies génériques qui peuvent être assignées à différents modules.

— **Modulaire**. Le logiciel Euler doit pouvoir servir de plate-forme de développement pour des laboratoires spécialisés sur un aspect du traitement de la parole. Sa modularité permet de réutiliser des ressources communes (données **et** algorithmes). Certains modules (comme MBROLA et MBRDICO, par exemple) sont par ailleurs utilisables isolément, à travers une interface spécifique et indépendante du projet. Ceci permet de valider ces modules indépendamment, et facilite l'intégration des résultats. Dans la mesure du possible, nous nous attacherons à pourvoir les autres modules de ce mode de fonctionnement.

— **Multilingue**. La réalisation de systèmes multilingues est grandement facilitée par les propriétés de généricité et de modularité des ressources. Partant d'un système de synthèse en français, il a par exemple été facile de produire une système similaire en arabe, et des embryons de synthétiseurs en néerlandais, anglais, espagnol, et turc.

— **Intégrable.** L'interface « intégrateur » du logiciel Euler a été conçue de façon à minimiser les caractéristiques à maîtriser pour parvenir à le coupler à une application.

— **Multi-plateforme**. Tous les composants développés doivent être fonctionnels sur plusieurs systèmes d'exploitation. Un réel effort a été entrepris pour rendre les codes sources indépendants du système d'exploitation cible. Ainsi, le logiciel Euler fonctionne d'ores et déjà sous Windows et Linux ; la version MacOs est à l'étude.

3.2 Modularité

A partir de ce cahier des charges, plusieurs architectures sont envisageables. Une toute première idée consiste à construire une architecture comportant toutes les ressources de base de toutes les langues cibles. Cette solution est difficile à mettre en œuvre car elle nécessite dès le départ des connaissances approfondies de toutes les langues. Le projet Euler est basé sur une autre approche : autour d'un noyau de

base (`Kernel`[1]) se rattache tout un ensemble de modules (`Modules`). Chacun de ces modules contient un algorithme et les données associées (`Database`) nécessaires à la réalisation d'une tâche, telle que l'analyse grammaticale ou la phonétisation. Chaque module assure en réalité deux fonctions : l'accès aux données internes des éléments (`Items`) et le lancement de l'algorithme de traitement de ces données. Cette stratégie de construction permet de s'attacher à la réalisation partagée d'un système TTS. Nous l'avons affinée en externalisant, quand c'était possible, la fonction algorithmique dans un **moteur** (`Engines`) indépendant du module. L'architecture générale des relations entre composants est illustrée à la figure 2.

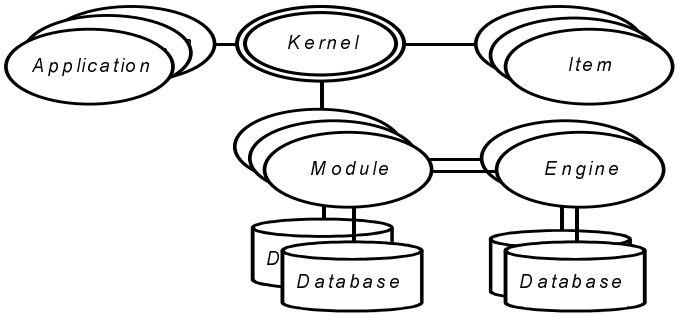

Figure. 2 Architecture modulaire d'Euler.

L'externalisation du traitement permet de tester aisément différentes technologies pour la réalisation d'un traitement linguistique donné : le module possède une fonctionnalité linguistique abstraite, et la façon dont cette fonctionnalité est implémentée dépend du moteur choisi.
Par exemple, la transcription phonétique d'un mot peut se faire soit par un moteur basé sur le parcours d'arbres (moteur ID3 décrit au paragraphe 4.4.1) de décision soit par un moteur basé sur un jeu de règles (moteur MLRR décrit au paragraphe 4.4.2), et cela ne modifie en rien l'accès des autres modules aux données graphémiques et phonétiques (voir figure 3).

[1] Dans le texte, les mots en en police courrier correspondent aux identificateurs internes du logiciel. Pour cette raison, ils conservent leurs notation anglo-saxonne.

Figure. 3.Moteur paramétrable pour le module de phonétisation.

L'utilisation de moteurs contribue également à la généricité d'Euler, dans la mesure ou une technologie donnée (et donc un moteur donné) peut intervenir dans plusieurs modules. Ainsi, un moteur basé sur un jeu de règles de réécritures peut être utilisé dans un module dédié à la traduction des nombres littéraux tout comme dans un module de phonétisation.

3.3 Souplesse

Définir un système TTS dans le logiciel Euler revient en fait à décrire la séquence des modules qu'il utilise, chaque module étant lui-même défini par les moteurs et les données associées.

Si une application doit intégrer plusieurs langues, voix, styles d'intonation, etc., elle peut définir plusieurs systèmes en parallèle. Ces systèmes partagent alors leurs modules et leurs bases de données, qui ne sont dès lors chargés qu'une seule fois en mémoire. L'application peut changer rapidement de système (par exemple passer d'une langue à une autre) en cours de fonctionnement.

Le fait de pouvoir partager les composants entre les systèmes permet de minimiser l'espace mémoire utile au lancement de plusieurs systèmes TTS en parallèle. Par exemple, le module de phonétisation est chargé une seule fois en mémoire, et peut être utilisé par plusieurs systèmes définis pour plusieurs langues. Seules les bases de données sont alors différentes suivant les langues.

