

Université de Marne-La-Vallée

Institut Gaspard-Monge

Thèse de Doctorat

Spécialité : Informatique Fondamentale

Présentée par

*Sung-Woo Choi*

Pour l'obtention du titre de  
Docteur de l'Université de Marne-La-Vallée

Sur le sujet

Implantation de dictionnaires électroniques du coréen par automates finis

Soutenue le mardi 7 Septembre 1999 devant le jury composé de :

Maxime Crochemore

Maurice Gross

Franz Güenthner (*Rapporteur*)

Éric Laporte (*Rapporteur*)

Dominique Perrin (*Directeur*)

# Résumé

Ce travail porte sur le traitement automatique d'un lexique du coréen. Comme le coréen est une langue agglutinante, les informations phonématiques sont essentielles pour le traitement des formes fléchies en coréen. Cependant le système de codage '*KSC5601-1987 Hangul WANSUNG*', couramment utilisé, n'en fournit pas en raison de sa structure même. Ainsi, le système '*Hangul Code Manager (HANCoM)*' a été développé pour pallier ce manque. Ce système permet également les conversions entre les différents codes utilisés pour représenter le coréen en machine.

Les informations statistiques sont utiles tant pour analyser des textes que pour le traitement des langues naturelles. C'est la raison pour laquelle de nombreuses caractéristiques statistiques des textes coréens sont examinées par le biais d'expériences sur un grand corpus. A travers ces expériences, nous avons étudié le nombre d'occurrences des syllabes et des mots. Les entropies des mots et des syllabes ont été estimées à partir des connaissances obtenues de la fréquence de leurs occurrences. L'entropie des digramme de séquences a été également calculée en utilisant un modèle de contexte-fini.

La loi de Zipf est étudiée ici sur les informations statistiques dans les textes coréens. Pour tester son application sur le coréen, on a proposé deux modèles, *modèle modifié de distribution de Mandelbrot* et *modèle des trois parties*. Les textes coréens se conforment à une variante de la loi de Zipf, à quelques différences près. Nous avons également vérifié que les types de la distribution de fréquence sont indépendantes de la nature et de la taille du corpus et ne dépendent que de la langue elle-même.

Les dictionnaires électroniques du coréen sont représentés par automate acyclique minimal. Une méthode utilisant *la table des transitions inverses* a été introduite pour minimiser l'automate acyclique déterministe. Deux nouveaux dictionnaires *DECO-RA* et *DECO-FlexAV* ont été élaborés pour engendrer comme entrées des formes fléchies coréennes *DECOF*. Les entrées du *DECOF* s'élèvent à un nombre énorme (plus de 148 millions). Pour construire des automates pour les entrées du *DECOF*, nous avons inventé une méthode faisant appel à deux automates et à une *matrice de cartographie* pour la combinaison. Ces démarches méthodiques nous permettent de représenter les dictionnaires électroniques du coréen par automates. Dans le cas du *DECOF*, la réduction d'espace dans la mémoire utilisée constitue une caractéristique très importante.

# Abstract

This work concerns the automatic treatment of Korean texts in the natural language processing. Since Korean is an agglutinative language, information at the level of phonemes is essential for the treatment of the Korean inflectional forms. Although the ‘KSC5601-1987 Hangul WANSUNG’ code system is widely used, it does not provide any information of phonemes because of its structure. The system ‘Hangul Code Manager (HANCoM)’ is developed to mitigate this lack. This system also allows a code conversion function for translating codes between the various codes used to represent Korean in the computers.

The statistical data are useful to analyze texts and for the treatment of the natural language. Some statistical characteristics of Korean texts are analyzed by experiments on large corpora. We obtain the number of occurrences of syllables and of words in Korean texts. The entropy of the words and the syllables are estimated thanks to the knowledge of the frequency of their occurrences by using finite context model. Digram and trigram entropy of syllables are also estimated.

We try to examine how Korean text obeys the well-known Zipf’s law. Two mathematical models are constructed by modifying Mandelbrot distribution and are simulated for Korean texts. The Korean text obeys a variant of Zipf’s law, though the model is somewhat different. We also checked that the shape of frequency distribution is independent of the kind and of the size of the corpus. It only depends on the language.

The Korean electronic dictionaries are represented by minimal acyclic automata. A method using the *reverse transition table* is introduced to minimize the deterministic acyclic automaton. Two new dictionaries *DECO-RA* and *DECO-FlexAV* are established to generate the entries of the dictionary of the Korean inflectional forms *DECOF*. The number of entries of the *DECOF* is enormous (more than 148 million). To establish the automata for the entries of the *DECOF*, a method by two automata and a *mapping matrix* for combination were introduced. Using these methods, we can represent the Korean electronic dictionaries by automata. In the case of the *DECOF*, the reduction of the memory space requirement is very important.

# Table des matières

|                                                                                     |           |
|-------------------------------------------------------------------------------------|-----------|
| <b>Introduction</b>                                                                 | <b>1</b>  |
| <b>1. <i>Hangul</i> et systèmes de codage du coréen</b>                             | <b>5</b>  |
| 1.1 Système d'écriture du coréen . . . . .                                          | 5         |
| 1.1.1 Alphabets . . . . .                                                           | 5         |
| 1.1.2 Syllabes . . . . .                                                            | 6         |
| 1.1.3 Mots . . . . .                                                                | 7         |
| 1.2 Systèmes de codages du coréen . . . . .                                         | 8         |
| 1.2.1 Système de codage sur <i>n</i> octets . . . . .                               | 8         |
| 1.2.2 Systèmes de codage sur deux octets <i>JOHAB</i> . . . . .                     | 9         |
| 1.2.3 Système de codage sur deux octets <i>Hangul WANSUNG</i> . . . . .             | 11        |
| 1.2.4 Structure du système de code ' <i>KSC5601-1987 Hangul WANSUNG</i> ' . . . . . | 13        |
| <b>2. Dictionnaires et Automates finis</b>                                          | <b>15</b> |
| 2.1 Dictionnaires électroniques du coréen . . . . .                                 | 16        |
| 2.1.1 <i>DECOS</i> . . . . .                                                        | 16        |
| 2.1.2 <i>DECOF</i> . . . . .                                                        | 18        |
| 2.1.3 <i>DECOC</i> . . . . .                                                        | 19        |
| 2.1.4 <i>DECO-RA</i> et <i>DECO-FlexAV</i> . . . . .                                | 20        |
| 2.2 Automates finis . . . . .                                                       | 20        |
| <b>I. Le système <i>HANCoM</i> et les statistiques</b>                              | <b>23</b> |
| <b>1. Le système <i>HANCoM</i></b>                                                  | <b>25</b> |
| 1.1 Système de gestion du code <i>Hangul (HANCoM)</i> . . . . .                     | 25        |
| 1.1.1 Les tableaux du système <i>HANCoM</i> . . . . .                               | 26        |
| 1.1.2 Etablissement de la table CIP . . . . .                                       | 28        |
| 1.1.3 Indexation entre des tableaux . . . . .                                       | 31        |

|            |                                                                        |           |
|------------|------------------------------------------------------------------------|-----------|
| 1.2        | Algorithmes . . . . .                                                  | 33        |
| 1.2.1      | Résolution . . . . .                                                   | 33        |
| 1.2.2      | Recombinaison . . . . .                                                | 34        |
| 1.3        | Applications prévues . . . . .                                         | 35        |
| <b>2.</b>  | <b>Les statistiques du coréen et la loi de Zipf</b>                    | <b>37</b> |
| 2.1        | Etablissement du corpus d'essai . . . . .                              | 37        |
| 2.2        | Expérimentations . . . . .                                             | 38        |
| 2.2.1      | Entropie . . . . .                                                     | 38        |
| 2.2.2      | Fréquences des syllabes en coréen . . . . .                            | 41        |
| 2.2.3      | Fréquences des mots en coréen . . . . .                                | 42        |
| 2.3        | La loi de Zipf . . . . .                                               | 45        |
| 2.3.1      | La loi de Zipf appliquée à l'anglais . . . . .                         | 45        |
| 2.3.2      | La loi de Zipf appliquée au français . . . . .                         | 47        |
| 2.3.3      | La loi de Zipf appliquée au coréen . . . . .                           | 49        |
| 2.3.4      | Le modèle modifié de distribution de <i>Mandelbrot</i> . . . . .       | 51        |
| 2.3.5      | Le modèle des trois parties . . . . .                                  | 52        |
| <b>II.</b> | <b>Implantation de dictionnaires du coréen<br/>par Automates finis</b> | <b>55</b> |
| <b>1.</b>  | <b>Stratégies</b>                                                      | <b>57</b> |
| 1.1        | Utilisation restreinte du code de syllabes . . . . .                   | 58        |
| 1.2        | Automate acyclique déterministe . . . . .                              | 59        |
| 1.3        | Minimisations . . . . .                                                | 60        |
| 1.4        | Compression de la table des transitions . . . . .                      | 61        |
| 1.5        | Matrice de cartographie pour <i>DECOF</i> . . . . .                    | 62        |
| <b>2.</b>  | <b>Construction de l'Automate pour <i>DECOS</i></b>                    | <b>63</b> |
| 2.1        | Construction de l'automate . . . . .                                   | 63        |
| 2.1.1      | Codes numériques du <i>DECOS</i> . . . . .                             | 63        |
| 2.1.2      | Indices du code de syllabes . . . . .                                  | 64        |
| 2.1.3      | Représentation des états terminaux . . . . .                           | 66        |
| 2.1.4      | Structure de la table des transitions . . . . .                        | 67        |
| 2.1.5      | Table des transitions inverses . . . . .                               | 68        |
| 2.1.6      | Variantes de la table des transitions inverses . . . . .               | 69        |

|           |                                                                |            |
|-----------|----------------------------------------------------------------|------------|
| 2.2       | Exemples . . . . .                                             | 71         |
| 2.2.1     | Déterminisation . . . . .                                      | 71         |
| 2.2.2     | Minimisation . . . . .                                         | 77         |
| 2.2.3     | Compression de la table des transitions . . . . .              | 85         |
| <b>3.</b> | <b>Construction des Automates pour <i>DECOF</i></b>            | <b>89</b>  |
| 3.1       | Organisation du dictionnaire <i>DECO-RA</i> . . . . .          | 89         |
| 3.1.1     | Extension du radical . . . . .                                 | 90         |
| 3.1.2     | Classification des radicaux . . . . .                          | 93         |
| 3.1.3     | Le dictionnaire <i>DECO-RA</i> . . . . .                       | 93         |
| 3.1.4     | Les entrées de l'automate du <i>DECO-RA</i> . . . . .          | 95         |
| 3.2       | Organisation du dictionnaire <i>DECO-FlexAV</i> . . . . .      | 96         |
| 3.2.1     | Classification des suffixes flexionnels . . . . .              | 96         |
| 3.2.2     | Etablissement des entrées du <i>DECO-FlexAV</i> . . . . .      | 98         |
| 3.3       | Automates pour <i>DECOF</i> . . . . .                          | 101        |
| 3.3.1     | Automates du <i>DECO-RA</i> et du <i>DECO-FlexAV</i> . . . . . | 102        |
| 3.3.2     | Matrice de Cartographie pour la Combinaison . . . . .          | 102        |
| 3.3.3     | Consultation sur les automates pour le <i>DECOF</i> . . . . .  | 104        |
| 3.3.4     | Quelques remarques sur les surcharges . . . . .                | 106        |
| 3.4       | Résultats sur les automates . . . . .                          | 108        |
|           | <b>Conclusion</b>                                              | <b>111</b> |
|           | <b>Bibliographie</b>                                           | <b>113</b> |
|           | <b>Annexe</b>                                                  | <b>117</b> |

## TRANSCRIPTION

Les coréen a 14 consonnes et 10 voyelles simples, transcrites phonétiquement de la façon suivante :

### *Consonnes*

|          |          |          |          |          |          |          |           |          |           |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|-----------|----------|-----------|----------|----------|----------|----------|
| ㄱ        | ㄴ        | ㄷ        | ㄹ        | ㅁ        | ㅂ        | ㅅ        | ㅇ         | ㅈ        | ㅊ         | ㅋ        | ㅌ        | ㅍ        | ㅎ        |
| <i>g</i> | <i>n</i> | <i>d</i> | <i>l</i> | <i>m</i> | <i>b</i> | <i>s</i> | <i>ng</i> | <i>j</i> | <i>ch</i> | <i>k</i> | <i>t</i> | <i>p</i> | <i>h</i> |

### *Voyelles*

|          |           |          |           |          |           |          |           |           |          |
|----------|-----------|----------|-----------|----------|-----------|----------|-----------|-----------|----------|
| ㅏ        | ㅑ         | ㅓ        | ㅕ         | ㅗ        | ㅛ         | ㅜ        | ㅠ         | ㅡ         | ㅣ        |
| <i>a</i> | <i>ya</i> | <i>e</i> | <i>ye</i> | <i>o</i> | <i>yo</i> | <i>u</i> | <i>yu</i> | <i>eu</i> | <i>i</i> |

et des consonnes et voyelles complexes :

### *Consonnes doubles*

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| ㄲ         | ㄸ         | ㅃ         | ㅆ         | ㅉ         |
| <i>gg</i> | <i>dd</i> | <i>bb</i> | <i>ss</i> | <i>jj</i> |

### *Autre Consonnes complexes*

|           |           |           |           |           |           |           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| ㄴㅇ        | ㄴㅅ        | ㄴㅎ        | ㄹㅇ        | ㄹㅅ        | ㄹㅎ        | ㄷㅇ        | ㄷㅅ        | ㄷㅎ        | ㄷㅇ        | ㄷㅅ        |
| <i>gs</i> | <i>nj</i> | <i>nh</i> | <i>lg</i> | <i>lm</i> | <i>lb</i> | <i>ls</i> | <i>lt</i> | <i>lp</i> | <i>lh</i> | <i>bs</i> |

### *Voyelles Complexes (deux lettres)*

|           |            |           |            |           |           |           |           |            |
|-----------|------------|-----------|------------|-----------|-----------|-----------|-----------|------------|
| ㅏㅣ        | ㅑㅣ         | ㅓㅣ        | ㅕㅣ         | ㅗㅏ        | ㅛㅑ        | ㅜㅓ        | ㅠㅕ        | ㅡㅣ         |
| <i>ai</i> | <i>yai</i> | <i>ei</i> | <i>yei</i> | <i>wa</i> | <i>oi</i> | <i>we</i> | <i>wi</i> | <i>eui</i> |

### *Voyelles Complexes (trois lettres)*

|            |            |
|------------|------------|
| ㅗㅣㅣ        | ㅛㅣㅣ        |
| <i>wai</i> | <i>wei</i> |

# Introduction

Cette étude a pour objet le traitement automatique d'un lexique du coréen. Certaines caractéristiques de la langue coréenne sont étudiées dans la première partie. La langue coréenne a son propre système d'écriture et son propre alphabet. Dans le chapitre 1 de la partie introductive, nous présentons l'alphabet coréen et le mécanisme de construction de la syllabe graphique. Ensuite, nous définissons le mot coréen par rapport au mot d'une langue écrite avec l'alphabet latin. Nous récapitulons les structures et les caractéristiques d'un certain nombre de systèmes de codage pour le coréen. Pour traiter les formes fléchies coréennes, le traitement des syllabes au niveau phonématique est essentiel, le système '*Hangul Code Manager (HANCoM)*' a été élaboré à cette fin.

Les informations statistiques sur la fréquence des lettres, la fréquence des mots, la longueur des mots et l'entropie sont utiles tant pour analyser des textes que pour le traitement des langues naturelles. Nous avons étudié certaines caractéristiques statistiques des syllabes coréennes et des mots coréens. Nous avons fondé la loi de Zipf appliquée au coréen sur les données statistiques du coréen.

Pour le traitement automatique d'un lexique du coréen, les dictionnaires électroniques sont des outils essentiels. Le système lexical *DECO*<sup>1</sup> du coréen a été établi de façon systématique et exhaustive par *Jee-sun NAM* [Nam94], [Nam96a], [Nam96b]. Le cadre théorique et méthodologique adopté dans le système *DECO* est le même que celui du système du dictionnaire électronique *DELA*<sup>2</sup> construit au *LADL*<sup>3</sup>. Le *DECOS* contient au total environ 35000 entrées sous forme normée. A chaque entrée sont associés des codes qui comportent des informations telles que certains traits sémantiques et syntaxiques, des noms et une partie du discours. Le *DECOS-NS* contient 15000 entrées qui sont recensées d'abord à partir de grands dictionnaires éditoriaux du coréen. Le *DECOS-VS* contient 7500 entrées. Le lexique des adjectifs simples *DECOS-AS* contient 5300 entrées.

---

<sup>1</sup> DECO : Dictionnaire Electronique du Coréen.

<sup>2</sup> DELA : Dictionnaire Electronique construit au L.A.D.L.

<sup>3</sup> LADL : Laboratoire d'Automatique Documentaire et Linguistique. Université Paris 7.

Deux nouveaux dictionnaires ont été établis à partir du système *DECO* pour représenter le dictionnaire de formes fléchies coréennes *DECOF* par automates. Le nombre d'entrées du dictionnaire *DECOF* est énorme, environ 148 millions.

Les automates acycliques sont une structure de données bien adaptée à la représentation des lexiques des langues naturelles. Le grand avantage de la représentation par automates réside dans le fait que la vitesse d'accès aux mots du dictionnaire est très rapide, l'accès s'effectue en un temps en  $O(|w|)$  où  $|w|$  est la longueur du mot considéré. Des méthodes de construction d'automate pour le *DECOS* et le *DECOF* sont proposées.

## Le système *HANCoM* et les statistiques

Comme le coréen est une langue agglutinante, un mot se compose d'éléments lexicaux et de zéro, un ou plusieurs suffixes. Pour cette raison, le traitement du code au niveau phonématique est essentiel pour analyser les formes fléchies en coréen. Les systèmes de codage pour le coréen sont généralement classifiés en trois principaux types: les systèmes de codage sur  $n$  octets, les systèmes de codage *JOHAB* sur 2 octets et les systèmes de codage *WANSUNG* sur 2 octets. Les systèmes de codage *JOHAB* et *WANSUNG* sont des codes de type syllabique. Le '*KSC 5601-1987 Hangul WANSUNG*' est un des plus utilisés de ces systèmes de codes. Cependant, il s'agit d'un système syllabique, qui ne fournit pas d'informations sur les phonèmes.

Nous avons élaboré une méthode simple qui permet d'associer à chaque syllabe les informations phonématiques correspondantes. Nous avons établi une table *CIP* (*Codes des Informations Phonématiques*) qui est utilisée par notre système '*Hangul Code Manager (HANCoM)*'. Ce système peut décomposer une syllabe en phonèmes et reconstituer une syllabe à partir de phonèmes. De plus, il est équipé d'une fonction de conversion qui traduit les codes entre les différents types de systèmes de codage coréens.

Pour les informations statistiques, le nombre d'occurrences de syllabes et de mots ont été obtenus sur le corpus du *KAIST*<sup>4</sup>. L'entropie peut être employée pour mettre en œuvre des systèmes de compression de textes ou pour traiter d'autres problèmes de décision dans les méthodes de traitement de textes qui doivent tenir compte de l'incertitude. À l'aide des résultats statistiques obtenus, la loi de Zipf est étudiée dans le cas du texte coréen. Deux modèles, *le modèle modifié de distribution de Mandelbrot* et *le modèle des trois parties*, sont proposés pour tester la loi de Zipf pour le coréen. Le texte coréen obéit à une variante de la loi de Zipf, bien que le modèle soit quelque peu différent. De plus, un certain nombre de données statistiques ont été rassemblées.

---

<sup>4</sup> KAIST : Korea Advanced Institute of Science and Technology

## Implantation de dictionnaire du coréen par automate fini

L'implantation des dictionnaires par automates est réalisée en deux étapes. Dans la première étape, on construit un automate qui est capable de reconnaître le langage utilisé dans la liste des entrées du dictionnaire. Dans la deuxième étape, on minimise cet automate. Nous avons construit un automate acyclique déterministe en prenant un *Trie* (arbre lexicographique) comme une structure de données. Pour représenter l'automate, nous avons utilisé un tableau bidimensionnel comme structure de données. Pour la minimisation de l'automate, nous avons adopté une méthode qui utilise *la table des transitions inverses*. Nous avons utilisé cette méthode pour le dictionnaire *DECOS*.

Les verbes et les adjectifs que l'on rencontre dans un texte donné ne sont pas sous forme canonique, mais sous forme fléchie. C'est-à-dire avec des suffixes, les combinaisons sont nombreuses et s'écrivent sans espace typographique. Le mécanisme de combinaison des suffixes de conjugaison varie en fonction des éléments prédicatifs. Les grammaires scolaires donnent quelques règles locales qui s'appliquent à certains éléments prédicatifs, mais elles sont très incomplètes et en partie implicites. Par ailleurs, les variations morphologiques aux frontières des suffixes et de la racine ne sont pas indiquées de façon systématique. Le système des suffixes de conjugaison est extrêmement riche et complexe. La construction de l'automate pour le dictionnaire des formes fléchies du coréen n'est pas simple, car le nombre d'entrées de *DECOF* est énorme (plus de 148 millions).

Les lexiques des formes fléchies du coréen *DECOF* sont engendrés à partir des *DECOS* par combinaison entre radicaux et suffixes. Une racine du *DECOS* ne peut pas toujours se combiner directement avec un *suffixe flexionnel*<sup>5</sup>. Certaines variations morphologiques aux frontières des suffixes et de la racine ont lieu. La prise en compte de ces variations nécessite la possibilité d'opérations sur les phonèmes. On ne peut donc pas construire l'automate du *DECOF* de la même manière celui du *DECOS*. Nous avons construit deux automates et une *matrice de cartographie* pour réaliser le *DECOF*. Nous avons engendré un nouveau dictionnaire des radicaux *DECO-RA* de *DECOS*. Nous avons remplacé les codes d'informations associées aux entrées de *DECO-POST* par les codes de combinaison. Nous avons aussi établi un nouveau dictionnaire de suffixes flexionnels *DECO-FlexAV* en fusionnant le *DECO-POSTA* et *DECO-POSTV* établis par [Nam96b] en un seul dictionnaire. Nous avons alors construit les automates pour le *DECOF* à l'aide du *DECO-RA* et du *DECO-FlexAV*.

---

<sup>5</sup> Ou 'postposition' au sens de [Nam96b].

# Chapitre 1

## *Hangul* et systèmes de codage du coréen

Ce chapitre est consacré à l'étude de l'alphabet coréen et des systèmes de code que l'on utilise pour représenter le coréen en machine.

### 1.1 Système d'écriture du coréen

#### 1.1.1 Alphabets

La langue coréenne a son propre système d'écriture et son propre alphabet. On a appelé le système d'écriture '*Hangul*'. Nous avons appelé l'alphabet coréen '*Hangul Jamo*', ce terme est employé dans le '*Unicode standard*' pour désigner l'alphabet coréen. L'alphabet coréen se compose de 14 consonnes simples et de 10 voyelles simples. Ces lettres simples peuvent être combinées en lettres complexes. Toutes les lettres complexes n'apparaissent pas dans les textes coréens. On y trouve 51 combinaisons de lettres incluant les lettres simples et les lettres complexes (30 pour les consonnes et 21 pour les voyelles).

##### *Consonnes* (30)

**14 simples** : ㄱ ㅋ ㆁ ㄷ ㅌ ㄴ ㄹ ㅍ ㅂ ㅅ ㅈ ㅊ ㅍ ㅎ

**5 doubles** : ㄲ ㅋ ㆁ ㅌ ㅍ

**11 complexes** : ㄴㅇ ㄹㅇ ㄹㅁ ㄹㅂ ㄹㅅ ㄹㅈ ㄹㅊ ㄹㅍ ㄹㅎ

##### *Voyelles* (21)

**10 simples** : ㅏ ㅑ ㅓ ㅕ ㅗ ㅛ ㅜ ㅠ ㅡ ㅣ

**9 complexes** : ㅖ ㅗ ㅛ ㅜ ㅠ ㅑ ㅓ ㅕ ㅕ (doubles lettres)

**2 complexes** : ㅖ ㅖ (triples lettres)

Dans les langues européennes, les lettres (caractères) de l'alphabet sont les unités élémentaires, et elles sont employées en tant qu'unités de base du système d'écriture. Bien que les lettres l'alphabet coréen (*Hangul Jamo*) soient les unités élémentaires, elles ne sont

pas employés comme unité de base dans le système d'écriture coréen. En coréen, la syllabe est l'unité de base de l'écriture.

### 1.1.2 Syllabes

Les lettres forment une syllabe qui est constituée de trois parties phonématiques. La première partie est une consonne qui provient d'une des 14 consonnes simples ou des 5 doubles. On a appelé cette première consonne '*Choseong*' (séquence de consonnes initiales). La deuxième partie est une voyelle médiale qui peut être de n'importe quel genre (10 simples, 9 doubles et 2 triples voyelles). Le nom de cette voyelle médiale est '*Jungseong*' (séquence de voyelles). La troisième partie est une consonne finale qui est l'une des 14 simples, 11 complexes et 2 doubles ( $\text{ㄱ}(gg)$ ,  $\text{ㄴ}(ss)$ ) consonnes ou bien elle est vide. Nous avons appelé la consonne finale '*Jongseong*' (séquence de consonnes finales). Nous pouvons représenter les combinaisons des lettres sous forme d'automate acyclique comme dans (Figure 1.1). Graphiquement, une syllabe coréenne occupe une place et possède une des structures représentées sur la (Figure 1.2). Les composants minimaux pour construire une syllabe sont la première et la deuxième partie (c'est-à-dire *Choseong* et *Jungseong* ; les cas (1) (2) (3) dans la Figure 1.2)). Nous ne pouvons pas construire de syllabe au moyen des seules consonnes (c'est-à-dire *Choseong* ou *Jongseong*). Cependant, les voyelles de la deuxième partie (*Jungseong*) peuvent former une syllabe à elles seules avec une consonne fictive 'ㅇ(o)' comme *Choseong*.

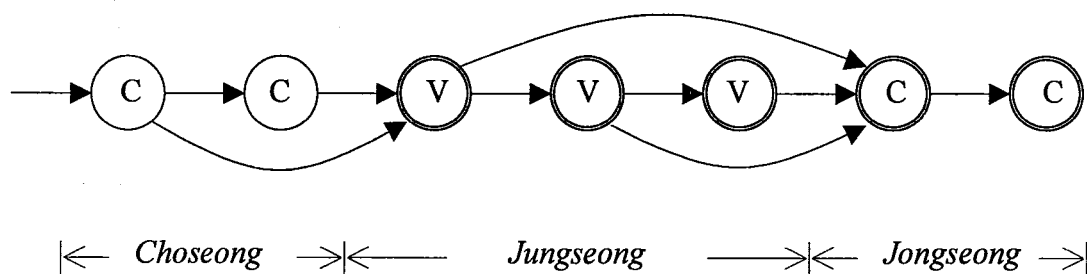


Fig. 1.1 La construction des syllabes coréennes

Mathématiquement, ces 30 consonnes et ces 21 voyelles produisent 11172 syllabes obtenues par combinaisons.

$$\begin{array}{ccccccc}
 19 \text{ consonnes} & \times & 21 \text{ voyelles} & \times & 28 \text{ consonnes} & = & 11172 \text{ syllabes} \\
 (\text{Choseong}) & & (\text{Jungseong}) & & (\text{Jongseong}) & & 
 \end{array}$$

Toutes ces combinaisons ne sont pas des syllabes réelles. Expérimentalement, nous trouvons 2139 syllabes différentes dans notre corpus.

Quand on tape des syllabes coréennes sur un clavier d'ordinateur, on entre des consonnes et des voyelles séquentiellement comme dans le cas de l'alphabet latin. Alors un module du logiciel installé dans l'ordinateur, '*l'automate des entrées de Hangul*', traduit selon leur ordre de saisie les consonnes et les voyelles en syllabes selon les règles de l'écriture coréenne et produit des codes de syllabes sur deux octets. L'unité de base de l'écriture du coréen est la syllabe. Autrement dit, on ne trouve pas les lettres de *Hangul Jamo* mais les syllabes dans les textes. C'est pourquoi on peut utiliser les codes de syllabes de la même manière que les codes des lettres de l'alphabet latin dans le domaine du traitement automatique des textes par ordinateur.

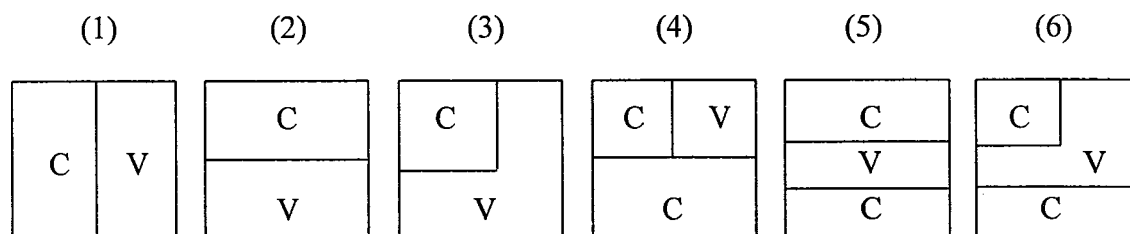


Fig.1.2 Les structures carrées graphiques des syllabes

### 1.1.3 Mots

Définir clairement la notion de mot est un problème. Dans le cas de l'anglais et d'autres langues écrites avec l'alphabet latin, pour l'analyse des textes nous pouvons définir un mot de la façon suivante. *Les mots sont généralement vus comme une suite de caractères sans l'espace.* Ainsi 'écrivain', 'écrivaine', 'écrivains' et 'écrivantes' sont tous différents, bien qu'ils partagent la même racine, puisque la forme graphique du mot n'est pas identique. Les homographes apparaîtront comme un même mot. Evidemment, 'écrivain.' et 'écrivain' sont des mots différents avec cette définition qui tient compte des séparateurs. Cette définition est raisonnable pour certaines applications comme la compression de textes. L'analyse de la lexicologie exige de considérer non seulement les espaces mais également les signes de ponctuation comme séparateurs des mots. Dans ce cas, 'écrivaine.' n'est pas un mot, mais 'écrivaine' en est un.

Dans les langues européennes, un texte est habituellement analysé à 3 niveaux : les lettres (ou caractères), les syllabes (mais les syllabes, pour des raisons culturelles et techniques ne sont pas considérées comme très importantes) et les mots. En coréen, un texte peut être analysé à 4 niveaux. Un niveau morphologique existe entre le niveau des syllabes et

le niveau du mot. Puisque le coréen est une langue agglutinante, un mot peut se composer d'éléments lexicaux et de zéro, un ou plusieurs suffixes. Par exemple dans la (Figure 1.3), les deux mots français 'à Paris' correspondent à un seul mot coréen 'Paris- Locatif', sans espace.

|                          |                |              |                                    |
|--------------------------|----------------|--------------|------------------------------------|
| 존-은                      | 파리-에           | 있다.          | ( 17 lettres, 7 syllabes, 3 mots ) |
| <i>jon-eun</i>           | <i>pali-ei</i> | <i>issda</i> |                                    |
| John-nominatif           | Paris-locatif  | est          | ( 5 éléments morphologiques )      |
| <b>John est à Paris.</b> |                |              | ( 13 lettres, 5 syllabes, 4 mots ) |

Fig. 1.3 Exemples d'éléments morphologiques

## 1.2 Systèmes de codage du coréen

On a besoin d'un minimum de 2 octets pour coder les syllabes coréennes, puisque le code ASCII occupe déjà un octet et que l'on a besoin de conserver ce code. Les lettres de l'alphabet coréen (*Hangul Jamo*) sont au nombre de 24, mais en pratique 50 combinaisons de lettres sont utilisées pour construire les syllabes coréennes. Supposons que l'on utilise simplement un octet pour coder une lettre, comme l'unité de base du coréen est la syllabe, il faudrait alors 3 octets pour représenter une syllabe coréenne. Nous présentons ici plusieurs méthodes permettant de coder les syllabes coréennes.

### 1.2.1 Le système de codage sur *n* octets

Depuis que les systèmes informatiques ont été présentés en Corée, beaucoup de systèmes de codes ont été développés pour représenter *Hangul* (les syllabes coréennes) en machine. Le système de code sur *n* octets est employé dans les ordinateurs de grande dimension. Ce genre d'ordinateurs a de nombreux terminaux d'utilisateurs. Dans ce système de codage, à chaque phonème (consonne et voyelle) est assigné 1 octet. Pour construire une syllabe, deux ou trois octets sont donc nécessaires. Le système de code sur *n* octets emploie le même secteur de codage que le code ASCII (7bit ASCII). Il doit donc utiliser des séparateurs pour distinguer le code de coréen du code ASCII. Il y a deux séparateurs : SO (*Shift On* : ^n, 0x0e) est utilisé comme marqueur de début du coréen et SI (*Shift In* : ^o, 0x0f) comme marqueur final. Nous montrons un exemple dans la (Figure 1.4).

|                    |   |                                     |   |   |   |   |   |           |         |         |         |         |        |
|--------------------|---|-------------------------------------|---|---|---|---|---|-----------|---------|---------|---------|---------|--------|
| <b>Exemple 1.1</b> |   | HANGUL 소 프 트 웨 어                    |   |   |   |   |   |           |         |         |         |         |        |
|                    |   | (so peu teu wei e)                  |   |   |   |   |   |           |         |         |         |         |        |
| Codes internes :   |   | H                                   | A | N | G | U | L | ^n        | U       | I       | ] z \ z | W u     | W f ^o |
| Taper              | : | H                                   | A | N | G | U | L | <u>SO</u> | ㅅ       | ㅍ       | ㅌ       | ㅍ       | ㅌ      |
|                    |   | ( s o p eu t eu E wei E e )         |   |   |   |   |   |           |         |         |         |         |        |
| n octets           | : |                                     |   |   |   |   |   | 2octets   | 2octets | 2octets | 2octets | 2octets |        |
| Total              | : | 10 octets pour 5 syllabes coréennes |   |   |   |   |   |           |         |         |         |         |        |

|                    |   |                                    |   |  |   |  |   |   |   |   |           |
|--------------------|---|------------------------------------|---|--|---|--|---|---|---|---|-----------|
| <b>Exemple 1.2</b> |   | 이 길 동                              |   |  |   |  |   |   |   |   |           |
|                    |   | (eu gil dong)                      |   |  |   |  |   |   |   |   |           |
| Codes internes :   |   | ^n                                 | W |  | A |  | I | G | I | W | ^o        |
| Taper              | : | <u>SO</u>                          | ㅇ |  | ㄱ |  | ㅇ | ㄷ | ㅇ | ㅇ | <u>SI</u> |
|                    |   | (E eu g i l d o ng)                |   |  |   |  |   |   |   |   |           |
| n octets           | : |                                    |   |  |   |  |   |   |   |   |           |
| Total              | : | 8 octets pour 3 syllabes coréennes |   |  |   |  |   |   |   |   |           |

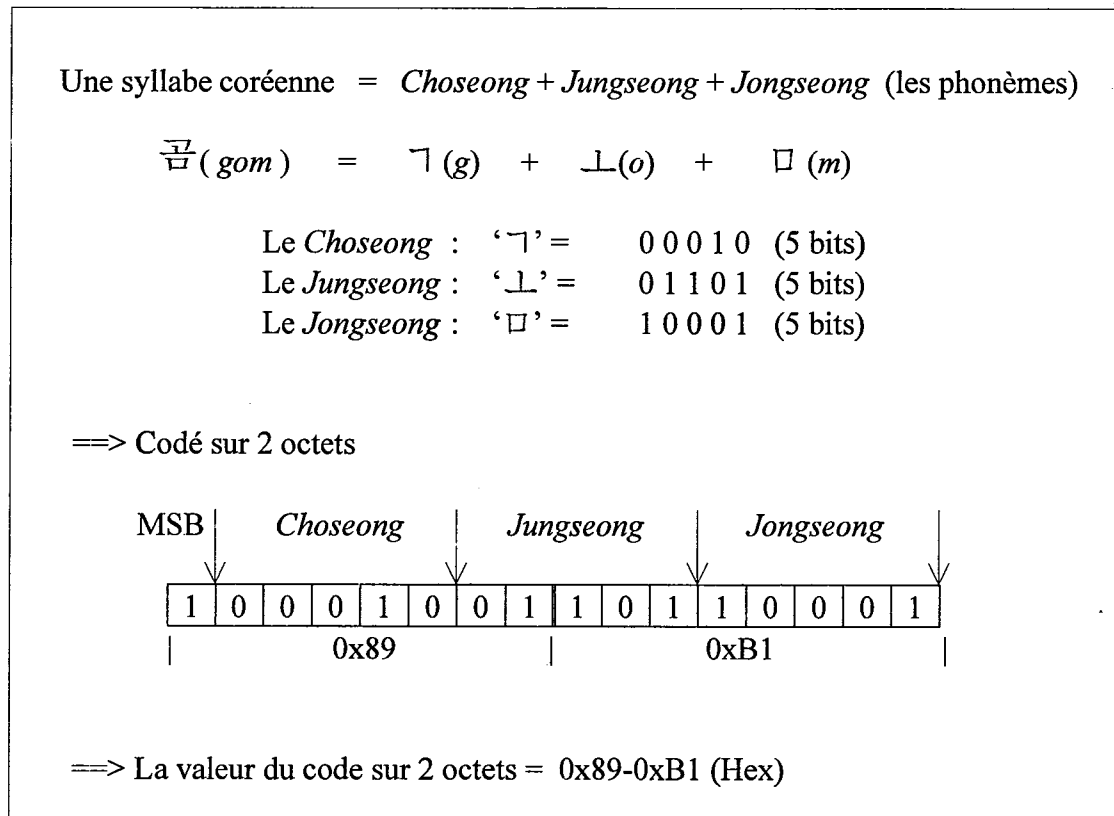
Fig. 1.4 Exemples du système de codage sur *n* octets

## 1.2.2 Les systèmes de codage sur deux octets JOHAB

Ces systèmes de codage de *Hangul* sont largement répandus par les programmeurs qui développent des logiciels en coréen. Il y a plusieurs types de systèmes de codage JOHAB construits avec le même mécanisme. Beaucoup de compagnies de logiciel ont composé et employé leur propre système de codage. C'est le cas par exemple, des systèmes suivants : 'le système de code JOHAB de COMMERCIAL', 'le système de code JOHAB de SAMSUNG', 'le système de code JOHAB de GOLDSTAR' et 'le système de code JOHAB de DOGAEBI', etc. Ils ont défini différentes tables de codes pour les phonèmes (*Choseong*, *Jungseong*, *Jongseong*), mais les mécanismes de combinaison sont identiques. Cependant, 'le système de code JOHAB de COMMERCIAL' était le plus largement répandu parmi les systèmes de codes JOHAB. Il existe depuis 1992 un standard pour le système de code JOHAB sur 2 octets, on l'appelle 'le système de code JOHAB de KSC5601-1992'. Nous comparons deux systèmes de codes JOHAB dans l'Annexe 1.

Une syllabe est codée sur deux octets par la méthode suivante. Le sens du mot coréen 'JOHAB' est 'combinaison'. Les lettres de *Jongseong* sont les plus nombreuses des trois parties phonématiques des syllabes coréennes. Elles sont au nombre de 28 en comptant le

code de remplissage qui correspond à l'absence de *Jongseong* dans la syllabe. On peut utiliser pour chaque partie phonématique 5 bits dans un octet. On combine alors ces trois 5-bits sur 2 octets comme dans la (Figure 1.5).



**Fig. 1.5** Le mécanisme de combinaison du codage *JOHAB*

Un code sur 2 octets pour une syllabe est finalement obtenu en plaçant dans le *MSB*<sup>1</sup> de l'octet supérieur le nombre binaire '1'. En regardant ce bit, on peut distinguer une série de 2 octets utilisés pour syllabe coréenne d'un octet utilisé pour autre chose comme par exemple le code ASCII. Cependant par construction, le *MSB* de l'octet inférieur peut être à '0' ou '1'. Cela peut poser des problèmes dans les systèmes de communication entre l'ordinateurs, parce que les octets utilisés pour le coréen dont le *MSB* est à '0' peuvent être confondus avec l'ASCII. On doit alors utiliser d'autres méthodes de codage pour pouvoir communiquer de données.

Les avantages de cette méthode sont les suivants. Le premier avantage est que cette méthode permet de coder toutes les syllabes possibles du coréen en combinant des phonèmes,

<sup>1</sup> MSB : Most Significant Bit ( *bit le plus significatif* ) : Le huitième bit dans un octet.

c'est-à-dire toutes les 11172 syllabes coréennes sont représentées par ce code système bien que beaucoup de syllabes ne soient pas utilisées. Le deuxième et le plus grand avantage est que ces codes fournissent des informations sur les phonèmes.

### 1.2.3 Le système de codage sur deux octets *Hangul WANSUNG*

De nos jours, le '*KSC5601-1987 Hangul WANSUNG*' est un des systèmes de codage standards les plus utilisés pour représenter le coréen en machine. Il a été créé et recommandé par le gouvernement en 1982 comme système standard de codage du *Hangul*. Il a été amélioré dans les années 1987-1989. Ce système de codage a été inventé pour communiquer *Hangul* à travers les réseaux d'ordinateurs, le système de codages *JOHAB* posant quant à lui des problèmes dans le cas des communications comme on l'a mentionné précédemment. Cependant, ce système de codage *Hangul WANSUNG* a le grand désavantage de ne pas fournir d'informations sur les phonèmes. Il s'impose à beaucoup de gens qui se sont engagés dans le développement de logiciels, en Corée, le gouvernement ayant adopté ce système de code comme système de code standard pour représenter *Hangul* dans les réseaux informatiques du gouvernement destinés à l'administration.

Ce code est établi selon le même mécanisme que celui du code ASCII. Les 2350 syllabes coréennes et 4888 caractères chinois fréquemment utilisés ont été choisis et triés selon l'ordre lexicographique du *Hangul Jamo*. Chaque syllabe et chaque caractère chinois est associé à un code selon l'ordre lexicographique. C'est pourquoi ce système code ne peut pas fournir d'informations phonématiques.

Pour coder ces 8000 caractères, on a besoin de beaucoup d'espace. Dans la table du code ASCII de 8 bits, on peut utiliser 128 codes à partir de 0x80 jusqu'à 0xFF. Il faut conserver les 32 codes de 0x80 à 0x9F pour coder les caractères de contrôle. On doit éviter les deux codes 0xA0 (espace) et 0xFF (DEL). Finalement on peut retenir les 8836 codes disponibles en combinant 2 octets de 8 bits. Nous montrons une méthode de prolongation de la table des codes dans la (Figure 1.6). Suivant cette dernière, on associe 94 syllabes à chaque page de la table des codes à partir de 0xA1 jusqu'à 0xEF. La valeur du *MSB* des codes de ce secteur est automatiquement le nombre binaire '1'. On peut alors utiliser ce *MSB* pour distinguer les codes *Hangul* des codes ASCII.

Nous montrons un extrait du système de code '*KSC5601-1987 Hangul WANSUNG*' dans la (Figure 1.7). On explique la structure et on détaille les caractéristiques de ce système de code dans la section suivante.

|   |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   | 0                                                                                                                                                                 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | <div>8 × 16 = 127      Caractères ASCII</div>                                                                                                                     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 2 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 4 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 6 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 7 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 8 | <div>2 × 16 = 32      Caractère de contrôle</div>                                                                                                                 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 9 |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A | <div>A0<br/><br/>( 6 × 16 ) – 2 = 94    Caractère graphique<br/>                                Utilisé pour les codes <i>Hangul</i><br/><br/>·<br/><br/>FF</div> |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| B |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| C |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| D |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| E |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| F |                                                                                                                                                                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

La combinaison de 2 octets : 94 × 94 = 8836 codes possibles

Exceptions : A0 = l'espace, FF = DEL  
Le *MSB* est fixé à '1' dans le secteur de 0x80 à 0xFF.



Fig. 1.6 Méthode de prolongation du code sur 8 bits

|          |             |         |
|----------|-------------|---------|
| Syllabes | Hexadécimal | Décimal |
| 가        | 0xB0A1      | 176-161 |
| 각        | 0xB0A2      | 176-162 |
| 간        | 0xB0A3      | 176-163 |
| 갈        | 0xB0A4      | 176-164 |
| 갈        | 0xB0A5      | 176-165 |
| :        | :           | :       |
| :        | :           | :       |

Fig. 1.7 Un extrait du tableau du code '*KSC5601-1987 Hangul WANSUNG*'

| HEX     | DEC       | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A  | B  | C  | D  | E  | F  |
|---------|-----------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
|         |           | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| B0 - A0 | 176 - 160 | 갈 | 가 | 각 | 간 | 갈 | 갈 | 갈 | 갈 | 갈 | 갈 | 갈  | 갈  | 갈  | 갈  | 갈  | 갈  |
| B0 - B0 | 176 - 176 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가  | 가  | 가  | 가  | 가  | 가  |
| B0 - C0 | 176 - 192 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가  | 가  | 가  | 가  | 가  | 가  |
| B0 - D0 | 176 - 208 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가  | 가  | 가  | 가  | 가  | 가  |
| B0 - E0 | 176 - 224 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가  | 가  | 가  | 가  | 가  | 가  |
| B0 - F0 | 176 - 240 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가 | 가  | 가  | 가  | 가  | 가  | 가  |
| B1 - A0 | 176 - 160 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관  | 관  | 관  | 관  | 관  | 관  |
| B1 - B0 | 176 - 176 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관  | 관  | 관  | 관  | 관  | 관  |
| B1 - C0 | 176 - 192 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관  | 관  | 관  | 관  | 관  | 관  |
| B1 - D0 | 176 - 208 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관  | 관  | 관  | 관  | 관  | 관  |
| B1 - E0 | 176 - 224 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관  | 관  | 관  | 관  | 관  | 관  |
| B1 - F0 | 176 - 240 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관 | 관  | 관  | 관  | 관  | 관  | 관  |
| B2 - A0 | 176 - 160 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸  | 갸  | 갸  | 갸  | 갸  | 갸  |
| B2 - B0 | 176 - 176 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸  | 갸  | 갸  | 갸  | 갸  | 갸  |
| B2 - C0 | 176 - 192 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸  | 갸  | 갸  | 갸  | 갸  | 갸  |
| B2 - D0 | 176 - 208 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸  | 갸  | 갸  | 갸  | 갸  | 갸  |
| B2 - E0 | 176 - 224 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸  | 갸  | 갸  | 갸  | 갸  | 갸  |
| B2 - F0 | 176 - 240 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸 | 갸  | 갸  | 갸  | 갸  | 갸  | 갸  |
| B3 - A0 | 176 - 160 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날  | 날  | 날  | 날  | 날  | 날  |
| B3 - B0 | 176 - 176 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날  | 날  | 날  | 날  | 날  | 날  |
| B3 - C0 | 176 - 192 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날  | 날  | 날  | 날  | 날  | 날  |
| B3 - D0 | 176 - 208 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날  | 날  | 날  | 날  | 날  | 날  |
| B3 - E0 | 176 - 224 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날  | 날  | 날  | 날  | 날  | 날  |
| B3 - F0 | 176 - 240 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날 | 날  | 날  | 날  | 날  | 날  | 날  |

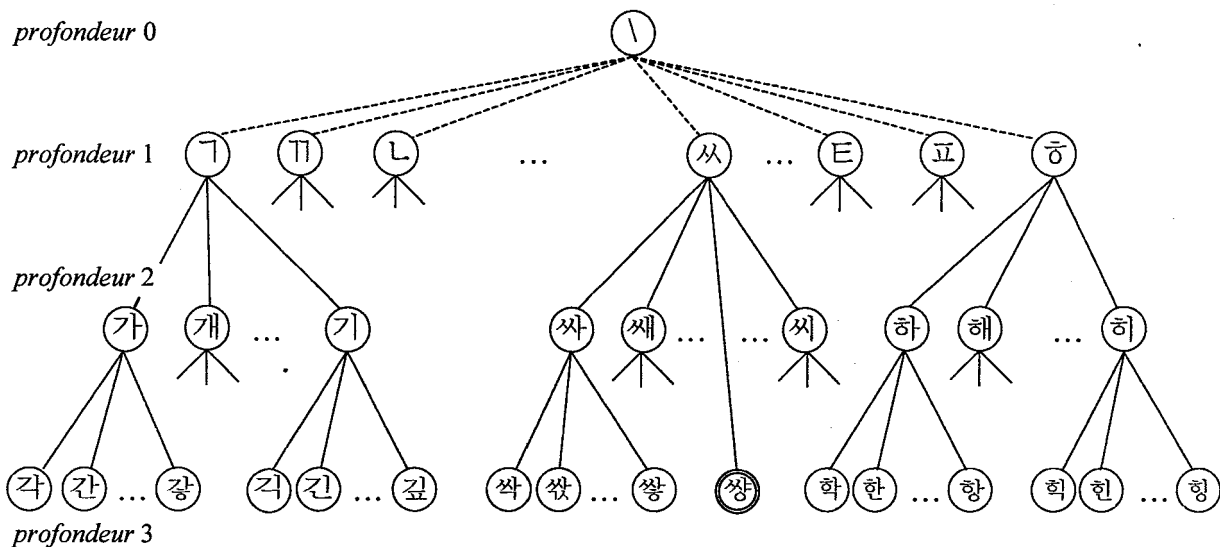
Fig. 1.7 (cont.) Un extrait du tableau du code 'KSC5601-1987 Hangul WANSUNG'

### 1.2.4 La structure du système de codage 'KSC5601-1987 Hangul WANSUNG'

On peut ranger les syllabes selon un ordre lexicographique comme dans le cas de l'alphabet latin, même si l'ordre lexicographique en coréen est un peu différent. Dans ce système de codage, les 2350 syllabes coréennes fréquemment utilisées sont énumérées séquentiellement selon l'ordre lexicographique. On peut représenter cet ordre par un arbre de hauteur 3 comme le montre la (Figure 1.8). Cet arbre expose la hiérarchie alphabétique coréenne qui conduit à cet ordre. On le simplifie en 'arbre HAC'. On peut obtenir l'ordre

lexicographique du coréen en considérant cet arbre comme 'un arbre de relation d'ordre strict'<sup>2</sup>. L'ordre sur les syllabes correspond à l'ordre sur les feuilles de cet arbre.

On dit qu'une syllabe coréenne a trois parties phonématiques. Ce sont *Choseong*, *Jungseong* et *Jongseong*. Les nœuds de profondeur 1 sont les consonnes qui sont utilisées comme première partie phonématique, *Choseong*. Les nœuds de profondeur 2 sont les voyelles *Jungseong*. Les nœuds de profondeur 3 sont les consonnes *Jongseong* qui sont utilisées comme troisième partie phonématique. Les syllabes sont donc deux types : les syllabes de profondeur 2 qui sont composées par combinaisons de *Choseong* et de *Jungseong*, et qui n'ont pas de troisième partie phonématique, *Jongseong*, et les syllabes de profondeur 3 qui sont composées des trois parties phonématiques. Le nombre maximum de feuilles qui ont le même père est 18.



- profondeur 1 : *Choseong* (consonnes)
- profondeur 2 : *Choseong* (consonnes) + *Jungseong* (voyelle)
- profondeur 3 : *Choseong* (consonnes) + *Jungseong* (voyelle) + *Jongseong* (consonnes)

Fig. 1.8 L'arbre HAC<sup>3</sup>

<sup>2</sup> preordered tree en anglais.

