NORME INTERNATIONALE

ISO/CEI 14651

Deuxième édition
2007-12-01

AMENDEMENT 1
2008-10-24

Technologies de l'information — Classement international et comparaison de chaînes de caractères — Méthode de comparaison de chaînes de caractères et description du modèle commun et adaptable d'ordre de classement

AMENDEMENT 1

Information technology — International string ordering and comparison — Method for comparing character strings and description of the common template tailorable ordering

AMENDMENT 1

Numéro de référence
ISO/CEI 14651:2001/Amd.1:2003(F)

PDF – Exonération de responsabilité

Le présent fichier PDF peut contenir des polices de caractères intégrées. Conformément aux conditions de licence d'Adobe, ce fichier peut être imprimé ou visualisé, mais ne doit pas être modifié à moins que l'ordinateur employé à cet effet ne bénéficie d'une licence autorisant l'utilisation de ces polices et que celles-ci y soient installées. Lors du téléchargement de ce fichier, les parties concernées acceptent de fait la responsabilité de ne pas enfreindre les conditions de licence d'Adobe. Le Secrétariat central de l'ISO décline toute responsabilité en la matière.

Adobe est une marque déposée d'Adobe Systems Incorporated.

Les détails relatifs aux produits logiciels utilisés pour la création du présent fichier PDF sont disponibles dans la rubrique General Info du fichier; les paramètres de création PDF ont été optimisés pour l'impression. Toutes les mesures ont été prises pour garantir l'exploitation de ce fichier par les comités membres de l'ISO. Dans le cas peu probable où surviendrait un problème d'utilisation, veuillez en informer le Secrétariat central à l'adresse donnée ci-dessous.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax. + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Publié en Suisse

Avant-propos

L'ISO (Organisation internationale de normalisation) et la CEI (Commission électrotechnique internationale) forment le système spécialisé de la normalisation mondiale. Les organismes nationaux membres de l'ISO ou de la CEI participent au développement de Normes internationales par l'intermédiaire des comités techniques créés par l'organisation concernée afin de s'occuper des domaines particuliers de l'activité technique. Les comités techniques de l'ISO et de la CEI collaborent dans des domaines d'intérêt commun. D'autres organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'ISO et la CEI participent également aux travaux. Dans le domaine des technologies de l'information, l'ISO et la CEI ont créé un comité technique mixte, l'ISO/CEI JTC 1.

Les Normes internationales sont rédigées conformément aux règles données dans les Directives ISO/CEI, Partie 2.

La tâche principale du comité technique mixte est d'élaborer les Normes internationales. Les projets de Normes internationales adoptés par le comité technique mixte sont soumis aux organismes nationaux pour vote. Leur publication comme Normes internationales requiert l'approbation de 75 % au moins des organismes nationaux votants.

L'attention est appelée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. L'ISO et la CEI ne sauraient être tenues pour responsables de ne pas avoir identifié de tels droits de propriété et averti de leur existence.

L'Amendement 1 à l'ISO/CEI 14651:2007 a été élaboré par le comité technique mixte ISO/CEI JTC 1, *Technologies de l'information*, sous-comité SC 2, *Jeux de caractères codés*.

Technologies de l'information — Classement international et comparaison de chaînes de caractères — Méthode de comparaison de chaînes de caractères et description du modèle commun et adaptable d'ordre de classement

AMENDEMENT 1

Page 1, article 1

Remplacer la deuxième puce, avec ses notes, par la suivante.

- une table-modèle commune de classement utilisée par la méthode de référence. Cette table décrit un ordre de base pour tous les caractères de l'ISO/CEI 10646:2003 jusqu'à son amendement 4. Tout cela permet de spécifier un ordre complètement déterministe. Cette table constitue le point de départ permettant de préciser un ordre de classement adapté aux règles de classement locales, sans qu'il soit nécessaire de connaître tous les systèmes d'écriture repris dans le JUC.

  NOTE 1 Cette table-modèle commune de classement est destinée à être modifiée pour satisfaire les besoins d'environnements locaux. L'avantage principal de cette pratique, sur le plan mondial, réside dans le fait que, pour d'autres systèmes d'écriture que celui de l'utilisateur, aucune modification n'est nécessaire et que cet ordre demeurera aussi cohérent que possible et prévisible dans un contexte international.

  NOTE 2 Le répertoire de caractères utilisé dans la présente Norme internationale est équivalent à celui du standard Unicode, version 5.1.

Page 2, article 3

Remplacer les références normatives par les suivantes.

ISO/CEI 10646:2003, *Technologies de l'information — Jeu universel de caractères codés sur plusieurs octets (JUC)*

ISO/CEI 10646:2003/Amd 1:2005, *Technologies de l'information — Jeu universel de caractères codés sur plusieurs octets (JUC) — Amendement 1 — Glagolitique, copte, géorgien et autres caractères*

ISO/CEI 10646:2003/Amd 2:2006, *Technologies de l'information — Jeu universel de caractères codés sur plusieurs octets (JUC) — Amendement 2 — N'Ko, phags-pa, phénicien et autres caractères*

ISO/CEI 10646:2003/Amd 3:2008, *Technologies de l'information — Jeu universel de caractères codés sur plusieurs octets (JUC) — Amendement 3 — Leptcha, ol tchiki, saurachtra, vaï et autres caractères*

ISO/CEI 10646:2003/FDAmd 4:2008, *Technologies de l'information — Jeu universel de caractères codés sur plusieurs octets (JUC) — Amendement 4 — Tcham, plaques de jeu et autres caractères*

Page 6, Paragraphe 6.2.2 Ajouter la note suivante après le premier paragraphe :

NOTE Les éléments de tri comportant plus de caractères sont préférés à ceux qui sont plus courts. Par exemple, si un élément de tri comportant plusieurs caractères est défini pour « abc » et qu'un autre est defini pour « ab » ou qu'un autre l'est pour « bc », alors, si « abc » se présente, l'élément de tri pour « abc » s'appliquera et non celui pour « ab » ou « bc ».