Les définitions de systèmes, en termes de modules, sont regroupées au sein d'un fichier script unique, lu au lancement du noyau. Un mécanisme de variables est intégré pour simplifier l'écriture. Le contenu de ce fichier est assez intuitif, comme on le constate ci-dessous :

```
#   Euler.ini DEMO 2.00 version

[SPEAKER]
 Thierry = French -Belgium -dataBase fr1 -volumeRatio 1 -durationRatio 1
 Vincent = French -French -dataBase fr3 -volumeRatio 1 -durationRatio 1
```

```
Nawfal  = Arabic -dataBase ar1 -volumeRatio 5 -durationRatio 0.8
…

 [SYSTEM]
 French = preprocFr lemmat grammar phonetFr postPhonetFr toneGeneratorFr
   melodyGenerator synthese
 Arabic = preprocAr taggerAr phonetAr prosodyAr synthese
…

[VARIABLE]
# for Windows, uncomment the next two lines
ext = .dll
slash = \
dba = ..%slash%..%slash%databases%slash%
frDba = %dba%french%slash%
arDba = %dba%arabic%slash%
…

 [MODULE]
# FRENCH
preprocFr        = rulepreprocessorfr%ext% -abrev %frDba%abrv.dba
…

# ARABIC
preprocAr        = RulePreProcessorAr%ext% -Arabic -lexicon mlrr%ext% -
   filename %arDba%arabic2latin.dba -NumberFile arDba%arabicNumber.dba
phonetAr =      phonetizer%ext%      -lexicon      id3%ext%      -tagFile
   %arDba%tags_phonet.rul       -zsFile       %arDba%ar.zs       -ID3File
   %arDba%arabic.tree
taggerAr         = taggerAr%ext% -tagFile %arDba%arabic.tag
prosodyAr        = fmprosodygeneratorAr -prosoArDll %arDba%proso%ext% -
        AlphabetFile      %arDba%arabic.alp      -RythmicDescriptionFile
        %arDba%arabic.des     -RythmicruleFile     %arDba%arabic.rul     -
        F0PatternFile %arDba%arabic.dat

# COMMON
synthese = mbrolainterface%ext%
```

Ainsi, pour le locuteur «Nawfal», défini dans section [SPEAKER], le noyau chargera les modules définis dans le système «Arabic» et utilisera les options de voix, de volume et de durée de la ligne de définition suivante :

```
Nawfal = Arabic -dataBase ar1 -volumeRatio 5 -durationRatio 0.8
```

Pour le système «Arabic», défini dans la section [SYSTEM], le noyau chargera les modules : «preprocAr», «taggerAr», «phonetAr», «prosodyAr», et «**synthese**» décrit dans la ligne de définition suivante :

```
Arabic = preprocAr taggerAr phonetAr prosodyAr synthese
```

Tous ces modules sont définis dans la section [MODULE]. Par exemple, pour le module «taggerAr», le noyau chargera le composant logiciel «taggerAr.dll[2]». Les arguments «arDba» et «arabicTag» définissent le chemin et le fichier de la base de données associé à ce composant. La description de ce module est présenté à ligne suivante :

```
taggerAr = taggerAr%ext% -tagFile %arDba%arabic.tag
```

Un mécanisme de variable permet de simplifier l'écriture du fichier d'initialisation. Les variables sont définies dans la section [VARIABLE] et peuvent être utilisées dans toutes les sections. Par exemple, les variables « ext » et « slash » contient les valeurs « .dll » et « \ » sous Windows. Il suffit uniquement de modifier ces valeur en « .so » et « / » pour utiliser la même configuration générale de Euler sous Linux.

3.4 Représentation interne des données

L'interdépendance des données internes produites par chacun des modules est forte et la représentation de celles-ci reste un véritable problème. La représentation interne est assez variable d'un synthétiseur à l'autre. Certains systèmes utilisent des flux de données linéaires, typiquement une chaîne de caractères [CAR76, HER82, MAC87, ALL87]. Chaque bloc de traitement reçoit un flux fourni par le bloc précédent et en extrait l'information nécessaire au traitement qu'il doit effectuer. A l'issue du traitement, il produit un flux de sortie composé du flux d'entrée augmenté par les données qu'il a généré, lesquelles se retrouvent imbriquées dans les données d'entrée. Cette solution est simple à définir et à étendre. Néanmoins, on remarque que la charge de travail utilisée pour le codage et le décodage est assez conséquente.
Il est de plus en plus courant d'organiser les données internes dans des structures de données plus complexes. L'une des approche utilisée est la représentation des données sous forme d'arbre. L'avantage de ce type de représentation est de mettre en valeur l'aspect structurel du texte analysé. Néanmoins, au fur et à mesure de l'analyse du texte, les nouvelles données sont ajoutées (majoritairement) sous forme de nouvelles branches aux feuilles existantes de l'arbre. Cette représentation implique donc de nombreux parcours en profondeur le l'arborescence. Une alternative de la représentation des données sous forme d'arbre utilisées par les systèmes TTS est la structure de données multi-niveaux (voir figure 4) où chaque niveau (ou couche) se spécialise dans la description d'une des organisations linguistiques de la phrase (séquence de mots, séquence d'étiquettes grammaticales,

---

[2] Le noyau de Euler, les modules, les éléments et les moteurs contiennent du code exécutable. Ces composants sont encapsulés dans des librairies logicielles dynamiques (« Dynamic Linked Libraries », ou .dll, pour Windows et « Shared Object », ou .so, pour Linux). La gestion mémoire de ces librairies est directement assurée par le système d'exploitation.

séquence de syllabes, séquences de phonèmes, séquences de tons, séquences d'unités prosodiques, etc.). Les éléments de ces niveaux de représentation, nommés « *item* », sont caractérisés par un ou plusieurs attributs (ex : phonème = nom + durée + f0 + …). Les données présentes dans différentes couches peuvent par ailleurs être synchronisées entre elles, comme on le retrouve dans la structure « grid » de SpeechMaker [LEU93] ou dans la structure « Multi-Level Data Structure » de Festival [BLA97], ainsi que, dans une certaine mesure, dans les systèmes SCYLA [LAZ87], DEPES [VAN89], et LIFT [FRE91]. La structure de données de Euler, appelée « Mutli-Layer Container : MLC » s'inspire de ces modèles.