<sup>3</sup> HAC : Hiérarchie Alphabétique Coréenne du système de code 'KSC5601-1987 Hangul WANSUNG')

## Chapitre 2

### Dictionnaires et Automates finis

Ce chapitre est consacré à la présentation des dictionnaires électroniques du coréen établis par *Jee-Sun NAM*. Ce travail s'appuie sur le système lexical du coréen *DECO*<sup>4</sup> établi de façon systématique et exhaustive. Le cadre théorique et méthodologique adopté dans le système *DECO* est le même que celui du système *DELA*<sup>5</sup> du dictionnaire électronique construit au *LADL*<sup>6</sup>. On se reportera à ([Nam94],[Nam96a],[Nam96b]) pour les détails de sa construction. Nous présentons le système *DECO* dans les sections suivantes. Le *DECO* est principalement constitué des trois lexiques suivants :

*DECOS* : Lexique des mots simples

*DECOF* : Lexique des formes fléchies

*DECOC* : Lexique des formes complexes

Ces lexiques comportent plusieurs sous-lexiques, répartis en fonction des parties du discours. Nous montrons l'organisation du système *DECO* dans la (Figure 2.1).

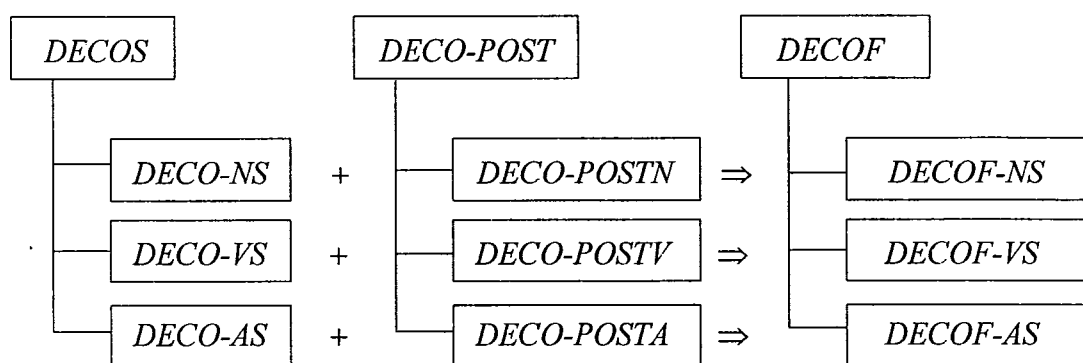


Fig. 2.1 Organisation du système *DECO*

<sup>4</sup> DECO : Dictionnaire Electronique du Coréen.

<sup>5</sup> DELA : Dictionnaire Electronique construit au L.A.D.L.

<sup>6</sup> LADL : Laboratoire d'Automatique Documentaire et Linguistique. Université Paris 7.

Le *DECO-NS* est constitué des noms simples, le *DECO-VS* des verbes simples et le *DECO-AS* est constitué des adjectifs simples. Le *DECO-POSTN* contient les suffixes flexionnels pour les noms, le *DECO-POSTV* les suffixes flexionnels pour verbes et le *DECO-POSTA* les suffixes flexionnels pour les adjectifs.

## 2.1 Dictionnaires électroniques du coréen

### 2.1.1 DECOS

Le *DECOS* est composé d'un lexique des noms simples (*DECOS-NS*), d'un lexique des verbes simples (*DECOS-VS*) et d'un lexique des adjectifs simples (*DECOS-AS*). Le *DECOS-NS* contient 15000 entrées qui sont recensées d'abord à partir de grands dictionnaires éditoriaux du coréen ([Sin92], [Lee92]). Dans le lexique des noms simples, les codes suivants doivent s'interpréter comme :

|        |                                                                                                                                                                                      |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NS     | Noms simples                                                                                                                                                                         |
| HUM    | Substantif <i>humain</i>                                                                                                                                                             |
| ANM    | Substantif désignant un <i>animal</i>                                                                                                                                                |
| PLT    | Substantif désignant une <i>plante</i>                                                                                                                                               |
| PRED1  | Substantif <i>prédicatif</i> associable avec 하다 ( <i>hada</i> ) dont l'ensemble est équivalent à un verbe <i>transitif</i> (i.e. N-하다 ( <i>hada</i> ))                               |
| PRED2  | Substantif <i>prédicatif</i> associable avec <i>hada</i> dont l'ensemble est équivalent à un verbe <i>intransitif</i> (i.e. N-하다 ( <i>hada</i> ))                                    |
| PRED3  | Substantif <i>prédicatif</i> associable avec <i>doida</i> dont l'ensemble est équivalent à un verbe <i>intransitif</i> (i.e. N-되다 ( <i>doida</i> ))                                  |
| PREDH  | Substantif <i>prédicatif</i> associable avec 하다 ( <i>hada</i> ) (i.e. N-하다 ( <i>hada</i> )), non indiqué dans les dictionnaires classiques                                           |
| PREDHA | Substantif <i>prédicatif</i> associable avec <i>hada</i> exigeant une postposition de l'accusatif (i.e. N-를 하다 ( <i>leul hada</i> )), non indiqués dans les dictionnaires classiques |
| ADJH   | Substantif dont dérive un adjectif en -하다 ( <i>hada</i> )                                                                                                                            |
| ETR    | Substantif transcrit d'un alphabet <i>étranger</i>                                                                                                                                   |
| NFA-M  | Nominalisation d'adjectif à l'aide du suffixe -음 ( <i>eum</i> )                                                                                                                      |
| NFV-M  | Nominalisation de verbe à l'aide du suffixe -음 ( <i>eum</i> )                                                                                                                        |
| NVK    | Nominalisation d'un élément <i>prédicatif</i> à l'aide du suffixe -음 ( <i>gi</i> ) (entrée enregistrée dans les dictionnaires éditoriaux, leur liste est donc incomplète)            |
| NVM    | Nominalisation d'un élément <i>prédicatif</i> à l'aide du suffixe -음 ( <i>eum</i> ) (entrée enregistrée dans les dictionnaires éditoriaux, leur liste est donc incomplète)           |
| NVS    | Nominalisation de verbe (ou d'adjectif) à l'aide d'un suffixe nominal                                                                                                                |

- \* comme -보 (*bo*) ou -개 (*gai*)  
Symbole désignant une nouvelle adjonction

Le *DECOS-VS* contient 7500 entrées auxquelles les codes suivants sont associés :

|       |                                                                    |
|-------|--------------------------------------------------------------------|
| VS    | Verbe simple                                                       |
| VC    | Verbe complexe                                                     |
| INT   | Verbe intransitif                                                  |
| TRA   | Verbe transitif                                                    |
| ADVHM | Verbe à suffixe 하다 ( <i>hada</i> ) basé sur un adverbe             |
| RHM   | Verbe à suffixe 하다 ( <i>hada</i> ) basé sur une racine lexicale    |
| ADVKM | Verbe à suffixe 거리다 ( <i>gelida</i> ) basé sur un adverbe          |
| RKM   | Verbe à suffixe 거리다 ( <i>gelida</i> ) basé sur une racine lexicale |
| JM    | Verbe à suffixe 지다 ( <i>jida</i> )                                 |
| TM    | Verbe à suffixe 뜨리다 ( <i>ddeulida</i> )                            |
| PS    | Verbe comportant un suffixe passif                                 |
| CS    | Verbe comportant un suffixe causatif                               |
| DE    | Verbe incomplet                                                    |

Le lexique des adjectifs simples *DECOS-AS* contient 5300 entrées. Certaines informations sont attachées aux entrées lexicales conformément au codage suivant :

|          |                                                                                          |
|----------|------------------------------------------------------------------------------------------|
| AADVHM   | Adjectif à suffixe 하다 ( <i>hada</i> ) basé sur un adverbe                                |
| ANHM     | Adjectif à suffixe 하다 ( <i>hada</i> ) basé sur un nom                                    |
| ARHM     | Adjectif à suffixe 하다 ( <i>hada</i> ) basé sur une racine lexicale                       |
| CM       | Adjectif à suffixe 적이다 ( <i>jegida</i> )                                                 |
| SM       | Adjectif à suffixe 스럽다 ( <i>seulebda</i> )                                               |
| IM       | Adjectif à suffixe 이다 ( <i>ida</i> )                                                     |
| JM       | Adjectif à suffixe 지다 ( <i>jida</i> )                                                    |
| LM       | Adjectif à suffixe 롬다 ( <i>lobda</i> )                                                   |
| MM       | Adjectif à suffixe 맞다 ( <i>majda</i> )                                                   |
| OYA      | Adjectif à suffixe 있다 / 없다 ( <i>issda/ebsda</i> )                                        |
| RAM/A-20 | Adjectif sans suffixe spécifique dont la classe est numérotée<br>(e. g. la classe A-20). |

Le *DECOS* contient au total environ 35000 entrées sous forme normée. A chaque entrée sont associés des codes qui comportent des informations comme certains traits

sémantiques et syntaxiques ; des noms et une partie du discours. Le *DECOS* utilise environ 141 codes distincts. Nous présentons (Figure 2.2) un extrait de ce dictionnaire.

가계 NS.  
 가공 NS. /PRED1/PRED3  
 가공적이다 ADJS. /CM  
 가관 NS.  
 가교 NS. /PRED2  
 가구 NS. /PREDHA  
 가군 NS. /HUM  
 가금 NS. /ANM  
 가까와지다 VS. /INT/JMA  
 가깝다 ADJS. /RM  
 가깝디가깝다 ADJS. /RM  
 가깝하다 ADJS. /HM  
 가꾸다 VS. /TRA  
 가꾸러뜨리다 VS. /TRA  
 가꾸러지다 VS. /INT/JM  
 가닐가닐하다 VS. /INT/ADVHM  
 가닐거리다 VS. /INT/ADVKM  
 가다 VS. /INT  
 가다랭이 NS.  
 가닥 NS.  
 가닥가닥하다 ADJS. /HM  
 가당찮다 ADJS. /RM  
 가대인 NS. /HUM  
 가도 NS.  
 가동 NS. /PRED2  
 가동가동하다 VS. /TRA/INT/ADVHM  
 가동거리다 VS. /TRA/INT/RKM

Fig. 2.2 Un extrait du *DECOS*

### 2.1.2 *DECOF*

Le dictionnaire des formes fléchies (*DECOF*) est engendré par la combinaison du *DECOS* et du *DECO-POST*. Nous présentons dans la (Figure 2.3) un extrait du dictionnaire *DECO-POSTV*. Le *DECOS-NS* et le *DECO-POSTN* permettent d'engendrer le *DECOF-NS* ; le *DECOS-AS* et le *DECO-POSTA* donnent le *DECOF-AS* ; le *DECOS-VS* et le *DECO-POSTV* produisent le *DECOF-VS*. Ce *DECOF* comporte toutes les formes fléchies des mots

simples que l'on rencontre dans des textes. Des informations morphologiques seront attachées aux entrées du *DECOS* et du *DECO-POST* pour engendrer des formes fléchies à partir des formes canoniques. Nous ne pouvons pas garder le *DECOF* sous la forme d'un texte combiné comme le *DELA*F pour le français, parce que le nombre d'entrées du *DECOF* est d'environ 100 millions<sup>7</sup>. Le nombre d'entrées du *DECOF* a été estimé comme suit par [Nam96c]. Nous montrons cette estimation dans la (Figure 2.4).

도록 \PV1\PV2\PV3\PV4\PV5 \PV6 \PV7 .Conj  
되 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj  
든 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj  
든말든 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj  
든지 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj  
들 \PV1 .Conj  
듯이 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj  
디 \PV1 \PV2 \PV3 \PV4 \PV6 \PV79 .Conj  
라 \PV1 .TmDec  
라 \PV1 .TmImp  
라거나 \PV1 \PV2 \PV5 \PV7 .CoDis  
라거늘 \PV1 \PV2 \PV5 \PV7 .CoDis

Fig. 2.3 Un extrait du dictionnaire *DECO-POST*.

|                | DECOS | DECO-POST | DECOF                 |
|----------------|-------|-----------|-----------------------|
| Nom            | 15000 | 1500      | 2.2 x 10 <sup>7</sup> |
| Verbe          | 7500  | 6000      | 4.5 x 10 <sup>7</sup> |
| Adjectif       | 5300  | 6000      | 3.2 x 10 <sup>7</sup> |
| Mot invariable | 7200  |           | 7200                  |
| Total          | 35000 | 13500     | 9.9 x 10 <sup>7</sup> |

Fig. 2.4 Le nombre d'entrées du *DECOF*

<sup>7</sup> Nombre d'entrées obtenu par estimation.

### 2.1.3 DECOC

Le *DECOC* est un dictionnaire comportant des entrées qui ne sont pas simples, ces items sont de deux natures : mots affixés et mots composés. Une entrée constituée d'un élément autonome (i.e. un mot simple) et d'un ou plusieurs éléments dépendants non autonomes (i.e. affixe(s)) est un mot affixé ; une entrée comportant au moins deux éléments autonomes est définie comme un mot composé. Les entrées du *DECOC* sont engendrées par l'association du *DECOS* et du *DECO-AFX* (*DECO-PF* préfixes, *DECO-PN* pseudo-noms, *DECO-SF* suffixes). Dans ce travail nous ne traitons pas le *DECOC*.

### 2.1.4 DECO-RA et DECO-FlexAV

Les entrées de la version du *DECOF* que nous avons utilisées pour nos expériences ne sont pas des formes combinées. Autrement dit, notre *DECOF* est fourni sous forme du *DECOS* et du *DECO-POST*. Nous ne pouvons pas utiliser directement ces dictionnaires pour notre travail. Pour construire le dictionnaire des formes fléchies *DECOF* par automate, il faut combiner les entrées du *DECOF* à partir du *DECOS* et du *DECO-POST*.

Pour cette raison, nous avons élaboré un nouveau dictionnaire des radicaux *DECO-RA* du *DECOS*. On peut en effet obtenir plusieurs radicaux à partir d'une racine en considérant les règles de combinaisons. Le dictionnaire des radicaux contient toutes les formes possibles des racines du *DECOS*. Donc, la taille de *DECO-RA* est 2.25 fois plus grande que celle de *DECOS*. Nous avons aussi ajouté des codes d'informations de combinaison à chaque entrée de *DECO-RA*. De la même manière que le *DECO-RA*, nous avons engendré *DECO-FlexAV* à partir du *DECO-POSTA* et du *DECO-POSTV* pour les suffixes flexionnels.

## 2.2 Automates finis

Un automate est une représentation du comportement d'un système mécanique fini. Un automate est formé par l'ensemble des différents états possibles du système, reliés entre eux par des conditions : le système passe d'un état dans un autre quand une condition donnée est vérifiée.

Un automate fini  $A$  est défini par la donnée d'un quintuplet d'ensembles  $(\Sigma, Q, I, T, E)$  avec

- $\Sigma$  un alphabet fini,
- $Q$  l'ensemble fini des états,
- $I \subseteq Q$  l'ensemble fini des états initiaux,
- $T \subseteq Q$  l'ensemble fini des états terminaux,
- $E \subseteq (Q \times \Sigma \times Q)$  l'ensemble fini des transitions.

On décrit des transitions de l'ensemble fini  $E$  par des triplets  $(p,a,q)$ , où  $p$  et  $q$  sont des états et  $a$  un symbole de l'alphabet  $\Sigma$ . Pour chaque état  $q$  et chaque caractère d'entrée  $a$ , il existe au plus une transition issue de l'état  $q$  et dont l'étiquette contient  $a$ . On peut noter une transition par  $\delta(p,a) = q$ . Un tel automate est dit déterministe. On parle d'automate acyclique quand il n'existe pas de suite de transitions dans l'automate qui passe deux fois par le même état.

Un mot est reconnu par un automate si et seulement si il est l'étiquette d'un chemin qui part de l'état initial et arrive à un état terminal. On lit le mot caractère après caractère, tout en se déplaçant dans l'automate en suivant les transitions étiquetées par le caractère courant. Si après avoir lu tout le mot on se trouve dans un état terminal de l'automate, le mot est reconnu ; sinon le mot n'est pas reconnu. On définit ainsi le langage reconnu par un automate, noté  $L(A)$ , comme l'ensemble des mots reconnus.

On peut représenter un automate par un graphe orienté dans lequel les nœuds représentent les états. Les arcs du graphe sont étiquetés par des ensembles de caractères.

*Exemple 2.1* On représente l'automate fini déterministe acyclique qui reconnaît le langage  $L(A) = \{abc, acb, baa, bac, bcb, c\}$  par

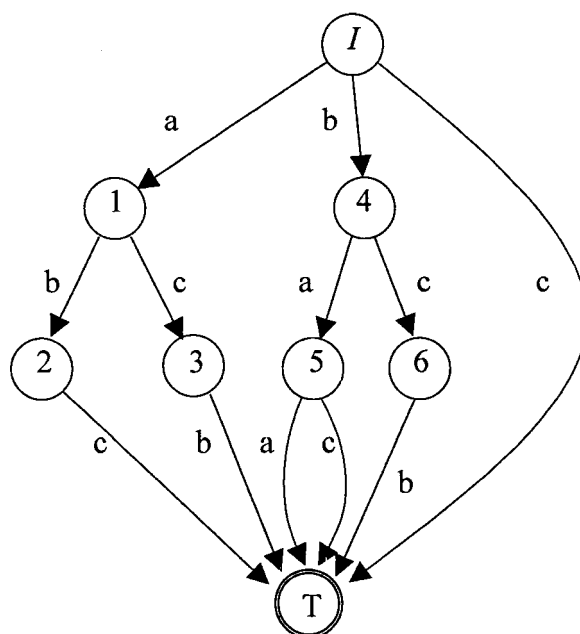


Fig. 2.5 Automate fini déterministe acyclique.

Les automates acycliques sont une structure de données bien adaptée à la représentation des lexiques de la langue naturelle. Dans ce travail, nous utilisons l'automate fini acyclique pour représenter des dictionnaires électroniques coréens.

## Première partie

### Le système *HANCoM* et les statistiques

# Chapitre 1.

## Le système *HANCoM*

### 1.1 Système de gestion du code *Hangul* (*HANCoM*)

Nous avons réalisé un nouveau système pour traiter le code du *Hangul* au niveau phonématique, les informations phonématiques sont essentielles pour analyser des formes fléchies en coréen. Le système *HANCoM*<sup>1</sup> permet : le *traitement phonématique* et la *conversion de codes*.

Le *traitement phonématique* du système *HANCoM* a deux fonctions principales : la *résolution* et la *recombinaison*. Nous montrons ce concept dans la (Figure 1.1).

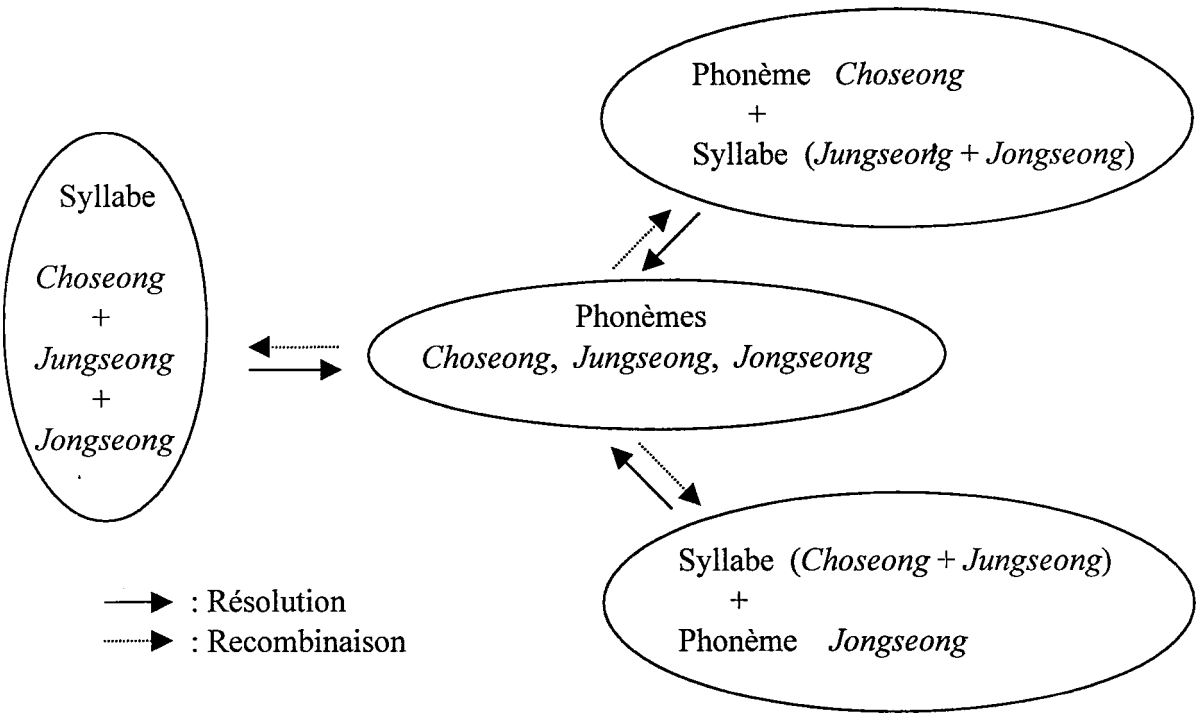


Fig. 1.1 Fonctions de traitement phonématique du système *HANCoM*

<sup>1</sup> *HANCoM* : Hangul Code Manager.

La première fonction de traitement phonématique est la résolution des syllabes. Autrement dit chaque syllabe est traduite en trois parties phonématiques (i.e. *Choseong*, *Jungseong* et *Jongseong*). L'autre fonction est la recombinaison des phonèmes en une syllabe dans le système de code '*KSC5601-1987 Hangul WANSUNG*'. On explique cette fonction sur des exemples dans la (Figure 1.2).

|                   |          |   |                                |   |           |   |           |
|-------------------|----------|---|--------------------------------|---|-----------|---|-----------|
| Exemple 1.1       | Syllabe  | ⇔ | Choseong                       | + | Jungseong | + | Jongseong |
|                   | ㅜㅓ       |   | ㄱ                              | + | ㅗ         | + | ㅁ         |
|                   | (gom)    |   | (g)                            |   | (o)       |   | (m)       |
| Le code WANSUNG : | (0xB0F5) |   | (0xA4A1)                       |   | (0xA4C7)  |   | (0xA4B1)  |
| Exemple 1.2       | Syllabe  | ⇔ | Syllabe (Choseong + Jungseong) | + | Jongseong |   |           |
|                   | ㅜㅓ       |   | ㅜㅓ                             | + | ㅁ         |   |           |
|                   | (gom)    |   | (go)                           |   | (m)       |   |           |
| Le code WANSUNG : | (0xB0F5) |   | (0xB0ED)                       |   | (0xA4B1)  |   |           |

Fig. 1.2 Exemples de traitement phonématique.

La conversion de codes est la fonction qui traduit un code d'un système dans un autre. Tous les processus de conversion sont basés sur le traitement au niveau phonématique. Nous le montrons dans la (Figure 1.3).

1.1.1 Les tableaux du système *HANCoM*

La conversion de codes doit être très rapide. Bien que beaucoup d'autres méthodes aient été considérées, nous avons finalement choisi une méthode basée sur les tableaux. La construction des tableaux exige beaucoup de temps, mais le temps de recherche est très réduit et les algorithmes sont très simples. Ce sont les avantages de cette méthode. Les éléments principaux qui ne changent jamais entre les systèmes de code du *Hangul* sont les consonnes et les voyelles du système d'alphabet coréen. Nous avons donc utilisé ces éléments du *Hangul Jamo* comme indices des tableaux.

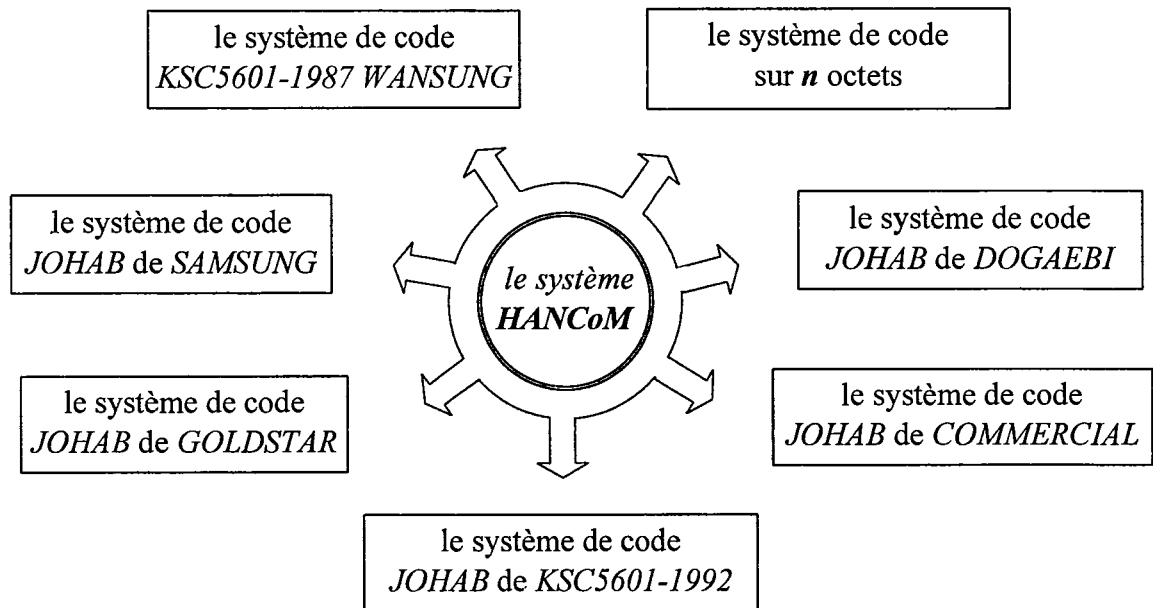


Fig. 1.3 La conversion de code du *HANCoM*

Nous avons créé des tableaux en utilisant les caractères du *Hangul Jamo* que nous avons mentionnés dans (la section 1.1 du chapitre 1. de l'Introduction). Les tableaux utilisés dans la fonction de *traitement phonématique* sont les suivants. Nous montrons ces tableaux dans l'Annexe 2 et l'Annexe 4.

- CIP : La table des codes des informations phonématiques
- ISP2 : Le tableau des indices des syllabes de profondeur 2 de l'arbre HAC<sup>2</sup>.
- ICC : La table des indices des consonnes de *Choseong*

Pour la fonction de *conversion des codes*, nous avons utilisé une paire de tableaux correspondant à une paire de systèmes de codes. Par exemple, si on fait une conversion entre le système de code 'KSC5601-1987 *Hangul WANSUNG*' et le système de code JOHAB de *COMERCIALE*, la paire de tableaux est la suivante. Nous montrons ces tableaux dans l'Annexe 3.

- WAN-JCOM : Le tableau de cartographie pour convertir les codes du KSC5601-1987 *Hangul WANSUNG* en codes du JOHAB de *COMMERCIAL*
- JCOM-WAN : Le tableau de cartographie pour convertir les codes du JOHAB *COMMERCIAL* en codes du KSC5601-1987 *Hangul WANSUNG*

<sup>2</sup> L'arbre de l'hierarchie alphabétique coréenne de système de code 'KSC5601-1987 *Hangul WANSUNG*'.

## 1.1.2 Etablissement de la table CIP

La table CIP est une liste simple de 2350 éléments représentée comme une matrice-ligne d'ordre 2350. Cette table est indicée (de 0 à 2349) par les syllabes coréennes. Les informations phonématiques concernant chaque syllabe y sont décrites. Nous employons la table CIP pour la résolution des syllabes et la recombinaison des phonèmes.

Nous représentons les informations phonématiques sur chaque syllabe par des nombres significatifs. Les trois parties phonématiques mentionnées ci-dessus sont remplacées par des nombres obtenus par un calcul simple. Nous obtenons les nombres de 1 à 30 pour les consonnes en soustrayant  $0xA0$  (Décimal 160) de l'octet inférieur des codes pour consonnes. Nous obtenons également les nombres de 1 à 21 pour les voyelles par la même méthode en soustrayant  $0xBE$  (Décimal 190).

(1) Dans le cas de consonne

$$c\_indice = (c\_code \& 0x00FF) - 0xA0 \quad (1.1)$$

$$c\_code = (0xA400) + (c\_indice + 0xA0) \quad (1.2)$$

(1) Dans le cas de voyelle

$$v\_indice = (c\_code \& 0x00FF) - 0xBE \quad (1.3)$$

$$v\_code = (0xA400) + (v\_indice + 0xBE) \quad (1.4)$$

Ces nombres que nous avons calculés ont les mêmes valeurs que les indices de la table du code du *Hangul Jamo* dans le système de code '*KSC5601-1987 Hangul WANSUNG*' (Figure 1.4). Nous illustrons ce fait par les exemples ci-dessous.

*Exemples 1.3* : Les calculs pour obtenir les nombres.

### Les consonnes

‘ㄱ’ (g) :  $0xA4-0xA1$  ( Décimal 164-161 )

↖ l'octet inférieur

$$0xA1 \text{ (déc. 161)} - 0xA0 \text{ (déc. 160)} = \underline{0x1} \text{ ( Décimal 1 )}$$

‘ㅇ’ (h) :  $0xA4-0xBE$  ( Décimal 164-190 )

↖ l'octet inférieur

$$0xBE \text{ (déc. 190)} - 0xA0 \text{ (déc. 160)} = \underline{0x1E} \text{ ( Décimal 30 )}$$

| Consonnes |             |       |         | Voyelles |              |       |         |
|-----------|-------------|-------|---------|----------|--------------|-------|---------|
|           | C           | Hex.  | Déc.    |          | V            | Hex.  | Déc.    |
| 1         | ㄱ <i>g</i>  | A4-A1 | 164-161 | 1        | ㅏ <i>a</i>   | A4-BF | 164-191 |
| 2         | ㄲ <i>gg</i> | A4-A2 | 164-162 | 2        | ㅑ <i>ai</i>  | A4-C0 | 164-192 |
| 3         | ㄴ <i>gs</i> | A4-A3 | 164-163 | 3        | ㅓ <i>ya</i>  | A4-C1 | 164-193 |
| 4         | ㄷ <i>n</i>  | A4-A4 | 164-164 | 4        | ㅕ <i>yai</i> | A4-C2 | 164-194 |
| 5         | ㄸ <i>nj</i> | A4-A5 | 164-165 | 5        | ㅗ <i>e</i>   | A4-C3 | 164-195 |
| 6         | ㄹ <i>nh</i> | A4-A6 | 164-166 | 6        | ㅛ <i>ei</i>  | A4-C4 | 164-196 |
| 7         | ㅁ <i>d</i>  | A4-A7 | 164-167 | 7        | ㅜ <i>ye</i>  | A4-C5 | 164-197 |
| 8         | ㅂ <i>dd</i> | A4-A8 | 164-168 | 8        | ㅠ <i>yey</i> | A4-C6 | 164-198 |
| 9         | ㅅ <i>l</i>  | A4-A9 | 164-169 | 9        | ㅡ <i>o</i>   | A4-C7 | 164-199 |
| 10        | ㅇ <i>lg</i> | A4-AA | 164-170 | 10       | ㅗ <i>wa</i>  | A4-C8 | 164-200 |
| 11        | ㅈ <i>lm</i> | A4-AB | 164-171 | 11       | ㅛ <i>wai</i> | A4-C9 | 164-201 |
| 12        | ㅊ <i>lb</i> | A4-AC | 164-172 | 12       | ㅜ <i>oi</i>  | A4-CA | 164-202 |
| 13        | ㅋ <i>ls</i> | A4-AD | 164-173 | 13       | ㅠ <i>yo</i>  | A4-CB | 164-203 |
| 14        | ㆁ <i>lt</i> | A4-AE | 164-174 | 14       | ㅓ <i>u</i>   | A4-CC | 164-204 |
| 15        | ㄷ <i>lp</i> | A4-AF | 164-175 | 15       | ㅕ <i>we</i>  | A4-CD | 164-205 |
| 16        | ㄹ <i>lh</i> | A4-B0 | 164-176 | 16       | ㅛ <i>wei</i> | A4-CE | 164-206 |
| 17        | ㅁ <i>m</i>  | A4-B1 | 164-177 | 17       | ㅕ <i>wi</i>  | A4-CF | 164-207 |
| 18        | ㅂ <i>b</i>  | A4-B2 | 164-178 | 18       | ㅠ <i>yu</i>  | A4-D0 | 164-208 |
| 19        | ㅅ <i>bb</i> | A4-B3 | 164-179 | 19       | ㅡ <i>eu</i>  | A4-D1 | 164-209 |
| 20        | ㅇ <i>bs</i> | A4-B4 | 164-180 | 20       | ㅓ <i>eui</i> | A4-D2 | 164-210 |
| 21        | ㄷ <i>s</i>  | A4-B5 | 164-181 | 21       | ㅣ <i>i</i>   | A4-D3 | 164-211 |
| 22        | ㄸ <i>ss</i> | A4-B6 | 164-182 |          |              |       |         |
| 23        | ㅇ <i>ng</i> | A4-B7 | 164-183 |          |              |       |         |
| 24        | ㅈ <i>j</i>  | A4-B8 | 164-184 |          |              |       |         |
| 25        | ㅊ <i>jj</i> | A4-B9 | 164-185 |          |              |       |         |
| 26        | ㅊ <i>ch</i> | A4-BA | 164-186 |          |              |       |         |
| 27        | ㅋ <i>k</i>  | A4-BB | 164-187 |          |              |       |         |
| 28        | ㅌ <i>t</i>  | A4-BC | 164-188 |          |              |       |         |
| 29        | ㅍ <i>p</i>  | A4-BD | 164-189 |          |              |       |         |
| 30        | ㅎ <i>h</i>  | A4-BE | 164-190 |          |              |       |         |

Fig. 1.4 Codes du *Hangul Jamo* dans le système de code *KSC5601-1987 Hangul WANSUNG*

Exemples 1.3 (cont.):

Les voyelles

‘ㅏ’ (a) : 0xA4-0xBF (Décimal 164-191)  
          ↖ l’octet inférieur  
          0xA1(déc. 191) – 0xBE (déc. 190) = 0x1 ( Décimal 1 )

‘ㅣ’ (i) : 0xA4-0xD3 (Décimal 164-211)  
          ↖ l’octet inférieur  
          0xA2 (déc. 211) – 0xBE (déc. 190) = 0x15 ( Décimal 21 )

Ces nombres composent les codes de la table CIP. Ils sont aussi employés pour l’indexation de la table ISP2.

Les éléments du CIP sont alors codés comme (Figure 1.5). Une syllabe coréenne se compose de deux ou de trois phonèmes. Dans le cas d’une syllabe de deux phonèmes, le *Jongseong* est absent. Une syllabe est résolue par des phonèmes, ensuite les nombres obtenus par la méthode ci-dessus sont associés aux phonèmes correspondants. On obtient finalement un code sur 2 octets en combinant ces trois nombres.

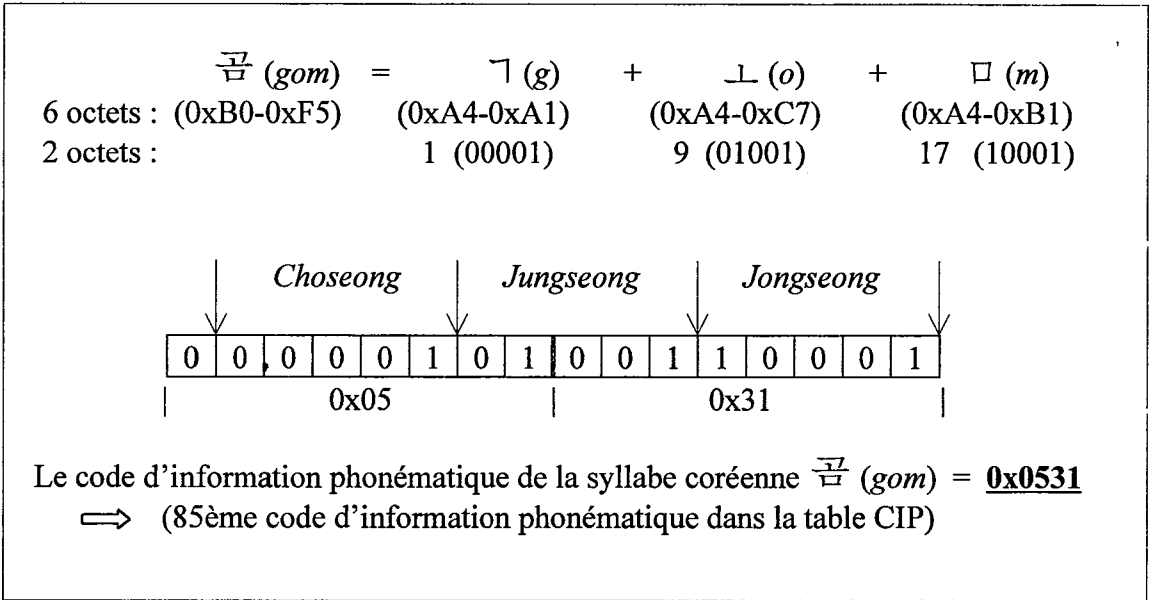


Fig. 1.5 Un exemple de calcul d’un code du CIP pour une syllabe coréenne

Maintenant, on ordonne les codes d'informations phonématiques que l'on a ainsi obtenus. Ces codes sont classés selon l'ordre lexicographique des syllabes correspondantes. Pour avoir un accès direct à un code donnant la valeur d'une syllabe, nous avons attaché aux codes des indices numériques de 0 à 2349. En construisant le système de code *KSC5601-1987 Hangul WANSUNG*, les 2350 syllabes sont séquentiellement arrangées en 25 ensembles de 94 syllabes. Avec cette structure, nous pouvons facilement obtenir les formules de traduction dans la (Figure 1.6) entre l'indice et le code de syllabe.

(1) Dans le cas d'une syllabe coréenne :

$$Indice = (((code \gg 8) - 0xB0) \times 94) + ((code \& 0x00FF) - 0xA1) \quad (1.5)$$

$$code = (((indice / 94) + 0xB0) < < 8) + ((indice \% 94) + 0xA1) \quad (1.6)$$

(2) Dans le cas d'un caractère chinois :

$$Indice = (((code \gg 8) - 0xCA) \times 94) + ((code \& 0x00FF) - 0xA1) \quad (1.7)$$

$$code = (((indice / 94) + 0xCA) < < 8) + ((indice \% 94) + 0xA1) \quad (1.8)$$

Fig. 1.6 Conversion entre un code de syllabe et un indice.

### 1.1.3 Indexation du tableau CIP

Pour minimiser le temps de recherche dans la table CIP, nous employons le tableau ISP2 pour la recombinaison des phonèmes. Les indices des colonnes désignent les codes des voyelles de 1 à 21. Les indices de ligne représentent les 19 codes des consonnes *Choseong*. Ces 19 nombres ne sont pas consécutifs, parce que certaines consonnes ne sont pas employées pour représenter le *Choseong*. Pour cette raison, nous avons établi la table ICC afin de réduire la taille du tableau ISP2. Les indices du tableau ISP2 sont un *Choseong* et un *Jungseong*. Ces deux indices déterminent donc un élément du tableau ISP2, et la valeur de cet élément indique la position d'un code de syllabe dans la table CIP. Cette syllabe est composée par le *Choseong* et le *Jungseong* ci-dessus. Plus précisément, l'élément du tableau ISP2 indice la première syllabe dans un sous-ensemble de syllabes qui ont le même père dans l'arbre HAC (voir la Figure 1.8 de la section 1.2.4 du chapitre 1 de l'Introduction). Le cardinal maximal d'un sous-ensemble est 18. On peut alors trouver la syllabe par un maximum de 17 comparaisons. Nous montrons l'indexation entre les tableaux dans la (Figure 1.7). Il y a cinq

syllabes particulières<sup>3</sup>. Ces syllabes sont des syllabes de profondeur 3 (*Choseong* + *Jungseong* + *Jongseong*) dans l'arbre HAC. Nous représentons ces codes dans le tableau ISP2 en additionnant 8000 à leurs codes pour les distinguer des autres.

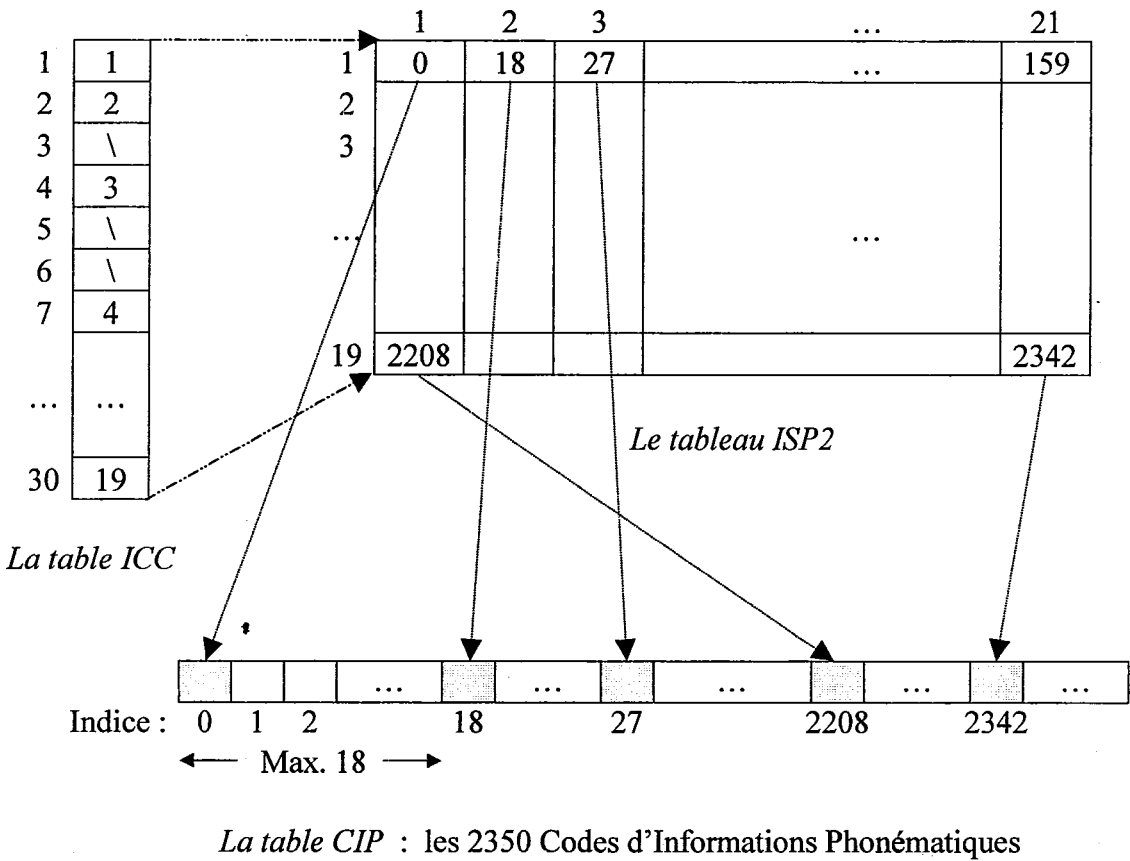


Fig. 1.7 L'indexation entre les tableaux : CIP, ISP2 et ICC

Nous montrons un exemple de recombinaison des phonèmes en une syllabe avec des tableaux. On considère les phonèmes de l'exemple de la (Figure 1.2).

*Choseong* : ㄱ (g) (0xA4-0xA1)  
*Jungseong* : ㅓ (o) (0xA4-0xC7)  
*Jongseong* : ㅁ (m) (0xA4-0xB1)

<sup>3</sup> ㄹ (lwaiss) (0xb7d9, CIP:174), ㅕ (ssyang) (0xbdd6, CIP:1285), ㅖ (ssyein)(0xbded, CIP:1298), ㅙ (ssyung) (0xb7d9, CIP:174), ㅞ (ssyein)(0xbded, CIP:1298), ㅟ (ssyung) (0xb7d9, CIP:174), ㅠ (ssyein)(0xbded, CIP:1298)

On peut obtenir les indices des trois phonèmes utilisant les formules (1.1) et (1.3). Les numéros du *Choseong*, du *Jungseong*, du *Jongseong* sont 1, 9 et 17. Avec deux indices 1 et 9, on trouve l'indice 76 du tableau ISP2. Cet indice 76 indique la première syllabe dans le sous-ensemble des syllabes qui ont le même *Choseong* et le *Jungseong*, c'est-à-dire, ayant le même père dans l'arbre HAC. Maintenant on cherche un code du CIP dont le nombre du *Jongseong* est 17. On trouve le 9<sup>ème</sup> code du CIP. On obtient finalement l'indice 84 pour la syllabe résultant de la combinaison des trois phonèmes ci-dessus. On obtient le code de la syllabe 0xB0-0xF5 par la formule (1.6) de la (Figure 1.6).

## 1.2 Algorithmes

### 1.2.1 Résolution

Nous pouvons obtenir la valeur d'indice à partir du code *Wansung* par la formule (1.5). Nous pouvons alors accéder directement à la table CIP avec l'indice obtenu ci-dessus. Nous obtenons un code d'information phonématique du CIP. Les trois codes de cinq bits sont convertis en *Choseong* (2 octets), *Jungseong* (2 octets) et *Jongseong* (2 octets) en ajoutant 0xA0, 0xBE et 0xA0. Nous montrons cet algorithme de résolution des syllabes dans la (Figure 1.8).

```

w_code      ; Code de syllabe du KSC5601-1987 Hangul WANSUNG
w_indice     ; Indice du w_code
w_cip        ; Code du CIP

begin
  w_indice ← ((w_code >> 8) - 0xB0) × 94 + ((w_code & 0x00FF) - 0xA1);
  w_cip ← la table CIP[w_indice];

  Choseong_indice ← w_cip & 001F ;
  Jungseong_indice ← (w_cip >> 5) & 001F ;
  Jongseong_indice ← (w_cip >> 10) & 001F ;

  Choseong_code ← (0xA400) + ( Choseong_indice + 0xA0) ;
  Jungseong_code ← (0xA400) + (Jungseong_indice + 0xBE) ;
  Jongseong_code ← (0xA400) + ( Jongseong_indice + 0xA0) ;

end

```

Fig. 1.8 L'algorithme de résolution des syllabes

## 1.2.2 Recombinaison

Nous montrons l'algorithme de recombinaison dans la (Figure 1.9). D'abord, nous pouvons convertir chaque code de phonème à 2 octets en indice phonématique en soustrayant 0xA0 pour *Choseong* et *Jongseong*, 0xBE pour *Jungseong* à l'octet inférieur du code de phonème. En cherchant dans le tableau ISP2 à l'aide de l'indice de *Choseong* et de l'indice de *Jungseong*, nous obtenons l'indice de la syllabe pour accéder à la table CIP. Si l'indice de *Jongseong* n'existe pas, la valeur d'indice que nous avons obtenue est l'indice du code que nous cherchions. Si l'indice de *Jongseong* existe, nous devons trouver la valeur d'indice de la syllabe dans la table GIP. Après, nous cherchons la syllabe en comparant avec *Jongseong*. Le nombre maximal de comparaisons n'excède pas 17 pour la recherche de la syllabe.

```

w_code    ; Code de syllabe de WANSUNG
w_indice   ; Indice de w_code

begin
  read (Choseong_code, Jungseong_code, Jongseong_code);
  Choseong_indice ← (Choseong_code & 0x00FF) - 0xA0;
  Jungseong_indice ← (Jungseong_code & 0x00FF) - 0xBE;
  Jongseong_indice ← (Jongseong_code & 0x00FF) - 0xA0;
  w_indice ← le tableau ISP2[Choseong_indice, Jungseong_indice];

  if ( Jongseong_indice n'existe pas ) then
    begin
      w_code ← (((w_indice / 94)+0xB0) << 8)+((w_indice % 94)+0xA1);
      write w_code;
    end;
  for i ← 1 step 1 until 18 do
    begin
      if ( la table CIP(w_indice + i) & 0x001F = Jongseong_indice ) then
        begin
          w_indice ← la table CIP(w_indice + i);
          w_code ← (((w_indice / 94)+0xB0) << 8)+((w_indice % 94)+0xA1);
          write w_code;
          goto END_P;
        end
      end
    end
  end
end

```

Fig. 1.9 Algorithme de recombinaison des phonèmes en une syllabe.

### 1.3 Applications prévues

Le système de code '*KSC5601-1987 Hangul WANSUNG*' ne contient pas d'informations phonématiques, bien qu'étant le système de code standard du coréen. L'essentiel des textes et des documents est écrit avec ce système de codage. Dans le domaine du traitement du langage naturel en coréen, les informations phonématiques sont très importantes. C'est pourquoi nous avons construit le système de gestion du code *Hangul* au niveau des phonèmes. Le système *HANCoM* offre de la flexibilité dans le traitement de textes en coréen. Par exemple, il peut être employé pour construire un dictionnaire électronique en coréen. Cette méthode permet d'utiliser les formes mélangeant représentations syllabiques et représentations phonématiques pour construire un automate du dictionnaire électronique. On peut ainsi utiliser ce système pour analyser les formes fléchies coréennes. Pour ce faire, il faut en effet découper un mot en un radical et un suffixe flexionnel. Dans cette procédure, les informations phonématiques sont essentielles.

## Chapitre 2

# Les statistiques du coréen et la loi de Zipf

Ce chapitre est consacré à l'étude de quelques caractéristiques statistiques de la langue coréenne. Les informations statistiques sur la fréquence des lettres, la fréquence des mots, la longueur des mots et l'entropie sont utiles tant pour analyser des textes que pour le traitement des langues naturelles. L'entropie peut être utilisée dans la mise en application des systèmes de compression de textes ou dans d'autres problèmes de décision lors de l'utilisation de méthodes de traitement de textes qui doivent tenir compte de l'incertitude. L'examen des régularités de la langue naturelle est également utile pour les autres traitements, par exemple pour estimer des conditions de stockage des listes de mots et des dictionnaires. Nous avons présenté l'alphabet coréen et la notion de syllabe dans le système d'écriture du coréen dans (la section 1.1 du chapitre 1. de l'Introduction). Les définitions d'alphabet, de lettre, de syllabe et de mot sont aussi données dans la section indiquée ci-dessus. Le mot est défini comme unité de base des phrases. Nous montrons maintenant les résultats statistiques de nos expérimentations sur le corpus de textes coréens. A l'aide des résultats statistiques, la loi de Zipf est étudiée dans le cas de textes coréens. De plus, un certain nombre de données probabilistes ont été rassemblées.

## 2.1 Etablissement du corpus d'essai

L'utilisation d'un corpus volumineux est indispensable pour un travail statistique. Il est très difficile de construire un corpus équilibré. Pour l'anglais, une collection de textes anglais américain connue sous le nom de '*Brown corpus*' a été largement répandue pour l'étude des statistiques de la langue. Ses 500 échantillons de 2000 mots chacun constituent un échantillon total de plus de 1 million de mots de textes de la langue naturelle représentant un éventail de modèles et d'auteurs. Malheureusement nous n'avons pas de collection de textes coréens comme le '*Brown corpus*'. L'étude de la langue naturelle en coréen est trop récente.

Il existe deux corpus de textes coréens qui sont publiés par KAIST<sup>1</sup> et ETRI<sup>2</sup>. Nous avons choisi le corpus du KAIST pour notre expérimentation. Il se compose de plus de mille

---

<sup>1</sup> KAIST : Korea Advanced Institute of Science and Technology

fichiers de textes. Selon [Han96], le corpus du KAIST couvre divers genres de textes écrits : journaux, textes scolaires, textes littéraires coréens et livres généraux, etc. Le corpus du KAIST que nous avons utilisé est une version sous forme de fichier de textes. Il contient des erreurs typographiques, des caractères chinois, des parenthèses et des symboles. Ceux-ci peuvent seulement avoir une influence négligeable sur notre résultat. Nous avons exclu quelques fichiers du corpus, parce qu'ils contenaient trop d'erreurs. Nous avons finalement choisi 1043 fichiers de textes pour composer notre corpus d'essai. Ce corpus contient 8768500 mots coréens et d'autres mots comme des mots anglais, des symboles, etc. Le nombre de syllabes est de 38385924 dans ce corpus. La taille du corpus est d'environ 44 Mo. Ce corpus d'essai a été codé dans le système de code '*KSC-5601 Hangul WANGUNG*', qui permet de coder les 2350 syllabes de *Hangul* et les 4880 caractères chinois qui sont utilisés dans les textes coréens.