Page 12, Paragraphe 6.3.2

Remplacer CF 4 par ce qui suit :

CF 4. Une *table_adaptée* doit contenir exactement un énoncé *début_ordre* suivi plus tard par exactement un énoncé *fin_ordre.* Après que tous les reclassements de lignes ont été effectués, l'énoncé *début_ordre* doit suivre toutes les *définition_de_symbole* et précéder tous les *poids_symbolique*.

Supprimer la ligne commençant pas CF 10. (Ceci impliquera la renumérotation de CF 11, CF 12 et CF 13 dans la prochaine édition).

Remplacer le début de CF 12 par ce qui suit :

CF 12. Tout *symbole_intervalle* doit contenir deux *symbole* remplissant les conditions suivantes : 1) Les deux *symbole* doivent avoir un préfixe commun de longueur 1. Le préfixe peut être n'importe quelle lettre sauf « U ».

Page 16, Paragraphe 6.5

Remplacer le paragraphe 6.5 par le suivant.

6.5 Nom de la table-modèle commune et déclaration de nom

Le nom ISO 14651_2008_TABLE1 doit être utilisé pour désigner la table-modèle commune dans toute référence externe à cette table comme base dans un contexte donné, que ce soit un processus, un contrat ou des exigences d'approvisionnement. Si un autre nom doit être utilisé pour des raisons pratiques, une déclaration de conformité doit indiquer la correspondance entre cet autre nom et le nom ISO 14651_2008_TABLE1.

L'utilisation d'un nom précis est nécessaire pour gérer les différents états de développement de cette table. Ceci découle de la nature du répertoire de caractères de référence, amenée à s'étendre pendant des années, voire des décennies.

Page 17, Annexe A

Remplacer l'Annexe A par la suivante.

Annexe A
(normative)

Table-modèle commune

Dans le but de minimiser les problèmes de formatage et les risques d'erreurs de reproduction, la table-modèle commune est fournie séparément, comme composante normative de la présente Norme internationale. Le nom de fichier dans cette version linguistique de la norme diffère du nom de référence normative précisé en 6.5 de la présente Norme internationale à cause de l'existence de versions du fichier avec des commentaires en d'autres langues. Le fichier pour cette version linguistique peut être récupéré du site web de l'ITTF à l'URL suivant:

http://www.iso.org/ittf/ISO14651_2008_TABLE1_fr.txt

Il existe une version anglaise officielle du fichier qui ne diffère que par les commentaires (le contenu technique est identique) et dont le nom est ISO 14651_2008_TABLE1_en.txt

NOTE 1 La présente norme internationale déconseille mais n'empêche pas la référence aux anciennes tables ISO 14651_2000_TABLE 1 et ISO 14651_2002_TABLE 1, qui contiennent les informations de tri des répertoires des versions précédentes de l'ISO/CEI 10646 et de leurs amendements. Les anciennes tables peuvent être récupérées des URL suivants:

- [informations de tri du répertoire de l'ISO/CEI 10646-1:1993 avec ses amendements 1 à 9]
  http://www.iso.org/ittf/ISO14651_2000_TABLE1.htm.
- Informations de tri du répertoire combiné de l'ISO/CEI 10646-1:2000 et de l'ISO/CEI 10646-2:2001
  http://www.iso.org/ittf/ISO14651_2002_TABLE1_fr.txt
- [informations de tri du répertoire de l'ISO/CEI 10646:2003]
  http://www.iso.org/ittf/ISO14651_2003_TABLE1_fr.txt
- [informations de tri du répertoire de l'ISO/CEI 10646:2003 incluant ses amendements 1 et 2]
  http://www.iso.org/ittf/ISO14651_2006_TABLE1_fr.txt

La table-modèle commune actuelle prend en compte les répertoires de caractères de l'ISO/CEI 10646:2003, y compris ses quatre premiers amendements, tel qu'indiqué dans le domaine d'application.

NOTE 2 Le répertoire visé par la présente norme internationale est équivalent au répertoire du standard Unicode, version 5.1 (voir la publication intitulée *The Unicode Standard Version 5.1*, publiée par le consortium Unicode).

Quand des données de tri applicables à d'autres amendements à l'ISO/CEI 10646:2003 seront disponibles, la présente Norme internationale, avec notamment sa table-modèle commune, sera amendée pour tenir compte du tri des caractères et écritures ajoutés. Pour satisfaire les exigences culturelles de communautés spécifiques, des déclarations de delta devront être appliquées à la table amendée telle que définie dans la présente Norme internationale.

La version courante de la présente Norme internationale désigne cette table sous le nom de **ISO 14651_2008_TABLE1.**

Page 21, remplacer dans la liste en désordre le mot « notres » par le mot « notre ».

Page 24, dans la 2e colonne du tableau, à la 4e ligne à partir de la fin, remplacer « Karl »(2e occurrence dans la table) par « karl ».Page 39. Ajouter l'article C.4 suivant.

C.4 Méthode proposée pour le prétraitement du hangûl

C.4.1 Introduction

La TMC d'inclut pas formellement de pondération pour les syllabes hangûl. Conséquemment, une adaptation de la table et un prétraitement des chaînes est requise pour trier les données hangûl selon l'algorithme donné dans la présente norme.

Une méthode consiste simplement à donner aux caractères syllabiques hangûl modernes des poids primaires en séquence croissante et de normaliser toute chaîne d'entrée contenant des jamos de sorte que toutes les chaînes d'entrée ne contiennent que des caractères syllabiques hangûl précomposés. Cette méthode est rapide et efficace pour des données ne contenant que du hangûl moderne, parce que les syllabes hangûl sont déjà codées dans leur ordre de tri coréen correct.

