

The Balinese Unicode Text Processing

Imam Habibi

*Informatics Engineering Department
Bandung Institute of Technology
Ganesha 10 Street Bandung 40132*

E-mail : if12042@students.if.itb.ac.id, imam_beckham@yahoo.com

Abstraction

In principal, the computer only recognizes numbers as the representation of a character. Therefore, there are many encoding systems to allocate these numbers although not all characters are covered. In Europe, every single language even needs more than one encoding system.

Hence, a new encoding system known as Unicode has been established to overcome this problem. Unicode provides unique id for each different characters which does not depend on platform, program, and language. Unicode standard has been applied in a number of industries, such as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, and Unisys. In addition, language standards and modern information exchanges such as XML, Java, ECMA Script (JavaScript), LDAP, CORBA 3.0, and WML make use of Unicode as an official tool for implementing ISO/IEC 10646.

There are four things to do according to Balinese script: the algorithm of transliteration, searching, sorting, and word boundary analysis (spell checking).

To verify the truth of algorithm, some applications are made. These applications can run on Linux/Windows OS platform using J2SDK 1.5 and J2ME WTK2 library. The input and output of the algorithm/application are character sequence that is obtained from keyboard punch and external file.

This research produces a module or a library which is able to process the Balinese text based on Unicode standard. The output of this research is the ability, skill, and mastering of:

1. Unicode standard (21-bit) as a substitution to ASCII (7-bit) and ISO8859-1 (8-bit) as the former default character set in many applications.
2. The Balinese Unicode text processing algorithm.
3. An experience of working with and learning from an international team that consists of the foremost experts in the area: Michael Everson (Ireland), Peter Constable (Microsoft-US), I Made Suatjana, and Ida Bagus Adi Sudewa.

Keywords: Unicode, transliteration, searching, sorting, word boundary analysis, canonical combining class, normalization, and Unicode Collation Element.

1. Introduction

Language and local script are the most precious cultural assets that have to be preserved for generations to come. Balinese script which can be used to writing Balinese language is threatened to extinction because

Balinese is rarely used and has less scope of usage. Efforts to preserve it have been attempted but met an obstacle, i.e. the lack of application to accommodate opinions using Balinese script. Basically, the more sophisticated the tool is, the more guaranteed the education and culture in the

future are. This tool refers to the computer which is capable to build software engineering easily in such a manner to produce and process Balinese script quickly and properly [YBG05].

The endeavor to computerize Balinese script is being conducted by Bali Galang Foundation. The first step is done by including the Balinese script in standard of Unicode character. The Unicode Consortium¹ and the ISO/IEC JTC1/SC2/WG2² committee have agreed in principle to include the Balinese script as per defined in the formal proposal written by Michael Everson and I Made Suatjana in the standards that they maintain[EVE05]. This proposal, numbered N2908, was presented to the committee submitted to the WG2 46th meeting in Xiamen, China, in January 2005. The information given in the proposal is by all means complete but will only be finalized during the next WG2 meeting in Sophia-Antipolis, France, in September 2005.

The conventional methods of processing a string of Latin text are not applicable to the Balinese text because there are at least three different areas for the Balinese Unicode text processing [EVE05], i.e.:

1. Searching algorithm should work on both pre-composed and decomposed strings. Searching for U+1B12 BALINESE LETTER OKARA TEDUNG ꦠꦺꦢꦸꦁ꧀ should be equivalent with searching for U+1B11 BALINESE LETTER OKARA ꦠꦺꦢꦸꦁ꧀ and U+1B35 BALINESE VOWEL SIGN TEDUNG ꦠꦺꦢꦸꦁ꧀ꦠꦺꦢꦸꦁ꧀.
2. Sorting algorithm should not be based purely on character code points. Vowels

should be ignored when comparing consonants, but vowels should be factored in only when the consonants are equal. Furthermore, there are two different sorting schemes exist: the traditional Balinese HANACARAKA ordering and the Sanskrit ordering.

3. The Balinese text does not use spaces as word separators. A spell-checking algorithm should be able to perform a dictionary-based lookup to determine word boundaries and to validate the spelling of the text.