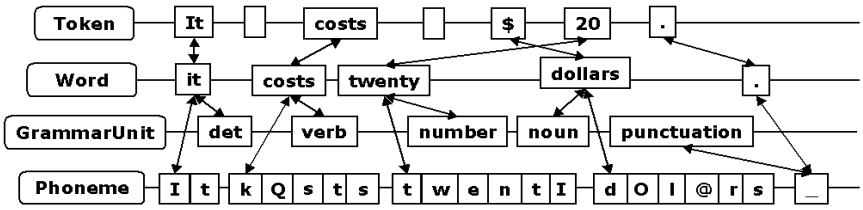

Figure. 4 *Extrait du contenu de la MLC pour la phrase « It costs $20. »*

Les principaux avantages de ces structures par rapport à un flux linéaire de données sont :

— Une **lisibilité** accrue des données.

— Une plus grande **traçabilité** : la recherche d'une donnée est simplifiée, grâce à un mécanisme de liens bidirectionnels entre données, et ceci aux travers des différentes couches.

— Une **vitesse** d'exécution accrue doublée d'une réduction de l'utilisation des ressources système due à l'absence de codage et de décodage des données ou de parcours d'arbre en profondeur.

— Une meilleure **extensibilité** : la structure MLC accepte toujours de nouvelles couches de données. De nouveaux modules peuvent ajouter de nouvelles couches de données sans influencer les modules existants.

— Un **parcours aisé** dans la structure, directement inspiré de la conception des itérateurs définis dans les « Standard Template Library (STL) » du langage C++.

4 Une réalisation concrète: les composants du français

Nous faisons ici l'inventaire de tous les modules et moteurs développés pour la langue française (illustré à la figure 5) et expliquons comment les adapter à d'autres langues.

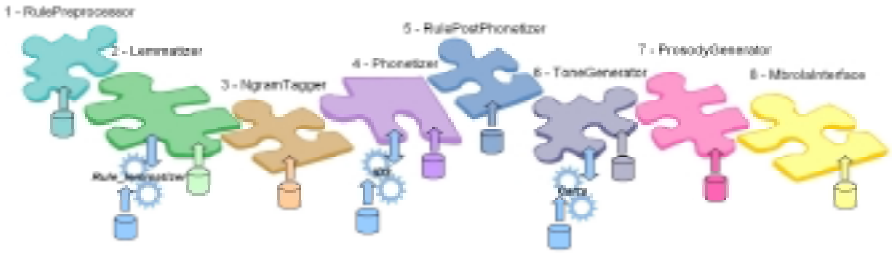

Figure. 5 *Inventaire des modules et moteurs développés pour la langue française*

4.1 Pré-traitement

L'étape de pré-traitement couvre classiquement la recherche des fins de phrases, la détection des acronymes, le traitement des nombres et des abréviations, etc. Dans l'implémentation actuelle du logiciel Euler, le traitement est confié à un petit analyseur à base de grammaires régulières (pour la détection des fins de phrases, des acronymes, et le décodage des nombres) et de lexiques (pour les abréviations).

Lors de la phase d'initialisation du module, la seule base de données chargée par ce module est celle qui contient les différentes abréviations qui seront reconnues par le système. Ce fichier contient les abréviations les plus connues de la langue française ainsi que leur transcription en toutes lettres. Contrairement aux abréviations, le traitement des nombres se fait à l'aide de règles codées dans le module.

Lors de son exécution, le module de pré-traitement crée deux couches dans la MLC :

— La couche «`Token`» qui contient les unités textuelles telles qu'elles apparaissent dans la phrase (ainsi que la ponctuation).

— La couche «`Word`» qui contient les mots du texte après analyse des «`Tokens`» (c'est-à-dire après la transcription des nombres, des abréviations etc.). Cet élément de type «`Word`» contient aussi un champ pour spécifier la nature grammaticale (*POS : part of speech*) associée à chaque mot.

Le champ « `POS` » de la couche «`Word`» est rempli par le module de pré-traitement. Si ce module détecte une abréviation ou un nombre par exemple, il affecte au champ « `POS` » respectivement les valeurs «`abrev`» et «`ordinal`». Il affecte la valeur «`unknown`» a tous les autres mots de la couche «`Word`». Le module d'analyse grammaticale se chargera par la suite de trouver la nature grammaticale correcte de tous les mots dont le champ «`POS`» est «`unknown`».

Il est bien connu que les étapes de pré-traitement sont généralement très dépendantes de la langue et de l'application visée. Il a donc été difficile de concevoir un module très général et adaptable à plusieurs langues par simple changement de fichiers de configuration : le module de pré-traitement de Euler pour le français est un des rares dont le code source soit particulier à la langue traitée. Cependant, ce code source étant distribué avec le logiciel, il est facile de s'en inspirer pour développer un pré-processeur dans une autre langue. Un module de pré-traitement en arabe a ainsi été

calqué sur le module français. Le traitement des nombres en français a par exemple été utile pour concevoir la transcription littérale des nombres arabes.
Très récemment, un nouveau pas vers la généricité a été franchi en confiant le traitement des nombres à un moteur générique qui implémente un noyau PROLOG et dont le fichier de données n'est autre que les règles de transcription littérale d'un nombre. Dans le même ordre d'idées, il est prévu de créer un autre moteur générique intégrant un noyau PERL.