Pour des données contenant de l'ancien hangûl, la situation est plus compliquée, parce qu'aucune forme de normalisation d'Unicode ne procure des chaînes d'entrée qui pourraient simplement être pondérées élément par élément pour produire des clés appropriées pour le tri. Un prétraitement supplémentaire peut alors être requis pour produire des clés pouvant être utilisées pour obtenir le comportement de tri désiré.

Le problème essentiel est que l'ordre de tri hangûl est syllabique, mais que les syllabes hangûl sont construites d'une séquence de trois jamos : un début de syllabe, un centre de syllabe (une voyelle), et une fin de syllabe optionnelle. Chacun de ces jamos, à son tour, peut consister en une à trois sous-parties, particulièrement en ancien hangûl, qui comporte de nombreux groupes de consonnes ou de voyelles représentés par des jamos simples.

La stratégie de base pour traiter de telles données hangûl consiste à s'assurer d'abord qu''elles sont entièrement représentées par des jamos, de sorte qu'il n'y ait pas de mélange en entrée entre les jamos liants et les caractères syllabiques hangûl précomposés. Cette étape peut se faire en utilisant la forme de normalisation Unicode D pour décomposer tous les caractères syllabiques hangûl précomposés. Par la suite, on a recours à un algorithme de détermination des frontières de syllabe pour identifier toutes ces dernières dans les données.

Une fois que toutes les frontières de syllable coréennes sont determinées dans les données d'entrée pour chaque syllabe, les jamos de début, de milieu et de fin de syllabe peuvent être pondérés pour produire des clés qui, lorsque elles sont comparées, donnent les résultats désirés pour le tri syllabique des chaînes.

.

Idéalement, les données de l'ancien hangûl prétraitées de la sorte pour les décomposer et déterminer les frontières de syllable contiendront exactement un jamo de début, un jamo central et optionnellement un jamo final pour chaque syllabe. Cependant, certains types de données d'entrée, une fois décomposées, pourraient

résulter en plus d'un jamo liant de début, et ainsi de suite. Dans de tels cas, le prétraitement peut impliquer une étape de mappage pour s'assurer que toute séquence de jamos de ce genre est traduite dans le jamo simple de début correspondant censé représenter ce groupe de consonnes (de deux ou de trois sous-parties) de l'ancien hangûl. Les mêmes considérations s'appliquent pour toute séquence de jamos centraux ou finaux dans les données.

Il existe plusieurs stratégies pour pondérer les jamos de début, de centre ou de fin du texte coréen prétraité pour produire l'ordre syllabique désiré. Une approche consiste à étendre chaque jamo initial, central et final en une séquence de trois poids, basés sur la composition interne de chaque jamo. Les jamos simples obtiennent un poids ; les jamos en deux parties en obtiennent deux et les jamos en trois parties en obtiennent trois. Toutes les positions restantes dans les neufs poids de chaque syllabe sont alors remplies de poids nuls (U0000).

Par exemple, pour la syllable hangûl précomposée U+AC01, la donnée pourrait produire par prétraitement la séquence jamo <1100, 1161, 11A8> et être pondérée avec la clé suivante :

U1100 U0000 U0000 U1161 U0000 U0000 U11A8 U0000 U0000

Pour une séquence contenant un jamo d'ancien hangûl en plusieurs parties représentant un groupe, les clés pourraient avoir des valeurs multiples. Par exemple, pour la séquence d'entrée <1123, 1161, 11A8>, le caractère U+1123 obtiendrait pour ses trois sous-parties les poids suivants :

U1107 U1109 U1103 U1161 U0000 U0000 U11A8 U0000 U0000

Le même genre d'expansion de poids pourrait aussi bien être fait pour tout jamo central ou final à plusieurs parties.

Quand une syllabe coréenne ne contient pas de jamo final, les trois derniers poids créés sont nuls

(U0000).

À chaque syllabe coréenne étendue à neuf poids par ce schème de prétraitement et de pondération, les poids de chaque syllabe sont alignés correctement aux frontières de syllabe et une comparaison directe des clés résultantes produisent les résultats de tris corrects.

Bien que cette stratégie de pondération fonctionne pour toutes les données hangûl, tant pour le moderne que pour l'ancien, les clés résultantes très étendues ne sont pas efficaces pour les applications de tri en production. Une fois les frontières de syllabe établies par prétraitement des données coréennes, d'autres approches de pondération des jamos peuvent produire des clés beaucoup plus compactes qui donneront aussi les mêmes résultats pour le tri.

Il est aussi pour une implantation de tri de produire à la volée l'équivalent de ce prétraitement de données hangûl en pondérant une chaîne d'entrée, et il n'est donc pas techniquement requis de prévoir une étape de prétraitement séparée pour le hangûl qui convertirait toutes les données d'entrée au départ dans leur forme prétraitée avant l'étape de pondération et de comparaison. Ceci est particulièrement important pour des algorithmes de comparaison incrémentiels, qui sont très sensibles en matière de performance, et qui ne peuvent généralement pas se permettre de prétraiter des chaînes entières pour une comparaison incrémentielle.

C4.1.1 FNB

Ce qui suit précise un jeu particulier de règles permettant de transformer des données hangûl

codées pour le JUC de telle sorte qu'elles puissent être triées par un programme qui soutient l'ISO/CEI 14651.

Puisque ces les règles de transformation sont précisées dans une notation très utilisée, une grammaire indépendante du contexte, la FNB (forme normale de Backus ou forme de Backus-Naur), une brève introduction de la notation FNB s'impose..

Les explications qui suivent sont extraites (et adaptées en français) de la publication *Compilers, Principles, Techniques, and Tools. Aho, Sethi, and Ullman, Addison-Wesley Publishing Company, 1985*. Quelques extraits ont été légèrement modifiés pour mieux les comprendre dans le contexte de l'ISO/CEI 14651.