2. Text Processing in Computer

Generally, information which flows from and into computer is in the form of text document, picture, audio, video, and combination among them. Text is used to submit information in language and written with understandable scripts by human being as either the subject or object of the information. In order to being processed in computer, these scripts need to be decoded in number since computer can only recognize in number. This number consists of binary numbers, i.e. 0 and 1, known as *bit*.

In fact, bit processing is processed on *octet* (from Latin word, *octo* which means eight), a bit combination of eight digits also called *byte*. Some methods of convention are made to look for solution of how to interpret octet and other series of octet on the way to represent data. For example, four series of octet are used to interpret real numbers. In this final project, octet series is used to interpret string. The simplest way which is still used widely to interpret character is by mapping one octet with one

¹ <http://www.unicode.org>

² <http://anubis.dkuug.dk/JTC1/SC2/WG2>

character according to the mapping table. In doing so, we can interpret 256 ($2^8=256$) characters. This number of characters exceeds those in *character set*³ used by Latin script, a script which has widely been used to write many languages around the world such as English and Indonesian. This technique is also used by ASCII character standard (*American Standard Character for Information Interchange*) which is developed at 1960's and has been being used up to now.

In general, text processing in computer works when user types with keyboard, and the keyboard sends its scan codes to the keyboard driver. Then, the driver transforms the scan codes into meaningful character sequence. In the case of non-roman input mode is on, the driver also checks the input sequences and rejects invalid sequences. After that, the text processor manipulates the characters. It may do searching, copy-pasting, sorting, word counting, line breaking, transliteration, etc. These characters are also stored in memory or other storage devices. In order to show character sequences, the rendering engine picks the glyph that represents the character. Then, the display such as monitor and printer displays the rendered glyphs.

3. Unicode as a Character Coding Standard

Unicode is a character coding standard for representing a written language in computer. Unicode was actually not the first coding standard, because it came as the answer to the problems arising from the previous coding standard for years [UNI03]. Therefore, Unicode is close to the previous existing coding standard. When Unicode version 1.0

³ The script character collection. It is not yet related to its code representation. For example, Indonesian alphabet with its punctuation mark.

was issued in 1991, ASCII and ISO-8859 had become the most well known standard.

The development of the Unicode character model follows 10 basic rules stated below [UNI03]. However, not all are actually fulfilled. Consistency can be sacrificed in order to keep simplicity, efficiency, and compatibility with the precious standard. The basic rules are:

1. Universality
2. Efficiency
3. Character, not glyphs
4. Semantics
5. Plain text
6. Logical order
7. Unification
8. Dynamic composition
9. Equivalent sequences
10. Convertibility

4. Balinese Script

The Balinese script is used for writing the Balinese language, the native language of the people of Bali. It is a descendent of the ancient Brahmic script from India; therefore it has some notable similarities with modern scripts of South Asia and Southeast Asia that also are descendent of the Brahmic script. The Balinese script is also used for writing *Kawi*, or Old Javanese, which had a heavy influence to Balinese language in the 11th century. Some Balinese words are also borrowed from Sanskrit, thus Balinese script is also used to write words from Sanskrit.

The basic elements of the alphabet are syllables. Each syllable has inherent sound of /a/ or /ĕ/ depending of the position of the syllable within a word.

The text direction of the Balinese script is from left to right, with vowel signs attached to either before, after, below or

above the syllable. Some vowel signs are split vowels, meaning that they appear at more than one position to the syllable.

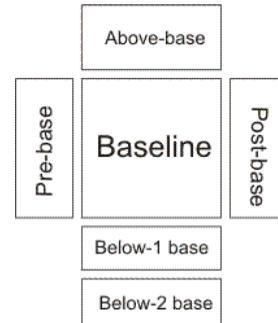
Writing system of Balinese script is more complex than Latin script. The alphabet consists of syllables. Every syllable ends up with vowel sound /a/.

Consonant cluster is a consonant group of syllable appearing without any vowels. In Balinese script, consonant naturally obtains the suffix of vowel sound /a/. In general, there are two ways to omit original vowel sound:

1. Utilizing consonant in the form of *gantungan* or *gempelang* attached to the next consonant. This *gantungan* or *gempelan* consonant is applied to omit the vowel on its left side, not the vowel on itself. For example, word  'bakta' (bring).
2. Utilizing *adeg-adeg*, U+1B44 BALINESE ADEG-ADEG. For example, word  'kadep' (sold).