4.2 Lemmatisation

La tâche de lemmatisation consiste à retrouver toutes les formes de base de chaque mot de la phrase pris isolément.
Le module de lemmatisation crée la couche «`GrammarUnit`» qui représente la nature grammaticale des mots ou groupes de mots indivisibles tels que les locutions ou nombres par exemple. Cette couche est constituée de 2 champs :
— Le champ «`ContextIndependentPOS`» qui contient toutes les formes de base d'un mot (ou d'une locution) indépendamment du contexte. Ce champ est rempli par le module de lemmatisation lui-même.
— Le champ «`POS`» qui contient la nature grammaticale finale du mot (ou d'une locution) après le traitement réalisé par le module d'analyse contextuelle.
Si la séquence de mots observée correspond à une locution d'un lexique (ex : « c'est-à-dire »), il suffit au lemmatiseur d'affecter la nature grammaticale à chacun des éléments de type « `Word` » qui composent cette locution, en remplissant leur attribut « `ContextIndependentPOS` », et d'ajouter dans la couche «`GrammarUnit`» une unité grammaticale contenant la nature de la locution associée à l'ensemble des mots de cette locution[3]. Dans le cas contraire, il faut traiter les mots individuellement. Pour le français, cela consiste en une décomposition des mots en racines et affixes (préfixes et suffixes). L'analyseur utilisé dans la version française d'Euler s'appuie sur une base de données contenant plus de 32 600 lemmes et 3085 suffixes regroupés dans 189 formes de dérivation de la plupart des mots français (chaque forme de dérivation est composée de la listes de suffixes possibles que l'on peut associer à un lemme). Cette base de données permet de retrouver toutes les natures grammaticales et toutes les flexions d'environ 400 000 mots. Elle est dérivée du corpus français Morlex [MERTE] développé à Leuven par Piet Mertens. Le cœur de l'algorithme de lemmatisation de mots isolés est assez classique (on le retrouve déjà dans [PIT83]). Il est implémenté sous forme d'un moteur générique qui nécessite 4 lexiques : un lexique de lemmes, de dérivations, de natures

[3] Le système actuel de reconnaissance des locutions ne préserve pas les ambiguïtés entre locutions et suites de mots; une solution unique est générée. Ainsi, pour la phrase "Nous le savons bien que tu ne l'admettes pas", si la locution "bien que" fait partie du lexique de locutions, la phrase sera interprétée comme : (Nous le savons) (bien que tu ne le donnes pas) ; dans le cas contraire, la même phrase sera interprétée comme : (Nous le savons bien) (que tu ne le donnes pas)

grammaticales et de flexions. L'algorithme mis en œuvre recherche toutes les décompositions possibles d'un mot en une racine et un suffixe. A chaque suffixe identifié du mot en question, correspond un suffixe de forme de base. Avec ce suffixe de base, un lemme, ou graphie de base, est généré et est recherché dans la table des lemme. Si le lemme existe, alors la graphie du mot est reconnue et sa(ses) nature(s) grammaticale(s) et sa(ses) flexion(s) sont produites. Par exemple, pour le mot « abaissons », le suffixe « ons » extrait du mot est présent dans la table de dérivation à l'index 6 et le suffixe « er » est proposé comme suffixe de lemme. Ainsi, le lemme « abaisser » peut être généré à partir de « abaissons » - « ons » + « er ». Ce lemme est présent dans la table des lemmes. Comme ce lemme accepte une terminaison d'index 6, le mot est accepté, sa natures grammaticale « V_a » et sa flexion « 1,p,IpPs » sont produites. Ces deux dernières informations sont deux clés internes dont les valeurs associées sont définies dans les tables de natures et de flexions. Ces deux tables sont utilisées pour traduire les symboles internes de nature et de flexion en symboles adaptés par les autres modules du système TTS en question. Des extraits des 4 lexiques sont donnés aux tables 1 à 4.

6	ons	er	1,p,IdPs
6	**ons**	**er**	**1,p,IpPs**
7	ça	cer	3,s,IdPa
7	çai	cer	1,s,IdPa

Table 1 : dérivations

meilleur	E	
même	E,Ma,Qt	
abaisser	**V_a**	**6**
abandonner	V_a	6

Table 3 : lemmes

V_a	**VERB**
V_ae	VERB
V_e	VERB
Conj_sub	CONJSUB

Table 2 : natures

f,p,PtPa	PARTPASSE
f,s	
f,s,PtPa	PARTPASSE
IfPr	INFINIT

Table 4 : flexions

Le module de lemmatisation actuel peut être adapté aux langues où la flexion modifie les derniers caractères des mots. Il faut alors construire tous les lexiques cités ci-dessus. Ce travail prend généralement un temps important. Il est à noter cependant que toutes les langues ne nécessitent pas un lemmatiseur complexe. Si on dispose d'un dictionnaire complet du lexique d'une langue (y compris toutes les formes fléchies), ce moteur peut être éventuellement remplacé par un accès direct à ce lexique.