Nous définissons 5 jeux de caractères pour notre analyse des textes. Ce sont respectivement les 2350 syllabes de *Hangul*, les 4880 caractères chinois, les 10 chiffres (0-9), les 52 lettres de l'alphabet latin dans le code ASCII (a-z, A-Z), le caractère d'espace et les 42 autres symboles imprimables dans le code ASCII.

## 2.2 Expérimentations

Nous avons réalisé des expérimentations sur la machine *HP9000/735* sous UNIX avec un terminal-X de *Tektronix*. Nous avons employé le logiciel graphique *Xgraphic* (version 3.93) pour tracer les graphes rang-fréquence, rang-probabilité et autres. Nous avons développé un programme de comptage en *langage C* sous UNIX, car il n'existe aucun outil utilisable pour compter les syllabes et les mots coréens dans les corpus. Ce programme de comptage permet à l'utilisateur d'obtenir des nombres d'occurrences en fonction des 5 jeux de caractères donnés.

### 2.2.1 Entropie

Alors que depuis longtemps l'idée d'une mesure quantitative de l'information était un sujet d'étude, la personne qui l'a mise en lumière sous le nom actuel de *théorie de l'information* est *Claude E. Shannon* [Sha48]. Il donne une définition mathématique de l'information. Une des idées fondamentales de la théorie de l'information est celle d'entropie, qui est une mesure de la quantité d'ordre ou de redondances dans un message. L'entropie est un nombre qui est petit quand il y a peu de désordre et grand quand il y a beaucoup de désordre. On le lie intimement à la compression de données car la longueur d'un message compressé devrait idéalement être égale à son entropie.

---

<sup>2</sup> ETRI : Electronic and Telecommunication Research Institute

Shannon a défini l'entropie de la manière suivante. On considère un ensemble d'événements possibles dont les probabilités d'occurrence sont  $p_1, p_2, \dots, p_n$  avec  $\sum_{i=1}^n p_i = 1$ . Ces probabilités sont connues mais c'est tout que nous savons. Pouvons-nous trouver une mesure du nombre de choix possibles ou de l'incertitude de cette situation? S'il existe une telle mesure, l'entropie  $H(p_1, p_2, \dots, p_n)$ , elle doit répondre aux exigences suivantes :

1.  $H$  doit être continu en les  $p_i$ .
2. Si tout les  $p_i$  sont égaux, alors  $H$  doit être une fonction croissante de  $n$ . Avec des événements équiprobables il y a plus de choix, ou d'incertitude, quand le nombre d'événements possibles est plus grand.
3. Si un choix est décomposé en deux choix successifs, le  $H$  original doit être la somme pondérée des différentes valeurs de  $H$ . La signification de ceci est illustrée sur la (Figure 2.1). A gauche, nous avons trois possibilités  $p_1 = \frac{1}{2}$ ,  $p_2 = \frac{1}{3}$ ,  $p_3 = \frac{1}{6}$ . Du côté droit, nous choisissons d'abord entre deux possibilités chacune avec la probabilité  $\frac{1}{2}$ , et si la seconde se produit nous faisons un autre choix avec les probabilités  $\frac{2}{3}$ ,  $\frac{1}{3}$ . Les résultats finals ont les mêmes probabilités que précédemment.  
Nous exigeons, dans ce cas

$$H\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right) = H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{2} H\left(\frac{2}{3}, \frac{1}{3}\right) \quad (2.1)$$

Le coefficient  $\frac{1}{2}$  apparaît car le deuxième choix ne se produit que la moitié du temps.

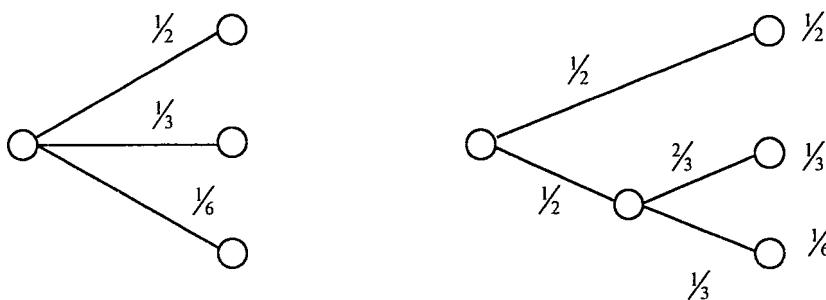


Fig. 2.1 Décomposition d'un choix entre trois possibilités.

Comme Shannon l'a démontré, la seule fonction qui répond à ces exigences est

$$H(p_1, p_2, \dots, p_n) = -k \sum_{i=1}^n p_i \log p_i, \quad (2.2)$$

où  $k$  est une constante positive qui régit les unités dans lesquelles l'entropie est mesurée. Normalement, les unités sont des 'bits', où  $k = 1$  et le logarithme utilisé est à base 2.

$$H = - \sum_{i=1}^n p_i \log_2 p_i \text{ bits.} \quad (2.3)$$

Le signe négatif reflète simplement le désir que l'entropie soit une quantité positive, tandis qu'étant inférieures à 1, les probabilités ont toujours des logarithmes négatifs.

Shannon définit la notion d'information inhérente (*Self information*)  $I(a_i)$  de la manière suivante. On considère un événement  $a_i$ , c'est-à-dire le résultat d'un essai aléatoire, si  $P(a_i)$  est la probabilité d'apparition de l'événement  $a_i$ , alors l'information inhérente associée à  $a_i$  est donnée par :

$$I(a_i) = \log_2 \frac{1}{P(a_i)} = -\log_2 P(a_i) \quad (2.4)$$

Par exemple, l'alphabet de l'anglais a 26 (a – z) lettres, si toutes les lettres sont équiprobables, son information inhérente est

$$I = -\log_2 \frac{1}{26} = 4.7 \text{ bits.}$$

Dans le cas du coréen, le cardinal de l'alphabet est 2350 (syllabes), son information inhérente  $I$  est alors 11.198 bits.

Pour calculer l'entropie des jeux de caractères, nous employons des modèles de *contexte fini* [Bel90]. Les fréquences des  $n$ -gramme peuvent être employées pour construire des modèles d'ordre  $n-1$ , où les  $n-1$  premières syllabes d'un  $n$ -grammes sont employées pour prévoir la  $n^{\text{ième}}$  syllabe. Nous avons calculé l'entropie au niveau des syllabes pour les modèle d'ordre 0 (syllabe simple), d'ordre 1 (digramme) et d'ordre 2 (trigramme) avec des fréquences de syllabes. Nous avons aussi calculé l'entropie au niveau des mots pour les modèles d'ordre 0 (mot simple) et les modèles d'ordre 1 (de digramme) des distributions de mots. Pour le modèle général des  $n$ -grammes, si  $X$  est l'ensemble des  $(n-1)$ -grammes et  $A$  est l'ensemble des syllabes d'entrées, l'entropie d'un modèle d'ordre  $n-1$  est

$$- \sum_{x \in X} p(x) \sum_{a \in A} p(a | x) \log_2 p(a | x) \quad (2.5)$$

2.2.2 Fréquences des syllabes en coréen

Quels caractères ou syllabes sont les plus fréquents dans un texte coréen? Cette question semble très intéressante. Bien que nous parlions, écrivions et lisions chaque jour, nous ne pensons pas à cela et ne le savons pas non plus. Le caractère le plus fréquent dans un texte coréen normal est l'espace. Les statistiques sur le nombre d'occurrences sont utiles pour le traitement de la langue naturelle. Nous avons défini 6 jeux de caractères: les 5 jeux de caractères définis dans la (section 2.1 du chapitre 2 de la première partie) et le caractère *d'espace*. Nous obtenons le nombre d'occurrences de chaque jeu de caractère et de chaque caractère dans les jeux. Quel est le caractère excepté *l'espace* qui apparaît le plus fréquent au niveau de la syllabe? D'après notre expérience le 'ㅇ'(i) est le plus fréquent dans les textes coréens. Il totalise 2.59 % de toutes les occurrences de caractères au niveau de la syllabe. Le caractère le plus fréquent, *l'espace*, occupe 25.19 % de toutes les occurrences. La longueur moyenne d'un mot coréen est d'environ 2.97 syllabes, ainsi nous nous attendons à ce qu'un espace apparaisse une fois toutes les 3.97 syllabes. Le nombre d'occurrences de chaque ensemble d'unités obtenues à partir de notre corpus d'essai est montré dans la (Figure 2.2). Les syllabes qui apparaissent seulement une fois dans le corpus sont au nombre de 705 et comptent pour 10.4 % du total. Les syllabes qui sont visibles moins de dix fois sont au nombre de 2819 et comptent pour 41.58 %. Au contraire, les 24 syllabes les plus fréquentes comptent pour 50 % des syllabes du corpus, mais seulement 0.37 % des 6779 syllabes différentes.

| Jeux de caractères                 | Nombre d'occurrences | Occupation (%) |
|------------------------------------|----------------------|----------------|
| syllabes coréennes                 | 26,250,624           | 68.39 %        |
| Caractères d'espace                | 9,668,071            | 25.19 %        |
| Symboles imprimables du code ASCII | 1,529,951            | 3.99 %         |
| Chiffres                           | 297,318              | 0.77 %         |
| Lettres de l'alphabet latin        | 354,998              | 0.92 %         |
| Caractères chinois etc.            | 284,962              | 0.74 %         |
| Nombre total d'unités              | 38,385,924           | 100 %          |

Fig. 2.2 Le nombre d'occurrences des Jeux de caractères

Les 2679 syllabes les plus fréquentes comptent pour 99.9 % des syllabes du corpus et pour 39.52 % de l'ensemble des différentes syllabes. L'entropie au niveau des syllabes est de 6.884 bits pour des monogrammes de syllabes. L'entropie des digrammes est de 4.872 bits. L'entropie des trigrammes est de 3.754 bits. L'information inhérente à la syllabe est de 12.84 bits. La (Figure 2.3) montre quelques syllabes et statistiques des *n*-grammes de notre corpus.

| (· : space character)      |          |                  |           |                              |          |
|----------------------------|----------|------------------|-----------|------------------------------|----------|
| Syllable                   | Prob.(%) | Digram           | Prob. (%) | Trigram                      | Prob.(%) |
| · ·                        | 25.1865  | · ·              | 2.4113    | · · ·                        | 1.6779   |
| 이 i                        | 2.5889   | 는· <i>neun</i> · | 1.7601    | 다· <i>da</i> ·               | 1.3771   |
| 다 da                       | 2.1813   | · ·              | 1.6227    | · · ·                        | 0.4802   |
| 는 <i>neun</i>              | 1.9242   | 다· <i>da</i> ·   | 1.4882    | 으로· <i>eu lo</i> ·           | 0.3114   |
| · ·                        | 1.8763   | 의· <i>eui</i> ·  | 1.3733    | 이다· <i>i da</i> ·            | 0.2476   |
| 의 <i>eui</i>               | 1.6299   | 을· <i>eul</i> ·  | 1.2791    | ·그 · <i>geu</i>              | 0.2461   |
| 에 <i>ei</i>                | 1.4401   | 이· <i>i</i> ·    | 1.1092    | ·것이 · <i>ges i</i>           | 0.2449   |
| 을 <i>eul</i>               | 1.3134   | 은· <i>eun</i> ·  | 0.8825    | 에서· <i>ei se</i> ·           | 0.2437   |
| 하 ha                       | 1.2472   | ·그 · <i>geu</i>  | 0.8796    | 었다· <i>ess da</i> ·          | 0.2381   |
| 가 ga                       | 1.1793   | 고· <i>go</i> ·   | 0.8624    | 는·것 <i>neun</i> · <i>ges</i> | 0.2340   |
| 고 go                       | 1.1764   | 에· <i>ei</i> ·   | 0.8420    | 하는· <i>ha neun</i> ·         | 0.2304   |
| 지 ji                       | 1.0171   | , ·              | 0.7488    | 고·있 <i>go</i> · <i>iss</i>   | 0.2145   |
| 그 <i>geu</i>               | 0.9393   | 로· <i>lo</i> ·   | 0.6614    | ·그· · <i>geu</i> ·           | 0.1941   |
| 한 <i>han</i>               | 0.9335   | ·이 · <i>i</i>    | 0.6521    | 하고· <i>ha go</i> ·           | 0.1753   |
| 은 <i>eun</i>               | 0.9263   | 한· <i>han</i> ·  | 0.6383    | ·있다 · <i>iss da</i>          | 0.1718   |
| 로 lo                       | 0.9161   | 를· <i>leul</i> · | 0.6338    | ·수· · <i>su</i> ·            | 0.1701   |
| 서 se                       | 0.8486   | ·있 · <i>iss</i>  | 0.6308    | ·있는 · <i>iss neun</i>        | 0.1571   |
| , ,                        | 0.7699   | 가· <i>ga</i> ·   | 0.6147    | 지· 않 <i>ji</i> · <i>anh</i>  | 0.1473   |
| 기 gi                       | 0.7554   | 서· <i>se</i> ·   | 0.5352    | 니다· <i>ni da</i> ·           | 0.1460   |
| 어 e                        | 0.7484   | ·것 · <i>ges</i>  | 0.5225    | 있는· <i>iss neun</i> ·        | 0.1436   |
| 나 na                       | 0.6983   | 도· <i>do</i> ·   | 0.4381    | ·사람 · <i>sa lam</i>          | 0.1295   |
| 도 do                       | 0.6864   | ·사 · <i>sa</i>   | 0.4264    | 있다· <i>iss da</i> ·          | 0.1275   |
| 사 sa                       | 0.6719   | 게· <i>gei</i> ·  | 0.3992    | 것이다 <i>ges i da</i>          | 0.1244   |
| 있 iss                      | 0.6470   | 으로 <i>eu lo</i>  | 0.3878    | ·이· · <i>i</i> ·             | 0.1243   |
| 를 <i>leul</i>              | 0.6412   | 지· <i>ji</i> ·   | 0.3370    | 수·있 <i>su</i> · <i>iss</i>   | 0.1203   |
| 것 <i>ges</i>               | 0.6393   | 에서 <i>ei se</i>  | 0.3248    | 적인· <i>jeg in</i> ·          | 0.1179   |
| ...                        |          | ...              |           | ...                          |          |
| Number of units            | 38385924 |                  | 38385923  |                              | 38385922 |
| Entropy<br>(bits/syllable) | 6.884    |                  | 4.872     |                              | 3.754    |

Fig. 2.3 Les statistiques sur les syllabes du corpus du KAIST

2.2.3 Fréquences des mots en coréen

Jusqu'ici nous avons étudié les statistiques sur les syllabes. Une autre composante naturelle des textes est le mot. Pour compter les mots, nous avons défini au niveau de syllabe 'les mots comme des suites de caractères sans espace'. Les espaces multiples ont été considérés comme des espaces simples. D'après les résultats de nos expériences, les 8768500

occurrences de mots dans le corpus du KAIST de textes coréens contiennent 1344018 mots différents. Quelques mots, les plus fréquents, du corpus du KAIST sont montrés dans la (Figure 2.4).

| Words              |                  | Prob.(%) | Digram words |                          | Prob.(%) |
|--------------------|------------------|----------|--------------|--------------------------|----------|
| 그                  | <i>geu</i>       | 0.8546   | 수 있는         | <i>su issseun</i>        | 0.1313   |
| 수                  | <i>su</i>        | 0.7427   | 수 있다.        | <i>su issda.</i>         | 0.1223   |
| 있는                 | <i>issneun</i>   | 0.5950   | 할 수          | <i>hal su</i>            | 0.0775   |
| 이                  | <i>i</i>         | 0.5513   | 수 없는         | <i>su ebsseun</i>        | 0.0593   |
| 있다.                | <i>issda.</i>    | 0.5415   | 볼 수          | <i>bol su</i>            | 0.0392   |
| 것이다.               | <i>gesida.</i>   | 0.5087   | 알 수          | <i>al su</i>             | 0.0385   |
| 한                  | <i>han</i>       | 0.3371   | 수 없다.        | <i>su ebsda.</i>         | 0.0353   |
| 것은                 | <i>geseun</i>    | 0.2944   | 수 있을         | <i>su isseul</i>         | 0.0345   |
| 그러나                | <i>geulena</i>   | 0.2799   | 것이다. 그러나     | <i>gesida. geulena</i>   | 0.0238   |
| 것이                 | <i>gesi</i>      | 0.2740   | 될 것이다.       | <i>doil gesida.</i>      | 0.0206   |
| 있었다.               | <i>issessda.</i> | 0.2603   | 될 수          | <i>doil su</i>           | 0.0202   |
| 하는                 | <i>haneun</i>    | 0.2400   | 수 있었다.       | <i>su issessda.</i>      | 0.0201   |
| 대한                 | <i>daihan</i>    | 0.2360   | 할 것이다.       | <i>ha gesida.</i>        | 0.0198   |
| 그리고                | <i>geuligo</i>   | 0.2272   | 않을 수         | <i>anheul su</i>         | 0.0197   |
| 것을                 | <i>geseul</i>    | 0.2237   | 이와 같은        | <i>ioa gateun</i>        | 0.0196   |
| 할                  | <i>hal</i>       | 0.2196   | 수 있게         | <i>su issgei</i>         | 0.0187   |
| 그는                 | <i>geuneun</i>   | 0.2115   | 있다. 이        | <i>issda. i</i>          | 0.0186   |
| 같은                 | <i>gateun</i>    | 0.2082   | 여러 가지        | <i>yele gaji</i>         | 0.0183   |
| 한다.                | <i>handa.</i>    | 0.2087   | 수 없었다.       | <i>su ebyessda.</i>      | 0.0183   |
| 나는                 | <i>naneun</i>    | 0.2031   | 수 있다는        | <i>su issdaneun</i>      | 0.0182   |
| 다른                 | <i>galeun</i>    | 0.1767   | 있는 것은        | <i>issneun geseun</i>    | 0.0173   |
| 이러한                | <i>ilehan</i>    | 0.1627   | 있는 것이        | <i>issneun gesi</i>      | 0.0167   |
| 모든                 | <i>modeun</i>    | 0.1621   | 말할 수         | <i>malhal su</i>         | 0.0149   |
| 또                  | <i>ddo</i>       | 0.1583   | 있기 때문이다.     | <i>issgi ddaimonida.</i> | 0.0147   |
| 없는                 | <i>ebsneun</i>   | 0.1562   | 더 이상         | <i>de isang</i>          | 0.0146   |
| 어떤                 | <i>edden</i>     | 0.1537   | 수 있도록        | <i>su issdolog</i>       | 0.0145   |
| 더                  | <i>de</i>        | 0.1526   | 것이다. 이       | <i>gesida. i</i>         | 0.0144   |
| ...                | ...              | ...      | ...          | ...                      | ...      |
| Number of units    |                  | 1344018  |              |                          | 6726025  |
| Entropy(bits/word) |                  | 16.048   |              |                          | 6.2756   |

Fig. 2.4 Les statistiques sur les mots du corpus du KAIST

Les mots fonctionnels courts (그 *geu*, 수 *su*, 있는 *issneun*, 이 *i*, ...) sont fréquents. Les longs mots comme les noms propres sont rares. Notre analyse prouve que la longueur moyenne des mots du corpus du KAIST est de 3.27 lettres. Le mot le plus court se compose d'une seule syllabe. Le plus long mot du corpus du KAIST a vingt-trois syllabes et est un nom

propre. La plupart des 160 mots les plus fréquents sont des mots fonctionnels et représentent 18 % des mots du corpus.

Dans le cas de l'anglais [Bel90], la longueur moyenne d'un mot est de 4.9 caractères. Les 100 mots les plus fréquents représentent 42 % des mots du corpus, mais seulement 0.1% des 100237 différents mots. Nous avons constaté que les 2782 mots les plus fréquents représentent 42 % des mots du corpus coréen, et 0.2 % de ses 1344018 mots différents. La (Figure 2.5) montre l'occupation des mots en pourcentage cumulé selon leur rang. Les mots qui n'apparaissent qu'une seule fois dans le corpus représentent 69 % des différents mots utilisés mais seulement 10.58 % des occurrences de mots. Les mots n'apparaissant pas plus de 10 fois représentent 94.43 % des différents mots mais seulement 24.27 % des occurrences.

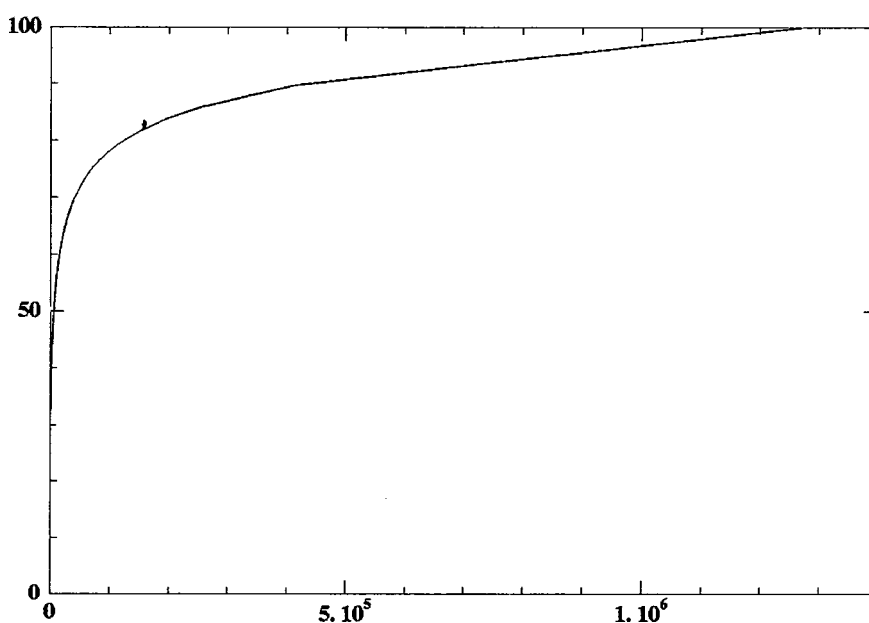


Fig. 2.5 L'occupation des mots en pourcentage cumulé

## 2.3. La loi de Zipf

### 2.3.1 La loi de Zipf appliquée à l'anglais

La loi de Zipf a été beaucoup étudiée dans le cas des textes anglais. Pour les détails, se reporter au chapitre 4 de [Bel90]. Nous pouvons ranger les mots selon l'ordre décroissant de leurs fréquences. Nous pouvons ainsi ordonner les mots, de sorte que le mot le plus fréquent soit le premier, le prochain mot fréquent est le deuxième et ainsi de suite. Si on représente le graphe de la relation entre le rang d'un mot et sa fréquence, on obtient une courbe approximativement hyperbolique. Maintenant on convertit ce graphe avec une échelle logarithmique, où cette forme hyperbolique apparaît comme une ligne approximativement droite. Cet effet est lié au fait que le produit du rang par le nombre d'occurrences du mot reste approximativement constant. *George Zipf*, un psycholinguiste, a observé cet effet, et l'a popularisé en 1949 dans son œuvre '*Human behavior and the principle of least effort* (Zipf, 1949)'. La loi de Zipf dit que le nombre d'occurrences  $f(r)$  du  $r^{\text{ième}}$  mot dans l'ordre des fréquences multiplié par le rang  $r$  est approximativement constant sur d'un texte donné, ou

$$r \cdot f(r) \cong k \quad (k : \text{constante}) \quad (2.6)$$

Nous pouvons exprimer cette relation de façon probabiliste. Soit  $p(r)$  la probabilité d'occurrence du  $r^{\text{ième}}$  mot dans l'ordre des fréquences, alors  $p(r)$  est égale à  $f(r)/N$  si  $N$  est le nombre total d'occurrences de tous les mot. La loi de Zipf sous forme probabiliste s'énonce alors :

$$p(r) = \frac{\mu}{r} \quad r = 1, 2, 3, \dots, n \quad (\mu : \text{constante de normalisation}) \quad (2.7)$$

Si  $n$  est le nombre de mots distincts, la constante de normalisation  $\mu$  peut être calculée comme suit:

$$\sum_{r=1}^n p(r) = \mu \sum_{r=1}^n \frac{1}{r} = 1 \quad (2.8)$$

La somme  $\sum_{r=1}^n \frac{1}{r}$  est connue comme étant le  $n^{\text{ième}}$  nombre harmonique  $H_n$  [Gra94].

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} = \sum_{k=1}^n \frac{1}{k} \quad \text{nombre entier } n \geq 0 \quad (2.9)$$

Sa valeur est:  $H_n = \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{\varepsilon_n}{120n^4} \quad (0 < \varepsilon_n < 1) \quad (2.10)$

Si nous supposons que  $n$  est suffisamment grand, nous avons :

$$H_n \cong \ln n + \gamma \quad \gamma = 0.5772156649... \text{ (constante d'Euler)} \quad (2.11)$$

Ainsi, 
$$\mu = \frac{1}{H_n} \cong \frac{1}{(\ln n + \gamma)} \quad (2.12)$$

Nous avons alors  $f(n) = 1$ , puisque le rang  $n$  est le dernier rang au-dessus de la distribution et que le mot  $n$ 'apparaît qu'une fois.

Par conséquent, 
$$p(n) = \frac{1}{N} = \frac{\mu}{n} \quad (2.13)$$

La relation entre le nombre total  $N$  d'occurrences de mots et le nombre  $n$  de mots distincts est ainsi:

$$N \cong n (\ln n + \gamma) \quad (2.14)$$

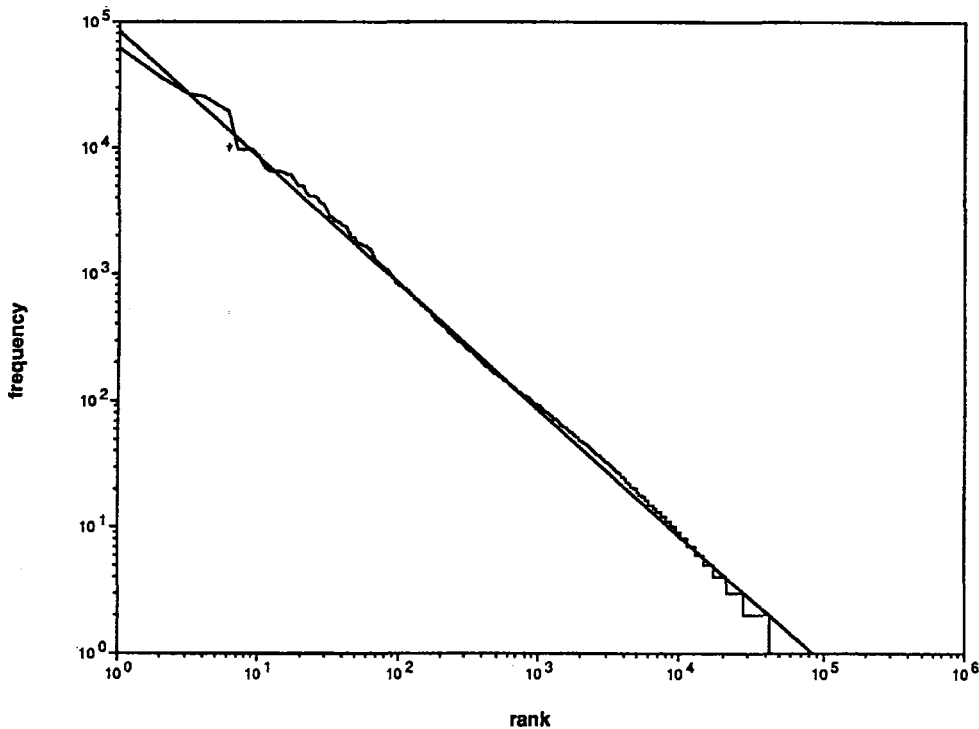


Fig. 2.6 Le graphe de rang-fréquence des mots anglais

La (Figure 2.6) montre le graphe rang-fréquence des 100237 mots différents. La constante de normalisation est calculée à partir de

$$\mu \cong \frac{1}{(\ln n + \gamma)} = 0.08270 \quad \text{avec le modèle de Zipf.}$$

Le nombre total d'occurrences de mots peut être estimé par l'équation (2.14).

$$N \cong n (\ln n + \gamma) = 1,212,117.$$

Pour améliorer l'ajustement de la distribution pour  $r$  petit, un paramètre  $c$  peut être introduit au dénominateur. Un autre paramètre  $B$  peut être ajouté pour améliorer l'ajustement pour  $r$  grand, soit :

$$p(r) = \frac{\mu}{(c+r)^B} \quad r = 1, 2, 3, \dots, n \quad (c > 0) \quad (2.15)$$

Selon *Mandelbrot* [Man52], dont cette distribution porte le nom,  $B > 1$  dans tous les cas habituels. Il a défini  $1/B$  comme la température de l'information du texte car  $1/B$  est d'autant plus grand que le vocabulaire est varié. Cependant, dans notre expérience, nous avons trouvé  $B < 1$  pour les mots fréquemment utilisés dans les textes coréens.

### 2.3.2 La loi de Zipf appliquée au français

La loi de Zipf en français a été étudiée par *Jean Sennellart* [Sén95]. Un coefficient  $B$  a été utilisé pour améliorer la précision pour tout  $r$  :

$$r^B \cdot f(r) \cong k \quad r = 1, 2, 3, \dots, n \quad (2.16)$$

(  $k$  : constante sur un texte donné )

Expérimentalement, la valeur du coefficient  $B$  est telle que  $1 < B < 2$ . Le nombre total d'occurrences de mots dans le corpus est :

$$N = \sum_{r=1}^n f(r) \cong k \sum_{r=1}^n \frac{1}{r^B} \quad (2.17)$$

La relation entre le nombre total  $N$  d'occurrences de mots et le nombre  $n$  de mots distincts peut être obtenue en employant l'inéquation suivante :

$$\sum_{r=a+1}^b f(r) \leq \int_a^b f(r) dr \leq \sum_{r=a}^{b-1} f(r) \quad (2.18)$$

La fonction  $f(r) = \frac{1}{r^B}$  et les variables  $a$  et  $b$  sont substituées dans (2.18) :

$$\int_1^{n+1} \frac{1}{r^B} dr \leq \sum_{r=1}^n \frac{1}{r^B} \leq 1 + \int_1^n \frac{1}{r^B} dr \quad (2.19)$$

Afin de déterminer la valeur de la somme, nous calculons les deux intégrales et puis leur moyenne arithmétique :

$$\sum_{r=1}^n \frac{1}{r^B} \cong \frac{B+1}{2(B-1)} - \frac{1}{n^{B-1}(B-1)} - \frac{1}{2n^B} \quad (2.20)$$

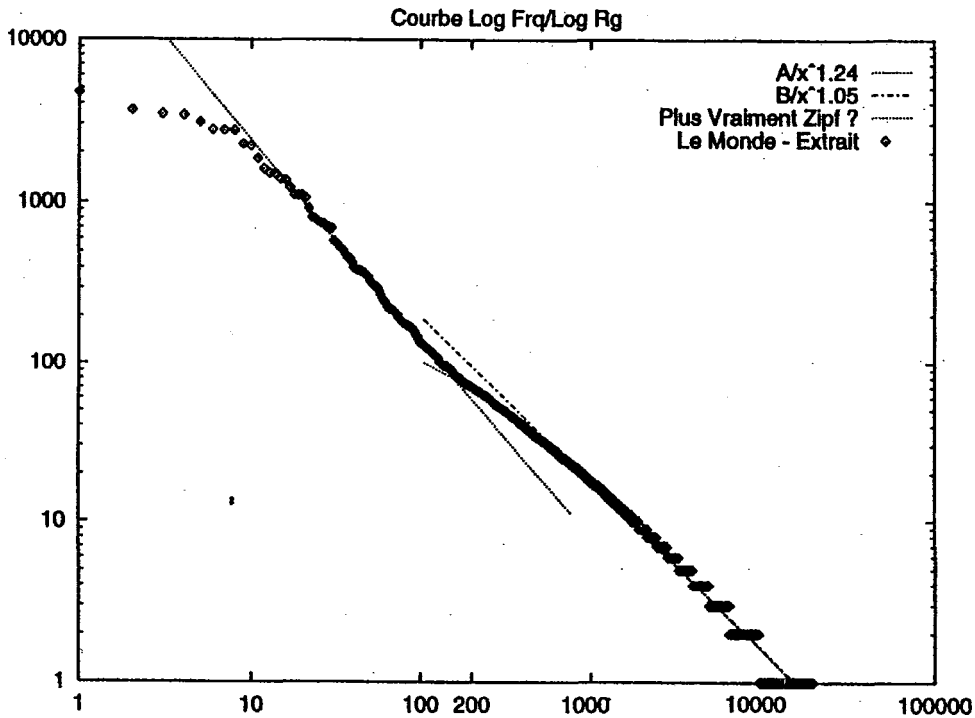


Fig. 2.7 Le graphe de rang-fréquence des mots français

Si  $r = n$ , alors  $n^B \cdot f(n) \cong k$ . Ici  $f(n)$  vaut 1. La relation entre le nombre total  $N$  d'occurrences de mots et le nombre  $n$  de mots distincts dans les textes français est :

$$N \cong n^B \frac{(B+1)}{2(B-1)} - \frac{n}{B-1} \quad (B > 1) \quad (2.21)$$

Dans la (Figure 2.7) de *Jean S nellart*, le coefficient  $B$  est exp rimentalement estim  entre 1.24 et 1.05 dans le cas de textes fran ais.

### 2.3.3 La loi de Zipf appliqu e au cor en

Nous avons fait des exp riences int ressantes avec des corpus de diverses tailles. Nous avons r duit la taille du corpus original   80 %, 60 %, 40 % et 20 %. Nous montrons la taille des corpus et leurs nombres de mots diff rents dans la (Figure 2.8). Les graphes rang-fr quence de ces 5 corpus sont montr s dans la (Figure 2.9), et les graphes rang-probabilit  sont montr s dans la (Figure 2.10). Les graphes des corpus de tailles diff rentes ont la m me forme sur l'ensemble des mots fr quents, autrement dit le param tre  $B$  a presque la m me valeur, en d pit des tailles diff rentes des corpus. Cette exp rience sugg re que la courbe est cintr e dans n'importe quel corpus cor en suffisamment volumineux. La courbe s'incline plus doucement que dans le cas de textes anglais ou de textes fran ais. En d'autres termes, '*la temp rature de l'information*' de *Mandelbrot*,  $1/B$ , devrait  tre plus grande qu'en anglais ou fran ais. Nous pouvons nous attendre   ce que le param tre  $B$  soit inf rieur   1.

| KAIST corpus | Total de mot | Mots diff rents | Taux (%) |
|--------------|--------------|-----------------|----------|
| Corpus 1     | 8,768,500    | 1,344,018       | 100 %    |
| Corpus 2     | 7,005,270    | 1,130,639       | 80 %     |
| Corpus 3     | 5,250,854    | 905,983         | 60 %     |
| Corpus 4     | 3,495,160    | 665,305         | 40 %     |
| Corpus 5     | 1,758,330    | 401,599         | 20 %     |

Fig. 2.8 Les tailles des corpus d'essai et les occurrences

Nous pouvons donner une explication intuitive de ce ph nom ne. Puisque le cor en est une langue agglutinante, les formes fl chies des verbes et des adjectifs sont tr s riches. Ils peuvent avoir beaucoup de suffixes diff rents, flexionnels ou non. Des milliers de combinaisons de suffixes flexionnels peuvent  tre appos es   la racine. Ces formes fl chies contribuent   la richesse du vocabulaire en cor en, parce que chaque combinaison produira un mot diff rent. Au contraire, dans le cas de l'anglais, le nombre d'occurrences de chaque forme fl chie est

relativement petit. Ces faits expliquent que la courbe s'incline plus doucement, et nous pensons qu'ils jouent un rôle dans le fait que la courbe est cintrée.

Afin de modéliser la distribution rang-fréquence du coréen, nous avons essayé deux méthodes. Le premier modèle est une *distribution modifiée de Mandelbrot*. Dans le deuxième modèle, nous divisons l'ensemble de la distribution en trois parties et adaptons la *distribution de Mandelbrot* avec  $c = 0$  et  $D = 0$  à chaque partie.

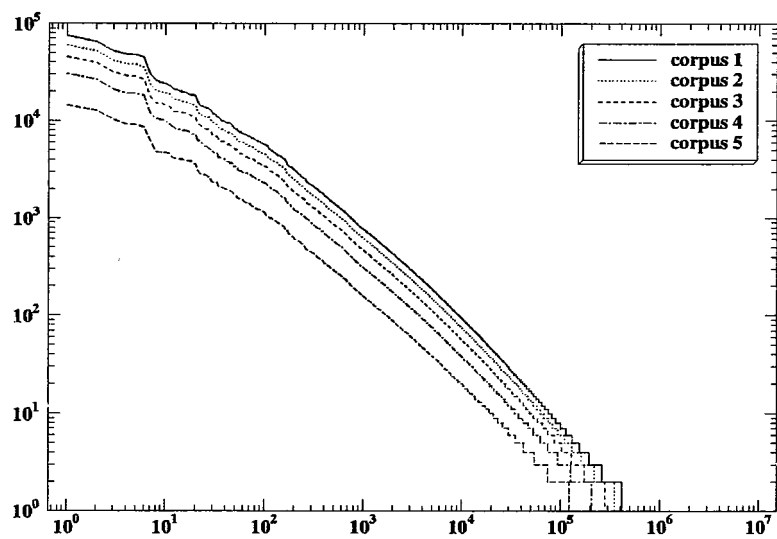


Fig. 2.9 Les graphes rang-fréquence du corpus coréen

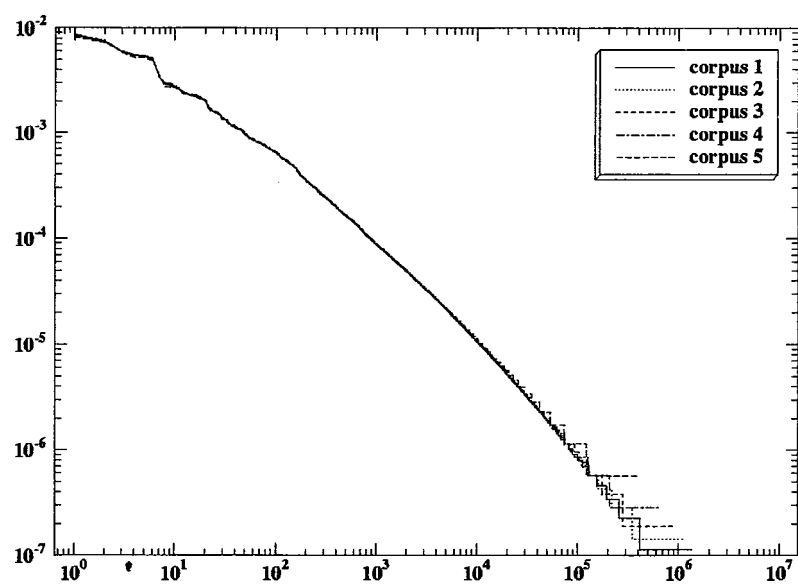


Fig. 2.10 Les graphes rang-probabilité du corpus coréen

### 2.3.4 Le modèle modifié de distribution de *Mandelbrot*

Nous pouvons adapter la *distribution de Mandelbrot* au texte coréen en ajoutant quelques facteurs. L'introduction de  $c$  rend la courbe logarithmique convexe pour les petites valeurs de  $r$ , mais la convexité de la courbe expérimentale est également importante pour les grandes valeurs de  $r$  : c'est pourquoi nous avons modifié la distribution de Mandelbrot afin de faire de  $B(r)$  un paramètre variable. La distribution modifiée de Mandelbrot est :

$$p(r) = \frac{\mu}{(c+r)^{B(r)}} \quad r = 1, 2, 3, \dots, n \quad (c : \text{constante}) \quad (2.22)$$

$$\text{où} \quad B(r) = \alpha + D \cdot e^{\left(1 - \frac{E}{r}\right)} \quad (\alpha > 0, D > 0, E > 0 : \text{constante}) \quad (2.23)$$

Nous avons appliqué ce modèle au corpus du KAIST (1344018 mots différents). Des valeurs appropriées des constantes ont été déterminées expérimentalement pour le corpus d'essai. La valeur de  $B(r)$  varie dans l'intervalle  $0.885 \leq B(r) \leq 0.9359$ .

$$p(r) = \frac{0.0397}{(11+r)^{(0.885+0.019 \cdot \exp(1-20000/r))}} \quad (2.24)$$

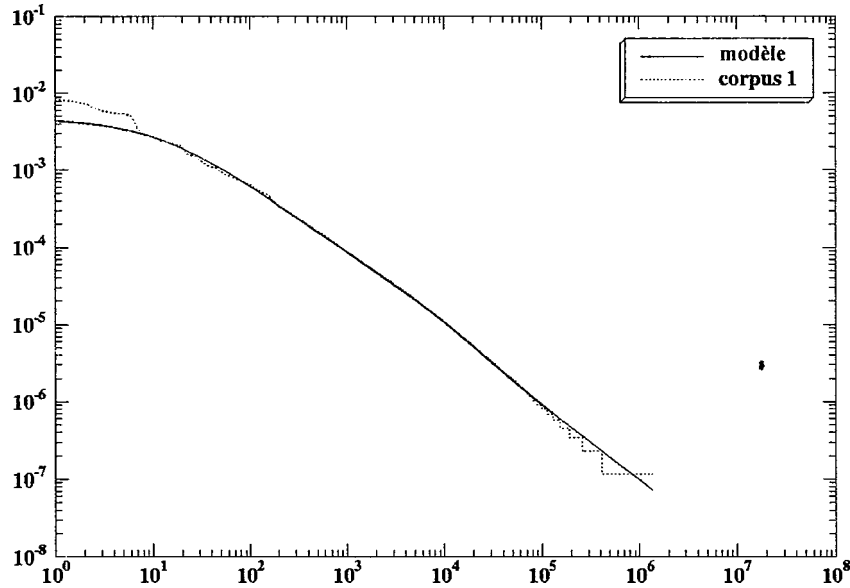


Fig. 2.11 Le graphe rang-probabilité des données réelles et du modèle

Nous représentons deux graphiques sur le même plan dans la (Figure 2.11). Le premier est le graphe rang-probabilité des données de corpus, et l'autre est le graphe obtenu avec la formule (2.24). Le graphe modèle concorde bien avec le graphe des données réelles sur la majeure partie excepté pour les mots moins fréquents. L'approximation est excellente mais il n'est pas facile de prévoir la relation entre le nombre total  $N$  d'occurrences de mots et le nombre  $n$  de mots distincts du modèle.

### 2.3.5 Le modèle des trois parties

Pour modéliser le corpus de textes coréens, nous divisons le graphe rang-fréquence en trois parties selon la forme du graphe. Nous obtenons une deuxième variante de loi de Zipf pour les textes coréens.

Pour chaque partie,

*Coefficient de la partie  $i$  :*  $B_i$

*Rang seuil :*  $R_0 = 1, R_1, R_2, R_3 = n$

*Intervalle de la partie  $i$  :*  $R_{i-1} \leq r \leq R_i$

*Nombre total de mots dans la partie  $i$  :*  $N_i$

*Nombre total de mots dans le corpus :*  $N = N_1 + N_2 + N_3$

La loi de Zipf est alors :

$$r^{B_i} \cdot f(r) \cong k_i \quad i = 1, 2, 3 ; \quad R_{i-1} \leq r \leq R_i \quad (2.25)$$

( $k$  : constante sur d'un texte donné)

Le nombre total  $N_i$  d'occurrences de mot dans la partie  $i$  est :

$$N_i = \sum_{r=R_{i-1}}^{R_i} f(r) \cong k_i \sum_{r=R_{i-1}}^{R_i} \frac{1}{r^{B_i}} \quad (2.26)$$

Nous avons effectué la même opération de calcul que dans le cas du français dans la (section 2.3.2 du chapitre 2 de la première partie) pour obtenir la relation entre  $N$  et  $n$ . La formule suivante est obtenue en approchant les sommes par des intégrales.

$$k_i \left( \frac{R_i^{(1-B_i)} - R_{i-1}^{(1-B_i)}}{1 - B_i} \right) \leq k_i \sum_{r=R_{i-1}}^{R_i} \frac{1}{r^{B_i}} \leq k_i \left( \frac{1}{R_{i-1}^{B_i}} + \frac{R_i^{(1-B_i)} - R_{i-1}^{(1-B_i)}}{1 - B_i} \right) \quad (2.27)$$

Nous prenons la valeur de la moyenne arithmétique pour déterminer  $N_i$ . Nous obtenons ainsi la formule suivante pour le nombre d'occurrences de mots  $N_i$  :

$$N_i \cong k_i \left( \frac{R_i^{(1-B_i)} - R_{i-1}^{(1-B_i)}}{1 - B_i} + \frac{g_i}{R_{i-1}^{B_i}} \right) \quad \begin{array}{l} B_i < 1 \text{ pour } i = 1, 2 \\ B_i > 1 \text{ pour } i = 3 \end{array} \quad (2.28)$$

où  $R_0 = 1, R_1 = n_1, R_2 = n_1 + n_2, R_3 = n$ .

Un paramètre  $g_i$  a été introduit et déterminé expérimentalement pour compenser l'approximation. Quand des sommes sont remplacées par des intégrales, un intervalle discret de longueur 1 est trop grand. Dans le cas du coréen, la formule reliant  $N$  et  $n$  ne peut pas être simplifiée plus. Par exemple, la relation entre  $N$  et  $n$  dans la partie 1 est:

$$N_1 \cong k_1 \left( \frac{n_1^{(1-B_1)} - 1}{1 - B_1} + g_1 \right) \quad \text{où } R_0 = 1. \quad (0 \leq g_i \leq 1, B_i < 1) \quad (2.29)$$

Il y a une constante de normalisation  $\mu_i$  pour chaque partie  $i$ , elle peut être obtenue avec  $R_i$  et  $R_{i-1}$  par la formule suivante:

$$\mu_i = \left( \frac{R_i^{(1-B_i)} - R_{i-1}^{(1-B_i)}}{1 - B_i} + \frac{g_i}{R_{i-1}^{B_i}} \right) \quad (0 \leq g_i \leq 1, i = 1, 2, 3) \quad (2.30)$$

Nous pouvons alors réécrire la loi de Zipf sous forme probabiliste en employant  $\mu_i$ .

$$p_i(r) = \frac{N_i}{N} \cdot \frac{\mu_i}{r^{B_i}} \quad (i = 1, 2, 3) \quad (2.31)$$

Nous avons expérimentalement déterminé les rangs-seuil :  $R_0 = 1, R_1 = 165, R_2 = 7441, R_3 = 1344019$ . Nous avons trouvé les valeurs de  $B_i$  et de  $k_i$  pour chaque partie du graphe rang-fréquence. Nous montrons ces valeurs dans la (Figure 2.12).

|       | Part 1              | Part 2                 | Part 3                     |
|-------|---------------------|------------------------|----------------------------|
| $r$   | $1 \leq r \leq 164$ | $165 \leq r \leq 7440$ | $7441 \leq r \leq 1344018$ |
| $B_i$ | 0.678               | 0.885                  | 1.07442                    |
| $k_i$ | 121049.977          | 347886.375             | 1882741.5                  |
| $g_i$ | 0.2408              | 0.0254                 | 0.124                      |
| $N_i$ | 1595389             | 2994778                | 4179374                    |

Figure 2.12 Les valeurs des paramètres obtenus expérimentalement

La (Figure. 2.13) montre le graphe de distribution rang-fréquence pour les données réelles du corpus du KAIST en le comparant avec le graphe de la distribution modèle.

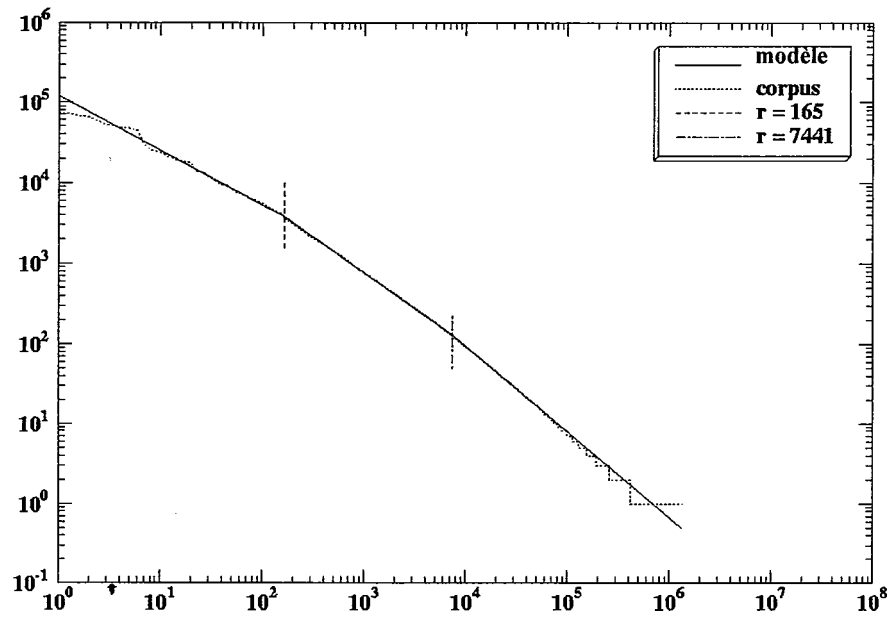


Fig. 2.13 La distribution rang-fréquence du modèle des trois parties pour les textes coréens

Pour conclure, les textes coréens se conforment à une variante de la loi de Zipf, à quelques différences du modèle près. Le paramètre  $B$  est  $B < 1$  dans les deux premières parties du modèle en trois parties, cependant, le paramètre  $B$  de la partie 3 est  $B > 1$  comme en français. Il n'y a aucune grande variation de la valeur  $B$  dans les deux premières parties malgré les variations de la taille de corpus. La même expérience a été exécutée avec des extraits de diverses tailles du même corpus. Nous avons également vérifié si les types de la distribution de fréquence sont indépendantes de la nature ou de la taille du corpus. Nous avons pu constater qu'il ne dépendent que de la langue elle-même.

Plusieurs données statistiques de base sur les textes coréens ont été construites. Nous espérons que ces données contribueront au développement des dictionnaires et des systèmes électroniques de traitement de textes comme la compression des textes ou les systèmes de recherche documentaire.

## Deuxième partie

Implantation de dictionnaires du coréen  
par Automates finis

# Chapitre 1

## Stratégies

Le but de ce travail est l'implantation du dictionnaire électronique du coréen représenté par automates. Les automates acycliques sont une structure de données bien adaptée à la représentation des lexiques des langues naturelles. Le grand avantage de la représentation par automates réside dans le fait que la vitesse d'accès aux mots du lexique ou du dictionnaire est très rapide, l'accès s'effectue en un temps en  $O(|w|)$  où  $|w|$  est la longueur d'un mot. Diverses méthodologies efficaces pour la construction d'automates acycliques ont été développées (voir [Dac98] page 30). Selon Dominique Revuz ([Rev91]), il n'existe pas d'algorithme rapide pour construire directement l'automate déterministe minimal d'un langage fini donné, basé sur des langages sous forme de liste. Il est donc nécessaire de procéder à la construction d'un automate acyclique en deux étapes. Dans la première étape, on élabore un automate qui reconnaît le langage utilisé dans la liste des entrées du dictionnaire. Puis dans la deuxième étape, on minimise cet automate. Il a proposé un algorithme de pseudo-minimisation pour élaborer des automates et un algorithme de minimisation des automates acycliques qui transforme un automate pseudo-minimal en automate minimal. Daciuk Jan, ([Dac98]), a présenté un procédé qui réduit la mémoire temporaire et le temps nécessaires à la construction en construisant l'automate minimal au fur et à mesure que les nouvelles données entrent. Ce sont des algorithmes efficaces pour construire un automate acyclique minimal.