Position adjustment in Balinese script writing is divided into several different areas (see picture 4-1), i.e.:

1. *Baseline area*: writing base of Balinese script. Consonants are written in this area.
2. Area on the left side or *pre-base marks (prem)* and on the right side or *post-base marks (pstm) baseline*: used to write *dependent vowel* and *gempelan*.
3. Area on the top side or *above-base marks (abvm)* and on the bottom side or *below-1 base marks (blw1m)* and *below-2 base marks (blw2m) baseline*: used to write *gantungan*, and *pengangge suara*.

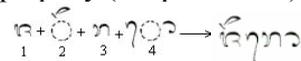


Picture 4-1. Writing position of Balinese script

5. Reordering and Split Vowel

Dependent vowel in Balinese script modifies base consonant syllable with several forms. A consonant or a cluster of consonants may have a dependant vowel to change the last vowel sound attached to it. Balinese script has various forms of dependant vowel, the spacing and the non-spacing one written on the previous, the next, the top side, or the bottom side of the base character. Yet the combination of them is also possible.

Unicode standard determines that the combining character is coded after its base character. Therefore, when a character sequence contains dependent vowels, reordering is necessary in the computer memory just before it is displayed on the screen. The function of the reordering is to make a change of the glyph order so that glyph component of dependent vowel is displayed properly (see picture 5-1).



Picture 5-1. Reordering

Split vowel is a vowel whose components appear on two different sides of its consonant. The component may

appear on either the top-right side, or the left and right side of its base consonant.

Glyph of vowels drawn on the bottom side of base character needs a special treatment because glyph selection depends on the context of the consonant frequency or the previous consonant cluster. These vowels are 1B38 BALINESE VOWEL SIGN SUKU (u) and 1B39 BALINESE VOWEL SIGN SUKU ILUT (uu). Both of these vowels have two different glyphs, i.e.: the one attached on the base consonants and their conjunct forms (*Pengangge Aksara*).

6. Ligatures

A glyph representing more than one character is called ligature, a script that is handwritten on a paper with no more than one scratch. Several Balinese scripts appearing adjacent to one another form ligature. Therefore, they seem on the screen as if they were only one glyph. For example, U+1B35 BALINESE VOWEL SIGN TEDUNG (aa) forms a ligature when attached to a syllable.

7. Line Breaking

Although Balinese script is written without any spaces between two successive words, line breaking cannot be conducted at random places. Hence, there are two common rules of line breaking, i.e.:

1. Line breaking may not be done between a syllable and any following combining characters.
2. Line breaking may not be done just before any punctuation.

8. Characteristics of Balinese Script

Like any other Unicode script, Balinese script has some unique characteristics (see

table 8-1). They are published in the proposal L2/05-008 which was approved by Unicode Consortium. However, the decomposition mapping property should be added to the proposal. Therefore, according to the Unicode standard, there should be ten characters requiring decomposition mapping (see algorithm 8-1).