Par exemple, un énoncé `if-else` en *langage C* prend la forme

`if` (*expression*) énoncé `else` énoncé

L'énoncé `if-else` est la concaténation du mot-clé `if`, d'une parenthèse ouvrante, d'une expression, d'une parenthèse fermante, d'un énoncé, du mot-clé `else`, et d'un autre énoncé. La structure peut être exprimée en FNB comme

<énoncé> -> `if` (<expr>) <énoncé> `else` <énoncé>

où la flèche peut se lire « peut prendre la forme de ». Une telle règle est appelée une « production ». Le mot-clé `if` et les parentèse sont appelés « unités lexicales ». <expr> et <énoncé> représentent une séquence d'unités lexicales et sont appelés non-terminaux.

Une grammaire indépendante du contexte comporte quatre composantes :

1) un ensemble d'unités lexicales, symboles connus comme « terminaux » ;

2) un ensemble de « non-terminaux » ;

3) un ensemble de « productions » où chacune comprend un non-terminal, appelé la partie gauche de la production, une flèche (« -> »), et une séquence d'unités lexicales ou de non-terminaux, appelés la partie droite de la production ;

4) la désignation d'un des non-terminaux comme symbole de départ.

On précise les règles de transformation (ou la grammaire) en listant leurs productions, en commençant par les productions du symbole de départ.

Ici, les non-terminaux sont délimités par une paire de chevrons, comme par exemple <ds>, <ds1>, <ds2>, <ds3>. Les terminaux ne sont pas délimités par des chevrons, comme par exemple U1100, U1162, où U1100 représente le HANGÛL TCH'ÔSONG KIYOK et U1162 le HANGÛL DJOUNGSONG È.

Les productions partageant le même non-terminal à gauche peuvent voir leurs parties droites groupées, les divers choix à droite séparatés par une barre verticale, qui représente la conjonction « ou ».

Exemple 1.1. Supposons des expressions consistant en deux chiffres séparés par le signe plus ou le signe moins (comme par exemple 9 + 2, et 3 – 1). La grammaire qui suit décrit la syntaxe de ces expressions. Les productions sont :

<expr> -> <terme> + <terme> **(production 1a)**

<expr> -> <terme> - <terme> **(production 1b)**

<terme> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 **(production 1c)**

Les parties droites des deux productions partageant le non-terminal <expr> dans la partie gauche peuvent être groupées de manière équivalente :

<expr> -> <terme> + <terme> | <terme> - <terme>

<expr> et <terme> sont des non-terminaux où <expr> est le non-terminal de départ parce que ses productions sont données en premier. +, -, 0, 1, ..., et 9 sont des terminaux (ou unités lexicales).

Une grammaire produit ses chaînes en commençant par un symbole de départ et en remplaçant à répétition un non-terminal par la partie droite d'une production pour ce non-terminal. Les chaînes qui peuvent être produites à partir du symbole de départ forment le langage défini par la grammaire.

Exemple 1.2. Le langage défini par la grammaire de l'exemple 1.1 consiste en deux chiffres séparés par un signe plus ou par un signe moins.

Les dix productions pour le non-terminal <chiffre> lui permettent de représenter n'importe lequel des dix chiffres décimaux 0, 1, ..., 9. De la production 1c, découle qu'un *chiffre* simple est par lui-même un *terme*. Les productions 1a et 1b expriment le fait que si nous prenons n'importe quel chiffre et que nous le faisons suivre par un signe plus ou un signe moins et par un autre chiffre, nous disposons d'une expression.

a) 9 est un <terme> en vertu de la production 1c

b) 9 - 5 est une <expr> en vertu de la production 1b, puisque 9 est un <terme> et que 5 est aussi un <terme>.

Exemple 1.3. L'alphabet latin tel qu'utilisé en français consiste en 26 lettres; en français, six lettres sont des voyelles et les autres sont des consonnes. Ceci peut s'exprimer comme suit :

<alphabet-latin-fr> -> <voyelle> | <consonne>

<voyelle> -> a | e | i | o | u | y

<consonne> -> b | c | d | f | g | h | j | k | l | m | n | p | q | r | s | t | v | w | x | z

C.4.1.2 Traduction dirigée par la syntaxe

Un schème de traduction est une grammaire indépendante du contexte dans laquelle des fragments de programme appelés « actions sémantiques » sont inscrites dans la partie droite des productions. La position où une action doit être exécutée est montrée en l'entourant d'accolades (« { } ») et en l'écrivant dans la partie droite d'une production, comme dans

```
<expr> -> <terme> + <terme> { produire('+') }      (production e1)
<expr> -> <terme> - <terme> { produire('-') }      (production e2)
<terme> -> 0 {produire ('0')}                      (production t0)
<terme> -> 1 {produire ('1')}                      (production t1)
...
<terme> -> 9 {produire ('9')}                      (production t9)
```

Un schème de traduction produit une sortie (à l'aide de la commande « produire ») pour chaque phrase x produite par la grammaire sous-jacente en exécutant les actions dans l'ordre où elles apparaissent.

Le schème de traduction ci-avant traduit une expression donnée dans une forme postfixée. Ce schème accepte des expressions comportant deux nombres séparés par un signe plus ou un signe moins. Par exemple, '9 + 5' ou '9 - 5' sont acceptables, mais '1 + 2 - 3' ou '9 - 8 - 7' ne le sont pas.

Des expressions telles que 3 + 5 ou 9 - 8 sont le fait d'une « notation infixe », puisqu'un signe plus ou un signe moins, c'est-à-dire un opérateur binaire, est écrit entre deux nombres. Dans une notation postfixe, l'opérateur binaire (un signe plus ou un signe moins) est placé après les deux nombres.