4.3 Analyse contextuelle

Le module d'analyse contextuelle résout les ambiguïtés liées aux natures grammaticales. Le lemmatiseur associe en effet à chaque mot toutes ses natures possibles, indépendamment du contexte dans lequel le mot se trouve. L'analyseur

contextuel se doit de choisir, parmi celles-ci, celle qui satisfait aux contraintes syntaxiques locales. Le résultat de l'analyse contextuelle est alors stocké dans le champ « POS » de la couche «GrammarUnit» de chaque mot.
L'analyse contextuelle implémentée dans Euler est basée sur une approche statistique désormais classique, suivant le formalisme des *n*-grammes. La phrase à analyser est représentée en interne sous forme d'un treillis[4] où chaque mot est modélisé par la liste des natures grammaticales qu'il peut prendre. Un algorithme de Viterbi est alors chargé de trouver le meilleur chemin dans ce treillis par maximisation de la somme des vraisemblances associées aux séquences de *n* classes composant ce chemin. Pour la version française de l'analyseur, un corpus initial d'environ 4 300 phrases, soit 51 400 mots, a été utilisé. Ce corpus a été initialement annoté par un analyseur grammatical automatique à base de règles. Les étiquettes grammaticales ont ensuite été complètement corrigées manuellement. Ces étiquettes grammaticales ont été regroupées en 38 classes dans le but d'augmenter leur représentativité statistique dans le corpus. Finalement une grammaire a été extraite de ce corpus de classes grâce aux outils de création de *n*-grammes au format ARPA [ROSEN]. Pour le français, ces outils ont généré une combinaison de 38 uni-grammes, 773 bi-grammes et 5734 trigrammes. Pour améliorer les performances de l'analyseur, un lexique de locutions prépositives et conjonctives a été utilisé.
Le principe de l'étiquetage grammatical par *n*-grammes a déjà été utilisé avec succès pour de nombreuses langues. Pour rendre Euler capable de réaliser l'analyse contextuelle d'une nouvelle langue, le plus simple est d'utiliser les outils d'entraînement de *n*-grammes disponibles avec le Statistical Language Modeling toolkit de CMU [CMU]. Le module d'analyse contextuelle fourni avec Euler est en effet compatible avec le format des fichiers de sorties de ces outils logiciels.

4.4 Phonétisation

Lors de la phase d'exécution, le module de phonétisation crée une couche «Phoneme". Chaque élément de type «Phoneme» comporte 2 champs : le nom du phonème et sa durée moyenne. Tous les mots de la couche «Word» sont phonétisés un à un grâce aux moteurs de phonétisation (ID3 ou MLRR, voir ci dessous). Les noms des phonèmes retournés sont ajoutés dans le champ correspondant «Name» de la couche phonème. Les liens entre les couches «Word» et «Phoneme» sont créés dans la MLC.
La tâche de phonétisation automatique est actuellement réalisable, dans Euler, de deux façons assez différentes : par arbres de décision entraînés automatiquement sur

[4] La structure même des éléments la MLC ne se prête facilement pas à la représentation des données sous forme de treillis tels qu'utilisés dans ce module. La structure d'un treillis peut se formaliser comme une liste d'ensembles de données élémentaires et chaque ensemble de cette liste peut contenir un nombre quelconque d'éléments. Or, les types des différents champs des éléments sont limités actuellement aux nombres et chaînes de caractères. La représentation d'ensembles dynamiques de données n'est donc pas implémentée en tant que telle dans les éléments d'Euler. Elle est par contre réalisée à l'intérieur du module d'analyse contextuelle.

de grands corpus ou par interprétation de règles de réécritures fournies par un expert. La phonétisation est réalisée mot-à-mot.

4.4.1 Le moteur ID3

Le moteur de phonétisation ID3 est basé sur le parcours d'arbres de décision. L'algorithme implémenté est ID3 [BLA98][PAG98]. L'implémentation qui en est faite dans Euler permet de prendre en compte les étiquettes grammaticales fournies pas l'analyse contextuelle, ceci afin de traiter les homographes hétérophones[5]. Le corpus d'entraînement utilisé contient environ 200 000 formes associées à leurs étiquettes grammaticales. Sur un ensemble de 1000 mots hors lexique extraits du journal Le Monde, le système transcrit 91% de ces mots conformément au lexique initial (ce taux monte à 99,02 % sur les mots du lexique). Lors d'une évaluation auditive il s'avère que 95% des transcriptions sont acceptables. Outre ses capacités de généralisation (pour la prononciation de mots non présents dans le lexique), cette méthode permet la compression du lexique. Ainsi, sur le corpus en français, le rapport de compression est de 22 pour 1.
Pour créer une nouvelle langue, il suffit de disposer d'un corpus de mots et de leur transcription phonétique, que l'on soumet à un outil d'entraînement qui construit un arbre de décision. Cet outil est disponible dans le projet MBRDICO [MBRDI], sous licence GNU. Ce projet (et donc automatiquement Euler) dispose actuellement d'arbres de décision dans 7 langues: l'arabe, l'anglais américain, l'anglais insulaire, l'espagnol, le turc, le français et le néerlandais.

4.4.2 Le moteur MLRR

Un algorithme basé sur des règles de réécritures régulières multi-niveaux (MLRR : *Multi-Level Rewrite Rules*) peut être utilisé pour les langues où l'on dispose de règles phonétiques.
L'initialisation du moteur se fait à l'aide d'un fichier qui contient les règles de phonétisation. Toutes les règles sont de la forme :

A / L _ R ; … ; An / Ln _ Rn ---> B

Une règle de ce type a la signification suivante : le symbole *A* est transcrit en un symbole *B* s'il est précédé par *L* et suivi par *R* dans la couche d'entrée, et à condition que les symboles *A2,..., An* (homologues de *A* sur d'autres couches de la MLC) soient précédés dans leurs couches respectives par *L2,…,Ln* et suivis respectivement par *R2,…,Rn.*
La base de phonétisation française regroupe 293 règles de ce type.