Nous avons élaboré une stratégie qui s'applique en particulier au cas des dictionnaires coréens. En effet, dans ce cas, l'établissement des entrées des dictionnaires est un sujet problématique. Le nombre d'entrées du *DECOS* n'est pas très grand (environ 27 mille), tandis que celui du *DECOF* est extrêmement grand (plus de 148 millions). Le cardinal de l'alphabet coréen (2350 syllabes) est plus grand que celui de l'alphabet latin (moins de 50). La stratégie que nous avons mise au point privilégie la rapidité du temps d'exécution pour la construction du dictionnaire par l'automate minimal. Nous avons ainsi considéré que la vitesse était un élément plus important que la consommation de mémoire temporaire pour les applications. Grâce aux nouvelles technologies de nos jours, les ordinateurs peuvent être équipés d'une grande capacité de mémoire centrale. Même les ordinateurs personnels (PC) sont couramment équipés de 128 Mo de mémoire centrale. D'ailleurs, dans les applications utilisant des dictionnaires représentés par automates,

l'opération de reconstruire ou de restructurer le dictionnaire n'est pas fréquente. Cependant, il va sans dire que la réduction de l'espace mémoire est tout de même importante à prendre en compte dans les approches théoriques.

Notre stratégie se développe de la manière suivante : dans un premier temps, nous avons construit un automate acyclique déterministe avec un *Trie* comme structure de données. Pour réaliser l'automate, nous avons choisi un tableau bidimensionnel comme structure de données. Pour la minimisation de l'automate, nous avons adopté une méthode qui utilise *la table des transitions inverses* que nous avons introduite. Toutes les structures de données que nous avons utilisées offrent la possibilité d'un accès direct. Grâce à cette stratégie, nous pouvons implanter le système de dictionnaires électroniques du coréen par automates.

## 1.1 Utilisation restreinte du code de syllabes

Il existe de nombreux systèmes de codes pour représenter les syllabes coréennes en machine, nous en avons parlé de façon détaillée à (la section 1.2 du chapitre 1 de l'Introduction). Parmi eux, nous avons choisi le système de codes sur 2 octets (appelé le système de code '*KSC5601-1987 Hangul Wansung*') comme code standard pour la représentation des éléments lexicaux du dictionnaire électronique coréen *DECO*. Ce système de codes contient 2350 codes attachés aux 2350 syllabes coréennes respectives. A l'heure actuelle, tous les codes de syllabes n'apparaissent pas dans les textes coréens. D'après nos observations dans (le chapitre 1 de l'Introduction), 2139 (91%) codes différents de ce système sont utilisés dans le corpus du KAIST. Il couvre 8768500 mots coréens et occupe 44 Mo d'espace de stockage sous forme de fichiers de textes. Le nombre de syllabes différentes attestées dans des dictionnaires est moins grand que celui de notre corpus. Nous pouvons trouver 1367 (58%) syllabes différentes dans le dictionnaire *DECOS* qui comprend les noms simples, les verbes simples et les adjectifs simples. Ce phénomène apparaît plus distinctement dans le dictionnaire *DECOF*. 1364 (58%) syllabes différentes y sont utilisées pour représenter les radicaux des verbes et des adjectifs. Les syllabes utilisées dans les dictionnaires pour représenter les suffixes flexionnels des verbes et des adjectifs coréens sont vraiment spécifiques en coréen. Seulement 127 (5.4%) syllabes différentes sont nécessaires.

Nous pouvons en tirer un avantage certain pour construire les automates en utilisant des tables de codes des syllabes employées distinctement dans chaque dictionnaire. Nous utilisons des tables de code de syllabes différentes pour chaque automate. Nous pouvons ainsi réduire l'espace mémoire nécessaire pour stocker la table des transitions que nous utilisons pour représenter l'automate dans notre système. Nous représentons la table des transitions de l'automate par une structure de tableau bidimensionnel. La taille de la table des transitions est  $|\Sigma| \times |Q|$ , où  $|Q|$  est le nombre d'états et  $|\Sigma|$  est le cardinal de l'alphabet. Dans le cas du coréen, les 2350 syllabes sont considérées comme l'alphabet. On peut donc dire que  $|\Sigma|$  est le nombre de syllabes coréennes. Si nous ne distinguons pas les majuscules,

$|\Sigma|$  sera égal à 26 pour l'alphabet latin. Dans le cas du coréen,  $|\Sigma|$  vaut 2350. Ainsi, nous devons réduire la taille de l'alphabet autant que possible. En utilisant la méthode de l'utilisation restreinte du code de syllabes, nous pouvons économiser 42 % d'espace mémoire dans la construction des automates pour le *DECOS*. Pour le cas du *DECO-FlexAV*, 94.6 % d'espace mémoire peut être économisé par rapport à l'alphabet normal (2350 syllabes coréennes).

Un autre avantage est la rapidité de consultation de l'automate. Si certains codes de syllabes du mot n'existent pas dans la table des codes de syllabes, le mot n'est pas une entrée du dictionnaire. Cette caractéristique est très utile lorsqu'on consulte l'automate pour le dictionnaire *DECO-FlexAV*, puisque très peu de syllabes sont utilisées dans ce dictionnaire. D'après nos expériences, plus de 65.68 % des consultations d'automate du *DECO-FlexAV* font partie de ce cas. Il faut reconstruire la table de codes de syllabes lorsqu'une nouvelle entrée est ajoutée au dictionnaire. C'est un désavantage de cette méthode, mais les avantages sont plus importants.

## 1.2 Automate acyclique déterministe

Nous avons choisi comme structure de données un arbre appelé un *Trie* pour la première étape. Un *Trie* est un automate acyclique déterministe dont le graphe de transitions est un arbre ayant sa racine comme état initial et dont toutes les feuilles sont des états finaux. Par exemple dans la (Figure 1.1), un *Trie lié* peut être transformé en automate. Le tableau du *Trie Indexé* est semblable à la table des transitions de l'automate acyclique déterministe. Les indices de la table du *Trie Indexé* sont des familles de nœuds qui correspondent aux états de l'automate. Les éléments du tableau indiquent la prochaine famille de nœuds du *Trie* où il faut aller quand une lettre donnée est lue. Ces éléments correspondent donc à l'état suivant et les lettres correspondent aux étiquettes des transitions de l'automate.

Nous représentons l'automate acyclique déterministe par un tableau bidimensionnel qui nous permet un accès direct à la table des transitions. Un des défauts de cette méthode est qu'elle utilise beaucoup de mémoire temporaire pendant la construction de l'automate. Mais, ce problème n'est pas critique, car le nombre d'entrées de chacun des dictionnaires représenté par automate n'est pas gros. De plus, la capacité de la mémoire centrale des ordinateurs modernes augmente de plus en plus. Les entrées du *DECOS* sont au nombre de 27150, celles du *DECO-RA* au nombre de 52839 et celles du *DECO-FlexAV* au nombre de 20651. L'accès direct à la table des transitions est important puisque les codes de l'alphabet (les codes de syllabes) du coréen sont très nombreux. Nous pouvons obtenir automatiquement et directement l'automate acyclique déterministe pendant que nous fabriquons la table des transitions grâce aux caractéristiques du *Trie*. Autrement dit, tous les préfixes communs doublés des entrées sont éliminés automatiquement, il ne reste qu'un seul préfixe. Un des autres avantages est qu'il n'est pas nécessaire que les entrées du dictionnaire soient triées par ordre lexicographique avant la construction de l'automate. Nous pouvons

ainsi facilement fabriquer un automate qui reconnaît le langage de la liste du dictionnaire à partir de la liste dans un ordre quelconque.

$$L = \{abc, acb, baa, bac, bcb, c\}$$

◇ : la famille de nœuds de *Trie lié*.  
( ) : l'état correspondant de l'automate

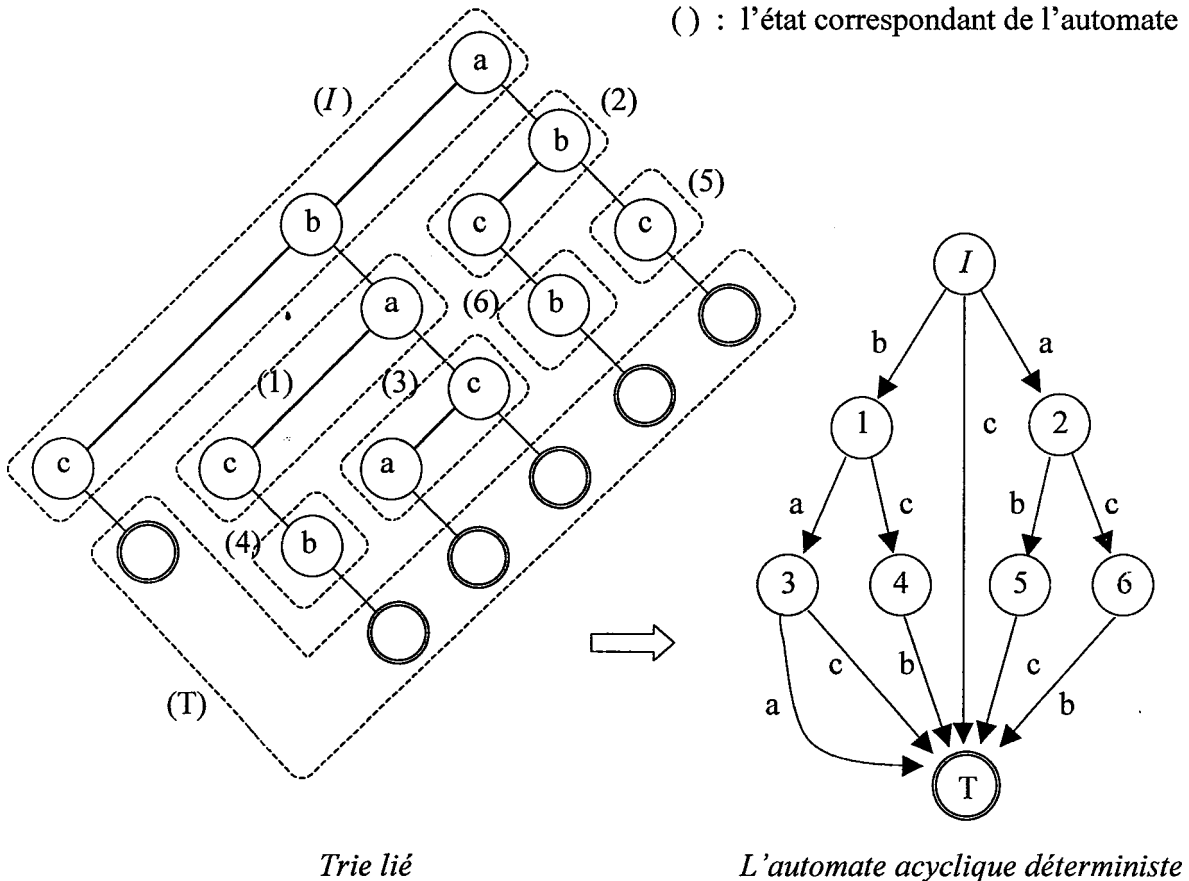


Fig. 1.1 Le *Trie lié* et l'*automate acyclique déterministe*

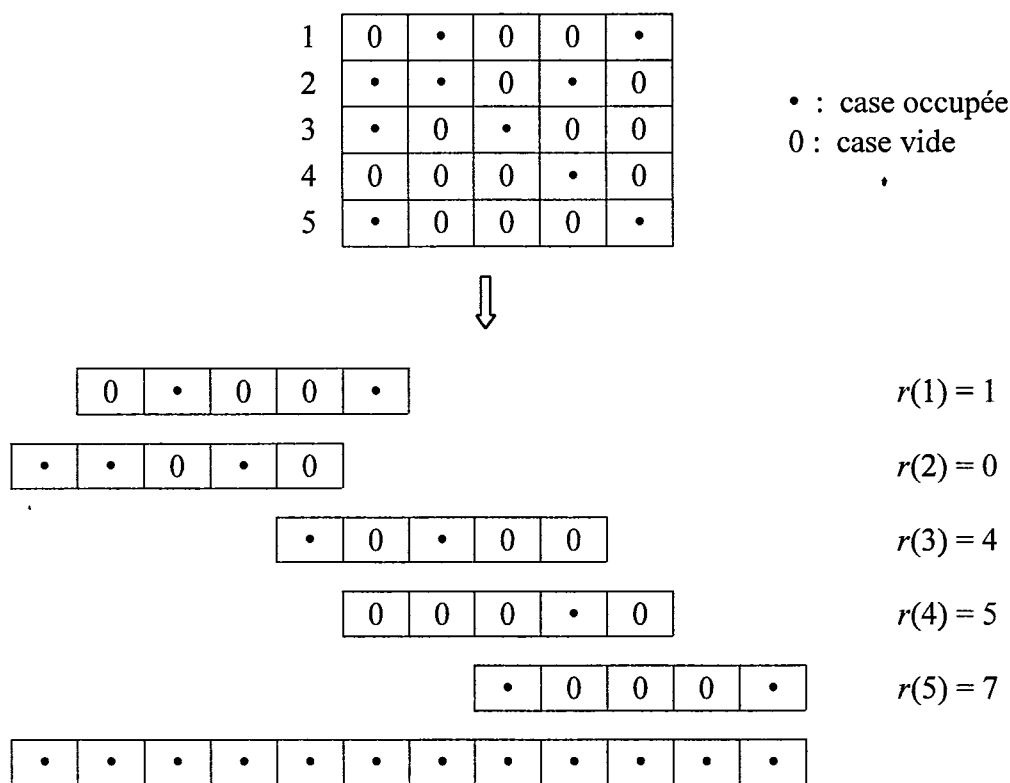
### 1.3 Minimisation

Pour minimiser l'automate déterministe, nous avons utilisé notre algorithme de la *table des transitions inverse* (voir la section 2.1.5, page 68). La structure de la table des transitions inverses est un tableau bidimensionnel normal. Si une transition est représentée par un triplet  $(p, a, q)$ , la transition inverse correspondante est représentée par un triplet  $(q, a, p)$ . La table des transitions inverses remplit trois fonctions : le positionnement, l'épreuve d'état redondant et la base de données des états redondants. Grâce à cette méthode, nous pouvons trouver facilement, sans examiner tous les états de l'automate, les états candidats à être fusionnés. Nous obtenons aussi les indices pour accéder directement dans la table des

transitions à l'état redondant qui doit être modifié ou effacé. Nous avons utilisé le concept de *hauteur* qui est la longueur du plus long chemin partant de l'état, et aboutissant à un état terminal. De plus, nous avons introduit un paramètre qui compte le nombre de transitions sortant d'un état. Après avoir construit l'automate minimal déterministe, nous avons compacté la table des transitions en un unique tableau comme nous l'expliquons dans la section suivante.

### 1.4 Compression de la table des transitions

La table des transitions contient beaucoup de cases inutiles. Nous avons utilisé l'algorithme très connu qui compresse un tableau incomplet. La méthode est décrite dans [Tar79]. Cette méthode comprime la matrice en arrangeant les lignes de sorte que les cases occupées ne se chevauchent pas. La matrice est comprimée en un vecteur. Cette méthode s'appelle une méthode de *'first-fit'*. Pour chaque ligne, une tentative est faite de la stocker dans le vecteur de telle manière que les transitions tombent dans les cases laissées vides par les autres lignes.



## 1.5 Matrice de cartographie pour *DECOF*

La taille du dictionnaire électronique du coréen des formes fléchies obtenues par combinaison entre des radicaux et des suffixes flexionnels est très grosse, environ 148 millions d'entrées occupant 2.55 Go sous forme texte. En raison de ce problème, le *DECOF* existe sous la forme séparée du *DECOS* et du *DECO-POST*. En raison de sa taille, il est très difficile de le construire à l'aide d'un seul automate.

Dans le cas de combinaisons d'adjectifs et de suffixes flexionnels adjectivaux ou de verbes et de suffixes flexionnels verbaux, les variations morphologiques se produisent dans les syllabes en association. Nous avons transformé le *DECOS-VS* et le *DECOS-AS* en le *DECO-RA*<sup>1</sup> qui contient les radicaux des verbes simples et des adjectifs simples. Un groupe de radicaux dans la liste du *DECO-RA* est obtenu par extension du radical de base selon sa syllabe finale (voir la section 3.1.1 du chapitre 3, page 90). Heureusement, il existe des règles de combinaison entre radicaux et suffixes flexionnels. Les codes du *DECO-FlexAV*<sup>2</sup> fournissent des informations sur les types morphologiques des adjectifs et des verbes. Nous avons aussi introduit des informations de combinaison dans le *DECO-RA*. Puis nous avons construit deux automates acycliques séparément pour les dictionnaires *DECO-RA* et *DECO-FlexAV* avec les codes qui contiennent les informations de combinaison.

Nous avons construit une matrice booléenne indicée respectivement par les codes du *DECO-RA* et du *DECO-FlexAV* donnant les informations de combinaison. Quand une chaîne de syllabes (une forme fléchie) est lue séquentiellement par deux automates, si le mot est accepté, deux automates émettent séparément les codes de la combinaison. Avec ces deux indices de matrice, on peut décider si le radical est accepté par l'automate du *DECO-RA* et si le suffixe flexionnel est accepté par l'automate du *DECO-FlexAV*. S'ils peuvent se combiner, le mot existe finalement dans le *DECOF*. Quelques surcharges existent dans cette méthode, mais il n'y en a pas trop, comme le montrent les résultats de nos expériences (voir la section 3.3.4 du chapitre 3 de la deuxième partie, page 106).

---

<sup>1</sup> *DECO-RA* : Dictionnaire électronique du coréen - radical

<sup>2</sup> *DECO-FlexAV* : Dictionnaire électronique du coréen – suffixes flexionnels d'adjectifs et de verbes

## Chapitre 2

# Construction de l'Automate pour *DECOS*

## 2.1 Construction de l'automate

### 2.1.1 Codes numériques de *DECOS*

Ce travail est effectué sur le système lexical du coréen *DECOS-NS*, *DECOS-VS*, *DECOS-AS*. Dans le dictionnaire, chaque entrée est associée un code exprimant une partie du discours et des informations grammaticales. (voir la (Figure 2.2) du chapitre 2 de l'Introduction, page 18). Il faut remarquer que le *DECOS* ainsi constitué contient nombre d'ambiguïtés, c'est-à-dire que plusieurs entrées peuvent avoir des clés d'accès identiques. Cela peut se produire pour un verbe et un adjectif ou pour les homographes d'un nom :

*Exemple 2.1* : le verbe et l'adjectif

|                                          |               |
|------------------------------------------|---------------|
| 가하다 ( <i>gahada</i> ) <b>ajouter</b>     | VS. /TRA /RHM |
| 가하다 ( <i>gahada</i> ) <b>raisonnable</b> | ADJS. /HM     |

*Exemple 2.2* : les homographes d'un nom

|                                                    |                   |
|----------------------------------------------------|-------------------|
| 가장 ( <i>gajang</i> ) <b>chef de famille</b>        | NS. /HUM /PREDHA  |
| 가장 ( <i>gajang</i> ) <b>feindre</b>                | NS. /PRED1 /PRED2 |
| 가장 ( <i>gajang</i> ) <b>enterrement provisoire</b> | NS. /PRED1 /PRED3 |

Le *DECOS* contient donc deux entrées avec 가하다 (*gahada*) et trois entrées avec 가장 (*gajang*) comme clé d'accès. La clé d'accès doit être unique pour représenter des dictionnaires par automate. Nous avons factorisé les clés d'accès dans le *DECOS*, ainsi le même extrait devient-il :

|                       |                                                |
|-----------------------|------------------------------------------------|
| 가하다 ( <i>gahada</i> ) | VS./TRA/RHM;ADJS./HM                           |
| 가장 ( <i>gajang</i> )  | NS./HUM/PREDHA;NS./PRED1/PRED2;NS./PRED1/PRED3 |

Nous avons construit un unique tableau de codes avec ces codes fusionnés. La valeur du code numérique qui est associée à l'entrée de dictionnaire est la clé d'accès à cet unique

tableau. Nous avons obtenu environ 330 codes différents pour le *DECOS* de cette manière. Les chaînes de codes, VS./TRA/RHM;ADJS./HM etc., sont en pratique transformées en codes numériques :

가하다 (*gahada*) 308

가장 (*gajang*) 60

Nous présentons un extrait de ce dictionnaire dans la (Figure 2.1).

가계 0  
 가공 1  
 가공적이다 4  
 가관 0  
 가교 5  
 가구 3  
 가군 2  
 가금 6  
 가까와지다 7  
 가깝다 8  
 가깝디가깝다 8  
 가깝하다 9  
 가꾸다 10  
 가꾸러뜨리다 10  
 가꾸러지다 11

Fig. 2.1 Un extrait de dictionnaire transformé en codes numériques.

La liste des entrées ainsi obtenue est utilisée pour la construction de l'automate pour *DECOS*. Ces codes numériques sont associés à des états finaux. Si un automate accepte un mot, le code numérique correspondant est utilisé pour chercher la chaîne de codes correspondant dans l'unique tableau de codes.

Le *DECOS* utilise 0.5Mo d'espace de stockage sous forme de fichiers texte. La version du *DECOS* utilisé dans notre travail ne contient pas encore le *DECOS-IS* (Lexique des mots invariables).

## 2.1.2 Indices du code de syllabes

Il s'agit maintenant de transformer des codes de syllabes en indices en utilisant le concept d'utilisation restreinte des codes de syllabes que nous avons introduit à (la section 1.1 du chapitre 2 de la deuxième partie, page 58). Les indices des colonnes de la table des transitions sont les symboles de l'alphabet. Nous avons représenté l'automate par un tableau, pour les besoins de la programmation les codes des symboles doivent être transformés en des indices numériques appropriés.

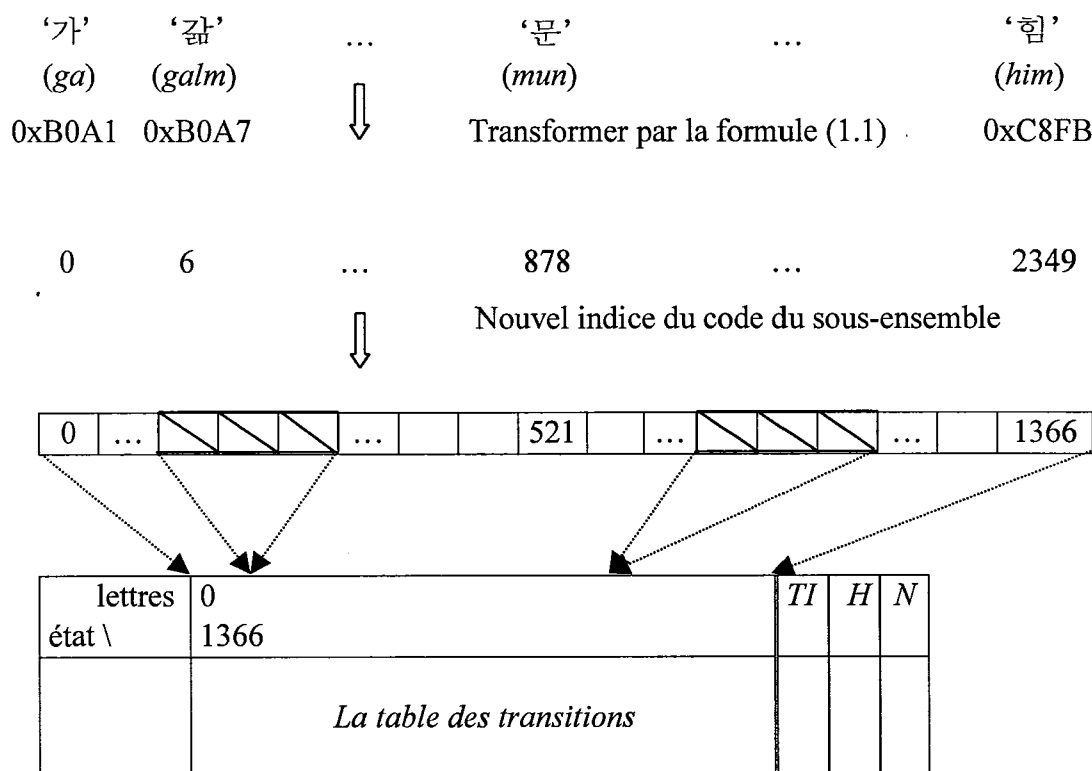


Fig. 2.2 Production des indices des codes des syllabes

Nous utilisons les syllabes coréennes comme symboles de la table des transitions. Nous avons montré le principe de la construction des pages de codes de syllabes coréennes à (la section 1.2.3 du chapitre 1 de l'Introduction, page 11). Grâce à cette structure, nous pouvons obtenir facilement une série d'indices (de 0 à 2349) par la formule suivante.

$$indice = (((code \gg 8) - 0xB0) \times 94) + ((code \& 0x00FF) - 0xA1) \quad (1.1)$$

$$code = (((indice / 94) + 0xB0) \ll 8) + ((indice \% 94) + 0xA1) \quad (1.2)$$

Où, '>>' est un opérateur de décalage à la droite et '&' est un opérateur de bit opération AND. Par exemple, '가'(ga) (code 0xB0A1) correspond à l'indice 0 et '힘'(him) (code 0xC8FB) correspond à l'indice 2346.

Nous avons créé un sous-ensemble des codes de syllabes qui sont dans la pratique utilisés dans la liste des entrées de dictionnaire. Nous avons construit une table de mise en correspondance autour de ce sous-ensemble pour obtenir une nouvelle série d'indices adaptés. La marche à suivre pour établir cette table de transformation est la suivante. Les codes du sous-ensemble sont triés par ordre croissant. On crée une liste de taille 2350.

Ensuite, nous assignons séquentiellement un nouvel indice, qui correspond aux syllabes du sous-ensemble, en commençant à partir de l'indice 0. Nous montrons cette transformation dans la (Figure 2.2). Ces indices sont utilisés dans tous les tableaux indicés de l'alphabet.

### 2.1.3 Représentation des états terminaux

Il y a deux sortes d'états terminaux dans notre automate déterministe acyclique ; *les états terminaux intermédiaires* qui sont des états terminaux placés entre l'initial et les états finaux (voir la (Figure 2.3) ; par exemple l'état '6' ).

|   | a | b    | c    | d    | TI   | H | N |
|---|---|------|------|------|------|---|---|
| 1 | 2 | 5    |      |      |      | 3 | 2 |
| 2 |   | 3    | 4    |      |      | 2 | 2 |
| 3 |   |      | 3020 |      |      | 1 | 1 |
| 4 |   | 3030 |      |      |      | 1 | 1 |
| 5 | 6 |      |      |      |      | 2 | 1 |
| 6 |   |      |      | 3041 | 3055 | 1 | 1 |
| 7 |   |      |      |      |      |   |   |

Les entrées sont :

$$L = \{(abc, 20), (bad, 41), (acb, 30), (ba, 55)\}$$

La valeur de référence est : 3000

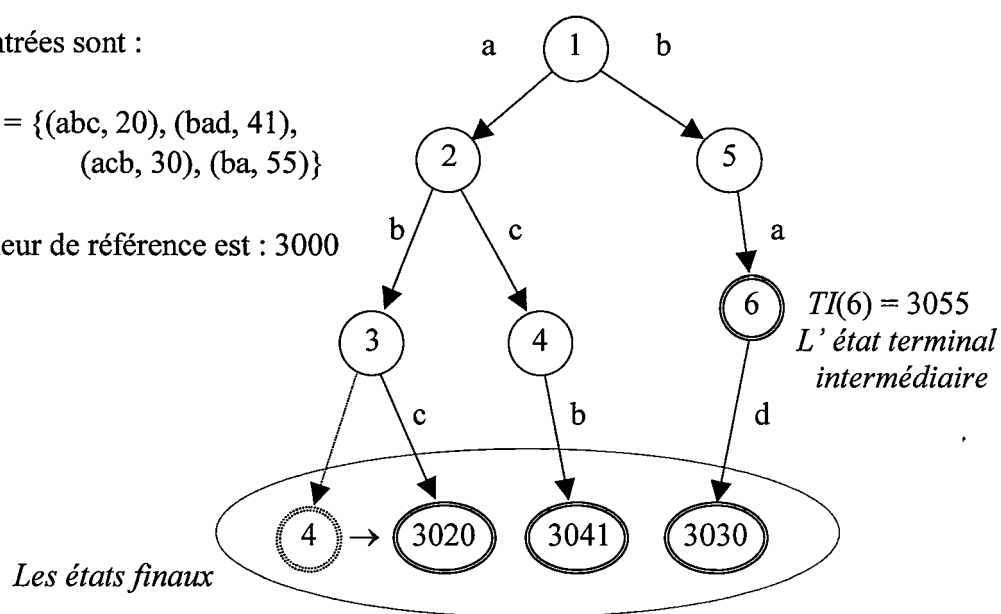


Fig. 2.3 Représentation des états terminaux

Il faut utiliser généralement un deuxième tableau ou une liste pour les états terminaux. Mais, nous n'utilisons pas de deuxième table. Nous avons joint simplement une colonne à la table des transitions pour *les états terminaux intermédiaires* et nous avons également introduit un *code terminal* qui est significatif d'état terminal et de code grammatical pour les états finals.

En fait, les états sont représentés par des numéros dans les cases de la table des transitions. Les codes grammaticaux aussi sont représentés par des numéros. Nous avons déjà transformé un code exprimant une partie du discours et les informations grammaticales en un code numérique. Pour représenter les deux dans la même table de transitions, il faut arriver à les distinguer. Nous pouvons ajouter une *valeur de référence* suffisamment grande à un code grammatical pour obtenir un *code terminal*, puisqu'à entrée n'est associée qu'un seul code numérique. Ce code terminal est utilisé comme numéro de l'état terminal à la place de son numéro normal. Par exemple, si l'on considère une entrée (abc, 20) dans la (Figure 2.3), les transitions normales sont (1, a, 2), (2, b, 3) et (3, c, 4). Cependant comme le numéro de l'état terminal est '4', on modifie la transition (3, c, 4) en (3, c, 3020). Par contre dans le cas de l'entrée (ba, 55), la transition (5, a, 6) est la transition finale et l'état '6' peut devenir un état terminal. La transition (5, a, 6) devrait être changée toutefois en (5, a, 3055), mais elle reste (5, a, 6), car état '6' est un état terminal intermédiaire.

' Pour la table des transitions inverses, nous numérotés tous les états finaux par le numéro spécial '0', par suite, une case de la ligne indicée par '0' peut contenir plusieurs états, ce qui signifie que si plusieurs états sont dans une même case de la ligne indicée par '0', ils deviennent candidats comme états redondants.

Nous pouvons directement savoir qu'un état est final grâce à la valeur référence dans la table des transitions. Dans le cas d'un état terminal intermédiaire, on trouve ce code terminal dans la colonne (*TI*) de la table des transitions. On peut récupérer facilement le code grammatical en soustrayant la valeur de référence indiquant un code terminal.

## 2.1.4 Structure de la table des transitions

La *table des transitions* est un tableau bidimensionnel. La taille de la table des transitions est obtenue par  $(|\Sigma| + 3) \times |Q|$ , où  $|\Sigma|$  est le nombre de codes du *sous-ensemble des syllabes* et  $|Q|$  est le nombre d'états. Nous avons ajouté trois colonnes à cette table; une colonne (*H*) pour représenter la *hauteur* de chaque état, une colonne (*TI*) pour indiquer *les états terminaux intermédiaires* et une colonne (*N*) pour représenter *le nombre de transitions sortant de chaque état*. Une transition est représentée par un triplet  $(p, a, q)$ , où  $p$  et  $q$  sont des états et  $a$  un symbole de l'alphabet  $\Sigma$ . On utilise la notion de hauteur  $H$  définie sur les états par

$$H(s) = \text{MAX}(|w|) ; s.w \text{ appartient à } T$$

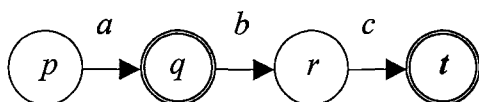
On peut calculer ces trois valeurs pendant la construction l'automate déterministe. Par exemple, si un mot  $w$  a  $n$  syllabes, on peut penser que la hauteur maximale de ce mot est la longueur du mot  $|w| = n$ . Autrement dit, la hauteur de la première syllabe est  $n$ , de la deuxième est  $n-1$  et de la troisième est  $n-2$ . Si la première syllabe conduit à l'état  $p$ , et que  $p$  n'est pas encore dans la table des transitions, la hauteur de l'état  $p$  devient  $n$ . Par contre, si l'état  $p$  existait déjà, la plus grande valeur entre la hauteur actuelle de  $p$  et  $n$  devient la hauteur de l'état  $p$ . On répète la même procédure pour les autres syllabes. On peut compter facilement *le nombre de transitions sortant d'un état*. Si une syllabe conduit à un état, et que cette transition est nouvelle, le nombre de transitions sortant de cet état doit alors être augmenté d'un. Cette procédure se répétera jusqu'à la fin de la liste des entrées. Les procédures décrites ci-dessus sont accomplies en même temps que la construction de l'automate déterministe. On n'a pas besoin de procédure supplémentaire pour obtenir ces trois paramètres. La minimisation d'un automate consiste à éliminer les états redondants, de manière à obtenir un nouvel automate ayant le même langage associé. Pour décider l'équivalence de deux états, les trois paramètres ci-dessus sont des conditions nécessaires. Ils sont donc très utiles à tester.

## 2.1.5 La table des transitions inverses

Nous avons introduit *la table des transitions inverses* pour minimiser l'automate déterministe acyclique. La structure de la table des transitions inverses est un normal tableau bidimensionnel. Si une transition est représentée par un triplet  $(p, a, q)$ , la transition inverse correspondante est représenté par un triplet  $(q, a, p)$ . Nous présentons la relation des deux tables des transitions dans la (Figure 2.4). Il y a deux intérêts à utiliser cette table.

Le premier est que nous pouvons facilement trouver des états candidats pour la minimisation. Ces états candidats apparaissent dans la même case de la table des transitions inverses. Dans le cas d'automate acyclique déterministe, chaque case de la table des transitions inverses doit être occupée par un seul état sauf celle correspond à l'état final. Le deuxième est que nous pouvons obtenir les informations sur la position de certains états dans la table des transitions. Cette information nous permet l'accès direct à la table des transitions. Ainsi, le temps de recherche d'un état redondant qui doit être éliminé est extrêmement rapide. Le problème est que cette table occupe d'un grand espace de mémoire tout comme la table des transitions (  $|Q| \times |\Sigma|$  ). Nous avons trouvé une méthode qui réduit l'espace mémoire occupé à un tableau de taille  $|Q| \times 2$ , un tableau de taille  $(2 \times |\Sigma|)$  et un tableau de taille  $m \times |\Sigma|$  pour la table des transitions inverses. Nous expliquerons cette méthode à la section suivante (section 2.1.6).

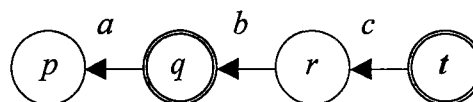
$$L = \{ a, abc \}$$



Transitions :  $F = \{(p,a,q),(q,b,r),(r,c,t)\}$

| lettre<br>état | a | b | c | TI | H | N |
|----------------|---|---|---|----|---|---|
| p              | q |   |   |    | 3 | 1 |
| q              |   | r |   | 1  | 2 | 1 |
| r              |   |   | t |    | 1 | 1 |
|                |   |   |   |    |   |   |

< La table des transitions >



Transitions inverses de  $F$

| lettre<br>état | a | b | c |
|----------------|---|---|---|
| t              |   |   | r |
| p              |   |   |   |
| q              | p |   |   |
| r              |   | q |   |

< La table des transitions inverses >

TI: état terminal intermédiaire

H : hauteur de chaque état

N : nombre de transitions sortantes de chaque état

Fig. 2.4 La relation entre transition et transition inverse

## 2.1.6 Variantes de la table des transitions inverses

Nous pouvons économiser l'espace mémoire nécessaire pour stocker la table des transitions inverses en éliminant les cases inutiles. Si l'automate acyclique déterministe est construit sans erreur, chaque case de la table des transitions inverses doit être occupée par un seul état sauf dans le cas des états finaux. Autrement dit, la case valide est unique à chaque ligne et les autres sont cases inutiles.

La table des transitions inverses remplit trois fonctions : le positionnement, l'épreuve d'état redondant et la base de données des états redondants. Plus précisément, nous avons créé :

- . un tableau de taille  $(|Q| \times 2)$  pour les positionnements, les lignes de ce tableau sont indicées par les états, les colonnes mentionnent respectivement l'état-but et l'étiquette d'une transition ;
- . un tableau de taille  $(2 \times |\Sigma|)$  pour l'épreuve d'état redondant qui est un tableau temporaire construit pour un état redondant potentiel et contient un indicateur booléen indiquant la présence ou non de plusieurs état-but ;

–. un tableau de taille  $(m \times |\Sigma|)$  pour la base de données des états redondants, les colonnes de ce tableau sont indicées par l'alphabet et contiennent tous les états atteints par les étiquetées par le symbole correspondant, à raison d'un état par case.

Nous pouvons ainsi représenter la table des transitions inverses par trois plus petits tableaux. Nous les avons mis dans la (Figure 2.5).

(1) Un tableau de taille  $(|Q| \times 2)$  pour les positionnements.

| lettre<br>état | <i>a</i> | <i>b</i> | <i>c</i> |
|----------------|----------|----------|----------|
| <i>t</i>       |          |          | <i>r</i> |
| <i>p</i>       |          |          |          |
| <i>q</i>       | <i>p</i> |          |          |
| <i>r</i>       |          | <i>q</i> |          |

forme normale :  $(|Q| \times |\Sigma|)$

$\Rightarrow$

| état     | but      | lettre   |
|----------|----------|----------|
| <i>p</i> |          |          |
| <i>q</i> | <i>p</i> | <i>a</i> |
| <i>r</i> | <i>q</i> | <i>b</i> |

forme modifiée :  $(|Q| \times 2)$

(2) Un tableau d'extension de la forme modifiée de taille  $(2 \times |\Sigma|)$  pour l'épreuve d'état redondant.

( par exemple : extension d'état '*q*' )

| lettre          | <i>a</i>      | <i>b</i> | <i>c</i> |
|-----------------|---------------|----------|----------|
| Indicateur      | <i>0 ou 1</i> |          |          |
| état = <i>q</i> | <i>p</i>      |          |          |

–. L'indicateur vaut 0 s'il n'y a qu'un seul état-but et 1 quand il y en a plusieurs.

(3) Un tableau de taille  $(m \times |\Sigma|)$  pour la base de données des états redondants.

| $\Sigma$ | <i>a</i> | <i>b</i> | <i>c</i> |
|----------|----------|----------|----------|
| 0        |          |          |          |
| ...      |          |          |          |
| <i>m</i> |          |          |          |

*m* : Le nombre maximum d'états redondants associée à une lettre

Fig. 2.5 Variantes de la table des transitions inverses

## 2.2 Exemples

### 2.2.1 Détermination

Ce paragraphe présente la minimisation d'un automate acyclique déterministe utilisant la table des transitions inverses. Nous l'expliquons sur l'exemple suivant. Nous utilisons la table des transitions inverses sous sa forme normale pour faciliter les explications. En pratique, nous utilisons la forme modifiée de la table des transitions inverses.

On suppose que :

- . l'alphabet  $|\Sigma| = \{ a, b, c, d, e \}$
- . le langage  $L = \{(baa,1), (bba,1), (eab,3), (eaba,1), (aba,1), (ac,1), (caa,1), (dc,1), (cba,1), (cab,1), (dba,1), (cbb,1), (eba,1), (eaa,1)\}$
- . la valeur de référence : 100

On commence par la première entrée (baa,1) (Figure 2.6). Si l'état initial est 1, la première transition est (1,b,2), et la suivante (2,a,3). La valeur de référence est 100, la transition finale est donc (3,a,101). Pour la table des transitions inverses, la transition finale est (0,a,3). Pour la deuxième entrée (bba,1) (Figure 2.7), la première transition est (1,b,2), parce que le préfixe 'b' existe déjà. Ensuite, l'état '2' n'a pas de transition qui corresponde au symbole 'b', on crée donc un nouvel état '4'. La deuxième transition est (2,b,4). La transition finale est (4,a,101).

(1) Les tables après (baa,1) =  $\{(1,b,2), (2,a,3), (3,a,101)\}$

|   | a   | b | c | d | e | TI | H | N |
|---|-----|---|---|---|---|----|---|---|
| 1 |     | 2 |   |   |   |    | 3 | 1 |
| 2 | 3   |   |   |   |   |    | 2 | 1 |
| 3 | 101 |   |   |   |   |    | 1 | 1 |
|   |     |   |   |   |   |    |   |   |

[ La table des transitions ]

|   | a | b | c | d | e |
|---|---|---|---|---|---|
| 0 | 3 |   |   |   |   |
| 1 |   |   |   |   |   |
| 2 |   | 1 |   |   |   |
| 3 | 2 |   |   |   |   |

[ La table des transitions inverses ]

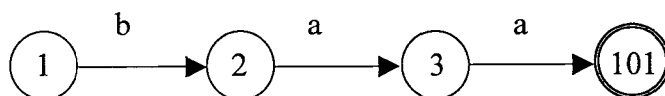


Fig. 2.6 Pour la première entrée (baa,1)

(2) Les tables après  $(bba,1) = \{(1,b,2), (2,b,4), (4,a,101)\}$

|   | a   | b | c | d | e | TI | H | N |
|---|-----|---|---|---|---|----|---|---|
| 1 |     | 2 |   |   |   |    | 3 | 1 |
| 2 | 3   | 4 |   |   |   |    | 2 | 2 |
| 3 | 101 |   |   |   |   |    | 1 | 1 |
| 4 | 101 |   |   |   |   |    | 1 | 1 |
|   |     |   |   |   |   |    |   |   |

|   | a   | b | c | d | e |
|---|-----|---|---|---|---|
| 0 | 3,4 |   |   |   |   |
| 1 |     |   |   |   |   |
| 2 |     | 1 |   |   |   |
| 3 | 2   |   |   |   |   |
| 4 |     | 2 |   |   |   |

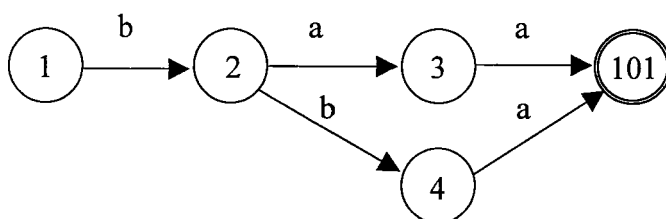


Fig. 2.7 Pour la deuxième entrée (bba,1)

On indique les informations relatives aux *états terminaux intermédiaires* dans la colonne *TI*. On considère d'abord l'entrée (eab,3) (Figure 2.8). Jusque là, les transitions sont normales ; (1,e,5), (5,a,6), (6,b,103).

(3) Les tables après  $(eab,3) = \{(1,e,5), (5,a,6), (6,b,103)\}$

|   | a   | b   | c | d | e | TI | H | N |
|---|-----|-----|---|---|---|----|---|---|
| 1 |     | 2   |   |   | 5 |    | 3 | 1 |
| 2 | 3   | 4   |   |   |   |    | 2 | 2 |
| 3 | 101 |     |   |   |   |    | 1 | 1 |
| 4 | 101 |     |   |   |   |    | 1 | 1 |
| 5 | 6   |     |   |   |   |    | 2 | 1 |
| 6 |     | 103 |   |   |   |    | 1 | 1 |
|   |     |     |   |   |   |    |   |   |

|   | a   | b | c | d | e |
|---|-----|---|---|---|---|
| 0 | 3,4 | 6 |   |   |   |
| 1 |     |   |   |   |   |
| 2 |     | 1 |   |   |   |
| 3 | 2   |   |   |   |   |
| 4 |     | 2 |   |   |   |
| 5 |     |   |   |   | 1 |
| 6 | 5   |   |   |   |   |

Fig. 2.8 Pour la troisième entrée (eab,3)

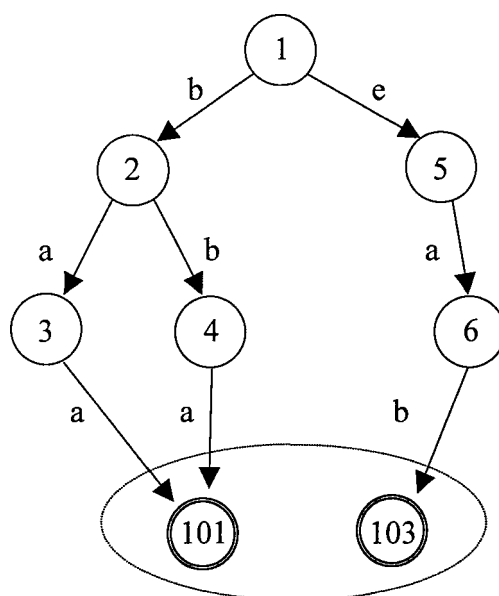


Fig. 2.8 (cont.) Pour la troisième entrée (eab,3)

Dès l'entrée (eaba,1), l'état '7' de la transition (7,a,101) dans la (Figure 2.9) est un état terminal intermédiaire. On le représente comme suit. Pour ajouter cette transition, la valeur dans la case (6,b), c'est-à-dire la transition (6,b,7), est modifiée par l'état '7'. On ajoute la transition (7,a,101) de façon normale. Ensuite, on indique que l'état '7' est un état terminal intermédiaire en mettant le code terminal '103', qui était dans la case (6,b), dans la case (7,TI). Il faut maintenant éliminer l'état '6' dans la case (0,b) de la table des transitions inverses, puisque l'état '7' dans la transition (6,b,7) n'est plus un état final, il a en effet changé son statut d'état final pour celui d'état terminal intermédiaire.

(4) Les tableaux après (eaba,1) = {(1,e,5), (5,a,6), (6,b,7), (7,a,101)}

|   | a   | b          | c | d | e | TI  | H | N |
|---|-----|------------|---|---|---|-----|---|---|
| 1 |     | 2          |   |   | 5 |     | 4 | 1 |
| 2 | 3   | 4          |   |   |   |     | 2 | 2 |
| 3 | 101 |            |   |   |   |     | 1 | 1 |
| 4 | 101 |            |   |   |   |     | 1 | 1 |
| 5 | 6   |            |   |   |   |     | 3 | 1 |
| 6 |     | 103<br>→ 7 |   |   |   |     | 2 | 1 |
| 7 | 101 |            |   |   |   | 103 | 1 | 1 |

|   | a     | b | c | d | e |
|---|-------|---|---|---|---|
| 0 | 3,4,7 | 6 |   |   |   |
| 1 |       |   |   |   |   |
| 2 |       | 1 |   |   |   |
| 3 | 2     |   |   |   |   |
| 4 |       | 2 |   |   |   |
| 5 |       |   |   |   | 1 |
| 6 | 5     |   |   |   |   |
| 7 |       | 6 |   |   |   |

éliminer

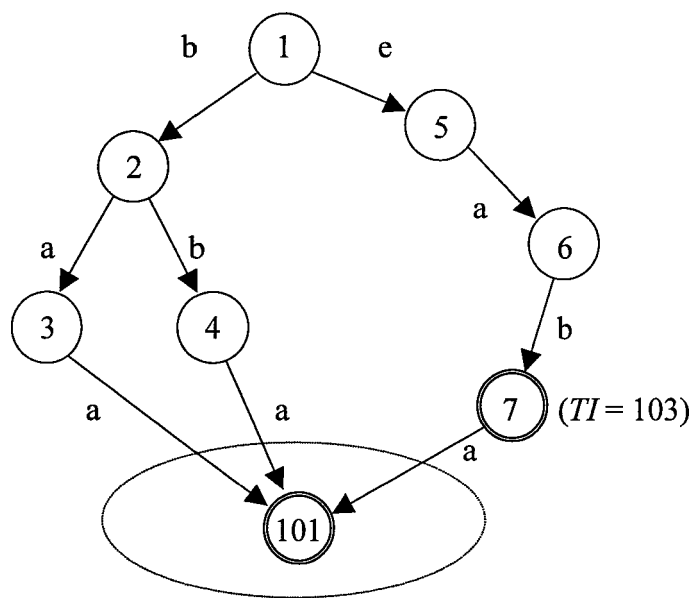


Fig. 2.9 Dans le cas de l'état *terminal intermédiaire*

On poursuit cette procédure jusqu'à la dernière entrée. Nous avons représenté la table des transitions obtenue dans la (Figure 2.10). On se sert en fait des deux petites tables (3) et (4) au lieu d'une grande table des transitions inverses pour réduire l'espace de mémoires temporaires utilisé.