Par exemple, la notation postfixe pour 3 + 5 s'écrit « 3 5 + » (le signe plus est placé après les deux nombres, et non entre les deux).

Voyons maintenant comment « 9 – 5 » est traduit en « 9 5 - ». Nous partons avec la production e1 « <expr> -> <terme> + <terme> {produire('+')} ». Le premier élément de la partie droite est <terme>. Alors la production t9 « <terme> -> 9 » correspond à « 9 » et '9' est produit (par la fonction « `produire` »). Maintenant le '+' de la partie droite ne correspond pas à '-'. Donc nous abandonnons e1 « <expr> -> <terme> + <terme> {`produire('+')`} ».

Maintenant nous essayons la prochaine production e2 « <expr> -> <terme> - <terme> {`produire('-')`} ». La production t9 « <terme> -> 9 » correspond à '9' et produit '9'. Alors '-' dans la production correspond à '-' ; toutefois rien n'est produit à ce stade. Maintenant la production t5 « <terme> -> 5 » correspond à '5' et produit '5'.

La production e2 « <expr> -> <terme> - <terme> {`produire('-')`} » correspond à la chaîne donnée '9 - 5'. À ce stade, '-' est produit. Le processus est terminé. En conséquence, la sortie finale est '9 5 -', ce qui constitue une notation postfixe pour l'expression donnée '9 - 5'.

C.4.2 Exemples montrant comment transformer les données

À l'article 1, nous avons étudié le concept de base de la FNB et du schème de traduction. Cela étant acquis, voyons des exemples montrant comment transformer les données dans le contexte des hangûls.

C.4.2.1 Exemple 1 (exemple simple qui n'utilise que quelques caractères hangûl)

- Pour la simplicité, nous avons inclus seulement deux caractères de début de syllabes, deux caractères de centre de syllabe, deux caractères de fin de syllabe, et deux caractères de bourre.

- Quelques exercices sont expliqués ci-après pour montrer comment les caractères de départ sont transformés selon les règles données.

- Deux notations sont utilisées ci-après.

Les chaînes de caractères entre '/*' et '*/' sont des commentaires et n'affectent pas le processus de transformation. Elles ne sont utiles qu'aux humains.

'==' définit des constantes. Par exemple, si 'BOURRE-DS == U115F', ;'on peut substituer U115F à BOURRE-DS partout où se trouve BOURRE-DS. La présence de BOURRE-DS à la place de U115F (un codet difficile à retenir), améliore la lisibilité.

Exemple 1 (abrégé par rapport à l'exemple 2 à des fins de démonstration)

```
/* constantes */

BOURRE-DS == U115F /* BOURRE de début de syllabe */

BOURRE-CS == U1160 /* BOURRE de centre de syllabe */


/* Syllabes hangûl */

/* PG: Partie gauche */

/* PD: Partie droite ou patron */

/* Le départ se fait à la <racine> */

/* ÉCHEC cancels temporary SORTIE */

/* 'finaliser SORTIE' finalise la SORTIE temporaire */

/* une action est montrée entre { }.  */

/* La plupart des actions produisent quelques caractères. */


/* La règle R01B accepte quatre combinaisons de caractères:
```

```
  U1100 U1161 | U1100 U1163 |  U1102 U1161 | U1102 U1163
          KA            KYA            NA            NYA */

---- -------    -------------------------------------
règle  PG        PD (ou patron)
---- -------    -------------------------------------
RACINE <racine>   -> <syl-hg> {finaliser SORTIE}
R01B <syl-hg> -> <ds> <cs> |
R01F           BOURRE-DS { produire('BOURRE-DS') } <fs>

/* <ds> : lettres simples de début de syllabe
   <ds1>: lettres simples de début de syllabe */
R11D <ds>  -> <ds1>
R12A <ds1> -> U1100 { produire('U1100') } |
R12B         U1102 { produire('U1101') }
          /* règles R12A et R12B: sortie sans transformation */

/* <cs> : lettres de centre de syllabe
   <cs1>: lettres simples de centre de syllabe */
R21D <cs>  -> <cs1>
R22A <cs1> -> U1161 { produire('U1161') } |
R22B         U1163 { produire('U1163') }
          /* règles R22A et R22B: sortie sans transformation */

/* <fs> : lettres de fin de syllabe
   <fs1>: lettres simples de fin de syllabe */
R31D <fs>  -> <fs1>
R32A <fs1> -> U11A8 { produire('U11A8') } |
R32B         U11AB { produire('U11AB') }
```

/* règles R32A et R32B: sortie sans transformation */

Exercice 1.1 Supposons que la chaîne de départ est « U1100 U1161 » (syllabe hangûl « GA »).

- Une chaîne de départ représente syllabe hangûl « GA ».

- À moins que « ÉCHEC » soit mentionné ci-après, la correspondance de patron existe.

- Quand « finaliser SORTIE » est exécuté, une sortie temporaire devient finale.

- Étant donné les règles qui précèdent, nous traiterons la chaîne de départ.

- Nous commençons avec la règle RACINE « <racine> -> <syl-hg> ».

- Alors nous examinons la première règle avec sa partie gauche <syl-hg>, c'est-à-dire la règle R01B « <syl-hg> -> <ds> <cs> ». Maintenant nous essayons la première composante de cette règle, qui est « <ds> ».

- Puis nous examinons la règle R11D « <ds> -> <ds1> », qui est la seule règle ayant comme partie gauche <ds>.

- Puis nous examinons la première règle ayant comme partie gauche <ds1>, c'est-à-dire la règle R12A « <ds1> -> U1100 { `produire` ('U1100') } ». Sa partie droite « U1100 » correspond au caractère de départ U1100. À ce stade, 'U1100' est produit par l'action { `produire` ('U1100') } de la règle R12A.