4.5 Liaisons, élisions et e muets

Les liaisons et e muets potentiels sont traités à ce stade, quand toutes les informations morphologiques, syntaxiques, et phonétiques sont disponibles. Ce

---

[5] De même que dans le traitement des locutions, dans le cas d'homographes hétérophones ambigus, une unique transcription phonétique sera produite et l'ambiguïté ne sera pas préservée.

traitement est très dépendant de la langue. En français, un module spécifique contenant un petit analyseur basé sur quelques règles régulières remplit ce rôle.

4.6 Génération des tons

Le module de génération des tons découpe en syllabes la suite des phonèmes produits par le phonétiseur, et associe à chaque syllabe un ton en fonction du champ «POS» du mot auquel elle appartient, des phonèmes qui constituent la syllabe, de sa position dans le mot et dans la phrase, du champ «POS» des mots voisins, etc.
Ce module est lui aussi très dépendant de la langue.
La version disponible actuellement est assez simple. L'assignation des tons est basée sur un découpage de la phrase en groupes intonatifs obtenus sur base de la distinction mot lexical/mot grammatical (cette approche est souvent appelée « chinks and chunks » ; voir par exemple [LIB92]).

4.7 Génération de la prosodie

Le module de génération de la prosodie génère la durée des phonèmes et la courbe intonative à y associer, en fonction des tons préalablement produits.
Ce module est issu des travaux de F. Malfère [MAL98] et est conçu pour être aussi indépendant de la langue que possible.
Les durées sont obtenues à l'aide d'un arbre de décision, entraîné sur un corpus de parole segmenté phonétiquement, et rendant essentiellement compte de la durée des phonèmes en fonction de leur position dans la syllabe, de la position de cette syllabe dans le mot, et des étiquettes grammaticales des mots environnants.
La production de l'intonation est quant à elle basée sur un principe de concaténation de patrons intonatifs naturels, extraits du même corpus d'entraînement, et correspondant aux mêmes groupes intonatifs que ceux considérés par le module de génération de tons. A la synthèse, les patrons utilisés sont choisis automatiquement, de façon à minimiser une fonction de coût qui fait intervenir à la fois un coût de cible (qui rend compte de la ressemblance entre les caractéristiques des patrons disponibles et celles du patron dont on cherche à établir l'intonation) et un coût de concaténation (qui rend compte de la compatibilité entre patrons successifs, de façon à assurer une certaine continuité des courbes intonatives). Finalement, les données de durée et de fréquence fondamentale ainsi produites sont enregistrées dans la MLC dans une couche prosodique créée par le module.

Pour adapter ces deux derniers modules (génération de tons et génération de prosodie) à d'autres langues, il est nécessaire de disposer d'un grand corpus de parole (typiquement, une heure de parole) segmenté phonétiquement, étiqueté grammaticalement, et sur lequel on a calculé l'intonation. Le plus simple est d'utiliser, pour la segmentation, les outils du projet MBROLIGN [MAL97], et, pour l'étiquetage grammatical, les modules d'Euler qui s'y rapportent (si ces modules ne sont pas disponibles pour la langue considérée, il faudra les créer). Il reste alors à produire les arbres de décision et les patrons intonatifs qu'il faudra fournir au module lors de son initialisation. Ceci requiert un outil logiciel développé à cet effet.

Cet outil n'est pas disponible dans le cadre du projet Euler; son utilisation nécessite au préalable un accord entre le propriétaire de l'outil (la Faculté Polytechnique de Mons) et les utilisateurs potentiels. Le contenu de cet accord est très semblable à celui qui est à la base du projet MBROLA : il s'agit essentiellement pour le propriétaire de s'assurer que les fichiers produits par les utilisateurs seront mis à la disposition de tous dans le cadre et les limites d'utilisation du projet Euler.

4.8 Synthèse acoustique

La génération des signaux acoustiques est actuellement confiée au synthétiseur MBROLA, basé sur la concaténation de diphones.
Il est cependant à noter, et ceci est d'ailleurs valable pour tous les modules cités précédemment, que le projet Euler n'impose aucunement l'utilisation d'une technologie ou d'un algorithme particulier. Tout composant logiciel construit en respectant l'interface requise par le logiciel Euler pour la communication entre noyau et modules (et éventuellement entre module et MLC) peut se substituer aux modules actuellement disponibles de Euler.

5 Comment participer au développement de Euler ?

Il est clair que le logiciel Euler est loin d'être la solution idéale à tous les problèmes d'ingénierie logicielle posés par la conception d'un système de synthèse vocale. Ainsi, le logiciel Euler ne dispose pas, dans sa version actuelle, d'un réel interpréteur de commandes, qui permettrait au concepteur d'interagir finement sur les modules et les données (pour tester un module isolément, par exemple). L'intelligibilité et le naturel des voix disponibles sont forcément limités par les hypothèses qui sous-tendent les technologies utilisées pour chacun des modules (pré-traitement assez rudimentaire, étiquetage statistique, prosodie par concaténation de patrons intonatifs, et synthèse par concaténation de diphones), ainsi que par la disponibilité de bases de données de taille et de qualité suffisante pour la mise en œuvre de ces technologies.
Plusieurs scénarii sont envisageables pour apporter de nouvelles ressources dans le projet Euler et contribuer ainsi à son extension.

5.1 Nouvelles bases de données

Le projet Euler peut être augmenté de nouvelles bases de données. C'est le cas typique de la réalisation d'une nouvelle voix ou de l'adaptation de lexiques à des cas particuliers. Ces nouvelles ressources doivent donc être adaptées au format de base de données actuel, puis simplement déclarées dans le fichier d'initialisation du noyau d'Euler pour être utilisées.