Langage  $L = \{(baa,1), (bba,1), (eab,3), (eaba,1), (aba,1), (ac,1), (caa,1), (dc,1), (cba,1), (cab,1), (dba,1), (cbb,1), (eba,1)\}$

|    | a   | b   | c   | d  | e | TI  | H | N |
|----|-----|-----|-----|----|---|-----|---|---|
| 1  | 8   | 2   | 10  | 12 | 5 |     | 4 | 1 |
| 2  | 3   | 4   |     |    |   |     | 2 | 2 |
| 3  | 101 |     |     |    |   |     | 1 | 1 |
| 4  | 101 |     |     |    |   |     | 1 | 1 |
| 5  | 6   | 15  |     |    |   |     | 3 | 2 |
| 6  | 101 | 7   |     |    |   |     | 2 | 2 |
| 7  | 101 |     |     |    |   | 103 | 1 | 1 |
| 8  |     | 9   | 101 |    |   |     | 2 | 2 |
| 9  | 101 |     |     |    |   |     | 1 | 1 |
| 10 | 11  | 13  |     |    |   |     | 2 | 2 |
| 11 | 101 | 101 |     |    |   |     | 1 | 2 |
| 12 |     | 14  | 101 |    |   |     | 2 | 2 |
| 13 | 101 | 101 |     |    |   |     | 1 | 2 |
| 14 | 101 |     |     |    |   |     | 1 | 1 |
| 15 | 101 |     |     |    |   |     | 1 | 1 |
|    |     |     |     |    |   |     |   |   |

(1) [ La table des transitions ]

|    | a                                 | b         | c        | d | e |
|----|-----------------------------------|-----------|----------|---|---|
| 0  | 3,4,7,<br>9,11,<br>13,14,<br>15,6 | 11,<br>13 | 8,<br>12 |   |   |
| 1  |                                   |           |          |   |   |
| 2  |                                   | 1         |          |   |   |
| 3  | 2                                 |           |          |   |   |
| 4  |                                   | 2         |          |   |   |
| 5  |                                   |           |          |   | 1 |
| 6  | 5                                 |           |          |   |   |
| 7  |                                   | 6         |          |   |   |
| 8  | 1                                 |           |          |   |   |
| 9  |                                   | 8         |          |   |   |
| 10 |                                   |           | 1        |   |   |
| 11 | 10                                |           |          |   |   |
| 12 |                                   |           |          | 1 |   |
| 13 |                                   | 10        |          |   |   |
| 14 |                                   | 12        |          |   |   |
| 15 |                                   | 5         |          |   |   |

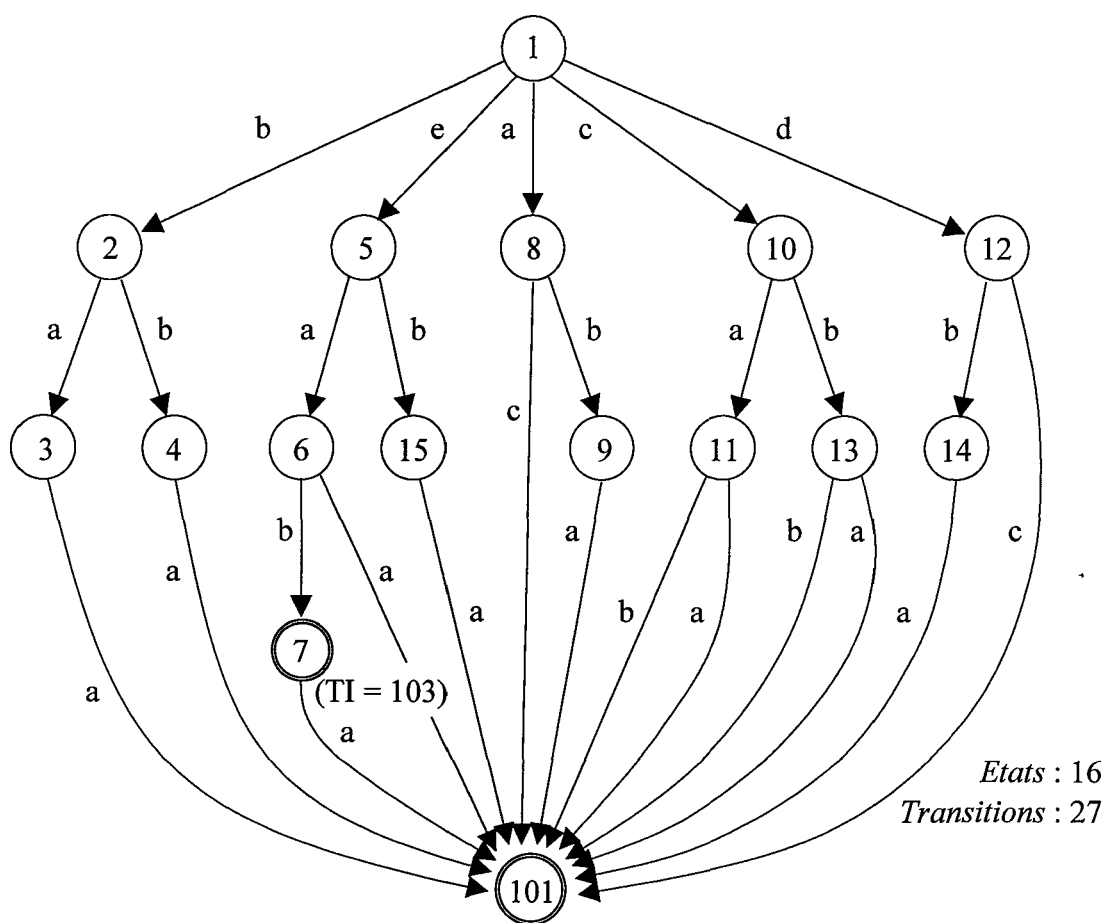
(2) [ La table des transitions *inverses* ]

| état   | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| lettre | b | a | b | e | a | b | a | b | c  | a  | d  | b  | b  | b  |
| but    | 1 | 2 | 2 | 1 | 5 | 6 | 1 | 8 | 1  | 10 | 1  | 10 | 12 | 5  |

(3) [La table des transitions inverses sous la forme modifiée]

|   | a  | b  | c  | d | e |
|---|----|----|----|---|---|
| 1 | 3  | 11 | 8  |   |   |
| 2 | 4  | 13 | 12 |   |   |
| 3 | 7  |    |    |   |   |
| 4 | 9  |    |    |   |   |
| 5 | 11 |    |    |   |   |
| 6 | 13 |    |    |   |   |
| 7 | 14 |    |    |   |   |
| 8 | 15 |    |    |   |   |
| 9 | 6  |    |    |   |   |

(4) [La base de données des états redondants]

Fig. 2.10 L'automate acyclique **déterministe** de *L*

## 2.2.2 Minimisation

La première étape de la minimisation consiste à trouver les états redondants dans l'automate. Nous avons introduit *la table des transitions inverse* dans ce but. Tous les états de l'automate, sauf l'état final, ont une seule transition entrante, puisque nous avons construit l'automate acyclique déterministe en utilisant le *Trie* comme structure de données. Ensuite, il faut tester si des états sont équivalents. Si des états sont équivalents, on ne conserve qu'une seule transition et on élimine les autres en modifiant la table des transitions.

Nous avons ajouté trois colonnes dans la table des transitions : une colonne pour la *hauteur* ( $H$ ), une colonne pour les *états terminaux intermédiaires* ( $TI$ ) et une colonne pour le *nombre de transitions sortant de l'état* ( $N$ ). Plus précisément,  $N(i)$  est le nombre d'éléments du sous-ensemble de l'alphabet  $\Sigma$  qui sont associés à un état  $i$ ,  $H(i)$  est la longueur du plus long chemin partant de l'état  $i$ , et aboutissant à un état terminal et si l'état  $i$  est un état terminal intermédiaire,  $TI(i)$  est le code terminal de l'état  $i$ . Si deux états,  $p$  et  $q$ , sont équivalents alors  $H(p) = H(q)$ ,  $TI(p) = TI(q)$  et  $N(p) = N(q)$ . Cependant ces trois conditions ne sont pas des conditions suffisantes pour que les états soient équivalents, mais ceci nous permet de tester l'équivalence plus facilement.

Soient deux états  $p$  et  $q$ . Ils sont équivalents si et seulement si pour toute lettre  $a$  de  $\Sigma$ , on a  $\delta(p,a) = \delta(q,a)$  et  $H(p) = H(q)$ .

On peut minimiser deux états qui sont équivalents, puisque les transitions associées à la même lettre partant de ces deux états conduit au même état but. Dans ce cas, cet état but plus de deux transitions entrantes. Ce résultat apparaît dans la table des transitions inverses. La case du tableau des transitions inverses, qui correspond à l'état but ci-dessus, contiendra plus de deux états. Par conséquent, si on veut chercher des états redondants, on peut les trouver facilement en examinant les cases du tableau des transitions inverses qui ont plus de deux états. Pendant la construction de l'automate, nous avons réuni tous les états terminaux de chaque chemin en un seul état final. Nous l'avons numéroté par '0' dans la table des transitions inverses. On peut commencer la minimisation à partir de l'état '0'.

On explique la méthode de minimisation sur un exemple. Si on regarde l'automate de la (Figure 2.10), on peut voir intuitivement que l'état '3', '4', '9', '14' et '15' sont équivalents. Mais l'état '7' n'est pas équivalent à ces cinq états, car c'est un état terminal intermédiaire. Il est pourtant associé à l'état '6' par la lettre 'a' comme les cinq états équivalents, mais sa hauteur est différente.

Dans un premier temps, on considère l'état '0' dans la table des transitions inverses. On peut obtenir un groupe d'états associés par la lettre 'a'; {3, 4, 7, 9, 13, 11, 14, 15, 6}, un autre groupe d'états associés par la lettre 'b'; {11, 13} et un autre groupe d'états associés par la lettre 'c'; {8, 12}. Pour tester la condition d'équivalence ;  $\delta(p,a) = \delta(q,a)$ , on regarde si les

états de chaque groupe satisfont au moins les trois conditions :  $H$ ,  $TI$  et  $N$ . Suivant ces conditions, le groupe associé à la lettre 'a' est séparé en quatre sous-groupes ; {3, 4, 9, 14, 15}, {11, 13}, {6} et {7}.

### A. Minimisation pour les états de hauteur 1

On commence par s'occuper des états de hauteur 1, ensuite des états de hauteur 2 et ainsi de suite. L'état '6' est traité à l'étape correspondant à la hauteur 2. L'état '7' ne peut pas être groupé avec {3, 4, 9, 14, 15}, parce qu'il est d'un type d'état différent. Les groupes {3, 4, 9, 14, 15} et {11, 13} sont distingués, parce que la valeur de  $N$  est différente entre les deux groupes. Le groupe associé par la lettre 'b' reste regroupé en {11, 13}.

|               | a              | b              | c   | d  | e | TI  | H            | N            |
|---------------|----------------|----------------|-----|----|---|-----|--------------|--------------|
| 1             | 8              | 2              | 10  | 12 | 5 |     | 4            | 1            |
| 2             | 3              | 4 → 3          |     |    |   |     | 2            | 2            |
| 3             | 101            |                |     |    |   |     | 1            | 1            |
| <del>4</del>  | <del>101</del> |                |     |    |   |     | <del>1</del> | <del>1</del> |
| 5             | 6              | 15 → 3         |     |    |   |     | 3            | 2            |
| 6             | 101            | 7              |     |    |   |     | 2            | 1            |
| 7             | 101            |                |     |    |   | 103 | 1            | 1            |
| 8             |                | 9 → 3          | 101 |    |   |     | 2            | 2            |
| <del>9</del>  | <del>101</del> |                |     |    |   |     | <del>1</del> | <del>1</del> |
| 10            | 11             | 13 → 11        |     |    |   |     | 2            | 2            |
| 11            | 101            | 101            |     |    |   |     | 1            | 2            |
| 12            |                | 14 → 3         | 101 |    |   |     | 2            | 2            |
| <del>13</del> | <del>101</del> | <del>101</del> |     |    |   |     | 1            | 2            |
| <del>14</del> | <del>101</del> |                |     |    |   |     | <del>1</del> | <del>1</del> |
| <del>15</del> | <del>101</del> |                |     |    |   |     | <del>1</del> | <del>1</del> |

|               | a                             | b             | c     | d | e |
|---------------|-------------------------------|---------------|-------|---|---|
| 0             | 3, 4, 7, 9, 11, 13, 14, 15, 6 | 11, 13        | 8, 12 |   |   |
| 1             |                               |               |       |   |   |
| 2             |                               | 1             |       |   |   |
| 3             | 2                             | 2, 8, 12, 5   |       |   |   |
| <del>4</del>  |                               | <del>2</del>  |       |   |   |
| 5             |                               |               |       |   | 1 |
| 6             | 5                             |               |       |   |   |
| 7             |                               | 6             |       |   |   |
| 8             | 1                             |               |       |   |   |
| <del>9</del>  |                               | <del>8</del>  |       |   |   |
| 10            |                               |               | 1     |   |   |
| 11            | 10                            | 10            |       |   |   |
| 12            |                               |               |       | 1 |   |
| <del>13</del> |                               | <del>10</del> |       |   |   |
| <del>14</del> |                               | <del>12</del> |       |   |   |
| <del>15</del> |                               | <del>5</del>  |       |   |   |

(1) [ La table des transitions ]

(2) [ La table des transitions inverses ]

Fig. 2.11 Les tableaux représentant la procédure de minimisation.

Le groupe associé par la lettre 'c' reste regroupé en {8, 12}, parce que sa hauteur est 2. Finalement à l'étape correspondant à hauteur 1, on teste la condition d'équivalence sur les deux groupes {3, 4, 9, 14, 15} et {11, 13}.

On considère les cinq états du groupe {3, 4, 9, 14, 15}, ils satisfont la condition d'équivalence. On décide de conserver l'état '3' et d'éliminer les autres. Les étapes sont les suivantes (Figure 2.11).

- (1) On élimine l'état '4' dans la table des transitions inverses. On déplace la case (4,b) à la case (3,b), puis on efface l'état '4'.
- (2) On obtient les indices (2,b) avec la table des transitions inverses pour accéder directement à l'état '4' de la transition (2,b,4) dans la table des transitions. Avec ces indices, on peut transformer rapidement la transition (2,b,4) en la transition (2,b,3).

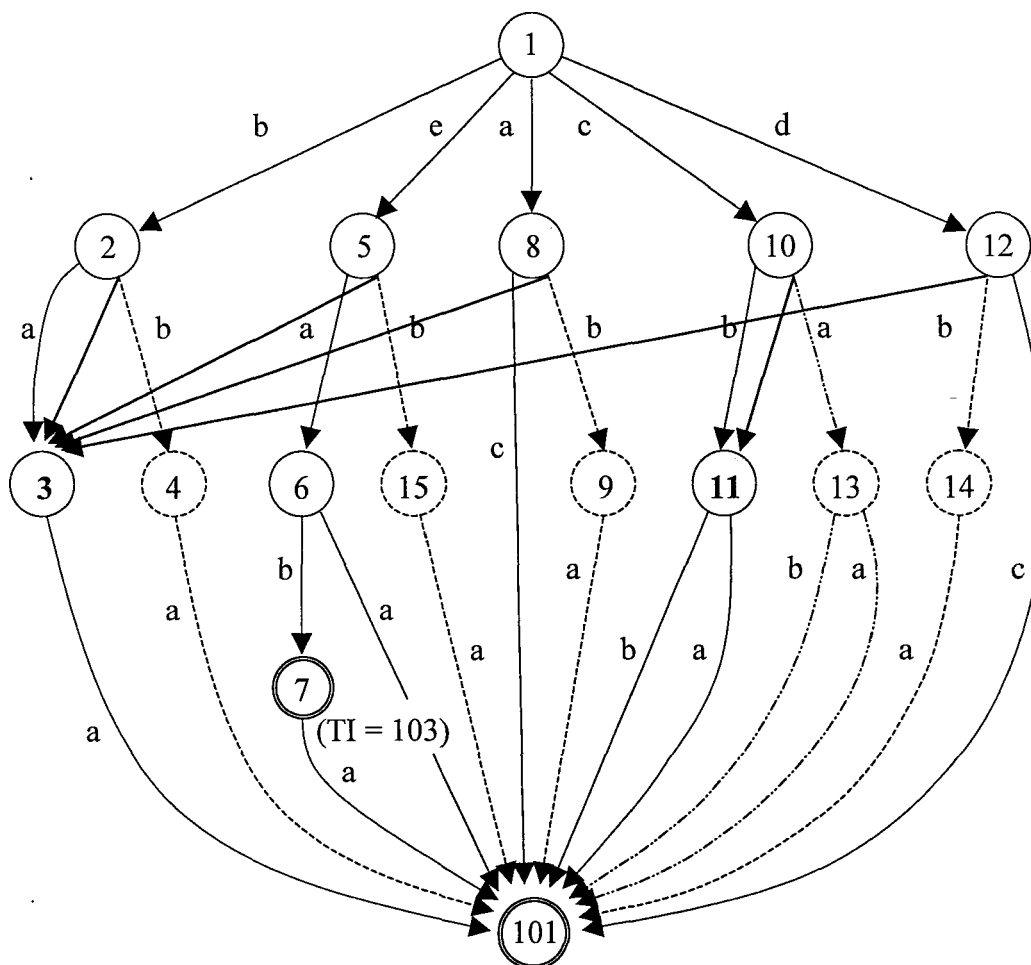


Fig. 2.12 Automate acyclique illustrant la procédure de minimisation.

- (3) On efface l'état '4' de la table des transitions.
- (4) On répète la même procédure pour les autres états du groupe {3, 4, 9, 14, 15}.
- (5) A la fin de procédure pour le groupe {3, 4, 9, 14, 15} on efface les états (3, 4, 9, 14, 15) dans la case (0,a) de la table des transitions inverses.

Pour le deuxième groupe {11, 13}, on conserve l'état '11' et on élimine l'état '13', car ils sont équivalents. On suit les mêmes étapes que précédemment.

- (1) On élimine l'état '13' dans la table des transitions inverses. On déplace la case (13,b) à la case (11,b), puis on efface l'état '13'.
- (2) On obtient les indices (10,b) avec la table des transitions inverses pour accéder directement à l'état '13' de la transition (10,b,13) dans la table des transitions. Avec ces indices, on peut transformer rapidement la transition (10,b,13) en la transition (10,b,11).
- (3) On efface l'état '13' de la table des transitions.

|               | a   | b            | c              | d                    | e | TI  | H            | N            |
|---------------|-----|--------------|----------------|----------------------|---|-----|--------------|--------------|
| 1             | 8   | 2            | 10             | <del>12</del> →<br>8 | 5 |     | 4            | 1            |
| 2             | 3   | 3            |                |                      |   |     | 2            | 2            |
| 3             | 101 |              |                |                      |   |     | 1            | 1            |
| 5             | 6   | 3            |                |                      |   |     | 3            | 2            |
| 6             | 101 | 7            |                |                      |   |     | 2            | 1            |
| 7             | 101 |              |                |                      |   | 103 | 1            | 1            |
| 8             |     | 3            | 101            |                      |   |     | 2            | 2            |
| 10            | 11  | 11           |                |                      |   |     | 2            | 2            |
| 11            | 101 | 101          |                |                      |   |     | 1            | 2            |
| <del>12</del> |     | <del>3</del> | <del>101</del> |                      |   |     | <del>2</del> | <del>2</del> |
|               |     |              |                |                      |   |     |              |              |

(1) [ La table des transitions ]

|               | a            | b                      | c                   | d            | e |
|---------------|--------------|------------------------|---------------------|--------------|---|
| 0             | <del>6</del> | <del>2, 8, 12, 5</del> | <del>8,</del><br>12 |              |   |
| 1             |              |                        |                     |              |   |
| 2             |              | 1                      |                     |              |   |
| 3             | 2            | 2, 8, 12, 5            |                     |              |   |
| 5             |              |                        |                     |              | 1 |
| 6             | 5            |                        |                     |              |   |
| 7             |              | 6                      |                     |              |   |
| 8             | 1            |                        |                     | 1            |   |
| 10            |              |                        | 1                   |              |   |
| 11            | 10           | 10                     |                     |              |   |
| <del>12</del> |              |                        |                     | <del>1</del> |   |

(2) [ La table des transitions *inverses* ]

Fig. 2.13 Les tables après minimisations pour les groupes de hauteur 1.

- (4) A la fin de procédure pour le groupe  $\{11, 13\}$ , on efface les états (11, 13) des cases (0,a) et (0,b) dans la table des transitions inverses.
- (5) Pour le groupe  $\{7\}$ , on efface l'état '7' de la case (0,a) dans la table des transitions inverses, parce qu'il est unique.

Nous montrons dans la (Figure 2.13) les deux tables obtenues après minimisations pour les groupes de hauteur 1.

### B. Minimisation pour les états de hauteur 2

A présent, les groupes de hauteur 1 sont minimisés. On minimise alors les groupes de hauteur 2. Dans la case (3,b) de la table des transitions inverses, on peut trouver les quatre nouveaux états candidats (2, 8, 12, 5) (Figure 2.13 (2)). Pour les groupes de hauteur 2, on trouve le groupe  $\{6\}$  associé à la lettre 'a', le groupe  $\{2\}$  associé aux lettres 'a' et 'b' et le groupe  $\{8, 12\}$  associé aux lettres 'c' et 'b'. Le groupe  $\{5\}$  est associé aux lettres 'a' et 'b' comme le groupe  $\{2\}$ , mais sa hauteur est différente. On déplace les états (2, 8, 12, 5) de la case (3,b) à la case (0,b) pour faciliter les explications.

|    | a   | b   | c   | d | e | TI  | H | N |
|----|-----|-----|-----|---|---|-----|---|---|
| 1  | 8   | 2   | 10  | 8 | 5 |     | 4 | 1 |
| 2  | 3   | 3   |     |   |   |     | 2 | 2 |
| 3  | 101 |     |     |   |   |     | 1 | 1 |
| 5  | 6   | 3   |     |   |   |     | 3 | 2 |
| 6  | 101 | 7   |     |   |   |     | 2 | 1 |
| 7  | 101 |     |     |   |   | 103 | 1 | 1 |
| 8  |     | 3   | 101 |   |   |     | 2 | 2 |
| 10 | 11  | 11  |     |   |   |     | 2 | 2 |
| 11 | 101 | 101 |     |   |   |     | 1 | 2 |

Fig. 2.14 Les tables représentant l'automate acyclique **minimal** de  $L$ .

Pour le groupe  $\{8, 12\}$ , on conserve l'état '8' et on élimine l'état '12', car ils satisfont la condition d'équivalence. On suit les même étapes que précédemment.

- (1) On élimine l'état '12' dans la table des transitions inverses. On déplace la case (12,d) à la case (8,d), puis on efface l'état '12'.
- (2) On peut obtenir les indices (1,d) avec la table des transitions inverses pour accéder directement à l'état '12' de la transition (1,d,12) dans la table des transitions. Avec ces indices, on peut transformer rapidement la transition (1,d,12) en la transition (1,d,8).
- (3) On efface l'état '12' de la table des transitions.

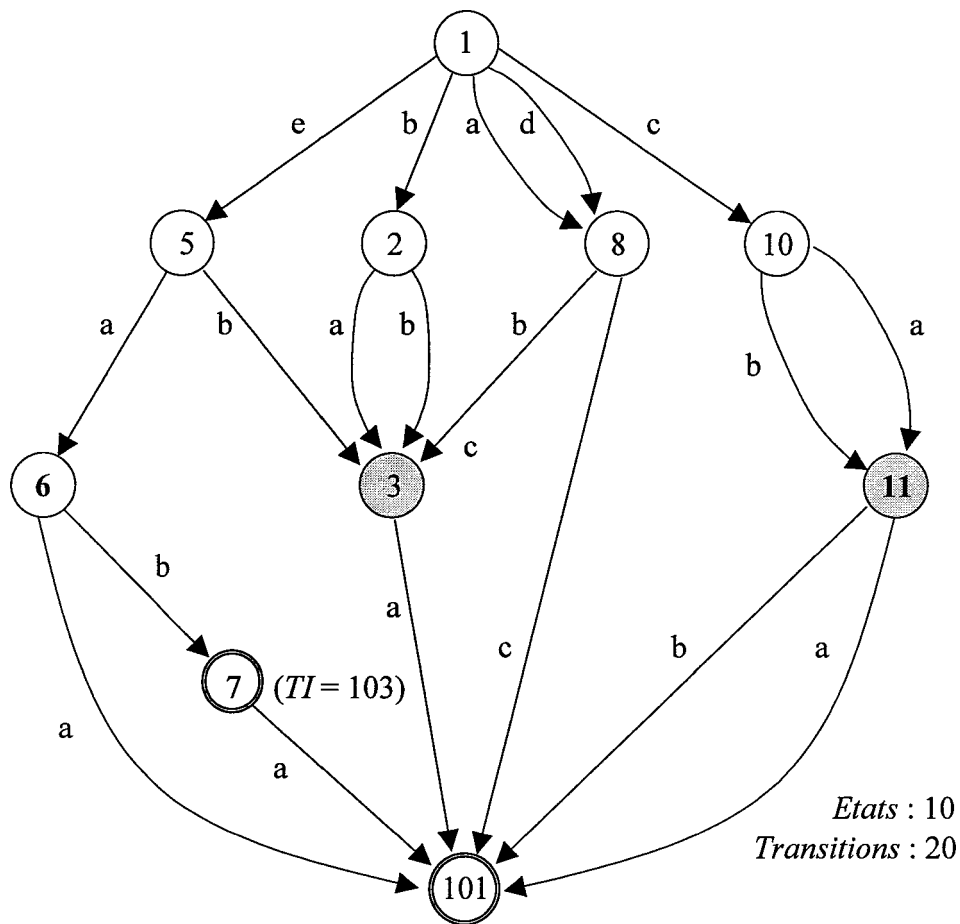


Fig. 2.15 L'automate acyclique **minimal** de  $L$

- (4) Pour finir la procédure pour le groupe  $\{8, 12\}$ , on efface les états (8, 12) dans la case de (0,d) de la table des transitions inverses.

Pour le groupe  $\{6\}$ , on efface l'état '6' dans la case (0,a) de la table des transitions inverses, car il est unique. On efface aussi l'état '2' dans la case (0,b).

### **C. Minimisation pour les états de hauteur 3**

Pour les groupes de hauteur 3, on efface l'état '5' du groupe  $\{5\}$ , parce qu'il est unique. La minimisation est alors finie, il n'y a en effet plus de case qui contienne au moins deux états. Il n'y a donc plus de candidat. Nous montrons les deux tables obtenues après minimisations dans la (Figure 2.14) et la (Figure 2.15).

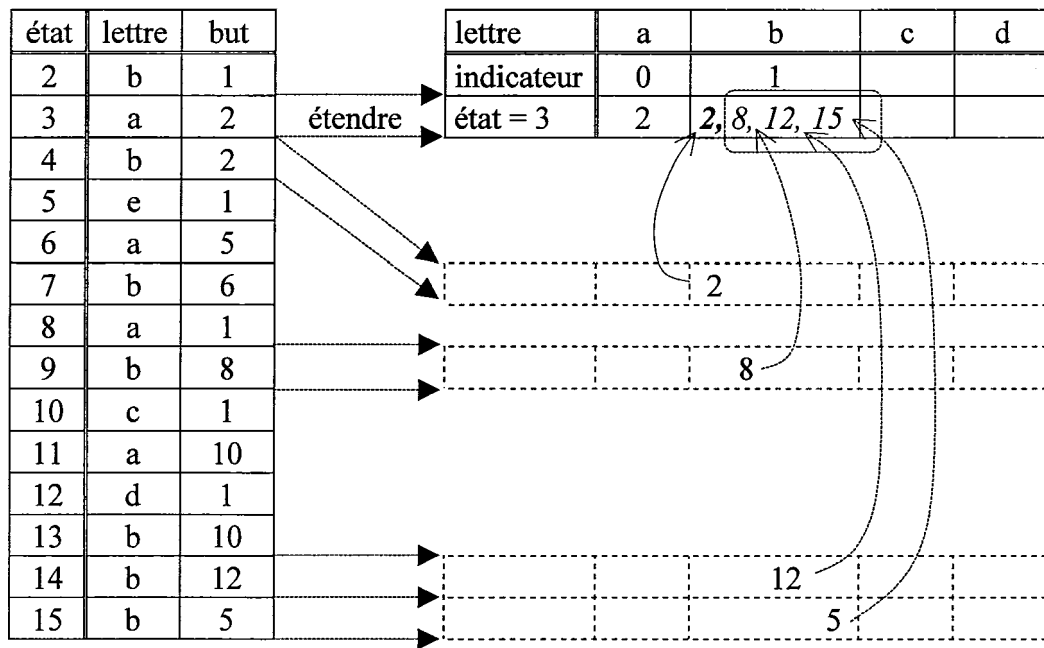
### **D. Minimisation avec les trois petits tableaux**

. En fait, nous avons utilisé trois petits tableaux au lieu de la grande table des transitions inverses. Ces trois tableaux sont le tableau de positionnements, le tableau de l'épreuve d'état redondant et le tableau de la base de données des états redondants. Le tableau de l'épreuve d'état redondant fonctionne comme indiqué sur la (Figure 2.16).

Par exemple, si l'on considère le groupe  $\{3, 4, 9, 14, 15\}$ , ces états doivent être fusionnés. On conserve l'état '3' et on élimine les autres. On obtient les indices (a,2) du tableau de positionnements avec l'état '3'. A ce moment, le tableau de l'épreuve est actualisé pour l'état '3'. On met d'abord l'état '2' dans la case (état,a) sur le tableau de l'épreuve. Ensuite, on obtient les indices (b,2) du tableau de positionnements pour l'état '4'.

Pour l'état '9', on met l'état '8' dans la case (état,b) du tableau de l'épreuve. Si la case (état,b) du tableau de l'épreuve était vide, on mettrait l'état '8'. Mais la case était occupée déjà par l'état '2' et l'indicateur correspondant (indicateur,b) est '0'. Dans ce cas, on ajoute les deux états '2' et '8' au tableau des états redondants et on met '1' dans la case (indicateur,b). Pour l'état '14' et '15', on ajoute les deux états '12' et '5' au tableau des candidats, parce que l'indicateur de la case (indicateur,b) est '1'.

De cette manière, on peut faire jouer le même rôle que celui de la table des transitions inverses aux trois petits tableaux. Quand il n'y a plus d'état dans le tableau des candidats, la minimisation est terminée.



[ le tableau de positionnements ]

[ le tableau de l'épreuve ]

|   | a             | b             | c  | d | e |
|---|---------------|---------------|----|---|---|
| 1 | <del>3</del>  | <del>11</del> | 8  |   |   |
| 2 | <del>4</del>  | <del>13</del> | 12 |   |   |
| 3 | <del>7</del>  | 2             |    |   |   |
| 4 | <del>9</del>  | 8             |    |   |   |
| 5 | <del>11</del> | 12            |    |   |   |
| 6 | <del>13</del> | 5             |    |   |   |
| 7 | <del>14</del> |               |    |   |   |
| 8 | <del>15</del> |               |    |   |   |
| 9 | 6             |               |    |   |   |



|   | a | b  | c  | d | e |
|---|---|----|----|---|---|
| 1 | 6 | 2  | 8  |   |   |
| 2 |   | 8  | 12 |   |   |
| 3 |   | 12 |    |   |   |
| 4 |   | 5  |    |   |   |

( Le table des candidats après  
minimisation pour les états  
de hauteur 1 )

[ le tableau des candidats ]

Fig. 2.16 Trois petits tableaux équivalant à la table des transitions inverses

2.2.3 Compression de la table des transitions

Nous compressons la table des transitions en utilisant la méthode que nous avons mentionnée dans (la section 1.4 du chapitre 1 de la deuxième partie). Nous transformons la table des transitions en une table unidimensionnelle : *Trans[]*. On considère la première transition  $(p,l,q)$  de la ligne  $p$  du tableau. Soit  $trou(p)$  l'indice de la première case vide dans la table *Trans[]* où les états de la ligne  $p$  peuvent se glisser sans chevauchement. Soit  $e\_indice(q)$  l'indice du premier état  $q$  dans la ligne  $p$ . On peut calculer la valeur  $r(p)$  correspondant au déplacement de la ligne  $p$  du tableau par la formule (2.1).

$$r(p) = trou(p) - e\_indice(q) + 1 \tag{2.1}$$

Pour adapter cette méthode de 'first-fit' au cas des automates. Les valeurs des  $r(p)$  doivent être distinctes, car sinon des états différents ayant la même valeur seraient fusionnés et apparaîtraient comme un seul état. Une entrée de la table *Trans[]* est un triplet (*lettre, déplacer, TI*) où *TI* représente un état Terminal Intermédiaire et *déplacer* prend la valeur de  $r(p)$ .

Exemple 2.3 : on considère le langage  $L = \{(baa,1), (bba,1), (eab,3), (eaba,1), (aba,1), (ac,1), (caa,1), (dc,1), (cba,1), (cab,1), (dba,1), (cbb,1), (eba,1)\}$

La construction de la table *Trans[]*

|            |     |     |      |     |     |     |     |       |     |     |    |    |    |
|------------|-----|-----|------|-----|-----|-----|-----|-------|-----|-----|----|----|----|
|            | 1   | 2   | 3    | 4   | 5   | 6   | 7   | 8     | 9   | 10  | 11 | 12 | 13 |
| $r(1) = 1$ | a,8 | b,2 | c,10 | d,8 | e,5 |     |     |       |     |     |    |    |    |
| $r(2) = 6$ |     |     |      |     |     | a,3 | b,3 |       |     |     |    |    |    |
| $r(3) = 8$ |     |     |      |     |     |     |     | a,101 |     |     |    |    |    |
| $r(5) = 9$ |     |     |      |     |     |     |     |       | a,6 | b,3 |    |    |    |

|              |       |     |       |     |       |      |      |       |       |    |    |    |    |
|--------------|-------|-----|-------|-----|-------|------|------|-------|-------|----|----|----|----|
|              | 11    | 12  | 13    | 14  | 15    | 16   | 17   | 18    | 19    | 20 | 21 | 22 | 23 |
| $r(6) = 11$  | a,101 | b,7 |       |     |       |      |      |       |       |    |    |    |    |
| $r(7) = 13$  |       |     | a,101 |     |       |      |      |       |       |    |    |    |    |
| $r(8) = 14$  |       |     |       | b,3 | c,101 |      |      |       |       |    |    |    |    |
| $r(10) = 17$ |       |     |       |     |       | a,11 | b,11 |       |       |    |    |    |    |
| $r(11) = 19$ |       |     |       |     |       |      |      | a,101 | b,101 |    |    |    |    |

Fig. 2.17 Le rangement des lignes de la table des transitions par la méthode de 'first-fit'

Nous illustrons cette méthode en utilisant l'automate de la (Figure 2.15). Nous associons des indices à chaque lettre :  $a = 0$ ,  $b = 1$ ,  $c = 2$ , ... etc. Au début, la table  $\text{Trans}[]$  est vide. On trouve  $\text{trou}(1) = 1$ ,  $e\_indice(8) = 1$  et  $l\_indice(a) = 0$  pour la première entrée (1,a,8) de la ligne 1 du tableau. On obtient donc  $r(1) = 1$ . On met (a,8) dans la case  $\text{Trans}[1]$  de la table, et les quatre états suivants dans les cases de  $\text{Trans}[2]$  à  $\text{Trans}[5]$ . Pour la ligne 2 du tableau, la première transition est (2,a,3). On obtient donc  $r(2) = 6$  avec  $\text{trou}(2) = 6$ ,  $e\_indice(3) = 1$  et  $l\_indice(a) = 0$ . On met (a,3) dans la case  $\text{Trans}[6]$  et (b,3) dans la case  $\text{Trans}[7]$ .

En continuant, on obtient, pour la ligne 8 du tableau,  $r(8) = 13$  selon la formule (1.1). Mais alors,  $r(8) = 13$  n'est pas unique, puisque  $r(7) = 13$  existe déjà. La valeur  $\text{trou}(8) = 14$  n'est donc pas valide. On essaye  $\text{trou}(8) = 15$  et obtient  $r(8) = 14$ . On continue cette procédure jusqu'à la dernière ligne de la table des transitions. On obtient la table  $\text{Trans}[]$  représentée sur la (Figure 2.17).

Ensuite, on remplace les numéros des états apparaissant dans les cases de la table  $\text{Trans}[]$  par les valeurs de la longueur  $r$  du déplacement calculée ci-dessus. On ajoute également l'information TI d'état intermédiaire. Autrement dit, dans la table les éléments (lettre, état) sont remplacés par (lettre,  $r(\text{état})$ ,  $TI(\text{état})$ ). Nous montrons la table finale de  $\text{Trans}[]$  dans la (Figure 2.18).

$\text{Trans}[k] = (\text{lettre}, \text{déplacer}, TI)$

| 0     | 1      | 2     | 3      | 4      | 5     | 6     | 7     |
|-------|--------|-------|--------|--------|-------|-------|-------|
| *,1,0 | a,14,0 | b,6,0 | c,17,0 | d,14,0 | e,9,0 | a,8,0 | b,8,0 |

| 8       | 9      | 10    | 11      | 12       | 13      | 14   |
|---------|--------|-------|---------|----------|---------|------|
| a,101,0 | a,11,0 | b,8,0 | a,101,0 | b,13,103 | a,101,0 | vide |

| 15    | 16      | 17     | 18     | 19      | 20      |
|-------|---------|--------|--------|---------|---------|
| b,8,0 | c,101,0 | a,19,0 | b,19,0 | a,101,0 | b,101,0 |

Fig. 2.18 La forme finale de la table  $\text{Trans}[]$  de la (Figure 2.17)

Pour lire les transition dans cette table, on procède de la manière suivante. Soit  $k_i$  l'indice de la table  $\text{Trans}[]$  pour la  $i^{\text{ème}}$  lettre  $l_i$  du mot entré et  $l\_indice(l_i)$  l'indice de la  $i^{\text{ème}}$  lettre  $l_i$ . On peut calculer l'indice  $k_{i+1}$  pour la lettre suivante  $l_{i+1}$  par

$$k_{i+1} = \text{Trans}[k_i].\text{déplacer} + l\_indice(l_{i+1}) \quad (2.2)$$

où  $\text{Trans}[k_i].\text{déplacer}$  représente le deuxième élément contenu dans le case  $\text{Trans}[k_i]$ . Pour la première lettre  $l_1$  du mot,  $\text{Trans}[k_0].\text{déplacer} = 1$ . L'ajout de la case  $\text{Trans}[0]$  indique en quelque sorte l'état initial.

On peut lire un mot dans l'automate représenté par la table  $\text{Trans}[]$  ci-dessus de la façon suivante. Pour une nouvelle lettre  $l_i$  du mot, on obtient l'indice  $k_i$  de la table  $\text{Trans}[]$  avec la formule (2.2). Si la lettre  $l_i$  et  $\text{Trans}[k_i].\text{lettre}$  sont identiques, alors la transition existe, sinon la transition n'existe pas.

*Exemple 2.4 :* On considère d'abord le mot (ada,1).

Pour la première lettre 'a' :  $k_1 = \text{Trans}[k_0].\text{déplacer} + l\_indice(l_1) = 1 + 0 = 1$   
 $\text{Trans}[1].\text{lettre} = a$  : Il existe donc une transition.

Pour la deuxième lettre 'd' :  $k_2 = \text{Trans}[k_1].\text{déplacer} + l\_indice(l_2) = 14 + 3 = 17$   
 $\text{Trans}[17].\text{lettre} = a$  : Il n'existe pas de transition.

Le mot (ada,1) n'appartient donc pas au langage reconnu.

*Exemple 2.5 :* On considère le mot (baa,1).

Pour la première lettre 'b' :  $k_1 = \text{Trans}[k_0].\text{déplacer} + l\_indice(l_1) = 1 + 1 = 2$   
 $\text{Trans}[2].\text{lettre} = b$  : Il existe donc une transition.

Pour la deuxième lettre 'a' :  $k_2 = \text{Trans}[k_1].\text{déplacer} + l\_indice(l_2) = 6 + 0 = 6$   
 $\text{Trans}[6].\text{lettre} = a$  : Il existe une transition.

Pour la dernière lettre 'a' :  $k_3 = \text{Trans}[k_2].\text{déplacer} + l\_indice(l_3) = 8 + 0 = 8$   
 $\text{Trans}[8].\text{lettre} = a$  : Il existe une transition.

Quand la dernière lettre du mot est atteinte, on peut savoir si l'état est terminal en regardant le prochain déplacement  $\text{Trans}[k_3].\text{déplacer}$ . La valeur de la référence était 100. Dans (l'Exemple 2.5),  $\text{Trans}[k_3].\text{déplacer}$  égal 101 qui est supérieur à la valeur de référence, cet état est donc un état terminal. Pour que le mot soit reconnu complètement par l'automate,

il faut encore vérifier le code associé au mot. On obtient ce code en soustrayant la valeur de la référence de  $\text{Trans}[k_3]$ . *déplacer*. Le mot (baa,1) est reconnu par l'automate.

## Chapitre 3

# Construction des Automates pour *DECOF*

Ce chapitre est consacré à la construction de l'automate pour le dictionnaire de formes fléchies du coréen (*DECOF*). Notre dictionnaire *DECOF* est établi à partir de deux dictionnaires séparés le *DECOS* et le *DECO-POST*. Une forme fléchie du coréen est obtenue en combinant un radical du *DECOS* et un suffixe flexionnel du *DECO-POST*. Une racine du *DECOS* ne peut pas se combiner directement avec un suffixe flexionnel. Certaines variations morphologiques aux frontières des suffixes et des radicaux ont lieu. De plus, ces variations nécessitent des opérations sur les phonèmes.

D'après ce qui est indiqué ci-dessus, on ne peut pas construire l'automate pour le *DECOF* de la même manière que pour le *DECOS*. Nous avons construit deux automates différents pour réaliser le *DECOF*. Nous avons engendré un nouveau dictionnaire des radicaux *DECO-RA* du *DECOS*. Dans notre travail, nous avons traité deux dictionnaires le *DECO-POSTA* et le *DECO-POSTV* parmi les *DECO-POSTs*. Nous avons remplacé les codes, d'information associés aux entrées du *DECO-POST* par les codes de combinaison. Nous avons aussi fusionné le *DECO-POSTA* et le *DECO-POSTV* en un seul dictionnaire de suffixes flexionnels. Nous avons obtenu un nouveau dictionnaire de suffixes flexionnels *DECO-FlexAV* résultant des deux opérations ci-dessus. Nous avons construit l'automate pour le *DECOF* à l'aide du *DECO-RA* et du *DECO-FlexAV*.

### 3.1 Organisation du dictionnaire *DECO-RA*

Combien d'entrées y a-t-il dans le dictionnaire des formes fléchies du coréen *DECOF*? Quel prédicat peut être combiné avec quelle série flexionnelle? Il n'y a pas d'information raisonnable sur les sujets jusqu'à présent. Selon ([Nam96c]) et ([Lee97]) dans le cas des verbes et des adjectifs, le nombre de combinaisons possibles serait d'environ 282 millions. Cette estimation est obtenue sans l'aire de restriction : le nombre de suffixes flexionnels des verbes est 20853 et celui des adjectifs est 18893. Habituellement, des restrictions sont faites, un prédicat peut alors être combiné avec environ 6000 suffixes flexionnels prédictifs. On obtient le nombre estimé suivant :

$$8763 \text{ (verbes)} \times 6000 + 5250 \text{ (adjectifs)} \times 6000 = 84078000$$

Mais, nous pouvons considérer que le nombre de combinaisons possibles est 148103215. Ce nombre n'est pas une estimation : ces combinaisons sont celles reconnues par les automates pour *DECOF* parmi les 1091136887 combinaisons possibles. Cependant, les 148 millions d'entrées que nous avons obtenues ne sont pas toutes des combinaisons de formes fléchies du coréen. Dans le dictionnaire *DECO-POST*, les entrées associées au code <POSTA/PA5> ou <POSTV/PV5>, se comportent de façon un peu particulière. Ce type d'entrées est formé d'un radical dont la syllabe finale est 'ha', il peut se combiner avec tous les suffixes flexionnels possibles. On pourra consulter [Nam96b] pour plus de détails. Si l'on considère ces types de combinaisons, les entrées du *DECOF* sont supérieures à 148 millions. Dans notre travail, nous n'avons pas traité ce type de combinaisons.

### 3.1.1 Extension du radical

Il s'agit de certaines variations morphologiques qui peuvent avoir lieu aux frontières des suffixes et de la racine. Pour représenter cette variation de la racine, le concept de *syllabe finale du radical* a été introduit. A titre d'exemple, considérons les variations morphologiques du verbe '나가다' (*nagada* sortir). Le radical est '나가' (*naga*). Nous montrons quelques formes fléchies dans ce qui suit.

Exemple 3.1 :

Pour la racine : '나가다' (*nagada*)

la forme fléchie = le radical + le verbe suffixe flexionnel : la syllabe finale du radical

- |     |                       |   |             |   |                    |   |           |     |             |
|-----|-----------------------|---|-------------|---|--------------------|---|-----------|-----|-------------|
| (1) | '나간지가'                | = | '나가'        | + | 'ㄴ지가'              | : | '가'       | --> | '간'         |
|     | <i>naganjiga</i>      |   | <i>naga</i> |   | <i>n-jiga</i>      |   | <i>ga</i> |     | <i>gan</i>  |
| (2) | '나갈수록'                | = | '나가'        | + | '르수록'              | : | '가'       | --> | '갈'         |
|     | <i>nagalsulog</i>     |   | <i>naga</i> |   | <i>l-sulog</i>     |   | <i>ga</i> |     | <i>gal</i>  |
| (3) | '나갔다가는'               | = | '나가'        | + | 'ss다가는'            | : | '가'       | --> | '갔'         |
|     | <i>nagassdaganeun</i> |   | <i>naga</i> |   | <i>ss-daganeun</i> |   | <i>ga</i> |     | <i>gass</i> |

La syllabe finale du radical est '가' (*ga*). Dans ce cas, certains phonèmes consonantiques sont ajoutés à la dernière syllabe. Dans le cas (1), le phonème 'ㄴ' (*n*) est ajouté à la syllabe finale du radical. Selon le caractère du suffixe flexionnel, le dernier phonème consonantique de la dernière syllabe est éliminé ou modifié. Pour l'adjectif '빨강다' (*bbalgahda* rouge), par exemple, le radical est '빨강' (*bbalgah*) et la syllabe finale

est ‘강’ (*gah*). Cette syllabe finale peut avoir cinq variations ; ‘강, 간, 갈, 가, 갓’ (*gah, gan, gal, ga, gaiss*).

Comme autres exemples, on considère les deux verbes ‘나가다’ (*nagada* **sortir**) et ‘내려가다’ (*nailyegada* **descendre**). Le radical du verbe ‘내려가다’ (*nailyegada*) est ‘내려가’ (*nailyega*). Les deux radicaux ont la même syllabe finale ‘가’ (*ga*). Ils ont également les même formes fléchies, ‘면서’, ‘더라도’, comme dans la (Figure 3.1).

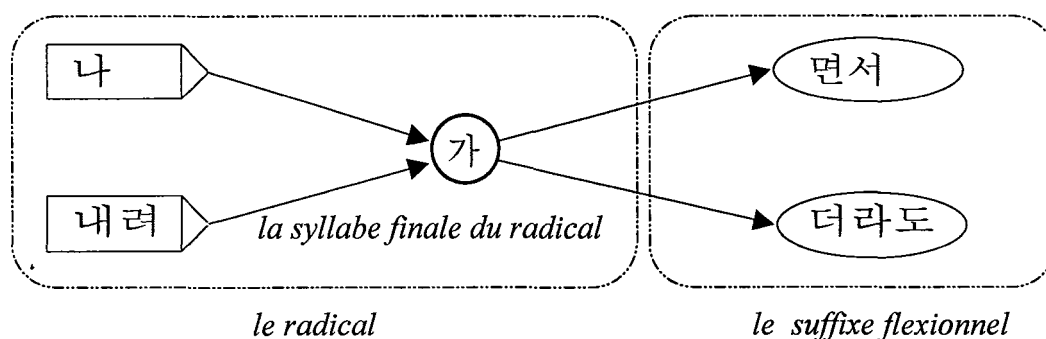


Fig. 3.1 Deux verbes dont les radicaux ont la même syllabe finale

Nous considérons des opérations sur les phonèmes de la syllabe finale du radical. Nous montrons un exemple d’opération sur les phonèmes ci-dessous dans la (Figure 3.2). Il faut modifier, éliminer ou ajouter un dernier phonème consonantique à la syllabe finale du radical pour faire la combinaison. Si la structure du système de codes du coréen était celle de l’alphabet latin, cette opération serait facile, on ajouterait ou en éliminerait simplement un octet du code. Les autres parties ne seraient pas changées. Dans le cas du coréen, le code *0xB0A1* représente la syllabe finale ‘가’ (*ga*). S’il est combiné avec un suffixe flexionnel, par exemple le phonème ‘ㄴ’ (*0xA4A4*), il devient ‘간’ (*gan*) et son code devient *0xB0A3*. C’est un code tout à fait différent.

Il n’y a pas d’informations phonématiques dans le système de code du coréen que nous utilisons dans ce travail. Nous avons déjà expliqué les problèmes et les caractéristiques de ce système de code syllabique dans (le chapitre 1 de l’Introduction). Grâce au système ‘*Hangul Code Manager (HANCoM)*’ que nous avons proposé dans (le chapitre 1 de la première partie), on pourrait faire ce type d’opération phonématique. Bien que l’on puisse utiliser ce système pour obtenir des informations phonématiques de syllabe, il n’est pas agréable pour construire un dictionnaire électronique par automates, en raison des surcharges.

Les syllabes finales du radical sont examinées une à une. Les types de syllabe finale du radical des verbes constituent 333 classes et ceux des adjectifs forment 151 classes. Dans

le dictionnaire *DECOF*, grâce à ces considérations, on peut classer les verbes et les adjectifs en plusieurs groupes selon le type de la syllabe finale du radical. Les radicaux qui ont la même syllabe finale se combinent alors avec les suffixes flexionnels qui appartiennent à un même groupe.

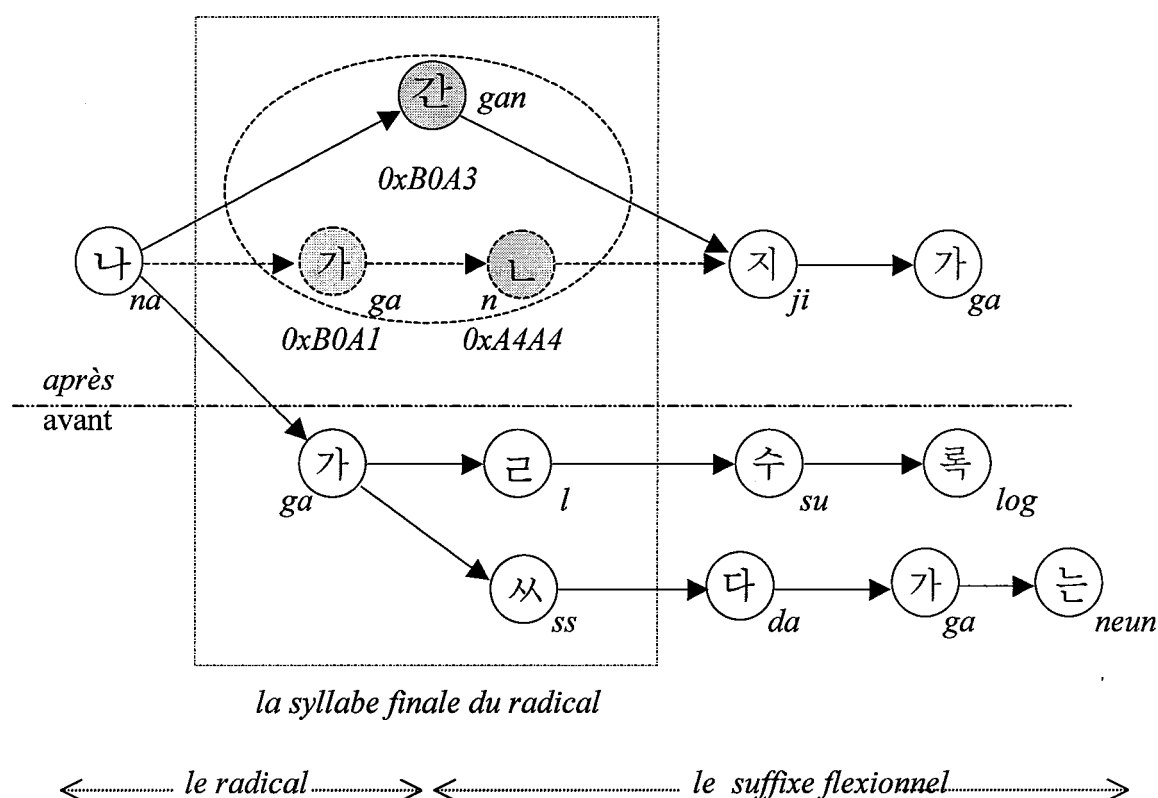


Fig. 3.2 Une opération sur les phonèmes de la syllabe finale du radical

Nous avons utilisé une autre solution pour le problème des opérations phonématiques. Les 333 classes de verbes et les 151 classes d'adjectifs ci-dessus représentent toutes les formes possibles de la syllabe finale des radicaux. Autrement dit, on peut trouver toutes les formes modifiées des radicaux. Si l'on utilise les différentes formes des radicaux comme entrées, il n'est pas nécessaire de faire des opérations sur les phonèmes. Par exemple, si l'on considère une racine du *DECOF* '나가다' (*nagada sortir*), son radical est '나가' (*naga*) et sa syllabe finale est '가' (*ga*). On peut trouver toutes les formes du radical '나가', ce sont les suivantes.

|     |    |                 |
|-----|----|-----------------|
| 나가다 | 나가 | VS./TRA/INT/VC. |
| 나가다 | 나간 | VS./TRA/INT/VC. |
| 나가다 | 나갈 | VS./TRA/INT/VC. |
| 나가다 | 나갔 | VS./TRA/INT/VC. |

On considère les quatre radicaux ci-dessus comme un groupe de radicaux associés à la racine '나가다'(nagada). En répétant cette procédure, nous avons obtenu environ 54 milles entrées de radicaux. Le nombre de radicaux des verbes augmente de 8761 à 35953 et celui des adjectifs augmente de 5250 à 18250.