Le traitement de la règle R12A « <ds1> -> U1100 » est terminé.

Puis nous remontons à R11D « <ds> -> <ds1> ». Le traitement de la règle R11D est terminé.

Puis nous remontons à R01B « <syl-hg> -> <ds> <cs> ». Le cas de <ds>, la première composante de la partie droite de R01B, est réglé.

Nous sommes maintenant prêts à examiner la deuxième composante de la partie droite de R01B, soit <cs>.

Nous examinons la règle R21D « <cs> -> <cs1> », qui est la seule règle ayant comme partie gauche <cs>.

Puis nous examinons la première règle ayant comme partie gauche <cs1>, soit la règle R22A « <cs1> -> U1161 { `produire` ('U1161') } ». Sa partie droite « U1161 » correspond au caractère de départ U1161. À ce stade, 'U1161' est produit par l'action { `produire`('U1161') } de la règle R22A.

Le traitement de la règle R22A « <cs1> -> U1161 { `produire` ('U1161') } » est terminé.

Puis nous remontons à la règle R21D « <cs> -> <cs1> » dont le traitement est terminé.

Puis nous remontons à la règle R01B « <syl-hg> -> <ds> <cs> ». Puisque le traitement de la deuxième composante de la partie droite (c'est-à-dire <cs>) est terminé, à ce stade, le cas de R01B « <syl-hg> -> <ds> <cs> » est réglé.

Puis nous remontons à la règle RACINE « <racine> -> <syl-hg> {finaliser SORTIE} » dont le traitement est terminé. À ce stade, la SORTIE temporaire est finalisée.

- Dans cet exercice, nous ne changeons rien. Nous n'essayons que de faire correspondre à la chaîne de départ des règles et de produire une sortie sans transformation. Le procesus qui précède peut se résumer comme suit :

règle (PG)	ÉGALITÉ/ÉCHEC	SORTIE en fonction des actions
RACINE <racine>		
R01B <syl-hg>		
R11D <ds>		
R12A <ds1>	ÉGALITÉ R12A <ds1> -> U1100	U1100
	ÉGALITÉ R11D <ds> -> <ds1>	
	ÉGALITÉ R01B <syl-hg> -> <ds>	
R21D <cs>		
R22A <cs1>	ÉGALITÉ R22A <cs1> -> U1161	U1161
	ÉGALITÉ R21D <cs> -> <cs1>	
	ÉGALITÉ R01B <syl-hg> -> <ds> <cs>	
	ÉGALITÉ RACINE <racine> -> <syl-hg>	finaliser sortie

* sortie finale: U1100 U1161

Exercice 1.2 Supposons que la chaîne d'entrée est « U115F U11A8 » (HANGÛL TCH'ÔSONG KIYOK).

---------- ---------- ----------

règle (PG) ÉGALITÉ/ÉCHEC SORTIE en fonction des actions

---------- ---------- ----------

RACINE <racine>

R01B <syl-hg>

R11D <ds>

R12A <ds1> ÉCHEC R12A <ds1> -> U1100

ÉCHEC R12B <ds1> -> U1102

ÉCHEC R11D <ds> -> <ds1>

ÉCHEC R01B <syl-hg> -> <ds>

R01F <syl-hg> ÉGALITÉ BOURRE-DS {`produire` ('U115F')} U115F

R31D <fs>

R32A <fs1> ÉGALITÉ R32A <fs1> -> U11A8 U11A8

ÉGALITÉ R31D <fs> -> <fs1>

ÉGALITÉ R01F <syl-hg> -> BOURRE-DS <fs>.

ÉGALITÉ RACINE <racine> -> <syl-hg> finaliser sortie

* sortie finale : U115F U11A8

C.4.2.2 Exemple 2

- Cet exemple transforme une syllabe hangûl en 9 postions de code : 3 positions de code pour chacun des caractères de début de syllabe, de centre de syllabe et de fin de syllabe, respectivement.

- Quelques caractères NULS (U0000) sont insérés intentionnellement de manière à pouvoir trier les hangûls correctement (ce qui est particulièrement utile pour trier les hangûls anciens).

Exemple 2

```
/* constantes */
BOURRE-DS == U115F /* caractère de bourre de début de syllabe */
BOURRE-CS == U1160 /* caractère de bourre de centre de syllabe */

/* Syllabes hangûl */
---- -------    -------------------------------------
règle   PG          PD (ou patron)
---- -------    -------------------------------------
RACINE <racine>   -> <syl-hg> { finaliser SORTIE }
R01B <syl-hg> -> <ds> <cs> { produire('U0000 U0000 U0000') }
             /* ÉCHEC annule une sortie temporaire concernée */
R01F          BOURRE-DS { produire('BOURRE-DS U0000 U0000 U0000 U0000 U0000') }
              <fs>

/* lettres de début de syllabe :  <ds1> lettre simple de début de syllabe */
R11D <ds>  -> <ds1> { produire('U0000 U0000') }
R12A <ds1> -> U1100 { produire('U1100') } |
R12B         U1102 { produire('U1102') }  /* sortie sans transformation */

/* lettres de centre de syllabe:  <cs1> lettre simple de centre de syllabe */
R21D <cs>  -> <cs1> { produire('U0000 U0000') }
R22A <cs1> -> U1161 { produire('U1161') } |
R22B         U1163 { produire('U1163') } /* sortie sans transformation */
```

/* lettres de fin de syllabe: <fs1> lettre simple de fin de syllabe */

R31D <fs> -> <fs1> { `produire`('U0000 U0000') }

R32A <fs1> -> U11A8 { `produire`('U11A8') } |

R32B U11AB { `produire`('U11AB') } /* sortie sans transformation */

Exercice 2.1 Supposons que la chaîne de départ est « U1100 U1161 » (syllabe hangûl « GA »).