5.2 Nouveaux algorithmes

Comme déjà mentionné plus haut, l'utilisation d'Euler n'impose pas l'usage d'algorithmes particuliers à l'intérieur des modules. Les algorithmes disponibles sont tout au plus une aide au développement rapide de systèmes TTS.

L'ajout d'un nouvel algorithme peut se faire selon plusieurs cas de figure :
— Si une implémentation de cet algorithme existe déjà dans un des langages couverts par les moteurs d'Euler (PROLOG, PERL ou en C++) , il suffit de l'interfacer avec le moteur en question.
— Dans le cas contraire, il faudra :
- soit traduire l'algorithme en langage C/C++ pour compiler un nouveau moteur
- soit traduire l'algorithme dans le langage d'un des moteurs existants et l'interfacer au moteur en question
- soit créer un moteur intégrant le langage dans lequel est écrit l'algorithme et y interfacer l'algorithme.

5.3 Nouveaux modules

Les modules déjà implémentés dans Euler ne couvrent pas nécessairement tous les traitements que l'on pourrait s'attendre à trouver dans un synthétiseur à partir du texte. Ainsi, par exemple, l'analyse contextuelle proposée suppose l'existence, en amont, d'un analyseur morphologique indépendant. Dans certains cas, il peut être intéressant de réaliser ces deux opérations en même temps (comme par exemple dans le système Mingus [MER98]). L'utilisation d'Euler nécessitera dans ce cas de créer un nouveau module.
Dans le même ordre d'idées, il se peut qu'un développeur veuille modifier la structure des éléments d'une couche. Il conviendra dans ce cas de créer une nouvelle couche. Les noms des couches étant définis dans les modules, créer une nouvelle couche (dont le nom peut alors être défini par le concepteur) impliquera de créer des modules qui la manipulent (si possible à partir de modules existant).
Nous fournissons à cet effet un petit utilitaire qui permet au concepteur de spécifier les couches utiles et crée au vol les sources d'un module ne réalisant aucun traitement particulier tout en donnant un accès facile aux couches demandées.

5.4 Nouvelles langues

Pour la création d'une nouvelle langue, il faut, autant que possible, réutiliser au maximum les ressources existantes (éléments, modules, moteurs et base de données) afin de minimiser le temps de création. L'existence préalable de corpus, de lexiques, et autres bases de données texte/parole, est un avantage certain. La compatibilité de composants Euler avec Festival au niveau des CARTS, avec Festival et CMU au niveau des *n*-grammes, avec MBRDICO au niveau des arbres de décision phonétiques, et avec MBROLA au niveau des bases de données de voix devrait contribuer à accélérer le processus. Au pire, il est nécessaire de créer de nouvelles ressources. Dans la mesure du possible, nous mettons les outils nécessaires à disposition.

5.5 Nouvelles fonctionnalités

Pour inclure de nouvelles fonctionnalités au projet Euler, il faut y intégrer toute une série de nouveaux composants, qui interagiront en général en profondeur avec le

noyau. Ainsi, l'ajout d'un «éditeur de MLC» requiert la maîtrise de l'objet sous-jacent ; l'ajout d'un interpréteur de commandes nécessite une bonne connaissance des communications entre le noyau et les modules d'Euler ; l'interfaçage avec de visages parlants impose de gérer la synchronisation entre le visage et le son produit en temps réel. Néanmoins, l'intégration de ces nouveaux composants permet de donner un nouvel élan à toutes les applications utilisant le noyau Euler. Par exemple, développer un visage parlant dans Euler permet de bénéficier de toutes les langues existantes, et d'autre part, le développement d'une nouvelle langue héritera automatiquement des fonctionnalités du visage parlant.

6 Conclusions et perspectives

La vocation du projet Euler est de constituer une plate-forme de recherche et de développement en synthèse vocale multilingue. Le logiciel Euler est gratuit et libre d'utilisation dans le cadre de la recherche (utilisation non commerciale et non militaire). La plus grande partie des sources (le noyau et la plupart des modules) est publique sous licence GNU.

Les codes sources, écrits en C++, ont été développés pour les plates-formes Windows (depuis la version 1.00), et Linux (depuis la version 2.00).

La protection de la propriété intellectuelle des développeurs est assurée sur plusieurs niveaux :

— les moteurs, modules et bases de données, sont systématiquement fournis avec des marques de copyright et licences appropriées ;

— les applications logicielles utilisant Euler affichent automatiquement ces informations au lancement de l'application.

Deux langues sont actuellement complètement réalisées: le français et l'arabe. La version 2.00 d'Euler offre, en outre, une synthèse embryonnaire (sans intonation) de l'anglais américain, de l'anglais insulaire, de l'espagnol, du turc et du néerlandais.

L'extension à d'autres langues est grandement facilitée par la généricité des composants logiciels d'Euler.

Une information plus complète est disponible dans la documentation d'Euler à l'adresse http://tcts.fpms.ac.be/synthesis/Euler/doc/index.html. Il est également intéressant de consulter le code du noyau et des différents modules d'Euler pour avoir plus de précisions sur les méthodes et les fonctions disponibles.

Parmi les extensions prévues, la synthèse vocale par sélection et concaténation automatique d'unités non uniformes de parole dans une grande base de données ouvre la porte vers de la parole de synthèse plus naturelle et intelligible [HUN96]. Cette nouvelle génération de synthétiseur basé sur ces principes sera intégrée dans Euler en tant que module.

Nous travaillons également à l'extension du projet Euler à d'autres langues, et envisageons de porter le logiciel sur MacOs pour une utilisation plus large.