### 3.1.2 Classification des radicaux

Nous avons classifié les radicaux que nous avons obtenus dans la section précédente (section 3.1.1 du chapitre 3 de la deuxième partie, page 90). Les radicaux qui ont la même syllabe finale sont évidemment rangés dans la même classe, parce qu'ils sont combinés avec les même classes de suffixes flexionnels. En plus, deux différentes classes de radicaux peuvent être regroupées si elles se combinent avec les mêmes classes de suffixes flexionnels. Dans la (Figure 3.3), nous montrons une classe de radicaux dont la syllabe finale diffère. Selon cette procédure, tous les radicaux sont groupés en 117 classes. Nous avons représenté ces classes par les codes suivants ; V-1-1, V-18-3, A-6-1, etc. Nous appelons ces codes : *les codes de combinaisons de radicaux*.

|       |     |             |       |      |             |
|-------|-----|-------------|-------|------|-------------|
| V-1-1 | 가라앉 | VS./INT/VC. | V-1-1 | 건어잡  | VS./TRA/VC. |
| V-1-1 | 가려잡 | VS./TRA/VC. | V-1-1 | 건잡   | VS./TRA/VC. |
| V-1-1 | 가로달 | VS./INT/VC. | V-1-1 | 건머말  | VS./TRA/VC. |
| V-1-1 | 가로막 | VS./TRA/VC. | V-1-1 | 건머잡  | VS./TRA/VC. |
| V-1-1 | 가로말 | VS./TRA/VC. | V-1-1 | 건터앉  | VS./INT/VC. |
| V-1-1 | 갈라말 | VS./TRA/VC. | V-1-1 | 것잡   | VS./TRA/VC. |
| V-1-1 | 갈앉  | VS./INT/VC. | V-1-1 | 건잡   | VS./TRA/VC. |
| V-1-1 | 갈앉  | VS./TRA     | V-1-1 | 건활   | VS./TRA/VC. |
| V-1-1 | 감   | VS./TRA     | V-1-1 | 고르잡  | VS./TRA/VC. |
| V-1-1 | 갈   | VS./TRA     | V-1-1 | 골라잡  | VS./TRA/VC. |
| V-1-1 | 갈   | VS./TRA     | V-1-1 | 곰삭   | VS./INT/VC. |
| V-1-1 | 거머안 | VS./TRA/VC. | V-1-1 | 곰살   | VS./TRA/VC. |
| V-1-1 | 거머잡 | VS./TRA/VC. | V-1-1 | 곰잡   | VS./TRA/VC. |
| V-1-1 | 건너막 | VS./TRA/VC. | V-1-1 | 구슬려살 | VS./TRA/VC. |

Fig. 3.3 Une classe de radicaux

### 3.1.3 Le dictionnaire DECO-RA

Maintenant, nous engendrons un dictionnaire des radicaux en ajoutant des codes de classes et des radicaux au DECOS. Nous l'appelons le *Dictionnaire Electronique CORéen des RADicaux*. A la place du DECOS, ce dictionnaire sera utilisé avec le DECO-FlexAV pour

construire les automates pour le *DECOF*. Nous montrons une partie du *DECO-RA* dans la (Figure 3.4).

---

|        |       |             |
|--------|-------|-------------|
| A-9-1  | 가공적이  | ADJS./CM    |
| A-9-2  | 가공적인  | ADJS./CM    |
| A-9-3  | 가공적일  | ADJS./CM    |
| V-17-1 | 가까와지  | VS./INT/JMA |
| V-17-2 | 가까와진  | VS./INT/JMA |
| V-17-3 | 가까와질  | VS./INT/JMA |
| V-17-4 | 가까와졌  | VS./INT/JMA |
| A-5-1  | 가까    | ADJS./RM    |
| A-5-2  | 가깝    | ADJS./RM    |
| A-5-1  | 가깝디가까 | ADJS./RM    |
| A-5-2  | 가깝디가깝 | ADJS./RM    |
| A-15-1 | 가깝하   | ADJS./HM    |
| A-15-2 | 가깝한   | ADJS./HM    |
| A-15-3 | 가깝할   | ADJS./HM    |
| A-15-4 | 가깝했   | ADJS./HM    |
| V-17-1 | 가꾸    | VS./TRA     |
| V-17-2 | 가꿀    | VS./TRA     |
| V-17-3 | 가꾼    | VS./TRA     |
| V-17-4 | 가꿔    | VS./TRA     |
| V-17-1 | 가꾸러뜨리 | VS./TRA     |
| V-17-2 | 가꾸러뜨린 | VS./TRA     |
| V-17-3 | 가꾸러뜨릴 | VS./TRA     |
| V-17-4 | 가꾸러뜨렸 | VS./TRA     |
| V-17-1 | 가꾸러지  | VS./INT/JM  |
| V-17-2 | 가꾸러진  | VS./INT/JM  |
| V-17-3 | 가꾸러질  | VS./INT/JM  |
| V-17-4 | 가꾸러졌  | VS./INT/JM  |

---

Fig. 3.4 Un extrait du *DECO-RA*

---

|         |         |
|---------|---------|
| 가 0     | 가꾸러졌 5  |
| 가공적이 1  | 가꾸러지 6  |
| 가공적인 2  | 가꾸러진 7  |
| 가공적일 3  | 가꾸러질 8  |
| 가까 4    | 가꾸로박혔 5 |
| 가까와졌 5  | 가꾸로박히 6 |
| 가까와지 6  | 가꾸로박힐 7 |
| 가까와진 7  | 가꾸로박 8  |
| 가까와질 8  | 가꾼 8    |
| 가깝 9    | 가꿀 7    |
| 가깝디가까 4 | 가꿔 5    |
| 가깝디가깝 9 | 가난하 10  |
| 가깝하 10  | 가난한 11  |
| 가깝한 11  | 가난할 12  |
| 가깝할 12  | 가난했 13  |
| 가깝했 13  | 가날뜨 14  |
| 가꾸 6    | 가날뜨 15  |
| 가꾸러뜨렸 5 | 가날뜨 16  |
| 가꾸러뜨린 6 | 가날뜨 17  |
| 가꾸러뜨릴 7 |         |
| 가꾸러뜨렸 8 |         |

---

Fig. 3.5 Un extrait des entrées de l'automate du *DECO-RA*

### 3.1.4 Les entrées de l'automate du *DECO-RA*

Pour construire un automate, il faut transformer les codes d'information en des codes numériques. Nous avons répété la procédure utilisée dans le cas du *DECOS* traité dans la (section 2.1.1 du chapitre 2 de la deuxième partie, page 63). Les homographes sont fusionnés et ces codes génèrent de nouveaux codes. Nous présentons un extrait des entrées de l'automate du *DECO-RA* dans la (Figure 3.5), et un extrait de la table des codes du *DECO-RA* dans la (Figure 3.6).

| <i>code</i> | <i>classe</i>              | <i>code</i> | <i>classe</i>                    |
|-------------|----------------------------|-------------|----------------------------------|
| 0           | V-19-1/V-7-1               | 127         | V-11-3/V-19-3                    |
| 1           | A-9-1                      | 128         | A-8-2/V-11-2/V-17-2/V-18-2       |
| 2           | A-9-2                      | 129         | A-8-3/V-11-3/V-17-3/V-18-3/V-4-2 |
| 3           | A-9-3                      | 130         | A-10-4/V-17-4                    |
| 4           | A-9-4                      | 131         | V-16-1/V-5-1                     |
| ...         |                            | 132         | V-5-2                            |
| 117         | V-13-1                     | 133         | A-14-1/V-19-1/V-7-1/V-8-1/V-9-1  |
| 118         | V-13-2                     | 134         | A-14-2/V-19-2/V-7-2/V-9-2        |
| 119         | V-13-3                     | 135         | A-14-3/V-19-3/V-7-3/V-8-2        |
| 120         | V-13-4                     | 136         | A-14-4/V-9-3                     |
| 121         | A-17-2/V-19-2              | 137         | V-11-1/V-13-1                    |
| 122         | A-17-3/V-19-3              | 138         | V-11-2/V-13-2                    |
| 123         | A-6-1/V-17-1               | ...         |                                  |
| 124         | A-8-1/V-11-1/V-17-1/V-18-1 | 230         | V-11-2/V-17-2                    |
| 125         | V-11-1/V-19-1              | 230         | V-11-3/V-17-3                    |
| 126         | V-11-2/V-19-2              |             |                                  |

Fig. 3.6 Un extrait de la table des codes du *DECO-RA*

## 3.2 Organisation du dictionnaire *DECO-FlexAV*

### 3.2.1 Classification des suffixes flexionnels

Les lexiques de suffixes flexionnels *DECO-POST* sont établis pour que l'on puisse engendrer les lexiques des formes fléchies *DECOF* à partir de *DECOS*. Dans notre travail, nous avons traité le *DECO-POSTV* et le *DECO-POSTA*. Les codes d'informations de classification sont associés à chaque entrée du *DECO-POST*. Ce sont, par exemple, \PV1, \PV2, \PV3, \PV4, \PV5, \PV6 et \PV7 pour les verbes et \PA1, \PA2, \PA3, \PA4, \PA5, \PA6 et \PA7 pour les adjectifs. Ces codes constituent le *code de combinaisons des suffixes flexionnels*.

```

도록 \PV1\PV2\PV3\PV4\PV5 \PV6 \PV7 .Conj
되 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj
든 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj
든말든 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj
든지 \PV1 \PV2 \PV3 \PV4 \PV5 \PV6 \PV7 .Conj
들 \PV1 .Conj

```

Fig. 3.7 Un extrait du *DECO-POSTV*

Ces codes contiennent des informations sur les combinaisons avec des radicaux. \PV1 indiquent que c'est un suffixe flexionnel qui peut être combiné avec des adjectifs et des verbes dont la syllabe finale a une voyelle claire. Par contre, \PV2 indiquent que c'est un suffixe flexionnel qui peut être combiné avec des adjectifs et des verbes dont la syllabe finale a une voyelle sombre.

Les combinaisons de suffixes sont nombreuses et s'écrivent sans espace typographique. Par exemple, les combinaisons de suffixes :

았, 겠, 다, 나, 던, 가  
*ess, geiss, da, nya, den, ga,*

sont représentées sous la forme d'un automate fini (Figure 3.8).

Comme nous le voyons dans la (Figure 3.9), plusieurs types de suffixes peuvent être attachés aux racines verbales. Les marqueurs de temps sont les suffixes présents, passés ou futurs ; les marqueurs d'aspect sont les suffixes inchoatifs, accomplis ou duratifs; les marqueurs de modalité sont les suffixes de probabilité ou de possibilité ; etc.

Nous pouvons trouver certaines régularités dans ce procédé de production. En regardant les premiers suffixes dans la (Figure 3.9), on peut savoir que plusieurs suffixes flexionnels commencent par la syllabe '았' (*ass*). Certains autres suffixes flexionnels commencent par la syllabe '겠' (*geiss*). Si les premières syllabes de deux suffixes flexionnels

sont différentes, ils peuvent se combiner avec différentes classes de radicaux, cependant ils ont le même code de combinaisons de suffixes flexionnels.

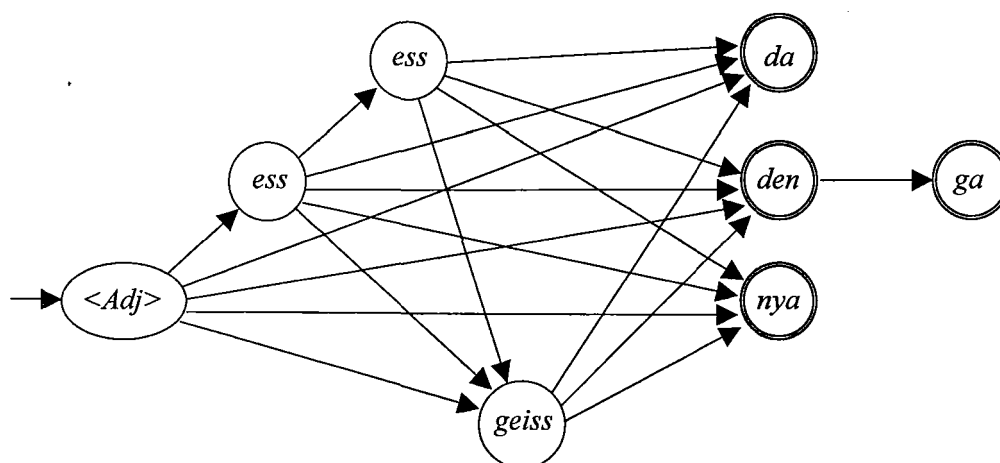


Fig. 3.8 Un automate fini de combinaisons des suffixes

L'automate de la (Figure 3.8) contient les 24 combinaisons suivantes :

|                            |                        |
|----------------------------|------------------------|
| <Adj>-ass-ess-da           | <Adj>-ass-geiss-da     |
| <Adj>-ass-ess-den          | <Adj>-ass-geiss-den    |
| <Adj>-ass-ess-den-ga       | <Adj>-ass-geiss-den-ga |
| <Adj>-ass-ess-nya          | <Adj>-ass-geiss-nya    |
| <Adj>-ass-ess-geiss-da     | <Adj>-da               |
| <Adj>-ass-ess-geiss-den    | <Adj>-den              |
| <Adj>-ass-ess-geiss-den-ga | <Adj>-den-ga           |
| <Adj>-ass-ess-geiss-nya    | <Adj>-nya              |
| <Adj>-ass-da               | <Adj>-geiss-da         |
| <Adj>-ass-den              | <Adj>-geiss-den        |
| <Adj>-ass-den-ga           | <Adj>-geiss-den-ga     |
| <Adj>-ass-nya              | <Adj>-geiss-nya        |

Fig. 3.9 Les 24 combinaisons des suffixes

Nous avons classifié les suffixes flexionnels avec les deux informations : la première syllabe et le code de combinaisons des suffixes flexionnels. Nous avons fusionné le *DECO-POSTA* et le *DECO-POSTV* en un seul dictionnaire pour construire les entrées. Pendant cette procédure, en raison des homographes de verbes et d'adjectifs, les codes de combinaisons de suffixes flexionnels sont transformés en d'autres codes. Nous avons classifié tous les

suffixes flexionnels en trois types. Le type *SfV* indique les suffixes flexionnels qui peuvent se combiner avec les radicaux de verbes seulement, et les suffixes flexionnels du type *SfA* qui peuvent se combiner avec les radicaux d'adjectifs. Les suffixes flexionnels qui peuvent se combiner avec les deux sont marqué par le type *SfU*. Chaque type de classes est séparé en plusieurs sous-classes plus précises. Le nombre de sous-classes de suffixes flexionnels est 93. Nous appelons ces 93 codes le *code de combinaisons des suffixes flexionnels*. Par exemple, les entrées classifiées par *SfU-1* sont les suivantes :

---

|                     |              |
|---------------------|--------------|
| 가 /TmInt;/Conj      | 뜻도 /Conj     |
| 가가 /Conj            | 뜻은 /Conj     |
| 가나 /Conj            | 뜻이 /Conj     |
| 가나마 /Conj           | 바 /Conj      |
| 가는 /Codis;/Conj     | 새 /Conj      |
| 가도 /Conj            | 즉 /Conj      |
| 가라도 /Conj           | 즉슨 /Conj     |
| 가마저 /Conj           | 지 /Conj      |
| 가마저도 /Conj          | 지가 /Conj     |
| 가만 /Conj            | 지나 /Conj     |
| 가만도 /Conj           | 지나마 /Conj    |
| 가만은 /Conj           | 지는 /Conj     |
| 가만이 /Conj           | 지도 /Conj     |
| 가만이라도 /Conj         | 지라 /Conj     |
| 가뿐만아니라 /Codis;/Conj | 지라도 /Conj    |
| 가뿐아니라 /Codis;/Conj  | 지마저 /Conj    |
| 가야 /Conj            | 지마저도 /Conj   |
| 가요 /TmInt           | 지만 /Conj     |
| 가조차 /Conj           | 지만도 /Conj    |
| 가조차도 /Conj          | 지만은 /Conj    |
| 가하 /Codis           | 지만이 /Conj    |
| 간 /Codis            | 지만이라도 /Conj  |
| 고 /TmInt            | 지뿐만아니라 /Conj |
| 데 /Conj             | 지뿐아니라 /Conj  |
| 데다가 /Conj           | 지야 /Conj     |
| 데도 /Conj            | 지조차 /Conj    |
| 들 /Conj             | 지조차도 /Conj   |
| 뜻 /Conj             |              |

---

Fig. 3.10 Les entrées classifiées par *SfU-1*

### 3.2.2 Etablissement des entrées du *DECO-FlexAV*

Nous avons classifié et codé les radicaux et les suffixes flexionnels. Maintenant, grâce à ces classifications, on peut représenter les relations de combinaison au niveau des classes. Par exemple, la classe de radicaux V-4-2 peut se combiner avec les classes de suffixes flexionnels *SfU2-3*, *SfU2-4*, *SfU2-5* et *SfV4-1*. Nous avons construit un tableau qui représente la relation de combinaisons entre des classes de radicaux et des classes de suffixes flexionnels. Un extrait de ce tableau est présenté dans la (Figure 3.11).

---

|       |                                                                                                                                                                                                                   |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| V-1-1 | /SfU1-5/SfU2-1/SfU2-2/SfU2-3/SfU2-4/SfU2-5/SfU3-1/SfU3-2/SfU3-3/SfU3-4/SfU3-5/SfU3-7<br>/SfU3-8/SfU3-12/SfU3-13/SfU3-14/SfU3-17/SfU3-18/SfU3-23/SfU3-24/SfU3-25/SfU3-26/SfU4-1<br>/SfV1-1/SfV1-3 /SfV3-1 /SfV4-1  |
| V-2-1 | /SfU1-5/SfU2-1/SfU2-2/SfU2-3/SfU2-4/SfU2-5/SfU3-1/SfU3-2/SfU3-3/SfU3-4/SfU3-5/SfU3-7<br>/SfU3-8/SfU3-12/SfU3-13/SfU3-14/SfU3-17/SfU3-18/SfU3-23/SfU3-24/SfU3-26/SfU4-1/SfV1-1<br>/SfV1-3/SfV3-1                   |
| V-3-1 | /SfU1-5/SfU1-6/SfU2-1/SfU2-2/SfU2-3/SfU2-4/SfU2-5/SfU3-1/SfU3-2/SfU3-4/SfU3-5/SfU3-7<br>/SfU3-8/SfU3-9/SfU3-10/SfU3-12/SfU3-13/SfU3-14/SfU3-15/SfU3-17/SfU3-21/SfU3-23/SfU3-24<br>/SfU4-1 /SfV1-1 /SfV1-3 /SfV4-1 |
| V-4-1 | /SfU1-5/SfU1-6/SfU2-1/SfU3-1/SfU3-2/SfU3-4/SfU3-5/SfU3-7/SfU3-8/SfU3-9/SfU3-10/SfU3-12<br>/SfU3-13/SfU3-14/SfU3-15/SfU3-17/SfU3-21/SfU3-23/SfU3-24/SfU4-1/SfV1-1/SfV1-3                                           |
| V-4-2 | /SfU2-2/SfU2-3/SfU2-4/SfU2-5/SfV4-1                                                                                                                                                                               |
| V-5-1 | /SfV6-1/SfV6-2/SfV6-3/SfV6-5/SfV6-6/SfV8-1                                                                                                                                                                        |
| V-5-2 | /SfU1-4/SfU1-5/SfU2-1/SfU3-1/SfU3-2/SfU3-3/SfU3-4/SfU3-5/SfU3-6/SfU3-7/SfU3-8/SfU3-9<br>/SfU3-10/SfU3-11/SfU3-12/SfU3-14/SfU3-15/SfU3-16/SfU3-17/SfU3-18/SfU3-26/SfU4-1/SfV1-1                                    |
| V-6-1 | /SfU1-5/SfU1-6/SfU1-9/SfU1-15/SfU1-16/SfU1-17/SfU1-18/SfU3-8/SfU3-12/SfV1-1                                                                                                                                       |
| V-6-2 | /SfU1-6/SfU1-9/SfU1-13/SfU2-3/SfU2-4/SfU2-5/SfU3-1/SfU3-2/SfU3-4/SfU3-5/SfU3-7/SfU3-14<br>/SfU3-17/SfU3-18/SfU3-21/SfU3-23/SfU3-26/SfU4-1                                                                         |

---

Fig. 3.11 Le tableau des relations de combinaisons entre les radicaux et les suffixes flexionnels

En regardant le tableau ci-dessus, on peut voir qu'une classe de suffixes flexionnels va se combiner avec plusieurs classes de radicaux, et qu'une classe de radicaux va se combiner avec plusieurs classes de suffixes flexionnels.

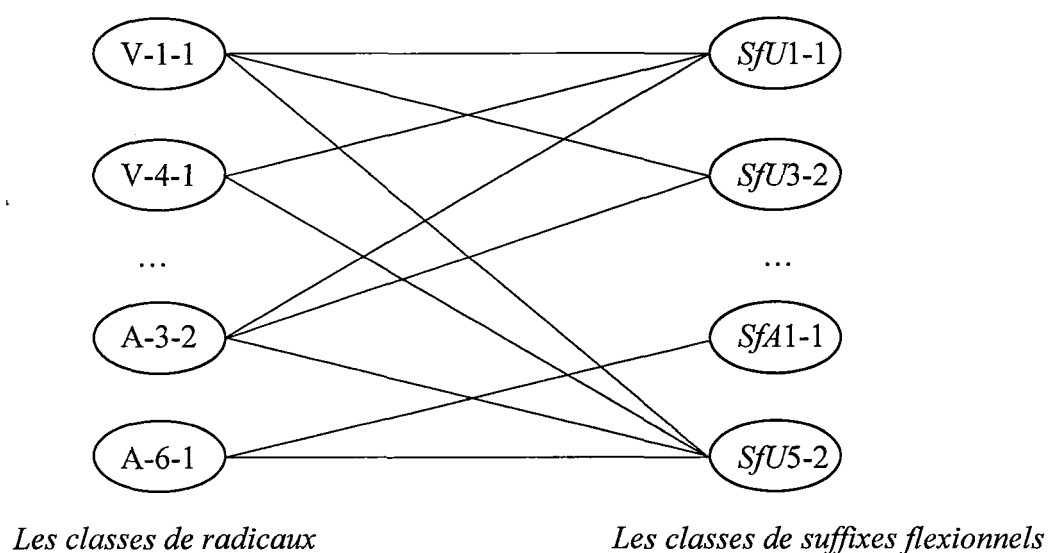


Fig. 3.12 Les relations de combinaisons entre les classes

Il faut que chaque entrée soit associée à un seul code numérique pour construire un automate. Chaque entrée du *DECO-RA* est codée par un seul code numérique. On doit maintenant coder aussi les entrées du *DECO-FlexAV*.

D'abord, on considère une entrée '지라도' (*jilado*) de la classe *SfU1-1*, cette entrée peut se combiner avec des radicaux de plusieurs classes de radicaux. Si on prend tous les codes numériques des radicaux comme codes de cette entrée '지라도' (*jilado*), on obtient :

'지라도' (*jilado*) \0 \2 \12 \14 \45 \87 \88 \109 \110 \120 \133 ...

On peut avoir une série de codes pour une entrée d'une classe de suffixes flexionnels. Chaque code ci-dessus indique la classe d'un radical. On répète cette procédure pour toutes les entrées des classes de suffixes flexionnels d'après le tableau des relations de combinaison de la (Figure 3.11). Le résultat de ce travail nous donne 72 différentes séries de codes de classes de radicaux. Pendant cette procédure, les homographes sont fusionnés et de nouveaux codes sont générés. Les entrées du *DECO-POSTV* (voir Figure 3.7) sont modifiées dans le *DECO-FlexAV* de la manière suivante :

---

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 지라도<br>( <i>jilado</i> )   | 103/104/11/112/114/118/119/12/121/122/126/127/128/129/134/135/138/139/141/144/<br>145/148/149/150/151/155/156/157/158/16/160/163/164/167/168/169/17/170/174/176/<br>177/179/180/182/183/184/185/187/188/191/192/194/195/196/198/199/2/20/202/204/<br>205/208/209/21/211/212/216/217/219/220/222/223/226/227/229/230/24/25/27/28/3/33/<br>34/40/43/44/47/48/51/52/56/57/61/62/64/65/68/69/7/72/73/76/77/8/81/82/84/85/86/88/<br>89/92/93/95/97/98   |
| 지마는<br>( <i>jimaneun</i> ) | 0/1/10/100/102/105/108/109/110/113/114/115/116/120/123/124/125/127/129/130/131/<br>132/133/135/136/139/14/141/142/143/149/15/151/154/157/158/159/160/161/162/166/<br>168/170/171/173/174/175/177/178/180/181/183/185/186/189/190/192/193/195/196/197/<br>200/201/203/205/207/209/210/215/217/218/22/220/221/224/225/228/23/230/25/26/30/<br>31/32/35/36/38/41/45/46/49/5/50/53/57/59/6/60/63/65/66/69/71/74/75/78/79/80/83/85/<br>9/90/94/95/96/99 |
| 지마저<br>( <i>jimaje</i> )   | 103/11/112/118/121/126/128/134/138/144/148/150/155/16/163/167/169/176/179/182/<br>184/187/191/194/196/198/2/20/204/208/211/216/219/222/226/229/24/27/33/43/47/51/<br>56/61/64/68/7/72/76/81/84/86/88/92/97                                                                                                                                                                                                                                         |
| 지마저도                       | 103/11/112/118/121/126/128/134/138/144/148/150/155/16/163/167/169 ...                                                                                                                                                                                                                                                                                                                                                                              |
| 지만                         | 0/1/10/100/102/103/105/108/109/11/110/112/113/114/115/116/118/120 ...                                                                                                                                                                                                                                                                                                                                                                              |
| 지만도                        | 0/1/10/100/102/103/105/108/109/11/110/112/113/114/115/116/118/120/121 ...                                                                                                                                                                                                                                                                                                                                                                          |

---

Fig. 3.13 Un extrait du *DECO-FlexAV*

On peut transformer ce code de combinaisons de suffixes flexionnels en code numérique comme dans le cas du *DECO-RA*. On construit un tableau dont les séries de codes sont les entrées. Après, on indice ce tableau par les classes de suffixes flexionnels, et chaque entrée mise sous forme de code numérique. Finalement, la liste des entrées de *DECO-FlexAV* que l'on utilise pour construire l'automate devient :

---

|        |    |
|--------|----|
| 지라도    | 38 |
| 지마는    | 71 |
| 지마저    | 0  |
| 지마저도   | 0  |
| 지만     | 70 |
| 지만도    | 70 |
| 지만은    | 70 |
| 지만이    | 0  |
| 지만이라도  | 0  |
| 지뿐만아니라 | 0  |
| 지뿐아니라  | 0  |
| 지야     | 70 |
| 지언정    | 2  |
| 지요     | 71 |
| 지조차도   | 0  |
| 진대     | 2  |
| 텐데     | 2  |
| 텐데도    | 2  |

---

Fig. 3.14 La liste des entrées du *DECO-FlexAV* pour construire l'automate

### 3.3 Automates pour *DECOF*

Pour les lexiques des formes fléchies (*DECOF*), nous avons construit deux automates, un pour les radicaux et un autre pour les suffixes flexionnels. Nous montrons le principe de l'automate du *DECOF* dans la (Figure 3.15). L'automate pour le *DECOF* se compose de trois parties. Ce sont l'automate du *DECO-RA*, l'automate du *DECO-FlexAV* et la *matrice de cartographie*. On utilise une méthode particulière pour savoir si l'automate pour le *DECOF* reconnaît une forme fléchie ou non. La forme fléchie est testée d'abord par l'automate du *DECO-RA*. Si l'automate du *DECO-RA* reconnaît une partie de la forme fléchie, le reste sera testé par l'automate du *DECO-FlexAV*. Si les deux parties, le radical et le suffixe flexionnel, de l'entrée sont reconnues par chaque automate, on doit alors tester si les deux peuvent se combiner. Pour cette épreuve, nous avons créé la *matrice de cartographie*.

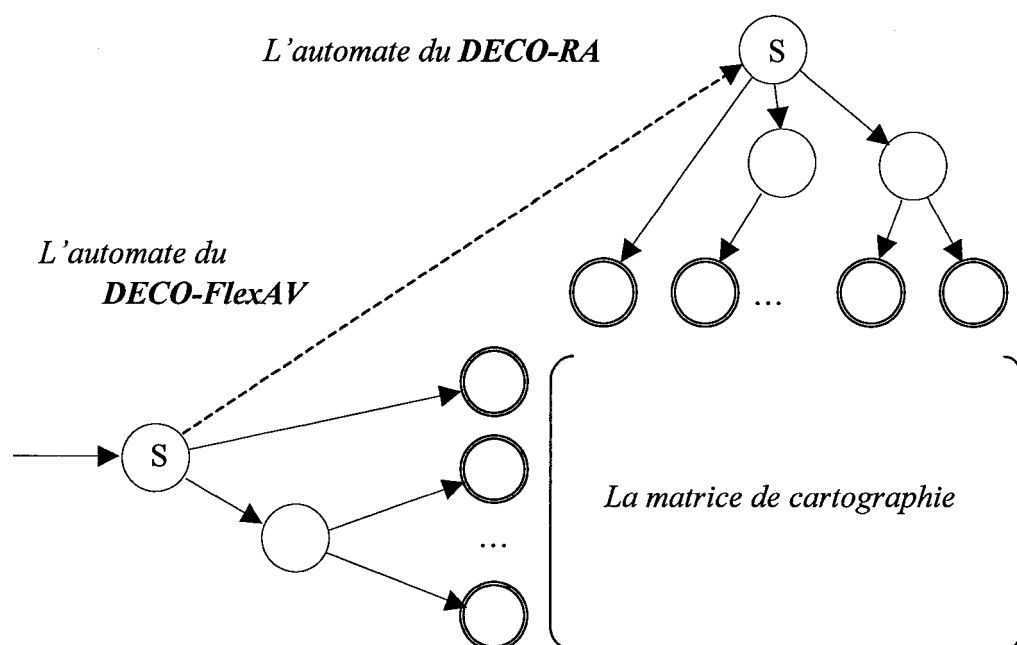


Fig. 3.15 Les automates pour le DECOF

### 3.3.1 Automates du DECO-RA et du DECO-FlexAV

La procédure de construction des automates pour le DECO-RA et le DECO-FlexAV est exactement la même que la procédure dans le cas du DECOS. On peut adapter les algorithmes utilisés pour le DECOS sans les modifier.

### 3.3.2 Matrice de Cartographie pour la Combinaison

Nous avons obtenu 72 différentes séries de codes de classes de radicaux comme codes de la combinaison. Nous avons établi la matrice de cartographie pour la combinaison à partir de ces informations. Par exemple, on considère l'entrée suivante du DECO-FlexAV :

‘지라도’ (*jilado*) \2 \3 \11 \12 \16 \17 \20 \48 \103 \118 \184 \204 ... conj

Dans la table des codes de suffixes flexionnels, on suppose que la série de codes ci-dessus correspond à l'indice '38', et cet indice est utilisé comme code d'entrée dans la liste que l'on utilise pour construire l'automate.

‘지라도’ (*jilado*) 38

Ce code '38' indique que l'entrée '지라도' (*jilado*) peut être combiné avec la radicaux des classes de la série des codes ci-dessus. De ce fait, si l'automate du *DECO-FlexAV* reconnaît l'entrée '지라도' (*jilado*), il émet un code '38'. En plus, si l'automate du *DECO-RA* reconnaît un radical, il émet un code, par exemple '48'. Ce code '48' existe dans la série de codes ci-dessus, on peut donc dire si cette forme fléchie a été reconnue par les automates pour le *DECOF*. Autrement dit, on peut décider de la possibilité de combinaison de ces deux codes.

On peut représenter la relation de combinaisons entre les radicaux et les suffixes flexionnels par un ensemble fini de paires  $(i, j)$ , dont l'élément  $i$  indique le code du suffixe flexionnel et l'élément  $j$  indique le code du radical comme ci-dessus. Par exemple le cas précédent, l'ensemble  $S$  est :

$$S = \{(38,2), (38,3), (38,11), (38,12), (38,16), (38,17), (38,20), (38,48), (38,103), (38,118), (38,204) \dots\}$$

On fabrique alors une matrice  $M = p \times r$ , dont les lignes  $p$  indiquent les codes de suffixes flexionnels et les colonnes  $r$  représentent les codes de radicaux. Dans ce cas, le nombre  $r$  est 231 et  $p$  est 72. Nous avons ainsi construit une matrice  $231 \times 72$  appelée '*la matrice de cartographie pour la combinaison*'. Nous avons utilisé 1 pour TRUE et 0 pour FALSE. On dit que TRUE indique une paire  $(i, j)$  qui est dans l'ensemble  $S$ , c'est-à-dire que la combinaison est possible, par contre FALSE indique que le radical et le suffixe flexionnel ne peuvent pas se combiner. Nous montrons un extrait de la matrice de cartographie dans la (Figure 3.16).

|                                    |   | les codes de radicaux |   |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |    |
|------------------------------------|---|-----------------------|---|---|---|---|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
|                                    |   | 0                     | 1 | 2 | 3 | 4 | ... |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |    |
| les codes des suffixes flexionnels | 0 | 0                     | 0 | 1 | 0 | 0 | 0   | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |    |
|                                    | 1 | 1                     | 1 | 0 | 0 | 0 | 1   | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  |
|                                    | 2 | 0                     | 0 | 0 | 1 | 0 | 0   | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | .. |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 0   | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 0   | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  |
|                                    |   | 1                     | 1 | 1 | 1 | 0 | 1   | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1  |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 1   | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  |
|                                    |   | 0                     | 0 | 0 | 0 | 0 | 0   | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1  |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 1   | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 0   | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | .. |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 0   | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1  |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 0   | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1  |
|                                    |   | 1                     | 1 | 0 | 0 | 0 | 0   | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  |
|                                    |   | 1                     | 0 | 0 | 0 | 0 | 0   | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  |
|                                    |   | ...                   |   |   |   |   |     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |    |

Fig. 3.16 Un extrait de *la matrice de cartographie*

### 3.3.3 Consultation sur les automates pour le *DECOF*

Les verbes et les adjectifs que l'on rencontre dans les textes coréens sont des formes fléchies et non des formes canoniques. De plus, il n'y a pas d'espace typographique dans les formes fléchies, mais on doit les découper pour identifier la partie correspondant au radical. Il est indispensable de faire une analyse automatique. Si l'on avait pu construire l'automate pour le *DECOF* avec les entrées sous formes entières, c'est-à-dire un seul automate, on aurait pu les consulter sur l'automate de la manière habituelle. A cause de la taille du dictionnaire *DECOF*, on n'a pas construit qu'un seul automate. On utilise donc une méthode particulière pour consulter un mot sur l'automate.

Quand on analyse une forme fléchie coréenne, généralement on utilise la méthode suivante. D'abord, on découpe la forme fléchie en plusieurs séries de combinaisons de radicaux et de suffixes flexionnels. On envisage toutes les possibilités de combinaisons. Si la longueur d'un mot est représenté par  $|w|$ , il existe  $(|w| - 1)$  de combinaisons possibles. Après, on les teste dans le dictionnaire jusqu'à trouver les bonnes combinaisons.

On considère la forme fléchie coréenne '굽신거리곶겠는가' (*gubsingelissyeyeissneunga*) : ( $|w| = 8$ ). En utilisant la méthode générale, on la découpe donc en 7 paires :

- (1) '굽' (*gub*) + '신거리곶겠는가' (*singelissyeyeissneunga*)
- (2) '굽신' (*gubsin*) + '거리곶겠는가' (*gelissyeyeissneunga*)
- (3) '굽신거' (*gubsinge*) + '리곶겠는가' (*lissyeyeissneunga*)
- (4) '굽신거리' (*gubsingeli*) + '곶겠는가' (*ssyegeissneunga*)
- (5) '굽신거리곶' (*gubsingelissyeye*) + '곶겠는가' (*geissneunga*)
- (6) '굽신거리곶곶' (*gubsingelissyeyeiss*) + '는가' (*neunga*)
- (7) '굽신거리곶곶곶' (*gubsingelissyeyeissneun*) + '가' (*ga*)

Fig. 3.17 Les découpages de la forme fléchie coréenne '굽신거리곶곶곶는가'

Pour trouver toutes les bonnes combinaisons, on doit la découper, puis on doit ensuite répéter 7 fois la même procédure de consultation des dictionnaires de radicaux et de suffixes flexionnels. Ici (1) et (4) sont les seuls bons radicaux, le suffixe flexionnel de (1) n'existe pas. Il en résulte que (4) est la bonne combinaison.

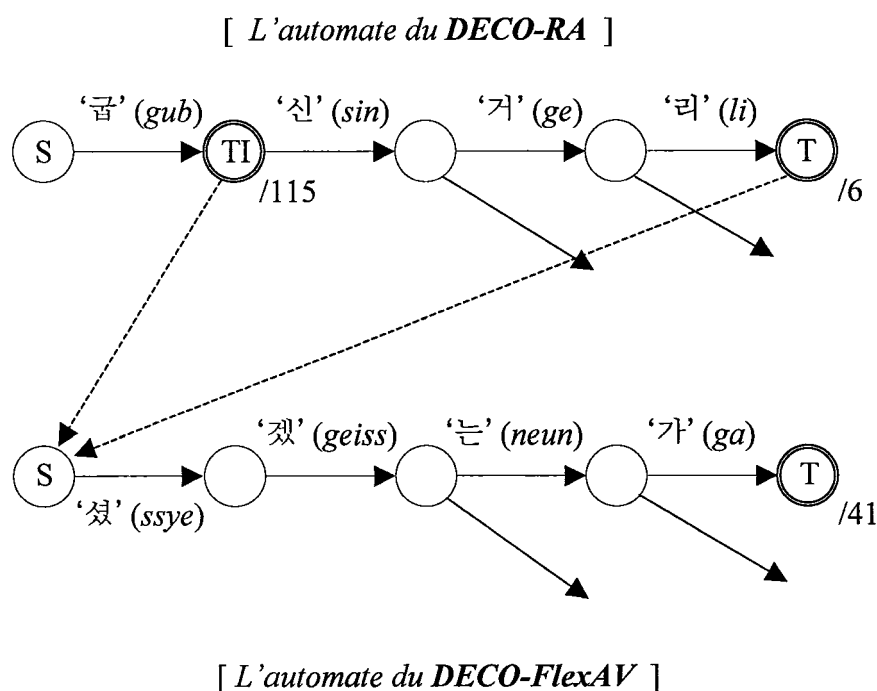
Avec nos dictionnaires *DECO-RA* et *DECO-FlexAV*, on peut obtenir ces résultats plus facilement et plus rapidement. En utilisant notre méthode, il n'est pas nécessaire de découper la forme fléchie avant la consultation. En trouvant le radical, le découpage est réussi automatiquement. Parmi les 7 radicaux ci-dessus, les bons radicaux sont '굽' (*gub*) et '굽신거리' (*gubsingeli*). Donc le dictionnaire *DECO-RA* ne contient que deux des radicaux de l'exemple. On montre les automates dans la (Figure 3.18). Quand la première syllabe '굽'

(*gub*) est lue, l'état atteint est un *état terminal intermédiaire*. L'automate du *DECO-RA* reconnaît ce radical '굽' (*gub*) et émet un code de radical (code 115). Le reste, à partir de la deuxième syllabe, devient le suffixe flexionnel candidat. La forme fléchie est découpée de la manière suivante ;

(1) '굽' (*gub*) + '신거리셨겠는가' (*singelissygeissneunga*)

Ensuite, on consulte l'automate du *DECO-FlexAV* pour le suffixe flexionnel. Dès que la première syllabe '신' (*sin*) est lue, le système de l'automate du *DECO-FlexAV* teste si la syllabe lue est valable. Comme on l'a mentionné à la (section 1.1 du chapitre 1 de la deuxième partie, page 58), seules 127 syllabes sont utilisées pour représenter le dictionnaire *DECO-FlexAV*. Dans cet exemple, la syllabe '신' (*sin*) n'est pas valable. C'est pour cette raison que le suffixe flexionnel n'est pas bon. On continue donc à consulter l'automate du *DECO-RA* pour la deuxième syllabe '신' (*sin*) de l'entrée. Après la lecture de la quatrième syllabe '리' (*li*), l'automate du *DECO-RA* reconnaît le radical '굽신거리' (*gubsingeli*) et émet un code de radical (code 6). Par suite, le deuxième découpage de la forme fléchie est le suivant :

(4) '굽신거리' (*gubsingeli*) + '셨겠는가' (*ssyegeissneunga*)



Le mot  $w = \text{'굽신거리셨겠는가'}$  (*gubsingelissygeiss neunga*)

Fig. 3.18 Un exemple d'automates pour le *DECOF*

On répète la même procédure pour le suffixe flexionnel. Cette fois, le suffixe flexionnel candidat est ‘췌췌는가’ (*ssyegeissneunga*). L’automate du *DECO-RA* reconnaît ce suffixe flexionnel et émet un code de suffixe flexionnel (code 41). Cette paire formée d’un radical et d’un suffixe flexionnel est reconnue par l’automate, on doit alors tester leur possibilité de combinaison.

La valeur de l’élément trouvé dans *la matrice de cartographie* avec les codes (41,6) est TRUE. Autrement dit la forme fléchie ‘굽신거리췌췌는가’ (*gubsingelissygeissneunga*) existe dans le dictionnaire *DECOF*, son radical est ‘굽신거리’ (*gubsingeli*) et son suffixe flexionnel est ‘췌췌는가’ (*ssyegeissneunga*).

Quand on trouve un radical, on peut décider si la procédure de consultation doit continuer ou non. Si l’état but est un *état terminal intermédiaire*, on doit continuer la procédure. Par contre, si l’état but est un *état final*, on arrête la procédure. Dans l’exemple ci-dessus, on arrête la procédure, parce que le deuxième état but était l’état final.

### 3.3.4 Quelques remarques sur les surcharges

Il est connu que la complexité de la consultation d’un mot  $w$  dans un automate est généralement en  $O(|w|)$  où  $|w|$  est la longueur du mot  $w$ . Dans le cas du *DECOF*, pour la raison que nous avons expliquée ci-dessus (dans la section 3.3.3), nous pouvons supposer que la complexité de la consultation de notre automate deviendra être supérieure à  $O(|w|)$ . En utilisant notre méthode, nous devons répéter plusieurs fois la même procédure de consultation pour un mot. Supposons que  $n$  indique le nombre maximum de découpages possibles d’une forme fléchie coréenne, on peut simplement décrire la complexité par  $O(n \cdot |w|)$ . Le nombre de découpages possibles  $n$  est décrit comme  $n = (|w| - 1)$ . Pourtant le nombre de combinaisons réelles que nous avons obtenu dans nos expériences est très inférieur à  $n$ . Nous considérons ce  $(n-1)$  comme une des surcharges.

Il existe des homographes, car les entrées sont établies par combinaison de radicaux et de suffixes flexionnels. Dans le cas habituel, ces homographes n’apparaissent pas. Ils sont représentés par une seule entrée et l’information concernant l’homographie est indiquée dans une partie du code grammatical. On considère que la présence de ces homographes est également une surcharge.

Nous avons introduit *la matrice de cartographie* pour tester la possibilité de combinaison. Cette matrice n’utilise pas l’automate général. Actuellement un élément de cette matrice est représenté par un octet et elle occupe 16.2 Ko de mémoire. Si on représente les éléments de cette matrice par des bits, elle occupe 2.04 Ko.

Nous avons fait des tests pour analyser les surcharges. Nous avons engendré toutes les combinaisons des radicaux du *DECO-RA* et des suffixes flexionnels du *DECO-FlexAV*. Les entrées que nous avons obtenues ont été testées sur les automates pour le *DECOF*. Nous avons mesuré les nombres d'occurrences pour les faits les plus importants. Nous avons divisé les résultats de nos tests en deux parties. La première partie représente les occurrences des toutes les combinaisons possibles, sans restriction du *DECOF*, c'est à dire tous les mots obtenus en associant tous les radicaux à tous les suffixes flexionnels, même si ce ne sont pas mots coréens. Les entrées des combinaisons reconnues par l'automate, sont décrites dans la deuxième partie. Nous les montrons dans la (Figure 3.19).

| éléments | Les descriptions concernant toutes entrées, sans restriction                                    | N° d'occurrences |
|----------|-------------------------------------------------------------------------------------------------|------------------|
| 1        | Nombre total de combinaisons dans le <i>DECOF</i>                                               | 1091136887       |
| 2        | Les entrées <i>non reconnues</i> par les automates pour le <i>DECOF</i>                         | 942818277        |
| 3        | Les entrées <i>reconnues</i> par l'automate pour <i>DECOF</i>                                   | 148318610        |
| 4        | La longueur moyenne d'un mot                                                                    | 9.24 (syllabes)  |
| 5        | Nombre d'accès à l'automate du <i>DECO-FlexAV</i> .                                             | 1885043440       |
| 6        | Moyenne du nombre d'accès à l'automate du <i>DECO-FlexAV</i> (N° total de découpages <i>n</i> ) | 1.728            |

| éléments | Les descriptions concernant les entrées <i>reconnues</i> par l'automate                   | N° d'occurrences |
|----------|-------------------------------------------------------------------------------------------|------------------|
| 3        | Les entrées <i>reconnues</i> par les automates pour le <i>DECOF</i>                       | 148318610        |
| 7        | Les entrées reconnues qui n'ont pas d'homographe                                          | 148081290        |
| 8        | Les entrées reconnues qui ont deux homographes                                            | 237320           |
| 9        | Les entrées reconnues qui ont trois homographes et de plus                                | 0                |
| 10       | Nombre d'accès sur l'automate du <i>DECO-FlexAV</i> .                                     | 260190929        |
| 11       | Nombre de cas si la 1 <sup>ère</sup> syllabe du suffixe flexionnel n'est pas valide.      | 73319559         |
| 12       | Nombre de cas où la combinaison impossible dans la <i>matrice de cartographie</i> .       | 846674           |
| 13       | Nombre total d'accès " <i>par erreur</i> " à l'automate du <i>DECO-FlexAV</i>             | 37468766         |
| 14       | Moyenne du nombre d'accès à l'automate du <i>DECO-FlexAV</i> (N° de découpages <i>n</i> ) | 1.754            |

Fig. 3.19 Les nombres d'occurrences pour les faits les plus significatifs.

La plus grande partie des entrées que l'on rencontre dans une application comme l'analyse d'un texte donné appartient au deuxième cas. On peut donc dire que le deuxième cas est en pratique beaucoup plus important que le premier. On discute le deuxième cas. Pour le nombre d'accès sur l'automate du *DECO-FlexAV*, même si on doit répéter la même procédure théoriquement  $n$  fois, la répétition n'excède pas 2 fois (voir l'élément 14 dans la Figure 3.19). La moyenne du nombre d'accès sur l'automate du *DECO-FlexAV* pour une entrée reconnue est 1.754 fois. On peut analyser cette surcharge en détails. Le nombre d'erreurs d'accès sur l'automate du *DECO-FlexAV* est la somme des lignes 11, 12 et 13. Il est de 111634999. Le cas 11 représente 65.68 % du total des erreurs. Il n'est pas considérable, car il n'est pas nécessaire d'accéder à l'automate pour tester ce type d'erreurs. Dans le cas 12, c'est une vraie surcharge. Pour tester ce genre d'erreurs, il faut continuer jusqu'à la fin du chemin de l'automate du *DECO-FlexAV*. Heureusement, ce genre d'erreurs est très peu fréquent et représente 0.76 % du total. Les surcharges considérables sont donc les cas 12 et 13. On peut constater que 14.73 % du nombre total d'accès à l'automate du *DECO-FlexAV* sont surcharge. Dans le premier cas, la moyenne du nombre d'accès à l'automate du *DECO-FlexAV* pour une entrée est 1.728 fois (voir l'élément 6 dans la Figure 3.19). Par conséquent, la surcharge concernant le nombre  $n$  de découpages possibles de la forme fléchie n'est pas très grave.

On considère la deuxième surcharge dues aux homographes. On peut trouver le nombre d'entrées qui ont des homographes dans l'élément 8. Ce nombre représente 0.16 % du total, ce cas est peu fréquent. Pour la *matrice de cartographie*, si l'on a besoin de réduire l'espace de mémoire, on peut représenter chaque élément de la matrice par un bit.

Les surcharges de la méthode que nous avons utilisée pour construire l'automate du dictionnaire *DECOF* sont acceptables.

### 3.4 Résultats sur les automates

Nous avons obtenu les automates des dictionnaires électroniques du coréen. La représentation d'un dictionnaire par automate est très efficace pour réduire l'espace occupé en mémoire et pour consulter rapidement un mot dans un dictionnaire. Dans le cas du coréen, l'automate du dictionnaire des mots simples *DECOS* ne réduit pas beaucoup l'espace utilisé. Par contre, la réduction de l'espace pour le dictionnaire des formes fléchies coréennes *DECOF* est remarquable. Par exemple, le *DECOS* contient 27150 entrées sur un alphabet de 1367 syllabes occupant 492.77 Ko de mémoire sous forme de textes. L'automate pour *DECOS* occupe 388.5 Ko de mémoire, soit 77 % de la taille du *DECOS* original. La réduction de l'espace pour le *DECOS* n'est pas très importante. Dans le dictionnaire *DECOS*, il n'existe pas beaucoup de préfixes et ni de suffixes communs. C'est la raison de la faible réduction de l'espace.

Dans le cas du *DECOF*, les entrées ne sont pas sous forme combinée. Les entrées sont établies par combinaison des radicaux du *DECO-RA* et des suffixes flexionnels du *DECO-FlexAV*. Si l'on engendre toutes les entrées du *DECOF* sous forme combinée, le *DECOF* contient 148.3 million d'entrées écrites sur un alphabet de 1375 syllabes. Il occuperait 2.55 Go de mémoire sous forme de textes. Grâce à notre méthode, nous avons pu représenter ces énormes entrées par deux automates et une matrice de cartographie qui n'occupent que 388.8 Ko de mémoire. Si l'on suppose que les automates sont construits à partir des entrées sous forme combinée, l'espace occupé par les automates pour le *DECOF* représente 0.0145 % de la taille totale des entrées. La représentation par automates pour le *DECOF* est plus de 6.8 mille fois plus petite. Les dictionnaires utilisés pour établir les entrées du *DECOF* sont le *DECO-RA* et le *DECO-FlexAV*. Ils occupent au total 5.0 Mo de mémoire sous forme de textes. On peut donc réellement dire que la représentation par automates du *DECOF* représente 0.76 % de la taille de la source. Nous montrons les résultats pratiques dans la (Figure 3.20).

|                                                                             | <i>DECOS</i> | <i>DECOF</i>   |                    |
|-----------------------------------------------------------------------------|--------------|----------------|--------------------|
|                                                                             |              | <i>DECO-RA</i> | <i>DECO-FlexAV</i> |
| Cardinal de l'alphabet ( syllabes)                                          | 1367         | 1364           | 127                |
| N° d'entrées dans la liste                                                  | 27150        | 52837          | 20651              |
| La taille sous forme des textes                                             | 492.77 Ko    | 1.44 Mo        | 3.56 Mo            |
| La taille sous forme d'automate                                             | 388.50 Ko    | 201.3 Ko       | 171.3 Ko           |
| N° de transitions de l'automate minimal                                     | 34259        | 29959          | 28368              |
| N° d'états de l'automate minimal                                            | 7828         | 5391           | 8766               |
| La taille maximum de la table Trans[]                                       | 55792        | 30705          | 22452              |
| Les cases non utilisées dans la table Trans[]                               | 23420        | 1761           | 55                 |
| Le temps moyen d'exécution pour la construction de l'automate minimal.      | 12 secondes  | 21 secondes    | 3 secondes         |
| Le temps moyen d'exécution pour la compression de la table des transitions. | 19 secondes  | 38 seconds     | 1 secondes         |

Fig. 3.20 Les résultats de l'expérimentation

Nous avons développé les programmes en *langage* C sous UNIX et les avons expérimentés sur la machine *HP9000/735* sous UNIX. Les temps pour l'exécution des programmes ont été mesurés en utilisant la fonction interne '*time()*' offert dans le *langage* C.