règle (PG)	ÉGALITÉ/ÉCHEC	SORTIE en fonction des actions
RACINE <racine>		
R01B <syl-hg>		
R11D <ds>		
R12A <ds1>	ÉGALITÉ R12A <ds1> -> U1100	U1100
	ÉGALITÉ R11D <ds> -> <ds1>	U0000 U0000
	ÉGALITÉ R01B <ds>	
R21D <cs>		
R22A <cs1>	ÉGALITÉ R22A <cs1> -> U1161	U1161
	ÉGALITÉ R21D <cs> -> <cs1>	U0000 U0000
	ÉGALITÉ R01B <syl-hg> -> <ds> <cs>	U0000 U0000 U0000
	ÉGALITÉ RACINE <racine> -> <syl-hg>	finaliser sortie

* sortie finale : U1100 U0000 U0000 U1161 U0000 U0000 U0000 U0000 U0000

Exercice 2.2 Supposons que la chaîne de départ est « U115F U11A8 ».

- le fichier d'entrée représente le HANGÛL TCH'ÔSONG KIYOK.

---

règle (PG) ÉGALITÉ/ÉCHEC SORTIE en fonction des actions

RACINE <racine>

R01B <syl-hg>

R11D <ds>

R12A <ds1> ÉCHEC R12A <ds1> -> U1100

ÉCHEC R12B <ds1> -> U1102

ÉCHEC R11D <ds> -> <ds1>

ÉCHEC R01B <syl-hg> -> <ds>

R01F <syl-hg> ÉGALITÉ <syl-hg> -> BOURRE-DS U115F U0000 U0000 U0000 U0000 U0000

R31D <fs>

R32A <fs1> ÉGALITÉ R32A <fs1> -> U11A8 U11A8

ÉGALITÉ R31D <fs> -> <fs1> U0000 U0000

ÉGALITÉ R01F <syl-hg> -> BOURRE-DS <fs>

ÉGALITÉ RACINE <racine> -> <syl-hg> finaliser sortie

* sortie finale : U115F U0000 U0000 U0000 U0000 U0000 U11A8 U0000 U0000

C.4.3 Prétraitement des hangûls modernes et anciens

C.4.3.1 Prétraitement des syllabes hangûl modernes (U+AC00 ~ U+D7A3)

Chacune des positions de l'espace de codage U+AC00 ~ U+D7A3 correspond à une syllabe hangûl moderne. Avant d'appliquer les règles de l'article C.4.3.2, nous devons décomposer chaque position de de l'espace de codage U+AC00 ~ U+D7A3 en deux ou trois positions de code.

Les positions de code correspondant aux syllabes comprenant une lettre de fin de syllabe seront décomposées en trois positions de code. Les positions de code correspondant à des lettres de début de syllabe, de centre de syllabe et de fin de syllabe sont concaténées dans cet ordre.

. Les positions de code correspondant à des syllabes sans une lettre de fin de syllabe seront décomposées en deux positions de code. Les positions de code correspondant à des lettres de début de syllabe et de centre de syllabe sont concaténées dans cet ordre.

Les positions de code pour les lettres de début de syllabe, de centre de syllabe et les lettres optionnelles de fin de syllabe sont calculées comme suit :

une position de code pour une lettre de début de syllabe = 0x1100 + (c / 588)

une position de code pour une lettre de centre de syllabe = 0x1161 + (c % 588) / 28

une position de code pour une lettre de fin de syllabe =

si (c % 28 != 0) alors (0x11A8 - 1) + c % 28

autrement rien

Note. '/' est un opérateur de division entière et '%' un opérateur « modulo ».

3.2 Prétraitement des jamos (U1100 ~ U11FF)

Les règles qui suivent peuvent prétraiter 11 172 syllabes hangûl modernes et d'autres syllabes incomplètes représentées à l'aide de jamos (U1100 ~ U11FF)

```
/* constante */
BOURRE-DS == U115F
BOURRE-CS == U1160

/* Syllabe hangûl */
/* syllabes complètes : R01A and R01B */
/* syllabes incomplètes : R01C, R01D, R01E and R01F */

RACINE <racine>   -> <syl-hg> {finaliser SORTIE}
R01A <syl-hg> -> <ds> <cs> <fs> |
R01B            <ds> <cs> { produire('U0000 U0000 U0000') } |
R01C            <ds> BOURRE-CS
               {produire('BOURRE-CS U0000 U0000 U0000 U0000 U0000') } |
R01D            BOURRE-DS { produire('BOURRE-DS U0000 U0000') } <cs> <fs> |
R01E            BOURRE-DS { produire('BOURRE-DS U0000 U0000') } <cs>
               { produire('U0000 U0000 U0000') } |
R01F            BOURRE-DS { produire('BOURRE-DS U0000 U0000 U0000 U0000 U0000') }
              <fs>


/* lettres de début de syllabe letters:
  <ds1> lettre simple de début de syllabe
  <ds2> lettre complexe de début de syllabe 2 (composée de 2 lettres simples)
  <ds3> lettre complexe de début de syllabe 3 (composée de 3 lettres simples)
*/

R11A <ds>  -> <ds1> <ds1> <ds1> |
R11B          <ds1> <ds1> { produire('U0000') } |
R11C          <ds1> <ds2> |
```