7 Remerciements

Les auteurs tiennent à remercier tous ceux qui ont contribué au développement de ce projet : Piet Mertens pour ses outils Morlex et Vertex, Richard Beaufort et Gilles Casse pour leurs contributions au dictionnaire phonétique français, BABEL Technologies S.A. pour son engagement en faveur de la mise à disposition gratuite d'outils à des fins non commerciales, Alan Black pour son soutien psychologique lors de l'élaboration de la MLC, nos collègues de l'Université d'Aix-en-Provence pour leurs encouragements lors de la genèse d'Euler, et Xavier Ricco pour son soutien logistique.

8 Bibliographie

[ALL87] Allen, J., S. Hunnicut, and D. Klatt, (1987), *From Text to Speech, The MITALK System*, Cambridge University Press, Cambridge.

[BLA97] A.W. Black, P. Taylor and R. Caley, 1997, *"The Festival Speech Synthesis System*", University of Edinburgh, http://www.cstr.ed.ac.uk/projects/festival

[BLA98] A. Black, K. Lenzo, V. Pagel, 1998, «Issues in Building General Letter to Sound Rules", *Proc. 3rd ESCA/COCSADA Workshop on Speech Synthesis*, Jenolan Caves, Australia, pp. 77-81.

[BOI00] R. Boite, H. Bourlard, T.Dutoit, J. Hancq, H. Leich, 2000, «*Traitement de la Parole"*, Presses Polytechniques Universitaires Romandes, Lausanne, 2000.

[CAR76] R. Carlson, and B. Granström, (1976), «A Text-to-Speech System Based Entirely on Rules", *Proceedings of ICASSP 76*, Philadelphia, pp. 686-688.

[CMU] *The CMU Statistical Language Modeling Toolkit*, http://www.speech.cs.cmu.edu/speech/SLM_info.html

[DUT96] T.Dutoit, V. Pagel, N. Pierret, F. Bataille, O. Van Der Vrecken, 1996, «The MBROLA project : towards a set of high quality speech synthetizers free of use for non commercial purpose", *Proc. ICSLP'96*, http://tcts.fpms.ac.be/synthesis

[FRE91] Frenkenberger, S., M. Kommenda, and S. Mossmüller, (1991), «A Multi-Linear Representation and Rule Formalism for Phonetics in Text-to-Speech Synthesis", *Proceedings of the XII'th International Congress of Phonetic Sciences,* Aix-en-Provence, vol. 2, pp. 518-521.

[HER82] Hertz, S.R., J. Kadin, and K. Karplus, (1985), «The DELTA Rule Development System for Speech Synthesis from Text", *IEEE Proceedings. on Acoustics, Speech, and Signal Processing*, n°73, pp. 1589-1601.

[HUN96] A. Hunt, A. Black, «Unit selection in a concatenative speech synthesis system using a large speech database", *Proc. ICASSP 96* Atlanta, Georgia, pp. 373-376.

[LAZ87] Lazzaretto, S., and S. Nebbia, (1987), «SCYLA: Speech Compiler for Your Language", *Proceedings of the European Conference on Speech Technology 87*, Edinburgh, vol. 1, pp. 381-384.

[LEU93] Van Leeuwen and E. Te Lindert, 1993*, «*Speech Maker: a flexible and general framework for text-to-speech synthesis, and its application to Dutch", *Computer Speech and Language*, pp. 149-167.

[LIB92] Liberman.J., and K.W. Church, (1992), «Text Analysis and Word Pronunciation in Text-to-Speech Synthesis", in *Advances in Speech Signal Processing*, S. Furui, M.M. Sondhi, eds., Dekker, New York, pp.791-831.

[MAL97] F.Malfrère and T.Dutoit, 1997, «High Quality Speech Synthesis for Phonetic Speech Segmentation", *Proceedings of EuroSpeech'97*, pp. 2631-2634.

[MAL98] F. Malfrère, T. Dutoit, P. Mertens, 1998, «Un générateur de Parole Tout Automatique", *Proc. XXIIèmes Journées d'Etudes sur la Parole*, Martigny, pp. 147-150.

[MBRDI] The MBRDICO Project : http://tcts.fpms.ac.be/synthesis/mbrdico

[MAC87] MAC ALLISTER, M., (1987), «The Structural Design of the CSTR Text-to-Speech System", *Proceedings of the European Conference on Speech Technology 87,* Edinburgh, vol. 1, pp. 59-62.

[MER98] P.Mertens, *"The MINGUS TTS*", http://bach.arts.kuleuven.ac.be/~piet/prosody/mingus.html

[MERTE] P. Mertens, «*Morlex I a lexical database for French",* http://bach.arts.kuleuven.ac.be/~piet/morlex/index.html

[PAG98] V. Pagel, K. Lenzo, A. Black, 1998, «Letter-to-Sound Rules for Accented Lexicon Compression", *Proc. ICSLP'98,* Sydney, Australia, pp. 252-255.

[PIT83] Pitrat, J., (1983), « Réalisation d'un Analyseur Lexicographique Général », *Rapport de recherche n°79/2*, GR22, Institut de programmation, Paris VI.

[ROSEN] R. Rosenfeld, P. Clarkson, *SLMT ToolKit*, Carnegie Mellon University, Cambridge university, http://svr-www.eng.cam.ac.uk/~prc14/toolkit.htm

[SPR98] R. W. Sproat. «Multilingual Text-to-Speech Synthesis: The Bell Labs Approach.» Dordrecht ; Boston : Kluwer Academic Publishers., 1998.

[VAN89] Van Coile, B.M.J., (1989), «The DEPES Development System for Text-to-Speech Synthesis, *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 89*, Glasgow, pp. 250-253.