## Conclusion

· Pour le traitement automatique d'un lexique du coréen, les informations sur les phonématiques sont essentielles, puisque le coréen est une langue agglutinante. On utilise le système de code 'KSC5601-1987 *Hangul WANSUNG*' pour représenter le coréen sur l'ordinateur, mais ce système de codes ne fournit pas d'information sur les phonèmes. Nous avons élaboré le système '*Hangul Code Manager (HANCoM)*' pour pallier à ce manque. Ce système permet également les conversions entre les différents codes utilisés pour représenter le coréen en machine.

Nous avons étudié certaines caractéristiques statistiques les syllabes coréennes et les mots coréens. La longueur moyenne d'un mot coréen est d'environ 2.97 syllabes, le caractère le plus fréquent, *l'espace*, occupe 25.19 % de toutes les occurrences. Les syllabes qui apparaissent moins de dix fois sont au nombre de 2819 et représentent 41.58 % des occurrences totales. Au contraire, les 24 syllabes les plus fréquentes représentent 50 % des occurrences des syllabes du corpus, mais seulement 0.37 % des 6779 syllabes différentes. La plupart des 160 mots les plus fréquents sont des mots fonctionnels et représentent 18 % des mots du corpus. L'entropie au niveau des syllabes est de 6.884 bits pour des syllabes de monogrammes. L'entropie des digrammes est de 4.872 bits.

· Une des caractéristiques de la langue naturelle, la loi de Zipf a été appliquée au coréen. Nous avons proposé deux modèles, *le modèle modifié de distribution de Mandelbrot* et *le modèle des trois parties*, pour tester la loi de Zipf sur le coréen. Les textes coréens se conforment à une variante de la loi de Zipf, à quelques différences du modèle près. Le paramètre  $B$  dans *le modèle des trois parties* est  $B < 1$  dans les deux premières parties du modèle des trois parties, alors que le paramètre  $B$  dans la troisième partie est  $B > 1$  comme en français. Nous avons également vérifié si des types de la distribution de fréquence sont indépendantes de la nature ou de la taille du corpus. Nous avons pu constater qu'ils ne dépendent que de la langue même.

Plusieurs données statistiques de base sur les textes coréens ont été construites. Nous espérons que ces données contribueront au développement des dictionnaires et des systèmes électroniques de traitement de textes comme la compression des textes ou les systèmes de recherche documentaire.

Les dictionnaires électroniques d'un lexique du coréen sont représentés par automates acycliques minimaux. Une méthode utilisant *la table des transition inverses* a été introduite pour minimiser les automates acycliques déterministes. Pour les mots simples, le *DECOS* contient 27150 entrées sur un alphabet de 1367 syllabes occupant 492.77 Ko de mémoire sous forme de textes. L'automate pour le *DECOS* occupe 388.5 Ko de mémoire, soit 77 % de la taille du *DECOS* original. La réduction de l'espace mémoire utilisé pour le *DECOS* n'est pas très importante. Dans le cas des mots simples coréen, il n'existe pas beaucoup de préfixes et de suffixes communs. C'est la raison pour laquelle la réduction de l'espace mémoire utilisé est faible.

Pour les formes fléchies coréennes, deux nouveaux dictionnaire *DECO-RA* et *DECO-FlexAV* ont été élaborés pour engendrer comme entrées du *DECOF* à partir du système *DECO*. Nous avons inventé une méthode qui permet de représenter le *DECOF* par deux automates et une matrice de la cartographie. Le dernier n'occupe alors que 388.8 Ko mémoire. Si on engendre toutes les entrées du *DECOF* sous forme combinée, le *DECOF* contient 148.3 million d'entrées sur un alphabet de 1375 syllabes. Il occuperait 2.55 Go de mémoire sous forme de textes. L'espace mémoire occupé par les automates pour le *DECOF* correspond à 0.0145 % de la taille totale des entrées.

Les dictionnaires utilisés pour engendrer les entrées du *DECOF* sont le *DECO-RA* et le *DECO-FlexAV*. Si l'on tient compte de la taille de ces deux dictionnaires, qui occupent au total 5.0 Mo de mémoire sous forme de textes, l'espace mémoire occupé par les automates représente alors 0.76 % de la taille de la source. La réduction de l'espace dans la mémoire utilisée pour le *DECOF* est vraiment significative.

Cependant il existe des surcharges lors de la consultation d'un mot dans l'automate que nous avons construit. En laissant la comparaison avec un automate normal, nous avons considéré que ces surcharges étaient acceptables.

## Bibliographie

- [Aho76] Aho, V. Alfred & Hopcroft, E. John & Ullman, D. Jeffrey. (1976). *The design and analysis of computer algorithms*. Addison wesley.
- [Bel90] Bell, T.C., & Cleary, J.G., & Written, I.H. (1990). *Text Compression*. Englewood Cliffs N.J. Prentice Hall.
- [Bro92] Brown, F. Peter et al. (1992). An Estimation of an Upper Bound for the Entropy of English, *Association for Computational Linguistics* (vol. 18, number 1).
- [Bye89] Byeon, jung-young. (1989). A study on optimum *HANGUL* code for information processing in Korean, *Proceedings of 1st conference of HANGUL and Information processing in Korean*.
- [Cho94] Choi, Key-Sun et al. (1994). A Two-Level Morphological Analysis of Korean, *Proceedings of The 15th International Conference on Computational Linguistics* (COLING '94), Kyoto, Japan.
- [Cho96] Choi, Sung-Woo (1996). Hangul Code Manager (for KSC5601-1987 Wansung Code), *Memoires du CERIL* N°14. Université de Marne-la-Vallée, IGM, Université Paris 7. France.
- [Cho98] Choi, Sung-Woo (1998). Some statistical properties of Korean text corpus and Zipf's law, *Rapport IGM 98-07*. Université de Marne-la-Vallée, France.
- [Cle93] Clemenceau, David. (1993). Structuration du lexique et reconnaissance de mots dérivés, *Thèse de Doctorat*. Université Paris 7. France.
- [Cro94] Crochemore, Maxime et Rytter, Wojciech. (1994). *Text Algorithms*. Oxford University Press.

- [Dac98] Daciuk, Jan. (1998). Incremental Construction of Finite-State Automata and Transducers, and their Use in the Natural Language Processing, *Thesis of Ph.D.* Technical University of **Gdańsk**.
- [Gra94] Graham, R.L., & Kunth, D.E., & Patashnik, O. (1994). *Concrete Mathematics* (pp.272-278). MA: Addison-Wesley
- [Gro87] Gross, Maurice, (1987). The use of finite automata in the lexical representation of natural language, *lecture Notes in Computer Science 377*. Springer-Verlag.
- [Gro89] Gross, Maurice, (1989). La construction de dictionnaires électroniques, *Annales des Télécommunications*, tome 44 N° 1:2, Issy-les-Moulineaux / Lannion : CNET.
- [Han96] Han, Y. S., & Park, H. R., & Sin, J. H., & Choi, K. S. (1996). An Upper Bound Estimate for the Entropy of Korean Texts, *Literary and Linguistic Computing* (vol. 11, No. 3) (pp. 141-146). Oxford University press.
- [Jun93] Jun, Gyung-Hun. (1993). An implementations of Chinese characters to *HANGUL* converter. *Rapport KAIST*. Korea.
- [Ker78] Kernighan, W. Brian & Ritchie, M. Dennis. (1978). *The C programming language*. Prentice-Hall software series.
- [Kim89] Kim, Chung-heui. (1989). Problems of *KS-WANSUNG* code, *Proceedings of 1st conference of HANGUL and Information processing* in Korean.
- [Kun73] Kunth, E. Donald. (1973). Sorting and Searching, *The art of Computer Programming* vol.3. 2nd edition. Addison Wesley.
- [Lap88] Laporte, Éric. (1988). Méthodes algorithmiques et lexicales de phonétisation de textes, Applications au français, *Thèse de doctorat*. Université Paris 7. France.
- [Lee97] Lee, Chang-Yeol. (1997). La construction de lexiques de formes fléchies et l'analyse morphologique du coréen, *Thèse de doctorat*. Université Paris 7. France.
- [Lia83] Liang, Franklin Mark. (1983). Word Hy-phen-a-tion by Com-put-er, *Thesis of Ph.D.* Stanford University. USA.
- [LKS82] Laboratory of Korean Standard. (1982). *KSC5601-1987 WANSUNG code table*.

- [Man52] Mandelbrot, B. (1952). An information theory of the statistical structure of language, *Symposium on Applications of Communication Theory* (Proc. September) (pp.486-500). Butterworth London.
- [Mic97] Sipser, Michael. (1997). *Introduction to the Theory of Computation*. PWS Publishing Company.
- [Moh93] Mohri, Mehryar. (1993). Analyse et représentation par automates de structures syntaxiques composées - Application aux complétives, *Thèse de doctorat*, Université Paris 7.
- [MS89] Monthly MicroSoftware. (1989). Problems and solution methodologies in HANGUL code, *Monthly magazine*. MicroSoftware Inc.
- [Nam94] Nam, Jee-Sun. (1994). Dictionnaire des noms simples du coreen, *Rapport technique*, N-46, LADL, University Paris 7.
- [Nam96a] Nam, Jee-Sun. (1996,a). Dictionary of Korean Simple Verbs, *Rapport technique*, N-49, LADL, University Paris 7.
- [Nam96b] Nam, Jee-Sun. (1996,b). Dictionary of Noun-Postpositions and Predicate-Postpositions in Korean: *DECOS-PostN / DECOS-PostA / DECOS-PostV*, *Rapport technique* N- 51, LADL, Université Paris 7.
- [Nam96c] Nam, Jee-Sun. (1996,c). Construction of the Korean electronic Lexical system DECO, *Rapport Technique* (IGM96-26). Université de Marne-la-Vallée.
- [Par96] Park, Se-Young et al. (1996). Corpus based on Korean News Papers. ETRI. Korea.
- [Per85] Perrin, Dominique et Berstel, Jean. (1985). *Theory of Codes*. Academic Press, Inc.
- [Per89] Perrin, Dominique. (1989). Automates et algorithmes sur les mots, *Annales des Télécommunications*, tome 44 N 1:2, Issy-les-Moulineaux, Lannion : CNET.
- [Rev91] Revuz, Dominique. (1991). Dictionnaires et Lexiques : Méthodes et algorithmes, *Thèse de doctorat*, Université Paris 7.
- [Roc93] Roche, Emmanuel. (1993). Analyse syntaxique transformationnelle du français par transducteurs et lexique-grammaire, *Thèse de doctorat*. Université Paris 7.
- [Roc97] Roche, Emmanuel et al. (1997). *Finite-State Language Processing*. The MIT Press.
- [Sén95] Sénellart, Jean. (1995). Outil statistique et Index, *Ecole polytechnique mémoire de fin d'étude*.

- [Sén96] Sénellart, Jean. (1996). Statistique Prudence, *Actes des premières journées INTEX* (pp. 95-101). LADL.
- [Sil93] Silberztein, Max. (1993). *Dictionnaires électroniques et analyse automatique de textes - le système INTEX*. Masson. Paris.
- [Sha48] Shannon, Claude E. (1948). A mathematical theory of communication, *The Bell system technical journal* (XXVII) (pp. 379-423, 623-656).
- [Sha51] Shannon, Claude E. (1951). Prediction and Entropy of Printed English, *The Bell system technical journal* (pp. 50-64). January.
- [Tar79] Tarjan, Robert Endre & Yao, Andrew Chi-Chih. (1979). Sorting a sparse table, *Communication of the ACM*, 22(11) (pp. 606-611). November.
- [Won86] Won, Kwang-Ho. (1986). *This is the HANGUL*. Samjungdang press. Korea.
- [Won76] Wong, K. L. & Poon, R. K. L. (1976). A comment on the entropy of the Chinese language, *IEEE Trans. Acoustics, Speech, and Signal Processing* (pp. 583-585). December.
- [Zipf49] Zipf, G.K. (1949). *Human behavior and the principle of least effort*. MA: Addison-Wesley.

## Annexes

1. Le système de code *JOHAB de KSC5601-1992* et *JOHAB de COMMERCIALE*
2. *CIP*
3. Les tableaux *WAN – JCOM* et *JCOM – WAN*
4. *ISP2 et ICC*
5. Un extrait des fréquences de syllabes ( les 600 les plus fréquentes )
6. Un extrait des fréquences de mots ( les 600 les plus fréquentes )

Annexe 1. Le système de code *JOHAB de KSC5601-1992* et *JOHAB de COMMERCIALE*[ *JOHAB de KSC5601-1992* ]

|    | bits  | Cho-seong | Jung-seong | Jong-seong |
|----|-------|-----------|------------|------------|
| 0  | 00000 |           |            |            |
| 1  | 00001 |           | *          | *          |
| 2  | 00010 |           | ㅏ a        | ㅑ g        |
| 3  | 00011 |           | ㅓ ai       | ㅕ gg       |
| 4  | 00100 |           |            | ㅗ gs       |
| 5  | 00101 |           | ㅛ ya       | ㅜ n        |
| 6  | 00110 |           | ㅝ yai      | ㅠ nj       |
| 7  | 00111 |           | ㅟ e        | ㅡ nh       |
| 8  | 01000 |           |            | ㅓ d        |
| 9  | 01001 | *         | ㅑ ei       | ㅓ l        |
| 10 | 01010 | ㅑ g       | ㅓ ye       | ㅓ lg       |
| 11 | 01011 | ㅕ gg      | ㅕ yei      | ㅓ lm       |
| 12 | 01100 | ㅜ n       |            | ㅓ lb       |
| 13 | 01101 | ㅓ d       | ㅓ o        | ㅓ ls       |
| 14 | 01110 | ㅕ dd      | ㅓ wa       | ㅓ lt       |
| 15 | 01111 | ㅓ l       | ㅓ wai      | ㅓ lp       |
| 16 | 10000 | ㅓ m       |            | ㅓ lh       |
| 17 | 10001 | ㅓ b       | ㅓ oi       | ㅓ m        |
| 18 | 10010 | ㅓ bb      | ㅓ yo       | ㅓ b        |
| 19 | 10011 | ㅓ s       | ㅓ u        | ㅓ bs       |
| 20 | 10100 | ㅓ ss      |            | ㅓ s        |
| 21 | 10101 | ㅓ ng      | ㅓ we       | ㅓ ss       |
| 22 | 10110 | ㅓ j       | ㅓ wei      | ㅓ ng       |
| 23 | 10111 | ㅓ jj      | ㅓ wi       | ㅓ j        |
| 24 | 11000 | ㅓ ch      |            | ㅓ ch       |
| 25 | 11001 | ㅓ k       | ㅓ yu       | ㅓ k        |
| 26 | 11010 | ㅓ t       | ㅓ eu       | ㅓ t        |
| 27 | 11011 | ㅓ p       | ㅓ eui      | ㅓ p        |
| 28 | 11100 | ㅓ h       |            | ㅓ h        |
| 29 | 11101 |           | ㅓ i        |            |
| 30 | 11110 |           |            |            |
| 31 | 11111 |           |            |            |

[ *JOHAB de COMMERCIALE* ]

| Cho-seong | Jung-seong | Jong-seong |
|-----------|------------|------------|
|           |            |            |
| *         |            | *          |
| ㅑ g       | *          | ㅑ g        |
| ㅕ gg      | ㅏ a        | ㅕ gg       |
| ㅜ n       | ㅓ ai       | ㅗ gs       |
| ㅓ d       | ㅛ ya       | ㅜ n        |
| ㅕ dd      | ㅝ yai      | ㅠ nj       |
| ㅓ l       | ㅟ e        | ㅡ nh       |
| ㅓ m       |            | ㅓ d        |
| ㅓ b       |            | ㅓ l        |
| ㅓ bb      | ㅑ ei       | ㅓ lg       |
| ㅓ s       | ㅓ ye       | ㅓ lm       |
| ㅓ ss      | ㅕ yei      | ㅓ lb       |
| ㅓ ng      | ㅓ o        | ㅓ ls       |
| ㅓ j       | ㅓ wa       | ㅓ lt       |
| ㅓ jj      | ㅓ wai      | ㅓ lp       |
| ㅓ ch      |            | ㅓ lh       |
| ㅓ k       |            | ㅓ m        |
| ㅓ t       | ㅓ oi       |            |
| ㅓ p       | ㅓ yo       | ㅓ b        |
| ㅓ h       | ㅓ u        | ㅓ bs       |
|           | ㅓ we       | ㅓ s        |
|           | ㅓ wei      | ㅓ ss       |
|           | ㅓ wi       | ㅓ ng       |
|           |            | ㅓ j        |
|           |            | ㅓ ch       |
|           | ㅓ yu       | ㅓ k        |
|           | ㅓ eu       | ㅓ t        |
|           | ㅓ eui      | ㅓ p        |
|           | ㅓ i        | ㅓ h        |
|           |            |            |
|           |            |            |

( \* : le code de remplissage )

## Annexe 2. CIP : La table des codes des informations phonématiques

0x0420, 0x0421, 0x0424, 0x0427, 0x0429, 0x042a, 0x042b, 0x0431, 0x0432, 0x0434, 0x0435, 0x0436, 0x0437, 0x0438, 0x043a, 0x043c, 0x043d, 0x043e, 0x0440, 0x0441, 0x0444, 0x0449, 0x0451, 0x0452, 0x0455, 0x0456, 0x0457, 0x0460, 0x0461, 0x0464, 0x0469, 0x0475, 0x0477, 0x0480, 0x0484, 0x0489, 0x04a0, 0x04a1, 0x04a4, 0x04a7, 0x04a9, 0x04ab, 0x04b1, 0x04b2, 0x04b5, 0x04b6, 0x04b7, 0x04b8, 0x04bc, 0x04bd, 0x04be, 0x04c0, 0x04c4, 0x04c9, 0x04d1, 0x04d2, 0x04d5, 0x04d6, 0x04d7, 0x04e0, 0x04e1, 0x04e2, 0x04e4, 0x04e7, 0x04e9, 0x04f1, 0x04f2, 0x04f5, 0x04f6, 0x04f7, 0x04fc, 0x0500, 0x0504, 0x0509, 0x0512, 0x0515, 0x0520, 0x0521, 0x0524, 0x0527, 0x0529, 0x052b, 0x052d, 0x0530, 0x0531, 0x0532, 0x0535, 0x0537, 0x0538, 0x0540, 0x0541, 0x0544, 0x0549, 0x054b, 0x0551, 0x0552, 0x0555, 0x0557, 0x0560, 0x0564, 0x0569, 0x0572, 0x0576, 0x0577, 0x0580, 0x0581, 0x0584, 0x0589, 0x0591, 0x0592, 0x0595, 0x0597, 0x05a0, 0x05a4, 0x05a9, 0x05b2, 0x05b5, 0x05c0, 0x05c1, 0x05c4, 0x05c7, 0x05c9, 0x05ca, 0x05cb, 0x05d0, 0x05d1, 0x05d2, 0x05d5, 0x05d7, 0x05d8, 0x05e0, 0x05e1, 0x05e4, 0x05e9, 0x05f6, 0x05f7, 0x0600, 0x0615, 0x0620, 0x0621, 0x0624, 0x0629, 0x0631, 0x0632, 0x0635, 0x0640, 0x0644, 0x0649, 0x0660, 0x0661, 0x0664, 0x0667, 0x0669, 0x066a, 0x0671, 0x0672, 0x0675, 0x0677, 0x0680, 0x06a0, 0x06a1, 0x06a4, 0x06a7, 0x06a9, 0x06ab, 0x06b1, 0x06b2, 0x06b5, 0x06b7, 0x06b8, 0x06bd, 0x06d1, 0x06d2, 0x06d5, 0x06d7, 0x06e0, 0x06e4, 0x06e9, 0x06f5, 0x06f6, 0x06fc, 0x0900, 0x0920, 0x0921, 0x0924, 0x0926, 0x0929, 0x0931, 0x0932, 0x0935, 0x0937, 0x0938, 0x093a, 0x0940, 0x0941, 0x0949, 0x0956, 0x0957, 0x0960, 0x0961, 0x0977, 0x0980, 0x0984, 0x0989, 0x0991, 0x0992, 0x0997, 0x09a0, 0x09c0, 0x09c1, 0x09c4, 0x09c9, 0x09d0, 0x09d1, 0x09d2, 0x09d5, 0x09d7, 0x09d8, 0x09e0, 0x09e9, 0x09f6, 0x09f7, 0x0a00, 0x0a01, 0x0a04, 0x0a09, 0x0a11, 0x0a12, 0x0a16, 0x0a20, 0x0a24, 0x0a29, 0x0a31, 0x0a32, 0x0a40, 0x0a60, 0x0a61, 0x0a64, 0x0a66, 0x0a69, 0x0a6b, 0x0a70, 0x0a71, 0x0a72, 0x0a75, 0x0a77, 0x0a7c, 0x0aa0, 0x0aa1, 0x0aa4, 0x0aa9, 0x0ab1, 0x0ab2, 0x0ab5, 0x0ab7, 0x1020, 0x1021, 0x1022, 0x1024, 0x1027, 0x1029, 0x102a, 0x102b, 0x1031, 0x1032, 0x1035, 0x1036, 0x1037, 0x1038, 0x103a, 0x103c, 0x103e, 0x1040, 0x1041, 0x1044, 0x1049, 0x1051, 0x1052, 0x1055, 0x1056, 0x1057, 0x1060, 0x1061, 0x1064, 0x1069, 0x1071, 0x1077, 0x10a0, 0x10a1, 0x10a3, 0x10a4, 0x10a9, 0x10ab, 0x10ac, 0x10b1, 0x10b2, 0x10b5, 0x10b6, 0x10b7, 0x10be, 0x10c0, 0x10c1, 0x10c4, 0x10c9, 0x10d1, 0x10d2, 0x10d5, 0x10d6, 0x10d7, 0x10e0, 0x10e1, 0x10e4, 0x10e9, 0x10f1, 0x10f2, 0x10f6, 0x10f7, 0x10fb, 0x1100, 0x1104, 0x1120, 0x1121, 0x1124, 0x1129, 0x112b, 0x1131, 0x1132, 0x1135, 0x1137, 0x113d, 0x113e, 0x1140, 0x1144, 0x1149, 0x1156, 0x1180, 0x1184, 0x1189, 0x1191, 0x1192, 0x1195, 0x11a0, 0x11a1, 0x11a4, 0x11a9, 0x11b2, 0x11b5, 0x11b7, 0x11c0, 0x11c1, 0x11c4, 0x11c7, 0x11c9, 0x11d1, 0x11d2, 0x11d5, 0x11d7, 0x11e0, 0x11f6, 0x1200, 0x1220, 0x1224, 0x1229, 0x1231, 0x1232, 0x1240, 0x1241, 0x1249, 0x1251, 0x1252, 0x1257, 0x1260, 0x1261, 0x1264, 0x1269, 0x126a, 0x126b, 0x1271, 0x1272, 0x1275, 0x1277, 0x1278, 0x127d, 0x1280, 0x1284, 0x1289, 0x12a0, 0x12a1, 0x12a4, 0x12a9, 0x12ab, 0x12b1, 0x12b2, 0x12b5, 0x12b7, 0x12bd, 0x1c20, 0x1c21, 0x1c22, 0x1c24, 0x1c27, 0x1c29, 0x1c2a, 0x1c2b, 0x1c2c, 0x1c30, 0x1c31, 0x1c32, 0x1c35, 0x1c36, 0x1c37, 0x1c38, 0x1c3a, 0x1c3e, 0x1c40, 0x1c41, 0x1c44, 0x1c49, 0x1c51, 0x1c52, 0x1c55, 0x1c56, 0x1c57, 0x1c60, 0x1ca0, 0x1ca1, 0x1ca2, 0x1ca4, 0x1ca7, 0x1ca9, 0x1cab, 0x1cac, 0x1cb1, 0x1cb2, 0x1cb5, 0x1cb7, 0x1cba, 0x1cbd, 0x1cc0, 0x1cc1, 0x1cc4, 0x1cc9, 0x1cd1, 0x1cd2, 0x1cd5, 0x1cd6, 0x1cd7, 0x1ce0, 0x1ce4, 0x1ce9, 0x1cf6, 0x1cf7, 0x1d00, 0x1d04, 0x1d20, 0x1d21, 0x1d24, 0x1d27, 0x1d29, 0x1d2b, 0x1d2d, 0x1d31, 0x1d32, 0x1d35, 0x1d37, 0x1d3a, 0x1d3c, 0x1d40, 0x1d44, 0x1d49, 0x1d60, 0x1d76, 0x1d80, 0x1d84, 0x1d89, 0x1d91, 0x1d92, 0x1d95, 0x1da0, 0x1dc0, 0x1dc1, 0x1dc4, 0x1dc9, 0x1dd1, 0x1dd2, 0x1dd5, 0x1dd7, 0x1de0, 0x1df6, 0x1e00, 0x1e17, 0x1e20, 0x1e24, 0x1e29, 0x1e32, 0x1e35, 0x1e37, 0x1e40, 0x1e44, 0x1e49, 0x1e51, 0x1e57, 0x1e60, 0x1e61, 0x1e64, 0x1e67, 0x1e69, 0x1e6b, 0x1e71, 0x1e72, 0x1e75, 0x1e77, 0x1e80, 0x1ea0, 0x1ea1, 0x1ea4, 0x1ea7, 0x1ea9, 0x1eb1, 0x1eb2, 0x1eb5, 0x1eb6, 0x1eb7, 0x1eb8, 0x2020, 0x2021, 0x2024, 0x2029, 0x2031, 0x2032, 0x2035, 0x2036, 0x2037, 0x2038, 0x2040, 0x2041, 0x2044, 0x2049, 0x2051, 0x2052, 0x2055, 0x2056, 0x2057, 0x20a0, 0x20a1, 0x20a4, 0x20a9, 0x20ab, 0x20ac, 0x20b1, 0x20b2, 0x20b5, 0x20b6, 0x20b7, 0x20be, 0x20c0, 0x20c1, 0x20c4, 0x20c9, 0x20d1, 0x20d2, 0x20d5, 0x20d6, 0x20d7, 0x20e0, 0x20f6, 0x2120, 0x2121, 0x2124, 0x2129, 0x2137, 0x2140, 0x2149, 0x2160, 0x2180, 0x2184, 0x21c0, 0x21c1, 0x21c4, 0x21c9, 0x21d0, 0x21d1, 0x21d7, 0x2200, 0x2220, 0x2224, 0x2229, 0x2231, 0x2232, 0x2237, 0x2260, 0x2261, 0x2264, 0x2267, 0x2269, 0x2271, 0x2272, 0x2275, 0x2280, 0x2284, 0x2289, 0x2291, 0x2292, 0x22a0, 0x22a4, 0x22a9, 0x22b1, 0x22b2, 0x22b5, 0x22b7, 0x2420, 0x2421, 0x2424, 0x2429, 0x2431, 0x2432, 0x2435, 0x2436, 0x2437, 0x2438, 0x243d, 0x243e, 0x2440, 0x2441, 0x2444, 0x2449, 0x2451, 0x2452, 0x2455, 0x2456, 0x2457, 0x2460, 0x2461, 0x2464, 0x2475, 0x2477, 0x24a0, 0x24a1, 0x24a4, 0x24a9, 0x24b1, 0x24b2, 0x24b5, 0x24b6, 0x24b7, 0x24be, 0x24c0, 0x24c1, 0x24c4, 0x24c9, 0x24d1, 0x24d2, 0x24d5, 0x24d7, 0x24e0, 0x24e1, 0x24e4, 0x24e9, 0x24f1, 0x24f2, 0x24f5, 0x24f6, 0x24f7, 0x2500, 0x2504, 0x2512, 0x2515, 0x2520, 0x2521, 0x2524, 0x2529, 0x2531, 0x2532, 0x2535, 0x2537, 0x2540, 0x2544, 0x2557, 0x2576, 0x2580, 0x2584, 0x2589, 0x2591, 0x2592, 0x2595, 0x2597, 0x25a0, 0x25a4, 0x25a9, 0x25b2, 0x25b5, 0x25b7, 0x25c0, 0x25c1, 0x25c4, 0x25c9, 0x25d1, 0x25d2, 0x25d5, 0x25d7, 0x25e0, 0x25f6, 0x2600, 0x2620, 0x2621, 0x2624, 0x2629, 0x2631, 0x2635, 0x2637, 0x2640, 0x2641, 0x2644, 0x2649, 0x2651, 0x2652, 0x2655, 0x2657, 0x2660, 0x2661, 0x2664, 0x2669, 0x2671, 0x2672, 0x2675, 0x2677, 0x2678, 0x267c, 0x267d, 0x26a0, 0x26a1, 0x26a4, 0x26a9, 0x26b1, 0x26b2, 0x26b5, 0x26b7, 0x4420, 0x4421, 0x4424, 0x4426, 0x4427, 0x4429, 0x442a,

0x442b,0x4431,0x4432,0x4435,0x4437,0x4438,0x443c,0x443e,0x4440,0x4441,0x4444,0x4449,0x4451,  
0x4452,0x4455,0x4456,0x4457,0x4458,0x4460,0x4461,0x4469,0x4477,0x44a0,0x44a1,0x44a4,0x44a9,  
0x44ab,0x44b1,0x44b2,0x44b5,0x44b7,0x44b8,0x44be,0x44c0,0x44c1,0x44c4,0x44c9,0x44d1,0x44d2,  
0x44d5,0x44d6,0x44d7,0x44e0,0x44e1,0x44e4,0x44e9,0x44f5,0x44f6,0x44f7,0x44fa,0x4500,0x4520,  
0x4521,0x4523,0x4524,0x4529,0x452b,0x4531,0x4532,0x4535,0x4537,0x4540,0x4544,0x4556,0x4557,  
0x4580,0x4584,0x4589,0x4592,0x4595,0x4597,0x45a0,0x45a4,0x45a9,0x45b2,0x45b5,0x45c0,0x45c1,  
0x45c2,0x45c4,0x45c7,0x45c9,0x45ca,0x45cb,0x45d1,0x45d2,0x45d5,0x45d7,0x45dc,0x45de,0x45e0,  
0x45e4,0x45e9,0x45f2,0x45f5,0x4600,0x4620,0x4624,0x4629,0x4640,0x4644,0x4649,0x4651,0x4655,  
0x4660,0x4664,0x4669,0x4671,0x4675,0x46a0,0x46a1,0x46a4,0x46a7,0x46a9,0x46ab,0x46b1,0x46b2,  
0x46b5,0x46b6,0x46b7,0x46ba,0x46bc,0x4820,0x4821,0x4822,0x4823,0x4824,0x4827,0x4829,0x482a,  
0x482b,0x482c,0x4831,0x4832,0x4835,0x4837,0x483c,0x4840,0x4841,0x4844,0x4849,0x4851,0x4852,  
0x4855,0x4856,0x4857,0x485c,0x4860,0x4861,0x4864,0x4872,0x48a0,0x48a1,0x48a4,0x48a7,0x48a9,  
0x48ab,0x48b1,0x48b2,0x48b5,0x48b7,0x48b8,0x48c0,0x48c1,0x48c4,0x48c7,0x48c9,0x48d1,0x48d2,  
0x48d5,0x48d6,0x48d7,0x48e0,0x48e1,0x48e4,0x48e9,0x48f2,0x48f5,0x48f6,0x48f7,0x48fc,0x4900,  
0x4904,0x4920,0x4921,0x4922,0x4924,0x4929,0x4931,0x4932,0x4935,0x4937,0x4940,0x4944,0x4956,  
0x4960,0x4976,0x4980,0x4981,0x4984,0x4989,0x4991,0x4992,0x49a0,0x49a4,0x49c0,0x49c1,0x49c4,  
0x49c7,0x49c9,0x49ca,0x49cb,0x49d1,0x49d2,0x49d5,0x49d7,0x49dc,0x49dd,0x49e0,0x49e9,0x49f6,  
0x4a00,0x4a20,0x4a21,0x4a24,0x4a29,0x4a37,0x4a40,0x4a44,0x4a49,0x4a51,0x4a55,0x4a57,0x4a60,  
0x4a61,0x4a64,0x4a69,0x4a71,0x4a72,0x4a75,0x4aa0,0x4aa1,0x4aa4,0x4aa9,0x4aab,0x4ab1,0x4ab2,  
0x4ab5,0x4ab7,0x4ab8,0x4aba,0x4ac20,0x4ac21,0x4ac24,0x4ac29,0x4ac2b,0x4ac31,0x4ac32,0x4ac35,0x4ac36,  
0x4ac37,0x4ac3e,0x4ac40,0x4ac41,0x4ac44,0x4ac49,0x4ac51,0x4ac52,0x4ac55,0x4ac56,0x4ac57,0x4ac60,0x4ac61,  
0x4ac71,0x4aca0,0x4ca1,0x4ca4,0x4ca7,0x4ca9,0x4cb1,0x4cb5,0x4cb6,0x4cb7,0x4cc0,0x4cd7,0x4ce0,  
0x4cel,0x4cf1,0x4cf2,0x4cf5,0x4cf6,0x4cf7,0x4d20,0x4d21,0x4d24,0x4d29,0x4d31,0x4d32,0x4d37,  
0x4d80,0x4da0,0x4db7,0x4dc0,0x4dc1,0x4dc4,0x4dc9,0x4dd1,0x4dd5,0x4dd7,0x4de0,0x4e57,0x4e60,  
0x4e64,0x4e69,0x4e71,0x4e72,0x4ea0,0x4ea1,0x4ea4,0x4ea9,0x4eb1,0x4eb2,0x4eb5,0x4eb7,0x5420,  
0x5421,0x5423,0x5424,0x5427,0x5429,0x542a,0x542b,0x5431,0x5432,0x5435,0x5436,0x5437,0x543c,  
0x5440,0x5441,0x5444,0x5449,0x5451,0x5452,0x5455,0x5456,0x5457,0x5460,0x5461,0x5464,0x5469,  
0x5471,0x5472,0x5475,0x5477,0x5480,0x5484,0x5489,0x5491,0x5497,0x54a0,0x54a1,0x54a2,0x54a3,  
0x54a4,0x54a7,0x54a9,0x54ab,0x54ac,0x54b1,0x54b2,0x54b5,0x54b6,0x54b7,0x54bd,0x54c0,0x54c1,  
0x54c4,0x54c9,0x54d1,0x54d2,0x54d5,0x54d6,0x54d7,0x54e0,0x54e1,0x54e4,0x54e9,0x54f1,0x54f2,  
0x54f5,0x54f6,0x54f7,0x5500,0x5504,0x5509,0x5517,0x5520,0x5521,0x5522,0x5524,0x5529,0x552b,  
0x5531,0x5532,0x5535,0x5537,0x553c,0x5540,0x5541,0x5544,0x5549,0x5557,0x5560,0x5564,0x5569,  
0x5571,0x5575,0x5576,0x5580,0x5584,0x5589,0x5591,0x5592,0x5595,0x55a0,0x55a1,0x55a4,0x55a9,  
0x55b1,0x55b2,0x55b5,0x55b7,0x55c0,0x55c1,0x55c4,0x55c7,0x55c9,0x55d1,0x55d2,0x55d5,0x55d7,  
0x55da,0x55dc,0x55dd,0x55e0,0x55f6,0x5600,0x5601,0x5604,0x5609,0x5611,0x5617,0x5620,0x5621,  
0x5624,0x5629,0x5631,0x5632,0x5635,0x5637,0x5640,0x5641,0x5649,0x5651,0x5655,0x5657,0x5660,  
0x5661,0x5664,0x5669,0x566a,0x5671,0x5672,0x5675,0x5677,0x56a0,0x56a1,0x56a4,0x56a7,0x56a9,  
0x56b0,0x56b1,0x56b2,0x56b5,0x56b7,0x56bd,0x5820,0x5821,0x5823,0x5824,0x5829,0x5831,0x5832,  
0x5836,0x5837,0x583e,0x5840,0x5841,0x5844,0x5849,0x5851,0x5852,0x5856,0x5857,0x5877,0x58a0,  
0x58a1,0x58a4,0x58a9,0x58ab,0x58b1,0x58b2,0x58b6,0x58b7,0x58c0,0x58c4,0x58c9,0x5904,0x5920,  
0x5921,0x5924,0x5927,0x5929,0x592b,0x5931,0x5932,0x5937,0x5940,0x5941,0x5944,0x5956,0x5960,  
0x5976,0x5980,0x5984,0x5989,0x5991,0x5992,0x59a0,0x59c0,0x59c1,0x59c4,0x59c9,0x59d1,0x59d2,  
0x59d7,0x59e0,0x59f6,0x5a00,0x5a20,0x5a24,0x5a57,0x5a60,0x5a61,0x5a64,0x5a69,0x5a6b,0x5a70,  
0x5a71,0x5a72,0x5a80,0x5a84,0x5a89,0x5a91,0x5aa0,0x5aa1,0x5aa4,0x5aa9,0x5ab1,0x5ab2,0x5ab5,  
0x5ab7,0x5ac20,0x5ac21,0x5ac24,0x5ac25,0x5ac26,0x5ac29,0x5ac2a,0x5ac2b,0x5ac30,0x5ac31,0x5ac32,0x5ac35,  
0x5ac36,0x5ac37,0x5ac3c,0x5ac3d,0x5ac40,0x5ac41,0x5ac44,0x5ac49,0x5ac51,0x5ac52,0x5ac55,0x5ac56,0x5ac57,  
0x5ac60,0x5ac61,0x5ac64,0x5ac69,0x5ac6c,0x5ac71,0x5ac72,0x5ac75,0x5ac77,0x5ac7c,0x5ac7e,0x5ac80,0x5ac84,  
0x5ac89,0x5ac92,0x5aca0,0x5aca1,0x5aca4,0x5aca5,0x5aca7,0x5aca9,0x5caa,0x5cab,0x5cb1,0x5cb2,0x5cb4,  
0x5cb5,0x5cb6,0x5cb7,0x5cb8,0x5cbb,0x5cbd,0x5cc0,0x5cc1,0x5cc4,0x5cc9,0x5cd1,0x5cd2,0x5cd5,  
0x5cd7,0x5ce0,0x5ce1,0x5ce2,0x5ce4,0x5ce9,0x5ceb,0x5cec,0x5cf1,0x5cf2,0x5cf4,0x5cf5,0x5cf6,  
0x5cf7,0x5cfc,0x5cfd,0x5cfe,0x5d00,0x5d04,0x5d09,0x5d11,0x5d12,0x5d15,0x5d16,0x5d20,0x5d21,  
0x5d24,0x5d29,0x5d2a,0x5d2b,0x5d2d,0x5d30,0x5d31,0x5d32,0x5d35,0x5d37,0x5d3a,0x5d40,0x5d41,  
0x5d44,0x5d49,0x5d51,0x5d52,0x5d55,0x5d56,0x5d57,0x5d60,0x5d61,0x5d64,0x5d71,0x5d75,0x5d77,  
0x5d80,0x5d81,0x5d84,0x5d89,0x5d91,0x5d92,0x5d95,0x5d97,0x5da0,0x5da1,0x5da4,0x5da9,0x5db1,  
0x5db2,0x5db5,0x5db7,0x5dc0,0x5dc1,0x5dc4,0x5dc9,0x5dca,0x5dcb,0x5dd1,0x5dd2,0x5dd5,0x5dd7,  
0x5de0,0x5de1,0x5de4,0x5de9,0x5df1,0x5df2,0x5df6,0x5df7,0x5e00,0x5e01,0x5e04,0x5e09,0x5e11,  
0x5e12,0x5e17,0x5e20,0x5e21,0x5e24,0x5e29,0x5e31,0x5e32,0x5e35,0x5e37,0x5e40,0x5e41,0x5e44,  
0x5e49,0x5e51,0x5e52,0x5e55,0x5e57,0x5e5a,0x5e60,0x5e61,0x5e64,0x5e69,0x5e6f,0x5e71,0x5e72,  
0x5e75,0x5e77,0x5e78,0x5e7a,0x5e7b,0x5e7c,0x5e7d,0x5e7e,0x5e80,0x5e84,0x5e89,0x5e91,0x5e95,  
0x5ea0,0x5ea1,0x5ea4,0x5ea9,0x5eaa,0x5eab,0x5eb0,0x5eb1,0x5eb2,0x5eb5,0x5eb6,0x5eb7,0x5eb8,  
0x5ebd,0x6020,0x6021,0x6024,0x6027,0x6029,0x602b,0x6031,0x6032,0x6036,0x6037,0x6038,  
0x6040,0x6041,0x6044,0x6049,0x6051,0x6052,0x6055,0x6056,0x6057,0x6060,0x6061,0x6064,  
0x6066,0x6069,0x6071,0x6077,0x6080,0x6084,0x6089,0x60a0,0x60a1,0x60a4,0x60a9,0x60ab,0x60b1,

0x60b2,0x60b5,0x60b7,0x60b8,0x60c0,0x60c1,0x60c4,0x60c9,0x60d1,0x60d2,0x60d5,0x60d7,0x60e0,  
0x60e4,0x60e9,0x60f1,0x60f2,0x60f6,0x60f7,0x6100,0x6120,0x6121,0x6124,0x6129,0x612b,0x6131,  
0x6132,0x6135,0x6137,0x6138,0x613a,0x613e,0x6140,0x6141,0x6149,0x6152,0x6155,0x6157,0x6160,  
0x6176,0x6177,0x6180,0x6184,0x6189,0x6191,0x6192,0x6195,0x6197,0x61a0,0x61a1,0x61a4,0x61b7,  
0x61c0,0x61c1,0x61c4,0x61c9,0x61ca,0x61cb,0x61d1,0x61d2,0x61d5,0x61d7,0x61e0,0x61f6,0x6200,  
0x6220,0x6221,0x6224,0x6229,0x6231,0x6232,0x6235,0x6240,0x6244,0x6249,0x6251,0x6260,0x6261,  
0x6264,0x6269,0x6271,0x6272,0x6275,0x6277,0x62a0,0x62a1,0x62a4,0x62a7,0x62a9,0x62ab,0x62b1,  
0x62b2,0x62b5,0x62b7,0x62b8,0x62bc,0x62bd,0x6420,0x6421,0x6424,0x6426,0x6429,0x642c,0x6431,  
0x6432,0x6435,0x6436,0x6437,0x6440,0x6441,0x6444,0x6449,0x6451,0x6452,0x6455,0x6456,0x6457,  
0x6460,0x6464,0x6477,0x64a0,0x64a1,0x64a4,0x64a9,0x64b1,0x64b2,0x64b5,0x64b6,0x64b7,0x64c0,  
0x64d7,0x64e0,0x64f6,0x6520,0x6521,0x6524,0x6529,0x6531,0x6532,0x6535,0x6537,0x653a,0x6540,  
0x6541,0x6549,0x6556,0x6560,0x6576,0x6580,0x6584,0x6589,0x6591,0x6592,0x65b7,0x65c0,0x65c1,  
0x65c4,0x65c9,0x65d1,0x65d2,0x65d7,0x65e0,0x65f6,0x65f7,0x6620,0x6640,0x6660,0x6671,0x6675,  
0x6677,0x66a0,0x66a1,0x66a4,0x66a9,0x66b1,0x66b2,0x66b7,0x66b8,0x66be,0x6820,0x6821,0x6824,  
0x6826,0x6829,0x6831,0x6832,0x6835,0x6836,0x6837,0x6838,0x6840,0x6841,0x6844,0x6849,0x6851,  
0x6852,0x6855,0x6856,0x6857,0x6860,0x6864,0x6866,0x6869,0x6871,0x6877,0x68a0,0x68a1,0x68a4,  
0x68a9,0x68b1,0x68b2,0x68b5,0x68b6,0x68b7,0x68c0,0x68c1,0x68c4,0x68c9,0x68d1,0x68d2,0x68d5,  
0x68d7,0x68e0,0x68e4,0x68f6,0x6900,0x6904,0x6917,0x6920,0x6921,0x6924,0x6929,0x6931,0x6932,  
0x6935,0x6937,0x6940,0x6944,0x6949,0x6957,0x6980,0x6984,0x6989,0x6991,0x6992,0x6995,0x6997,  
0x69a0,0x69b1,0x69c0,0x69c1,0x69c4,0x69c9,0x69d1,0x69d2,0x69d5,0x69d7,0x69e0,0x69f6,0x6a00,  
0x6a04,0x6a20,0x6a24,0x6a29,0x6a31,0x6a32,0x6a35,0x6a37,0x6a40,0x6a44,0x6a49,0x6a51,0x6a57,  
0x6a60,0x6a61,0x6a64,0x6a69,0x6a71,0x6a72,0x6a75,0x6a77,0x6aa0,0x6aa1,0x6aa4,0x6aa7,0x6aa9,  
0x6aaa,0x6ab1,0x6ab2,0x6ab5,0x6ab7,0x6c20,0x6c21,0x6c24,0x6c29,0x6c31,0x6c32,0x6c35,0x6c37,  
0x6c40,0x6c41,0x6c44,0x6c49,0x6c51,0x6c52,0x6c55,0x6c56,0x6c57,0x6c60,0x6c61,0x6c77,0x6ca0,  
0x6ca1,0x6ca4,0x6ca7,0x6ca9,0x6cb1,0x6cb2,0x6cb5,0x6cb6,0x6cb7,0x6cc0,0x6cc1,0x6cc4,0x6cc9,  
0x6cd1,0x6cd2,0x6cd5,0x6cd7,0x6ce0,0x6ce4,0x6ce9,0x6cf1,0x6cf2,0x6cf5,0x6cf6,0x6cf7,0x6d00,  
0x6d20,0x6d21,0x6d24,0x6d29,0x6d31,0x6d32,0x6d35,0x6d37,0x6d40,0x6d41,0x6d44,0x6d49,0x6d51,  
0x6d57,0x6d60,0x6d77,0x6d80,0x6d89,0x6da0,0x6dc0,0x6dc1,0x6dc4,0x6dc9,0x6dd1,0x6dd2,0x6dd5,  
0x6dd7,0x6de0,0x6de4,0x6de9,0x6df7,0x6e00,0x6e17,0x6e20,0x6e21,0x6e24,0x6e29,0x6e31,0x6e32,  
0x6e35,0x6e37,0x6e40,0x6e44,0x6e49,0x6e51,0x6e60,0x6e61,0x6e64,0x6e69,0x6e71,0x6e72,0x6e77,  
0x6ea0,0x6ea1,0x6ea4,0x6ea9,0x6eb1,0x6eb2,0x6eb5,0x6eb7,0x7020,0x7021,0x7024,0x7029,0x702a,  
0x7031,0x7032,0x7035,0x7036,0x7037,0x7040,0x7041,0x7044,0x7049,0x7051,0x7052,0x7055,0x7056,  
0x7057,0x7060,0x7077,0x70a0,0x70a1,0x70a4,0x70a9,0x70ab,0x70b1,0x70b2,0x70b5,0x70b6,0x70b7,  
0x70c0,0x70c1,0x70c4,0x70c9,0x70d1,0x70d2,0x70d5,0x70d7,0x70e0,0x70e4,0x70f6,0x7100,0x7104,  
0x7120,0x7121,0x7124,0x7129,0x7131,0x7132,0x7135,0x7137,0x713d,0x7140,0x7144,0x7160,0x7180,  
0x7184,0x7195,0x7197,0x71a0,0x71c0,0x71c1,0x71c4,0x71c9,0x71d1,0x71d2,0x71d5,0x71d7,0x71e0,  
0x71f6,0x7200,0x7220,0x7221,0x7224,0x7229,0x7231,0x7232,0x7237,0x7240,0x7244,0x7249,0x7251,  
0x7257,0x7260,0x7261,0x7264,0x7267,0x7269,0x726b,0x7271,0x7272,0x7275,0x7280,0x7284,0x7289,  
0x7291,0x7292,0x72a0,0x72a1,0x72a4,0x72a9,0x72b1,0x72b2,0x72b5,0x72b7,0x7420,0x7421,0x7422,  
0x7424,0x7429,0x742b,0x7431,0x7432,0x7435,0x7436,0x7437,0x743c,0x7440,0x7441,0x7444,0x7449,  
0x7451,0x7452,0x7455,0x7456,0x7457,0x7460,0x7461,0x74a0,0x74a1,0x74a4,0x74a9,0x74b1,0x74b2,  
0x74b5,0x74b6,0x74b7,0x74c0,0x74c1,0x74c4,0x74c9,0x74d1,0x74d2,0x74d5,0x74d7,0x74e0,0x74e4,  
0x74e9,0x74f1,0x74f2,0x74f6,0x74f7,0x7500,0x7509,0x7512,0x7515,0x7520,0x7521,0x7524,0x7529,  
0x7531,0x7532,0x7535,0x7537,0x7540,0x7557,0x7580,0x7584,0x75a0,0x75a4,0x75a9,0x75b2,0x75b6,  
0x75c0,0x75c1,0x75c4,0x75c7,0x75c9,0x75cb,0x75d1,0x75d2,0x75d5,0x75d7,0x75e0,0x75f7,0x7620,  
0x7624,0x7629,0x7631,0x7635,0x7640,0x7644,0x7649,0x7651,0x7655,0x7657,0x7660,0x7664,0x7669,  
0x7671,0x7672,0x7675,0x76a0,0x76a1,0x76a4,0x76a9,0x76b1,0x76b2,0x76b5,0x76b7,0x7820,0x7821,  
0x7824,0x7829,0x782e,0x7831,0x7832,0x7835,0x7837,0x7840,0x7841,0x7844,0x7849,0x7851,0x7852,  
0x7855,0x7856,0x7857,0x7860,0x7877,0x78a0,0x78a1,0x78a4,0x78a9,0x78ab,0x78b1,0x78b2,0x78b5,  
0x78b7,0x78c0,0x78c1,0x78c4,0x78c9,0x78d1,0x78d2,0x78d5,0x78d7,0x78e0,0x78e1,0x78e4,0x78e9,  
0x78f1,0x78f2,0x78f5,0x78f6,0x78f7,0x7900,0x7904,0x7909,0x7912,0x7920,0x7921,0x7924,0x7929,  
0x792e,0x7931,0x7932,0x7935,0x7937,0x793c,0x7940,0x7941,0x7944,0x7949,0x7955,0x7957,0x7960,  
0x7961,0x7964,0x7975,0x7977,0x7980,0x7981,0x7984,0x7989,0x7992,0x7995,0x7997,0x79a0,0x79a4,  
0x79a9,0x79b2,0x79b5,0x79c0,0x79c1,0x79c4,0x79c9,0x79ce,0x79d1,0x79d5,0x79d7,0x79e0,0x79e4,  
0x79e9,0x79f1,0x79f7,0x7a00,0x7a01,0x7a04,0x7a09,0x7a17,0x7a20,0x7a21,0x7a24,0x7a29,0x7a31,  
0x7a32,0x7a35,0x7a37,0x7a40,0x7a41,0x7a44,0x7a49,0x7a51,0x7a55,0x7a57,0x7a60,0x7a61,0x7a64,  
0x7a66,0x7a67,0x7a69,0x7a6a,0x7a71,0x7a72,0x7a75,0x7a77,0x7a7c,0x7a80,0x7a84,0x7a89,0x7a91,  
0x7a92,0x7a97,0x7aa0,0x7aa1,0x7aa4,0x7aa9,0x7ab1,0x7ab2,0x7ab5,0x7ab7,▲