R11D <ds1> { produire('U0000 U0000') } |

R11E <ds2> <ds1> |

R11F <ds2> { produire('U0000') }

/* R11G <ds3>: aucun ds3 pour le hangûl moderne */

R12A <ds1> -> U1100 { produire('U1100') } |

R12B U1102 { produire('U1102') } |

R12C U1103 { produire('U1103') } |

R12D U1105 { produire('U1105') } |

R12E U1106 { produire('U1106') } |

R12F U1107 { produire('U1107') } |

R12G U1109 { produire('U1109') } |

R12H U110B { produire('U110B') } |

R12I U110C { produire('U110C') } |

R12J U110E { produire('U110E') } |

R12K U110F { produire('U110F') } |

R12L U1110 { produire('U1110') } |

R12M U1111 { produire('U1111') } |

R12N U1112 { produire('U1112') }

/* sortie sans transformation */

R13A <ds2> -> U1101 { produire('U1100 U1100') } |

R13B U1104 { produire('U1103 U1103') } |

R13C U1108 { produire('U1107 U1107') } |

R13D U110A { produire('U1109 U1109') } |

R13E U110D { produire('U110C U110C') }

```
/* R14 <ds3> ->  aucune ds3 pour les hangûls modernes */

/* lettres de centre de syllabe :
  <cs1> lettre simple de centre de syllabe
  <cs2> lettre complexe de centre de syllabe 2 (composée de 2 lettres simples)
  <cs3> lettre complexe de centre de syllabe 3 (composée de 3 lettres simples) */

R21A <cs>  -> <cs1> <cs1> <cs1> |
R21B          <cs1> <cs1> { produire('U0000') } |
R21C          <cs1> <cs2> |
R21D          <cs1> { produire('U0000 U0000') } |
R21E          <cs2> <cs1> |
R21F          <cs2> { produire('U0000') } |
R21G          <cs3>

R22A <cs1> -> U1161 { produire('U1161') } |
R22B          U1163 { produire('U1163') } |
R22C          U1165 { produire('U1165') } |
R22D          U1167 { produire('U1167') } |
R22E          U1169 { produire('U1169') } |
R22F          U116D { produire('U116D') } |
R22G          U116E { produire('U116E') } |
R22H          U1172 { produire('U1172') } |
R22I          U1173 { produire('U1173') } |
R22J          U1175 { produire('U1175') }

R23A <cs2> -> U1162 { produire('U1161 U1175') } |
R23B          U1164 { produire('U1163 U1175') } |
R23C          U1166 { produire('U1165 U1175') } |
R23D          U1168 { produire('U1167 U1175') } |
```

```
R23E          U116A { produire('U1169 U1161') } |
R23F          U116C { produire('U1169 U1175') } |
R23G          U116F { produire('U116E U1165') } |
R23H          U1171 { produire('U116E U1175') } |
R23I          U1174 { produire('U1173 U1175') }

R24A <cs3> -> U116B { produire('U1169 U1161 U1175') }
R24B          U1170 { produire('U116E U1165 U1175') }



/* lettres de fin de syllabe :
  <fs1> lettre simple de fin de syllabe
  <fs2> lettre complexe de fin de syllabe 2 (composed of 2 simple letters)
  <fs3> lettre complexe de fin de syllabe 3 (composée de 3 lettres simples) */

R31A <fs>  -> <fs1> <fs1> <fs1> |
R31B          <fs1> <fs1> { produire('U0000') } |
R31C          <fs1> <fs2> |
R31D          <fs1> { produire('U0000 U0000') } |
R31E          <fs2> <fs1> |
R31F          <fs2> { produire('U0000') }
/* R31G <fs3>: pas de fs3 pour le hangûl moderne */

R32A <fs1> -> U11A8 { produire('U11A8') } |
R32B          U11AB { produire('U11AB') } |
R32C          U11AE { produire('U11AE') } |
R32D          U11AF { produire('U11AF') } |
R32E          U11B7 { produire('U11B7') } |
```

```
R32F        U11B8 { produire('U11B8') } |
R32G        U11BA { produire('U11BA') } |
R32H        U11BC { produire('U11BC') } |
R32I        U11BD { produire('U11BD') } |
R32J        U11BE { produire('U11BE') } |
R32K        U11BF { produire('U11BF') } |
R32L        U11C0 { produire('U11C0') } |
R32M        U11C1 { produire('U11C1') } |
R32N        U11C2 { produire('U11C2') }  /* sortie sans transformation */

R33A <fs2> -> U11A9 { produire('U11A8 U11A8') } |
R33B        U11AA { produire('U11A8 U11BA') } |
R33C        U11AC { produire('U11AB U11BD') } |
R33D        U11AD { produire('U11AB U11C2') } |
R33E        U11B0 { produire('U11AF U11A8') } |
R33F        U11B1 { produire('U11AF U11B7') } |
R33G        U11B2 { produire('U11AF U11B8') } |
R33H        U11B3 { produire('U11AF U11BA') } |
R33I        U11B4 { produire('U11AF U11C0') } |
R33J        U11B5 { produire('U11AF U11C1') } |
R33K        U11B6 { produire('U11AF U11C2') } |
R33L        U11B9 { produire('U11B8 U11BA') } |
R33M        U11BB { produire('U11BA U11BA') }

/* R34  <fs3> ->    aucune fs3 pour les hangûls modernes */
```

Le prétraitement des hangûls anciens est très similaire à celui des hangûls modernes.

C.4.4 Conclusions

Les articles normatifs actuels de l'ISO/CEI 14651 ne peuvent pas trier directement les données hangûl de l'ISO/CEI 10646, en particulier les vieux hangûls anciens. Sans prétraitement, on aboutit à un résultat incorrect.

Ce qui précède propose une méthode de prétraitement des données hangûl de l'ISO/CEI 10646 de telle sorte que le résultat peut être utilisé comme entrée d'un programme conforme à l'ISO/CEI 14651, qui pourra alors trier les hangûls correctement.

Les règles de l'article C.4.3.2 transforment une syllabe hangûl moderne (y compris les syllabes incomplètes) en 9 positions de code. Quand les données hangûl sont transformées de cette façon, elles peuvent être triées coretmenet par un programme conforme à l'ISO/CEI 14651.

Les règles de l'article C.4.3.2 peuvent être aisément étendues pour le prétraitement des hangûls anciens en fonction de ses règles de tri.