Table 8-1. Characteristics of Balinese Script

| |
|---|
| 1B00;BALINESE SIGN ULU RICEM;Mn;230;NSM;;;;;N;;ardhacan dra;;; |
| 1B01;BALINESE SIGN ULU CANDRA;Mn;230;NSM;;;;;N;;candrab indu;;; |
| 1B02;BALINESE SIGN CECEK;Mn;230;NSM;;;;;N;;anusvara ;;; |
| 1B03;BALINESE SIGN SURANG;Mn;230;L;;;;;N;;repha;;; |
| 1B04;BALINESE SIGN BISAH;Mc;226;L;;;;;N;;visarga;;; |
| 1B05;BALINESE LETTER AKARA;Lo;0;L;;;;;N;;a;;; |
| 1B06;BALINESE LETTER AKARA TEDUNG;Lo;0;L;1B05 1B35;;;;;N;;aa;;; |
| 1B07;BALINESE LETTER IKARA;Lo;0;L;;;;;N;;i;;; |
| 1B08;BALINESE LETTER IKARA TEDUNG;Lo;0;L;1B07 1B35;;;;;N;;ii;;; |
| 1B09;BALINESE LETTER UKARA;Lo;0;L;;;;;N;;u;;; |

```

1B0A;BALINESE LETTER UKARA
TEDUNG;Lo;0;L;1B09
1B35;;;N;;uu;;;

1B0B;BALINESE LETTER RA
REPA;Lo;0;L;;;;N;;vokalic r;;;

1B0C;BALINESE LETTER RA REPA
TEDUNG;Lo;0;L;1B0B
1B35;;;N;;vokalic rr;;;

1B0D;BALINESE LETTER LA
LENGA;Lo;0;L;;;;N;;vokalic l;;;

1B0E;BALINESE LETTER LA LENGA
TEDUNG;Lo;0;L;;;;N;;vokalic ll;;;

1B0F;BALINESE LETTER
EKARA;Lo;0;L;;;;N;;e;;;

1B10;BALINESE LETTER
AIKARA;Lo;0;L;;;;N;;ai;;;

1B11;BALINESE LETTER
OKARA;Lo;0;L;;;;N;;o;;;

1B12;BALINESE LETTER OKARA
TEDUNG;Lo;0;L;1B11
1B35;;;N;;au;;;

1B13;BALINESE LETTER
KA;Lo;0;L;;;;N;;;;;

1B14;BALINESE LETTER KA
MAHAPRANA;Lo;0;L;;;;N;;kha;;;

1B15;BALINESE LETTER
GA;Lo;0;L;;;;N;;;;;

1B16;BALINESE LETTER GA
GORA;Lo;0;L;;;;N;;gha;;;

1B17;BALINESE LETTER
NGA;Lo;0;L;;;;N;;;;;

1B18;BALINESE LETTER
CA;Lo;0;L;;;;N;;;;;

```

```

1B19;BALINESE LETTER CA
LACA;Lo;0;L;;;;N;;cha;;;

1B1A;BALINESE LETTER
JA;Lo;0;L;;;;N;;;;;

1B1B;BALINESE LETTER JA
JERA;Lo;0;L;;;;N;;jha;;;

1B1C;BALINESE LETTER
NYA;Lo;0;L;;;;N;;;;;

1B1D;BALINESE LETTER TA
LATIK;Lo;0;L;;;;N;;tta;;;

1B1E;BALINESE LETTER TA MURDA
MAHAPRANA;Lo;0;L;;;;N;;ttha;;;

1B1F;BALINESE LETTER DA MURDA
ALPAPRANA;Lo;0;L;;;;N;;dda;;;

1B20;BALINESE LETTER DA MURDA
MAHAPRANA;Lo;0;L;;;;N;;ddha;;;

1B21;BALINESE LETTER NA
RAMBAT;Lo;0;L;;;;N;;nna;;;

1B22;BALINESE LETTER
TA;Lo;0;L;;;;N;;;;;

1B23;BALINESE LETTER TA
TAWA;Lo;0;L;;;;N;;tha;;;

```

| | | |
|--------------|---|--------------------------|
| U+1B06 (ꦱꦺꦴ) | → | [U+1B05, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B08 (ꦱꦺꦴ) | → | [U+1B07, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B0A (ꦱꦺꦴ) | → | [U+1B09, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B0C (ꦱꦺꦴ) | → | [U+1B0B, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B12 (ꦱꦺꦴ) | → | [U+1B11, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B3B (ꦱꦺꦴ) | → | [U+1B3A, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B3D (ꦱꦺꦴ) | → | [U+1B3C, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B40 (ꦱꦺꦴ) | → | [U+1B3E, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B41 (ꦱꦺꦴ) | → | [U+1B3F, U+1B35] (ꦱ, ꦺꦴ) |
| U+1B43 (ꦱꦺꦴ) | → | [U+1B42, U+1B35] (ꦱ, ꦺꦴ) |

Symbol → showing that the left sided element has to be equivalent to the right side element

Algorithm 8-1. Decomposition mapping of Balinese script [CON05]

9. Canonical Combining Class of Balinese Script

The purpose of canonical combining classes is to establish appropriate equivalence classes under Unicode normalizations for character sequences that involve combining marks. Specifically:

1. Given a pair of combining marks that interact typographically (i.e., that nominally occupy the same position relative to the base), different encoded orders correspond to visually-distinct relative positions of the marks, hence are semantically distinct. By assigning these marks to the same canonical combining class (zero or non-zero), the nonequivalence of differently-ordered sequences is established under normalization.
2. Given a pair of combining marks that do not interact typographically (i.e., that occupy distinct positions relative to the base), different encoded orders are visually identical, hence not semantically distinct. By assigning these marks to different, non-zero canonical combining classes, the equivalence of differently-ordered sequences is established under normalization.

In canonical combining class, the class 0 has special behavior in the Unicode normalization algorithms: if a sequence contains a combining mark in class 0 and a mark in a non-zero class n , equivalence classes are defined as though the class-0 mark belonged to class n ; i.e., that sequence is not equivalent to the sequence containing those marks in the opposite order.

Using the canonical combining classes proposed in L2/05-008, there is only one pair of combining marks for which distinct orders would be considered canonically equivalent (see table 9-1).

Table 9-1. Canonical combining class of Balinese script proposed in L2/05-008

```
< 1B34 SIGN REREKAN (ccc=7), 1B44
ADEG-ADEG (ccc=9) > ≡
< 1B44 ADEG-ADEG (ccc=9), 1B34
SIGN REREKAN (ccc=7) >
```

In proposal L2/05-008, Combinations of syllable-modifier signs (1B00–1B03), REREKAN and vowel signs, at least, are linguistically valid. Because all of these but REREKAN are assigned to class 0, differently-ordered sequences of these marks, which would be visually distinct, are not canonically equivalent. Thus, the use of class 0 provides appropriate results in these cases.

In this case, < 1B35 BALINESE VOWEL SIGN TEDUNG, 1B04 BALINESE SIGN BISAH > is a linguistically plausible combination. Assuming it is normal use as a vowel killer, ADEG-ADEG should not co-occur with either of the other two marks. Again, though, different encoded orders of a combination of these marks are possible in principle and would be visually distinct, and so the use of class 0 provides appropriate results in these cases.

In the cases described above, the use of class 0 is sufficient to cause differently-ordered combinations of marks that do interact typographically (having different visual results) to be considered not canonically equivalent. However, the assignment of marks to class 0 breaks down because it is in failing to cause differently-ordered combinations of

marks that do not interact typographically to be considered canonically equivalent.

According to Unicode standard, a suggestion is made by assigning each classes 220, 224, 226, 230 to every character whose position relative to the base is bottom, left, right, or top [CON05]. However, character U+1B34 BALINESE SIGN REREKAN and U+1B44 BALINESE SIGN VIRAMA is assigned to fixed-position class 7 and 9.

10. Normalization of Balinese Script

There are four Unicode normalization forms, i.e.:

1. Normalized Form D: canonical decomposition.
2. Normalized Form C: canonical decomposition, followed by canonical composition.
3. Normalized Form KD: compatibility decomposition.
4. Normalized Form KC: compatibility decomposition, followed by canonical composition.

Considering the efficiency and performance of the Unicode normalization algorithm, Balinese script is processed in the normalized form D. Besides, the normalized form C is basically obtained through the same steps as the normalized form D.

Normalized form D consists of two phases. First, in every Balinese word is the decomposition mapping done. Second, canonical reordering is done according to the canonical combining class of each character. Decomposition mapping has recursive property, so the correct character sequences are obtained (see algorithm 10-1). For example, U+1B40 turns into

[U+1B3E,U+1B35], so  <U+1B13 KA,U+1B40 VOWEL SIGN TALING TEDUNG> \equiv <U+1B13 KA,U+1B3E VOWEL SIGN TALING,U+1B35 VOWEL SIGN TEDUNG>.

| |
|--|
| <p>If: $X \rightarrow [Y,Z]$ (defined) If: $Y \rightarrow [Y_1,Y_2]$ (defined) Then: $X \rightarrow [Y_1,Y_2,Z]$ (conclusion)</p> |
|--|

Symbol \rightarrow showing that the left sided element has to be equivalent to the right side element

Algorithm 10-1. Decomposition mapping algorithm

11. Unicode Collation Algorithm (UCA)

Basically, UCA is a process to create sorting keys possessing particular priority value. All strings are first processed on the primary level. They all are then compared. If they equal to each other, the comparison is continued to the secondary level and later to the tertiary level when necessary.