Annexe 3. Les tableaux de cartographie : *WAN-JCOM* et *JCOM-WAN*[ *WAN-JCOM* ]

| Index | <i>Choseong</i> | <i>Jungseon</i> | <i>Jongseon</i> |
|-------|-----------------|-----------------|-----------------|
| 1     | 2               | 3               | 2               |
| 2     | 3               | 4               | 3               |
| 3     |                 | 5               | 4               |
| 4     | 4               | 6               | 5               |
| 5     |                 | 7               | 6               |
| 6     |                 | 10              | 7               |
| 7     | 5               | 11              | 8               |
| 8     | 6               | 12              |                 |
| 9     | 7               | 13              | 9               |
| 10    |                 | 14              | 10              |
| 11    |                 | 15              | 11              |
| 12    |                 | 18              | 12              |
| 13    |                 | 19              | 13              |
| 14    |                 | 20              | 14              |
| 15    |                 | 21              | 15              |
| 16    |                 | 22              | 16              |
| 17    | 8               | 23              | 17              |
| 18    | 9               | 26              | 19              |
| 19    | 10              | 27              |                 |
| 20    |                 | 28              | 20              |
| 21    | 11              | 29              | 21              |
| 22    | 12              |                 | 22              |
| 23    | 13              |                 | 23              |
| 24    | 14              |                 | 24              |
| 25    | 15              |                 |                 |
| 26    | 16              |                 | 25              |
| 27    | 17              |                 | 26              |
| 28    | 18              |                 | 27              |
| 29    | 19              |                 | 28              |
| 30    | 20              |                 | 29              |

[ *JCOM-WAN* ]

| Index | <i>Choseong</i> | <i>Jungseon</i> | <i>Jongseon</i> |
|-------|-----------------|-----------------|-----------------|
| 1     | 0               |                 | 0               |
| 2     | 1               | 0               | 1               |
| 3     | 2               | 1               | 2               |
| 4     | 4               | 2               | 3               |
| 5     | 7               | 3               | 4               |
| 6     | 8               | 4               | 5               |
| 7     | 9               | 5               | 6               |
| 8     | 17              |                 | 7               |
| 9     | 18              |                 | 9               |
| 10    | 19              | 6               | 10              |
| 11    | 21              | 7               | 11              |
| 12    | 22              | 8               | 12              |
| 13    | 23              | 9               | 13              |
| 14    | 24              | 10              | 14              |
| 15    | 25              | 11              | 15              |
| 16    | 26              |                 | 16              |
| 17    | 27              |                 | 17              |
| 18    | 28              | 12              |                 |
| 19    | 29              | 13              | 18              |
| 20    | 30              | 14              | 20              |
| 21    |                 | 15              | 21              |
| 22    |                 | 16              | 22              |
| 23    |                 | 17              | 23              |
| 24    |                 |                 | 24              |
| 25    |                 |                 | 26              |
| 26    |                 | 18              | 27              |
| 27    |                 | 19              | 28              |
| 28    |                 | 20              | 29              |
| 29    |                 | 21              | 30              |
| 30    |                 |                 |                 |

Annexe 4. *ISP2* et *ICC*

|    |   |    | ㅏ    | ㅑ    | ㅓ           | ㅕ    | ㅗ    | ㅛ    | ㅜ    | ㅠ           | ㅡ    | ㅣ    | ㅗ           | ㅛ    | ㅜ           | ㅠ    | ㅡ    | ㅣ    | ㅗ    | ㅛ           | ㅜ    | ㅠ    | ㅡ    | ㅣ |
|----|---|----|------|------|-------------|------|------|------|------|-------------|------|------|-------------|------|-------------|------|------|------|------|-------------|------|------|------|---|
|    |   |    | 1    | 2    | 3           | 4    | 5    | 6    | 7    | 8           | 9    | 10   | 11          | 12   | 13          | 14   | 15   | 16   | 17   | 18          | 19   | 20   | 21   |   |
| 1  | ㄱ | 1  | 0    | 18   | 27          | 33   | 36   | 51   | 59   | 71          | 76   | 89   | 98          | 104  | 112         | 117  | 130  | 136  | 138  | 145         | 148  | 158  | 159  |   |
| 2  | ㄴ | 2  | 171  | 183  | 192         | 0    | 195  | 205  | 211  | 217         | 218  | 229  | 234         | 237  | 243         | 244  | 254  | 258  | 265  | 270         | 271  | 0    | 283  |   |
| 4  | ㄷ | 3  | 291  | 308  | 317         | 0    | 323  | 336  | 345  | 354         | 356  | 367  | 0           | 371  | 377         | 384  | 393  | 395  | 396  | 401         | 407  | 419  | 422  |   |
| 7  | ㄹ | 4  | 432  | 450  | 459         | 0    | 460  | 474  | 483  | 488         | 490  | 503  | 506         | 508  | 514         | 515  | 523  | 525  | 527  | 533         | 538  | 548  | 549  |   |
| 8  | ㅁ | 5  | 560  | 570  | 0           | 0    | 579  | 591  | 600  | 0           | 602  | 607  | 609         | 610  | 0           | 612  | 0    | 619  | 620  | 0           | 626  | 634  | 639  |   |
| 9  | ㅂ | 6  | 646  | 658  | 667         | 0    | 672  | 682  | 690  | 699         | 703  | 711  | <u>8714</u> | 715  | 722         | 728  | 736  | 738  | 739  | 746         | 754  | 0    | 765  |   |
| 17 | ㅅ | 7  | 773  | 788  | 798         | 0    | 802  | 813  | 822  | 830         | 831  | 841  | 0           | 845  | 851         | 856  | 870  | 875  | 876  | 879         | 884  | 0    | 889  |   |
| 18 | ㅆ | 8  | 902  | 917  | 927         | 0    | 931  | 942  | 952  | 961         | 963  | 972  | 975         | 977  | 983         | 985  | 998  | 1001 | 1002 | 1007        | 1013 | 0    | 1020 |   |
| 19 | ㅈ | 9  | 1031 | 1042 | 1051        | 0    | 1054 | 1063 | 1065 | 0           | 1072 | 0    | 0           | 1079 | 1080        | 1082 | 0    | 0    | 0    | 1089        | 1091 | 0    | 1096 |   |
| 21 | ㅊ | 10 | 1104 | 1118 | 1127        | 1135 | 1140 | 1155 | 1164 | 1173        | 1177 | 1188 | 1193        | 1199 | 1205        | 1213 | 1225 | 1227 | 1233 | 1241        | 1247 | 0    | 1256 |   |
| 22 | ㅋ | 11 | 1267 | 1277 | <u>9285</u> | 0    | 1286 | 1295 | 0    | <u>9298</u> | 1299 | 1308 | 1312        | 1314 | 1319        | 1320 | 1327 | 1329 | 1330 | <u>9332</u> | 1333 | 1341 | 1345 |   |
| 23 | ㆁ | 12 | 1353 | 1369 | 1378        | 1389 | 1393 | 1410 | 1418 | 1434        | 1441 | 1454 | 1463        | 1469 | 1477        | 1485 | 1495 | 1503 | 1510 | 1518        | 1527 | 1542 | 1547 |   |
| 24 | ㄷ | 13 | 1561 | 1574 | 1583        | 1590 | 1593 | 1603 | 1611 | 1618        | 1619 | 1631 | 1637        | 1640 | 1647        | 1651 | 1661 | 1663 | 1664 | 1671        | 1675 | 0    | 1683 |   |
| 25 | ㄸ | 14 | 1696 | 1707 | 1716        | 0    | 1719 | 1728 | 1730 | 0           | 1732 | 1741 | 1745        | 1747 | <u>9752</u> | 1753 | 1760 | 0    | 1763 | 1764        | 1765 | 0    | 1769 |   |
| 26 | ㅌ | 15 | 1778 | 1789 | 1798        | 0    | 1804 | 1813 | 1821 | 1824        | 1827 | 1835 | 0           | 1839 | 1846        | 1848 | 1856 | 1858 | 1860 | 1867        | 1872 | 0    | 1880 |   |
| 27 | ㅍ | 16 | 1890 | 1898 | 1907        | 0    | 1910 | 1920 | 1928 | 1936        | 1937 | 1945 | 1951        | 1953 | 1955        | 1956 | 1964 | 1968 | 1970 | 1978        | 1982 | 0    | 1989 |   |
| 28 | ㅑ | 17 | 1997 | 2007 | 2016        | 0    | 2018 | 2028 | 2036 | 2039        | 2041 | 2050 | 2052        | 2053 | 2057        | 2058 | 2066 | 2068 | 2069 | 2076        | 2081 | 2090 | 2095 |   |
| 29 | ㅓ | 18 | 2103 | 2115 | 2124        | 0    | 2126 | 2135 | 2143 | 2150        | 2154 | 2162 | 0           | 2164 | 2166        | 2171 | 2181 | 0    | 2183 | 2188        | 2194 | 0    | 2200 |   |
| 30 | ㅕ | 19 | 2208 | 2217 | 2226        | 0    | 2228 | 2237 | 2245 | 2254        | 2258 | 2268 | 2274        | 2279 | 2286        | 2291 | 2299 | 2304 | 2309 | 2317        | 2324 | 2336 | 2342 |   |

← ICC →

ISP2

- ISP2 : Le tableau des indices des syllabes de profondeur 2 de *l'arbre HAC*.
- ICC : La table des indices des consonnes de *Choseong*.

## Annexe 5. Les statistiques sur les syllabes du corpus du KAIST

( sur 38385924 syllabes différentes : les 600 les plus fréquentes )

(• : espace)

|    |           |    |          |     |         |     |         |
|----|-----------|----|----------|-----|---------|-----|---------|
| 1  | 9668073 • | 38 | 167922 수 | 75  | 75844 " | 112 | 55770 회 |
| 2  | 993771 이  | 39 | 162856 해 | 76  | 75588 위 | 113 | 55658 명 |
| 3  | 837318 다  | 40 | 150717 었 | 77  | 75493 스 | 114 | 54280 선 |
| 4  | 738614 는  | 41 | 149570 정 | 78  | 75476 할 | 115 | 53423 분 |
| 5  | 720239 .  | 42 | 147015 니 | 79  | 74193 거 | 116 | 53344 각 |
| 6  | 625639 의  | 43 | 142192 과 | 80  | 72721 했 | 117 | 53316 체 |
| 7  | 552785 예  | 44 | 133641 일 | 81  | 72348 계 | 118 | 52945 바 |
| 8  | 504173 을  | 45 | 133258 주 | 82  | 72142 국 | 119 | 52761 식 |
| 9  | 478764 하  | 46 | 128434 만 | 83  | 72090 때 | 120 | 52708 안 |
| 10 | 452695 가  | 47 | 128066 보 | 84  | 71995 조 | 121 | 52658 개 |
| 11 | 451571 고  | 48 | 127911 면 | 85  | 70731 화 | 122 | 52607 연 |
| 12 | 390431 지  | 49 | 124328 여 | 86  | 70519 비 | 123 | 52513 까 |
| 13 | 360541 그  | 50 | 124089 제 | 87  | 70284 간 | 124 | 50441 방 |
| 14 | 358330 한  | 51 | 120464 부 | 88  | 70257 경 | 125 | 50017 공 |
| 15 | 355562 은  | 52 | 117955 상 | 89  | 69746 학 | 126 | 49894 교 |
| 16 | 351662 로  | 53 | 115314 되 | 90  | 69659 l | 127 | 49877 운 |
| 17 | 325752 서  | 54 | 106594 전 | 91  | 68209 야 | 128 | 49645 단 |
| 18 | 295538 ,  | 55 | 105824 성 | 92  | 67383 세 | 129 | 49443 르 |
| 19 | 289952 기  | 56 | 105603 문 | 93  | 66946 치 | 130 | 48431 속 |
| 20 | 287272 어  | 57 | 104503 러 | 94  | 66115 실 | 131 | 48372 두 |
| 21 | 268035 나  | 58 | 104434 ) | 95  | 66079 앓 | 132 | 47463 던 |
| 22 | 263474 도  | 59 | 101855 ( | 96  | 65890 려 | 133 | 46880 용 |
| 23 | 257927 사  | 60 | 99309 구  | 97  | 65383 물 | 134 | 45652 었 |
| 24 | 248371 있  | 61 | 97374 우  | 98  | 65038 관 | 135 | 45473 당 |
| 25 | 246138 를  | 62 | 91897 생  | 99  | 64626 오 | 136 | 44569 같 |
| 26 | 245415 것  | 63 | 90560 신  | 100 | 64321 미 | 137 | 44484 작 |
| 27 | 244038 리  | 64 | 90278 장  | 101 | 64253 유 | 138 | 43876 발 |
| 28 | 233814 자  | 65 | 89453 동  | 102 | 64174 ' | 139 | 42768 산 |
| 29 | 228649 아  | 66 | 88040 내  | 103 | 63279 중 | 140 | 42529 더 |
| 30 | 225895 들  | 67 | 85695 마  | 104 | 60883 음 | 141 | 42445 통 |
| 31 | 208320 인  | 68 | 84299 무  | 105 | 60562 데 | 142 | 41859 불 |
| 32 | 205041 으  | 69 | 84131 소  | 106 | 59034 람 | 143 | 41501 본 |
| 33 | 192392 적  | 70 | 83507 없  | 107 | 57765 원 | 144 | 40789 았 |
| 34 | 188946 대  | 71 | 81041 말  | 108 | 57542 히 | 145 | 40543 결 |
| 35 | 182154 라  | 72 | 79204 요  | 109 | 57296 된 | 146 | 40422 년 |
| 36 | 177505 시  | 73 | 77841 와  | 110 | 56789 며 | 147 | 40033 재 |
| 37 | 174244 게  | 74 | 77374 모  | 111 | 56699 진 | 148 | 39894 입 |

|     |       |   |     |       |     |     |       |   |     |       |   |
|-----|-------|---|-----|-------|-----|-----|-------|---|-----|-------|---|
| 149 | 39371 | 행 | 192 | 28490 | 살   | 235 | 20906 | 트 | 278 | 17937 | 될 |
| 150 | 39128 | 0 | 193 | 28396 | 달   | 236 | 20882 | 근 | 279 | 17843 | 늘 |
| 151 | 39075 | 민 | 194 | 28328 | 알   | 237 | 20814 | 루 | 280 | 17794 | 판 |
| 152 | 38969 | 반 | 195 | 28057 | 중   | 238 | 20718 | 향 | 281 | 17716 | 크 |
| 153 | 38832 | 심 | 196 | 27742 | 드   | 239 | 20692 | 럼 | 282 | 17513 | 격 |
| 154 | 38273 | ? | 197 | 27555 | 배   | 240 | 20661 | 저 | 283 | 17414 | 레 |
| 155 | 38119 | 저 | 198 | 27204 | 버   | 241 | 20586 | 임 | 284 | 17237 | 누 |
| 156 | 37765 | 현 | 199 | 27086 | 느   | 242 | 20489 | 복 | 285 | 17134 | 급 |
| 157 | 37749 | 처 | 200 | 26358 | 3   | 243 | 20450 | 권 | 286 | 17020 | 네 |
| 158 | 36949 | 습 | 201 | 26276 | 후   | 244 | 20322 | 8 | 287 | 16979 | 너 |
| 159 | 36683 | 래 | 202 | 26071 | 호   | 245 | 20300 | 번 | 288 | 16958 | 손 |
| 160 | 36256 | 영 | 203 | 25945 | 날   | 246 | 20265 | 독 | 289 | 16924 | 증 |
| 161 | 36198 | 역 | 204 | 25704 | 설   | 247 | 20264 | 너 | 290 | 16813 | 필 |
| 162 | 35313 | 노 | 205 | 25597 | 난   | 248 | 20175 | 죽 | 291 | 16628 | 준 |
| 163 | 35233 | 터 | 206 | 25570 | 령   | 249 | 20142 | 매 | 292 | 16544 | 6 |
| 164 | 35156 | 차 | 207 | 25435 | 함   | 250 | 20107 | 포 | 293 | 16361 | 디 |
| 165 | 34976 | 2 | 208 | 25259 | 강   | 251 | 20054 | 초 | 294 | 16272 | 책 |
| 166 | 34864 | 금 | 209 | 25203 | a   | 252 | 20030 | 외 | 295 | 15910 | 랑 |
| 167 | 34763 | 양 | 210 | 24943 | 업   | 253 | 20000 | 돌 | 296 | 15904 | 병 |
| 168 | 34325 | 력 | 211 | 24620 | 술   | 254 | 19706 | 록 | 297 | 15892 | 확 |
| 169 | 33905 | 른 | 212 | 24562 | i   | 255 | 19652 | t | 298 | 15757 | 걸 |
| 170 | 33881 | 질 | 213 | 24402 | 란   | 256 | 19509 | 출 | 299 | 15715 | 극 |
| 171 | 33413 | 런 | 214 | 24319 | 언   | 257 | 19446 | 피 | 300 | 15692 | 듯 |
| 172 | 33412 | 9 | 215 | 23574 | 합   | 258 | 19438 | 석 | 301 | 15602 | 7 |
| 173 | 33174 | 집 | 216 | 23464 | 새   | 259 | 19405 | 챌 | 302 | 15526 | 잘 |
| 174 | 33106 | - | 217 | 23458 | 표   | 260 | 19398 | 순 | 303 | 15520 | 올 |
| 175 | 32692 | e | 218 | 23354 | 변   | 261 | 19238 | 존 | 304 | 15463 | 앞 |
| 176 | 32605 | 등 | 219 | 23261 | 목   | 262 | 19178 | 4 | 305 | 15368 | 환 |
| 177 | 32318 | 형 | 220 | 23200 | 파   | 263 | 19169 | 린 | 306 | 15307 | 프 |
| 178 | 31888 | 직 | 221 | 22521 | o   | 264 | 19146 | 울 | 307 | 14831 | 왔 |
| 179 | 31668 | 건 | 222 | 22189 | 편   | 265 | 19122 | 름 | 308 | 14796 | 죽 |
| 180 | 31335 | 든 | 223 | 22139 | 5   | 266 | 18868 | 활 | 309 | 14681 | 품 |
| 181 | 31290 | 또 | 224 | 21995 | 많   | 267 | 18822 | 온 | 310 | 14582 | 겨 |
| 182 | 30869 | 남 | 225 | 21949 | r   | 268 | 18697 | 추 | 311 | 14544 | 월 |
| 183 | 30348 | 법 | 226 | 21734 | 받   | 269 | 18633 | 키 | 312 | 14482 | 창 |
| 184 | 30338 | 예 | 227 | 21718 | 약   | 270 | 18532 | 특 | 313 | 14392 | 별 |
| 185 | 30254 | 론 | 228 | 21713 | 머   | 271 | 18325 | 님 | 314 | 14349 | 토 |
| 186 | 29938 | 태 | 229 | 21679 | 능   | 272 | 18280 | 얼 | 315 | 14347 | 백 |
| 187 | 29891 | 점 | 230 | 21622 | ... | 273 | 18181 | s | 316 | 14298 | 담 |
| 188 | 29660 | 타 | 231 | 21451 | n   | 274 | 18144 | 눈 | 317 | 14274 | 면 |
| 189 | 29173 | 감 | 232 | 21207 | 절   | 275 | 18049 | 평 | 318 | 14154 | 망 |
| 190 | 29053 | 따 | 233 | 21170 | 천   | 276 | 17994 | 열 | 319 | 14148 | 청 |
| 191 | 28506 | 못 | 234 | 21063 | 군   | 277 | 17954 | 길 | 320 | 14018 | 중 |

|     |       |   |     |       |   |     |      |   |     |      |   |
|-----|-------|---|-----|-------|---|-----|------|---|-----|------|---|
| 321 | 13996 | 침 | 364 | 11391 | 젖 | 407 | 9666 | 색 | 450 | 7433 | 옥 |
| 322 | 13988 | 움 | 365 | 11381 | 즉 | 408 | 9654 | 왜 | 451 | 7391 | 떨 |
| 323 | 13954 | 위 | 366 | 11378 | 투 | 409 | 9551 | 령 | 452 | 7276 | 봉 |
| 324 | 13945 | 애 | 367 | 11263 | 논 | 410 | 9417 | 므 | 453 | 7237 | 옥 |
| 325 | 13935 | 철 | 368 | 11200 | 항 | 411 | 9244 | 긴 | 454 | 7212 | 빛 |
| 326 | 13757 | l | 369 | 11145 | 냐 | 412 | 9119 | 완 | 455 | 7168 | 숙 |
| 327 | 13663 | 참 | 370 | 11108 | 갖 | 413 | 9094 | > | 456 | 7085 | 효 |
| 328 | 13556 | 협 | 371 | 11079 | 승 | 414 | 9071 | < | 457 | 7051 | 얕 |
| 329 | 13448 | 농 | 372 | 11032 | 귀 | 415 | 8947 | 잠 | 458 | 7039 | 송 |
| 330 | 13400 | 친 | 373 | 10931 | 접 | 416 | 8923 | 혁 | 459 | 6962 | g |
| 331 | 13094 | 곳 | 374 | 10889 | 코 | 417 | 8906 | 맞 | 460 | 6817 | 익 |
| 332 | 13059 | 흔 | 375 | 10858 | 규 | 418 | 8875 | 답 | 461 | 6812 | 총 |
| 333 | 12971 | ! | 376 | 10854 | · | 419 | 8844 | 몇 | 462 | 6770 | 몰 |
| 334 | 12957 | 잡 | 377 | 10845 | 료 | 420 | 8741 | 엄 | 463 | 6719 | 착 |
| 335 | 12898 | 뿐 | 378 | 10741 | 별 | 421 | 8723 | 끝 | 464 | 6648 | 먼 |
| 336 | 12884 | 견 | 379 | 10639 | u | 422 | 8722 | 덕 | 465 | 6639 | 웃 |
| 337 | 12864 | 박 | 380 | 10586 | 불 | 423 | 8678 | 농 | 466 | 6637 | 찰 |
| 338 | 12843 | 런 | 381 | 10551 | 씨 | 424 | 8651 | h | 467 | 6578 | 곧 |
| 339 | 12834 | 렸 | 382 | 10542 | 충 | 425 | 8530 | 축 | 468 | 6559 | 돈 |
| 340 | 12735 | 갈 | 383 | 10529 | 왕 | 426 | 8469 | 범 | 469 | 6558 | 십 |
| 341 | 12605 | 족 | 384 | 10445 | 굴 | 427 | 8362 | 깨 | 470 | 6539 | 택 |
| 342 | 12531 | 황 | 385 | 10395 | m | 428 | 8291 | 찬 | 471 | 6539 | 층 |
| 343 | 12494 | [ | 386 | 10371 | 찾 | 429 | 8262 | 닌 | 472 | 6536 | 늘 |
| 344 | 12465 | ] | 387 | 10231 | 삼 | 430 | 8222 | 북 | 473 | 6531 | 곧 |
| 345 | 12399 | 쟁 | 388 | 10206 | 몸 | 431 | 8202 | 났 | 474 | 6513 | 측 |
| 346 | 12371 | 씨 | 389 | 10195 | 림 | 432 | 8179 | 땡 | 475 | 6446 | 꾸 |
| 347 | 12357 | 악 | 390 | 10166 | 쓰 | 433 | 8161 | d | 476 | 6366 | 짓 |
| 348 | 12350 | 먹 | 391 | 10164 | 밖 | 434 | 8129 | 혹 | 477 | 6360 | 량 |
| 349 | 12278 | 최 | 392 | 10129 | 테 | 435 | 8089 | 글 | 478 | 6303 | 염 |
| 350 | 12256 | 막 | 393 | 10127 | 엇 | 436 | 8076 | 억 | 479 | 6291 | 암 |
| 351 | 12224 | c | 394 | 10034 | 광 | 437 | 8042 | 럽 | 480 | 6263 | 슨 |
| 352 | 12165 | 깨 | 395 | 10032 | 김 | 438 | 8025 | 빠 | 481 | 6256 | 슬 |
| 353 | 12070 | 옥 | 396 | 10004 | 취 | 439 | 7950 | 켜 | 482 | 6253 | 싸 |
| 354 | 12000 | 채 | 397 | 9981  | 끼 | 440 | 7944 | 응 | 483 | 6231 | 불 |
| 355 | 11923 | 넘 | 398 | 9977  | 갸 | 441 | 7889 | p | 484 | 6224 | 풀 |
| 356 | 11902 | 쳐 | 399 | 9916  | 락 | 442 | 7838 | 득 | 485 | 6167 | 검 |
| 357 | 11898 | : | 400 | 9900  | 카 | 443 | 7795 | 「 | 486 | 6156 | 베 |
| 358 | 11815 | 줄 | 401 | 9842  | 큰 | 444 | 7771 | 」 | 487 | 6126 | 넘 |
| 359 | 11800 | 뒤 | 402 | 9771  | 째 | 445 | 7723 | 둘 | 488 | 6120 | 커 |
| 360 | 11665 | 떠 | 403 | 9707  | 힘 | 446 | 7709 | 싶 | 489 | 6077 | 땅 |
| 361 | 11567 | 립 | 404 | 9686  | 희 | 447 | 7654 | 풍 | 490 | 6032 | 탄 |
| 362 | 11519 | 류 | 405 | 9683  | 혀 | 448 | 7616 | 뜻 | 491 | 6006 | 삶 |
| 363 | 11419 | 허 | 406 | 9677  | 밀 | 449 | 7535 | 높 | 492 | 5983 | 밤 |

|     |      |   |     |      |   |     |      |   |
|-----|------|---|-----|------|---|-----|------|---|
| 493 | 5941 | 큼 | 536 | 4655 | S | 579 | 3890 | M |
| 494 | 5831 | 릴 | 537 | 4635 | 밭 | 580 | 3868 | b |
| 495 | 5817 | 례 | 538 | 4634 | 압 | 581 | 3861 | 쉽 |
| 496 | 5802 | 얼 | 539 | 4579 | 숨 | 582 | 3859 | 휘 |
| 497 | 5797 | 숨 | 540 | 4524 | 획 | 583 | 3821 | 젊 |
| 498 | 5780 | 밋 | 541 | 4493 | 증 | 584 | 3803 | 랏 |
| 499 | 5765 | 척 | 542 | 4475 | 티 | 585 | 3795 | 뛰 |
| 500 | 5761 | 깊 | 543 | 4442 | 양 | 586 | 3790 | 혜 |
| 501 | 5733 | 틀 | 544 | 4439 | 호 | 587 | 3764 | 렵 |
| 502 | 5716 | 협 | 545 | 4435 | T | 588 | 3720 | 럿 |
| 503 | 5715 | ; | 546 | 4423 | 엔 | 589 | 3702 | 케 |
| 504 | 5689 | 곤 | 547 | 4404 | 죄 | 590 | 3694 | 퍼 |
| 505 | 5612 | 민 | 548 | 4402 | 척 | 591 | 3690 | 옷 |
| 506 | 5595 | 궁 | 549 | 4395 | B | 592 | 3665 | 킨 |
| 507 | 5592 | 징 | 550 | 4350 | 꽃 | 593 | 3654 | % |
| 508 | 5585 | 좁 | 551 | 4335 | 흥 | 594 | 3653 | 탕 |
| 509 | 5564 | 패 | 552 | 4333 | 넌 | 595 | 3643 | 찌 |
| 510 | 5509 | 윤 | 553 | 4327 | 헌 | 596 | 3621 | 쳐 |
| 511 | 5505 | 메 | 554 | 4268 | 퍼 | 597 | 3594 | 작 |
| 512 | 5490 | 벽 | 555 | 4236 | 빨 | 598 | 3582 | 혈 |
| 513 | 5438 | A | 556 | 4234 | 씩 | 599 | 3580 | 냥 |
| 514 | 5373 | 칠 | 557 | 4218 | 섯 | 600 | 3569 | 옛 |
| 515 | 5370 | 잔 | 558 | 4209 | { |     |      |   |
| 516 | 5355 | 팔 | 559 | 4196 | 들 |     |      |   |
| 517 | 5287 | 갑 | 560 | 4186 | 폐 |     |      |   |
| 518 | 5283 | 폭 | 561 | 4181 | — |     |      |   |
| 519 | 5182 | ” | 562 | 4159 | 벗 |     |      |   |
| 520 | 5170 | 탈 | 563 | 4138 | 웠 |     |      |   |
| 521 | 5131 | “ | 564 | 4138 | 객 |     |      |   |
| 522 | 5078 | 곡 | 565 | 4131 | 읽 |     |      |   |
| 523 | 5039 | 짐 | 566 | 4109 | 빈 |     |      |   |
| 524 | 5026 | 피 | 567 | 4065 | } |     |      |   |
| 525 | 5017 | 혜 | 568 | 4062 | 옥 |     |      |   |
| 526 | 4898 | C | 569 | 4057 | 홀 |     |      |   |
| 527 | 4881 | 끝 | 570 | 4045 | 균 |     |      |   |
| 528 | 4834 | 놈 | 571 | 4014 | 닥 |     |      |   |
| 529 | 4816 | 탁 | 572 | 4002 | P |     |      |   |
| 530 | 4766 | 흔 | 573 | 3991 | 짜 |     |      |   |
| 531 | 4728 | 울 | 574 | 3990 | 롭 |     |      |   |
| 532 | 4716 | 뜨 | 575 | 3958 | 쉬 |     |      |   |
| 533 | 4711 | 첫 | 576 | 3943 | 좌 |     |      |   |
| 534 | 4706 | 브 | 577 | 3914 | f |     |      |   |
| 535 | 4667 | y | 578 | 3911 | / |     |      |   |

Annexe 6. Les statistiques sur les mots du corpus du KAIST  
 ( sur 1344018 mots différents : les 600 les plus fréquents )

|    |       |     |    |       |      |     |      |     |
|----|-------|-----|----|-------|------|-----|------|-----|
| 1  | 76536 | 그   | 39 | 11149 | 된다   | 77  | 6887 | 않았다 |
| 2  | 65507 | 수   | 40 | 11070 | 다시   | 78  | 6884 | 여러  |
| 3  | 53598 | 이   | 41 | 10346 | 했다   | 79  | 6833 | 그렇게 |
| 4  | 52745 | 있는  | 42 | 10283 | 그것은  | 80  | 6758 | 함께  |
| 5  | 50837 | 있다  | 43 | 10043 | 그런   | 81  | 6664 | 모두  |
| 6  | 46907 | 것이다 | 44 | 9688  | 이런   | 82  | 6603 | 안   |
| 7  | 30765 | 한   | 45 | 9577  | 내가   | 83  | 6535 | 그래서 |
| 8  | 26950 | 것은  | 46 | 9549  | 있을   | 84  | 6494 | 다   |
| 9  | 25508 | 그러나 | 47 | 9539  | 가장   | 85  | 6473 | 수가  |
| 10 | 24296 | 것이  | 48 | 9487  | 많은   | 86  | 6432 | 또는  |
| 11 | 23538 | 있었다 | 49 | 9480  | 제    | 87  | 6423 | 불   |
| 12 | 21769 | 하는  | 50 | 9385  | 죽    | 88  | 6421 | 의해  |
| 13 | 20846 | 대한  | 51 | 9324  | 잘    | 89  | 6386 | 에   |
| 14 | 20642 | 그리고 | 52 | 9029  | 우리는  | 90  | 6375 | 그런데 |
| 15 | 20014 | 것을  | 53 | 8768  | 내    | 91  | 6367 | 그가  |
| 16 | 19759 | 한다  | 54 | 8605  | 바로   | 92  | 6354 | 이미  |
| 17 | 19468 | 할   | 55 | 8504  | 또한   | 93  | 6309 | 것도  |
| 18 | 19083 | 나는  | 56 | 8279  | 것으로  | 94  | 6281 | 을   |
| 19 | 19063 | 그는  | 57 | 8270  | 될    | 95  | 6222 | 가지  |
| 20 | 18638 | 같은  | 58 | 8044  | 때문이다 | 96  | 6180 | 가지고 |
| 21 | 15764 | 다른  | 59 | 7992  | 어느   | 97  | 6178 | 데   |
| 22 | 15226 | 때   | 60 | 7896  | 없었다  | 98  | 6106 | 된   |
| 23 | 14565 | 모든  | 61 | 7756  | 큰    | 99  | 6087 | 아니다 |
| 24 | 14498 | 년   | 62 | 7717  | 따라서  | 100 | 6057 | 않는  |
| 25 | 14422 | 이러한 | 63 | 7689  | 위해   | 101 | 5995 | 되어  |
| 26 | 14394 | 또   | 64 | 7689  | 따라   | 102 | 5943 | 속에서 |
| 27 | 14391 | 하고  | 65 | 7577  | 는    | 103 | 5884 | 속에  |
| 28 | 13937 | 없는  | 66 | 7518  | 자신의  | 104 | 5834 | 일이  |
| 29 | 13912 | 때문에 | 67 | 7516  | 않고   | 105 | 5829 | 위한  |
| 30 | 13798 | 어떤  | 68 | 7450  | 어떻게  | 106 | 5818 | 사람은 |
| 31 | 13532 | 더   | 69 | 7339  | 있다는  | 107 | 5809 | 있고  |
| 32 | 13168 | 그의  | 70 | 7322  | 되는   | 108 | 5803 | 자기  |
| 33 | 13039 | 것   | 71 | 7308  | 같이   | 109 | 5769 | 그것을 |
| 34 | 11823 | 두   | 72 | 7217  | 새로운  | 110 | 5733 | 우리가 |
| 35 | 11699 | 아니라 | 73 | 7099  | 사람이  | 111 | 5720 | 등   |
| 36 | 11348 | 없다  | 74 | 7090  | 되었다  | 112 | 5682 | 대해  |
| 37 | 11343 | 우리  | 75 | 7057  | 이렇게  | 113 | 5679 | 밋   |
| 38 | 11300 | 의   | 76 | 6929  | 말을   | 114 | 5649 | 그들은 |

|     |      |      |     |      |      |     |      |     |
|-----|------|------|-----|------|------|-----|------|-----|
| 115 | 5641 | 가    | 158 | 4462 | 사람들이 | 201 | 3430 | 얼마나 |
| 116 | 5639 | 몇    | 159 | 4430 | 관한   | 202 | 3429 | 오히려 |
| 117 | 5599 | 하지만  | 160 | 4420 | 동안   | 203 | 3417 | 저   |
| 118 | 5507 | 일    | 161 | 4413 | 거의   | 204 | 3416 | 너무  |
| 119 | 5495 | 를    | 162 | 4408 | 그들의  | 205 | 3411 | 작은  |
| 120 | 5467 | 것입니다 | 163 | 4396 | 알고   | 206 | 3410 | 말이  |
| 121 | 5450 | 것이였다 | 164 | 4365 | 라는   | 207 | 3389 | 마치  |
| 122 | 5294 | 더욱   | 165 | 4324 | 매우   | 208 | 3343 | 있는데 |
| 123 | 5279 | 특히   | 166 | 4292 | 은    | 209 | 3341 | 있기  |
| 124 | 5278 | 이것은  | 167 | 4283 | 않을   | 210 | 3329 | 사실을 |
| 125 | 5229 | 없이   | 168 | 4274 | 보고   | 211 | 3309 | 이는  |
| 126 | 5198 | 우리의  | 169 | 4272 | 좋은   | 212 | 3281 | 전혀  |
| 127 | 5179 | 물론   | 170 | 4219 | 혹은   | 213 | 3269 | 그대로 |
| 128 | 5115 | 않은   | 171 | 4213 | 온    | 214 | 3243 | 나   |
| 129 | 5042 | 인간의  | 172 | 4180 | 통해   | 215 | 3235 | 않으면 |
| 130 | 5021 | 수도   | 173 | 4171 | 있던   | 216 | 3207 | 크게  |
| 131 | 5007 | 말했다  | 174 | 4156 | 나의   | 217 | 3202 | 동시에 |
| 132 | 5001 | 곧    | 175 | 4091 | 세    | 218 | 3200 | 위에  |
| 133 | 4967 | 알    | 176 | 4077 | 게    | 219 | 3198 | 주는  |
| 134 | 4960 | 하나의  | 177 | 4066 | 사람의  | 220 | 3182 | 해도  |
| 135 | 4947 | 보면   | 178 | 4039 | 들어   | 221 | 3174 | 중   |
| 136 | 4944 | 중요한  | 179 | 4013 | 지금   | 222 | 3154 | 전에  |
| 137 | 4943 | 라고   | 180 | 4008 | 하면   | 223 | 3150 | 있지  |
| 138 | 4934 | 월    | 181 | 3972 | 사회적  | 224 | 3139 | 있었던 |
| 139 | 4917 | 그것이  | 182 | 3819 | 후    | 225 | 3087 | 위하여 |
| 140 | 4864 | 않는다  | 183 | 3819 | 채    | 226 | 3082 | 결국  |
| 141 | 4850 | 있습니다 | 184 | 3813 | 이를   | 227 | 3080 | 수는  |
| 142 | 4805 | 이제   | 185 | 3810 | 있으며  | 228 | 3074 | 네   |
| 143 | 4784 | 하지   | 186 | 3803 | 되고   | 229 | 3072 | 때는  |
| 144 | 4761 | 많이   | 187 | 3800 | 이와   | 230 | 3052 | 자신이 |
| 145 | 4754 | 있어서  | 188 | 3793 | 보다   | 231 | 3052 | 그를  |
| 146 | 4750 | 왜    | 189 | 3771 | 해서   | 232 | 3046 | 사람  |
| 147 | 4728 | 있다고  | 190 | 3727 | 경우   | 233 | 3045 | 눈을  |
| 148 | 4687 | 무슨   | 191 | 3693 | 와    | 234 | 3010 | 결코  |
| 149 | 4678 | 일을   | 192 | 3677 | 줄    | 235 | 3007 | 못   |
| 150 | 4636 | 서로   | 193 | 3639 | 난    | 236 | 3004 | 이에  |
| 151 | 4618 | 사람들은 | 194 | 3619 | 먼저   | 237 | 2994 | 해   |
| 152 | 4578 | 그녀는  | 195 | 3521 | 나를   | 238 | 2987 | 고   |
| 153 | 4552 | 있어   | 196 | 3505 | 다음과  | 239 | 2984 | 하나  |
| 154 | 4493 | 하였다  | 197 | 3493 | 하여   | 240 | 2983 | 다음  |
| 155 | 4476 | 그러한  | 198 | 3488 | 좁    | 241 | 2930 | 사회  |
| 156 | 4472 | 아주   | 199 | 3481 | 그러므로 | 242 | 2905 | 것과  |
| 157 | 4466 | 역시   | 200 | 3480 | 것처럼  | 243 | 2905 | 갖고  |

|     |      |      |     |      |      |     |      |      |
|-----|------|------|-----|------|------|-----|------|------|
| 244 | 2902 | 에서   | 287 | 2606 | 말할   | 330 | 2222 | 없다는  |
| 245 | 2896 | 한다는  | 288 | 2603 | 스스로  | 331 | 2219 | 아무리  |
| 246 | 2887 | 하며   | 289 | 2599 | 그들이  | 332 | 2213 | 이들   |
| 247 | 2882 | 보는   | 290 | 2591 | 손을   | 333 | 2210 | 있지만  |
| 248 | 2880 | 못하고  | 291 | 2561 | 이상   | 334 | 2201 | 훨씬   |
| 249 | 2873 | 가진   | 292 | 2553 | 과    | 335 | 2200 | 생각이  |
| 250 | 2868 | 등의   | 293 | 2540 | 완전히  | 336 | 2200 | 다만   |
| 251 | 2859 | 같다   | 294 | 2535 | 계속   | 337 | 2198 | 사람들의 |
| 252 | 2844 | 되면   | 295 | 2511 | 우선   | 338 | 2197 | 어떠한  |
| 253 | 2834 | 아직   | 296 | 2505 | 없고   | 339 | 2188 | 이후   |
| 254 | 2827 | 사람을  | 297 | 2503 | 그때   | 340 | 2184 | 것에   |
| 255 | 2822 | 의한   | 298 | 2495 | 사는   | 341 | 2177 | 사회의  |
| 256 | 2813 | 높은   | 299 | 2493 | 앞에   | 342 | 2175 | 쉽게   |
| 257 | 2806 | 날    | 300 | 2488 | 듯    | 343 | 2161 | 만일   |
| 258 | 2800 | 없을   | 301 | 2429 | 전    | 344 | 2158 | 반드시  |
| 259 | 2790 | 등을   | 302 | 2424 | 사이에  | 345 | 2153 | 서    |
| 260 | 2783 | 아닌   | 303 | 2412 | 사실   | 346 | 2152 | 말은   |
| 261 | 2763 | 왜냐하면 | 304 | 2402 | 정말   | 347 | 2150 | 것이며  |
| 262 | 2761 | 개의   | 305 | 2394 | 위해서  | 348 | 2149 | 길을   |
| 263 | 2755 | 못한   | 306 | 2390 | 일은   | 349 | 2144 | 것인가  |
| 264 | 2744 | 몸을   | 307 | 2388 | 대하여  | 350 | 2139 | 않다   |
| 265 | 2736 | 가는   | 308 | 2385 | 문제가  | 351 | 2134 | 모른다  |
| 266 | 2712 | 있게   | 309 | 2383 | 아니   | 352 | 2131 | 깊은   |
| 267 | 2703 | 본    | 310 | 2381 | 일본   | 353 | 2130 | 필요한  |
| 268 | 2701 | 각    | 311 | 2376 | 위해서는 | 354 | 2130 | 말하는  |
| 269 | 2699 | 년에   | 312 | 2360 | 앞으로  | 355 | 2122 | 자기의  |
| 270 | 2693 | 정치적  | 313 | 2328 | 언제나  | 356 | 2117 | 아니었다 |
| 271 | 2686 | 가운데  | 314 | 2290 | 마음을  | 357 | 2106 | 일이다  |
| 272 | 2673 | 항상   | 315 | 2289 | 수밖에  | 358 | 2105 | 갑자기  |
| 273 | 2671 | 번    | 316 | 2285 | 두고   | 359 | 2102 | 가서   |
| 274 | 2669 | 그녀의  | 317 | 2282 | 왔다   | 360 | 2095 | 아니면  |
| 275 | 2658 | 것이라고 | 318 | 2272 | 한번   | 361 | 2091 | 모습을  |
| 276 | 2655 | 이라는  | 319 | 2271 | 불구하고 | 362 | 2088 | 그래   |
| 277 | 2653 | 시작했다 | 320 | 2270 | 소리를  | 363 | 2082 | 정도로  |
| 278 | 2646 | 예를   | 321 | 2259 | 생각을  | 364 | 2079 | 단지   |
| 279 | 2638 | 로    | 322 | 2256 | 세계   | 365 | 2073 | 주고   |
| 280 | 2637 | 영향을  | 323 | 2251 | 앉아   | 366 | 2070 | 등이   |
| 281 | 2637 | 여기서  | 324 | 2249 | 문제를  | 367 | 2068 | 장    |
| 282 | 2628 | 이것이  | 325 | 2248 | 꼭    | 368 | 2063 | 이라고  |
| 283 | 2621 | 당시   | 326 | 2245 | 해야   | 369 | 2057 | 자신을  |
| 284 | 2613 | 말    | 327 | 2239 | 뿐만   | 370 | 2052 | 앞에서  |
| 285 | 2610 | 합니다  | 328 | 2239 | 만들어  | 371 | 2048 | 대해서  |
| 286 | 2610 | 만    | 329 | 2227 | 만큼   | 372 | 2045 | 의미를  |

|     |      |      |     |      |       |     |      |      |
|-----|------|------|-----|------|-------|-----|------|------|
| 373 | 2032 | 의하여  | 416 | 1850 | 그리하여  | 459 | 1694 | 사실은  |
| 374 | 2032 | 년대   | 417 | 1849 | 문제는   | 460 | 1690 | 향해   |
| 375 | 2023 | 적이   | 418 | 1844 | 관심을   | 461 | 1690 | 주었다  |
| 376 | 2022 | 필요가  | 419 | 1842 | 처음    | 462 | 1686 | 잠시   |
| 377 | 2022 | 그러면  | 420 | 1842 | 실제로   | 463 | 1684 | 커다란  |
| 378 | 2015 | 인간   | 421 | 1842 | 달리    | 464 | 1683 | 속으로  |
| 379 | 2013 | 힘을   | 422 | 1837 | 아니고   | 465 | 1683 | 대해서는 |
| 380 | 2012 | 뿐    | 423 | 1819 | 물었다   | 466 | 1674 | 의하면  |
| 381 | 2007 | 고개를  | 424 | 1813 | 차     | 467 | 1673 | 대    |
| 382 | 1998 | 삶의   | 425 | 1809 | 정도    | 468 | 1666 | 거야   |
| 383 | 1987 | 얼굴을  | 426 | 1809 | 만약    | 469 | 1664 | 하면서  |
| 384 | 1975 | 말한다  | 427 | 1808 | 으로    | 470 | 1664 | 나와   |
| 385 | 1972 | 지난   | 428 | 1805 | 남아    | 471 | 1659 | 생각해  |
| 386 | 1970 | 이야기를 | 429 | 1800 | 관계를   | 472 | 1651 | 한다고  |
| 387 | 1966 | 역사적  | 430 | 1799 | 듯이    | 473 | 1649 | 있다면  |
| 388 | 1962 | 세기   | 431 | 1787 | 의해서   | 474 | 1644 | 바    |
| 389 | 1955 | 살고   | 432 | 1787 | 눈에    | 475 | 1643 | 돈을   |
| 390 | 1955 | 되지   | 433 | 1784 | 안에    | 476 | 1640 | 시간   |
| 391 | 1952 | 누가   | 434 | 1783 | 통하여   | 477 | 1640 | 새    |
| 392 | 1940 | 미국의  | 435 | 1775 | 놓고    | 478 | 1638 | 그들   |
| 393 | 1934 | 뿐이다  | 436 | 1774 | 여전히   | 479 | 1637 | 그냥   |
| 394 | 1915 | 한편   | 437 | 1774 | 늘     | 480 | 1633 | 이처럼  |
| 395 | 1908 | 있음을  | 438 | 1772 | 듯한    | 481 | 1632 | 점을   |
| 396 | 1908 | 보이는  | 439 | 1771 | 주로    | 482 | 1627 | 하더라도 |
| 397 | 1907 | 들고   | 440 | 1765 | 인간은   | 483 | 1627 | 보인다  |
| 398 | 1903 | 건    | 441 | 1762 | 현재    | 484 | 1622 | 사이의  |
| 399 | 1900 | 비록   | 442 | 1760 | 아직도   | 485 | 1620 | 입을   |
| 400 | 1900 | 걸    | 443 | 1752 | 대로    | 486 | 1619 | 살아   |
| 401 | 1896 | 같았다  | 444 | 1750 | 인간이   | 487 | 1618 | 개    |
| 402 | 1889 | 그렇지  | 445 | 1749 | 경제적   | 488 | 1616 | 미국   |
| 403 | 1887 | 직접   | 446 | 1745 | 집에    | 489 | 1613 | 못했다  |
| 404 | 1884 | 만한   | 447 | 1741 | 그렇다면  | 490 | 1612 | 모르는  |
| 405 | 1881 | 어려운  | 448 | 1736 | 다양한   | 491 | 1597 | 보이지  |
| 406 | 1880 | 젊은   | 449 | 1734 | 있었고   | 492 | 1595 | 받고   |
| 407 | 1880 | 살    | 450 | 1731 | 한국    | 493 | 1592 | 있는가  |
| 408 | 1877 | 소리가  | 451 | 1731 | 집     | 494 | 1589 | 분명히  |
| 409 | 1867 | 약    | 452 | 1726 | 하기    | 495 | 1587 | 흔히   |
| 410 | 1866 | 일본의  | 453 | 1720 | 일에    | 496 | 1580 | 후에   |
| 411 | 1866 | 역할을  | 454 | 1718 | 보였다   | 497 | 1579 | 말이다  |
| 412 | 1865 | 보았다  | 455 | 1717 | 못하는   | 498 | 1572 | 아    |
| 413 | 1862 | 아무런  | 456 | 1716 | 마음이   | 499 | 1571 | 점에서  |
| 414 | 1853 | 이것을  | 457 | 1704 | 무엇을   | 500 | 1569 | 명의   |
| 415 | 1850 | 아무   | 458 | 1701 | 마찬가지로 | 501 | 1567 | 밖에   |

|     |      |      |     |      |       |     |      |     |
|-----|------|------|-----|------|-------|-----|------|-----|
| 502 | 1566 | 시간이  | 545 | 1474 | 생각하고  | 588 | 1376 | 이상의 |
| 503 | 1565 | 타고   | 546 | 1473 | 했던    | 589 | 1376 | 단순한 |
| 504 | 1565 | 있어야  | 547 | 1473 | 백     | 590 | 1375 | 보니  |
| 505 | 1561 | 오는   | 548 | 1472 | 없다고   | 591 | 1374 | 있도록 |
| 506 | 1560 | 체대로  | 549 | 1472 | 생각하는  | 592 | 1374 | 나온  |
| 507 | 1554 | 어린   | 550 | 1469 | 오늘    | 593 | 1373 | 눈이  |
| 508 | 1551 | 경우가  | 551 | 1467 | 중에서   | 594 | 1372 | 벌써  |
| 509 | 1547 | 자    | 552 | 1466 | 알게    | 595 | 1371 | 오래  |
| 510 | 1542 | 지니고  | 553 | 1463 | 않게    | 596 | 1369 | 많다  |
| 511 | 1542 | 있으면  | 554 | 1463 | 뒤에    | 597 | 1366 | 마음  |
| 512 | 1535 | 제가   | 555 | 1461 | 시간을   | 598 | 1365 | 중의  |
| 513 | 1535 | 들었다  | 556 | 1461 | 나오는   | 599 | 1365 | 싫은  |
| 514 | 1533 | 그녀가  | 557 | 1460 | 통해서   | 600 | 1365 | 순간  |
| 515 | 1532 | 아는   | 558 | 1459 | 하계    |     |      |     |
| 516 | 1531 | 만든   | 559 | 1449 | 좀더    |     |      |     |
| 517 | 1529 | 천    | 560 | 1449 | 방법을   |     |      |     |
| 518 | 1529 | 점점   | 561 | 1448 | 그와    |     |      |     |
| 519 | 1526 | 자기가  | 562 | 1447 | 나도    |     |      |     |
| 520 | 1526 | 그럼   | 563 | 1444 | 모두가   |     |      |     |
| 521 | 1521 | 듣고   | 564 | 1442 | 갔다    |     |      |     |
| 522 | 1521 | 그래도  | 565 | 1440 | 갈     |     |      |     |
| 523 | 1517 | 머리를  | 566 | 1439 | 여기에   |     |      |     |
| 524 | 1517 | 뒤    | 567 | 1436 | 빨리    |     |      |     |
| 525 | 1516 | 아닌가  | 568 | 1430 | 책을    |     |      |     |
| 526 | 1514 | 자주   | 569 | 1427 | 그럴    |     |      |     |
| 527 | 1510 | 경우에는 | 570 | 1414 | 때문이였다 |     |      |     |
| 528 | 1509 | 저는   | 571 | 1413 | 나라    |     |      |     |
| 529 | 1509 | 그저   | 572 | 1413 | 갖는    |     |      |     |
| 530 | 1503 | 과연   | 573 | 1411 | 말하고   |     |      |     |
| 531 | 1501 | 것이고  | 574 | 1405 | 죽은    |     |      |     |
| 532 | 1498 | 나서   | 575 | 1404 | 정치    |     |      |     |
| 533 | 1497 | 조금   | 576 | 1402 | 하다    |     |      |     |
| 534 | 1497 | 말고   | 577 | 1401 | 받은    |     |      |     |
| 535 | 1493 | 그에게  | 578 | 1393 | 자체가   |     |      |     |
| 536 | 1492 | 오직   | 579 | 1393 | 무엇인가  |     |      |     |
| 537 | 1490 | 와서   | 580 | 1389 | 위에서   |     |      |     |
| 538 | 1490 | 먹고   | 581 | 1385 | 하여금   |     |      |     |
| 539 | 1487 | 하나는  | 582 | 1385 | 문제에   |     |      |     |
| 540 | 1486 | 그렇지만 | 583 | 1384 | 그에    |     |      |     |
| 541 | 1481 | 비로소  | 584 | 1382 | 마침내   |     |      |     |
| 542 | 1479 | 이다   | 585 | 1380 | 곳에    |     |      |     |
| 543 | 1479 | 오늘날  | 586 | 1379 | 아름다운  |     |      |     |
| 544 | 1479 | 말았다  | 587 | 1379 | 물을    |     |      |     |