In general, the sorting keys of UCA are:

1. Alphabet/base character
2. Diacritic mark/accent
3. Uppercase and lowercase

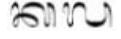
12. Comparison Algorithm of Balinese Script

Comparison algorithm of Balinese script consists of four phases, i.e.:

1. Every Balinese script input is first processed with the normalized form D.
2. The result from step (1) is then continued to the UCA according to the chosen sorting method, either HANACARAKA or SANSKRIT.

3. The next step is separating the values of collation element and inserting a particular value from level separator while recombining those values.
4. The last step is comparing the values obtained from the step (3) using binary comparison algorithm.

13. Transliteration of Balinese Script

Transliteration is a mapping from one writing system to another one, e.g. from Balinese script to Latin and vice versa by considering both accent and grammar of them [WIK05]. The main criterion is the lossless information so that user should be capable of retransforming the information to its original format. Therefore, transliteration is different from transcription which only focuses on voice mapping from one language to another one. The use of transliteration is for helping people who can not read the Balinese script. For example,  [U+1B13 KA,U+1B2E LA] becomes 'kala' (time).

Transliteration is performed by building a table to map characters from Balinese script to Latin and vice versa. To get it done, a complex conversion is needed in order to overcome the change of shape and special case of letters in source script. The input for transliteration is two digits from keyboard in hexadecimal format (0,1,2,...,A,B,C,D,E,F) with padding bit 0 if the number of digits is odd. The algorithm of transliteration utilizes data structure of inversion list (including basic functions: invert, union, intersection, set difference, adding, and deleting) in order to save more spaces in memory. The performance of those operations is faster due to random access for every element.

Character U+1B33 BALINESE LETTER HA  serves as a neutral place for vowel. Therefore, when a vowel is written on the first letter of word, either independent vowel or character U+1B3 followed by appropriate dependent vowel may be used.

Character U+1B05 BALINESE LETTER AKARA  serves as a neutral place when followed by appropriate character or sign, so the character may be transliterated to ‘e’, ‘i’, ‘o’, ‘u’, ‘ī’, ‘ū’, ‘ē’, and ‘ö’.

Character *nania* (U+1B2C) is used in consonant cluster between words using appended form ‘ya’, so this character may be transliterated to ‘ia’, e.g. ‘siap’ and ‘tabia’.

Character *suku kembang* (U+1B2F) is used in consonant cluster between words using appended form ‘wa’, so this character may be transliterated to ‘ua’.

Transliteration algorithm calls *translate string* function receiving both string input and output (see algorithm 13-1 and 13-2), for example, given an string input *s* with $n = \text{length}(s)$, then an iteration for *n* characters is performed.

In transliteration algorithm, the most dominant function is searching character in I/O file. The comparison is performed at Balinese script block (U+1B00–U+1B7F). In the worst case, the comparison is performed 128 times.

Given *m* is the number of comparison at lookup table in average, the transliteration algorithm theoretically has a complexity of $O(n*m)$.

2. Perform iteration from the first character to the last one.
3. If the character is the base character, then go to step 4. Otherwise, go to step 7.
4. If the character displays glyph in the appended form, it means that there is a usage of character virama.
5. Verify whether the character has a special case as mentioned before.
6. If the character is character rerekan, then the previous character should be validated. Besides, the previous output should be changed as well.
7. If the character is dependent vowel, then the previous vowel of base character should be turned into an appropriate character.
8. Finally, perform character matching using mapping table in appropriate form. The provided forms are normal form, appended form, case 1...8 forms, and UNKNOWN_TRANSLATE.

Algorithm 13-1. Character transliteration of Balinese script

| |
|---|
| <p>Input: character sequence of Balinese script</p> <p>Output: character sequence of Latin script</p> <p>1. Perform a block validation of Balinese script in Unicode.</p> |
|---|

Input: character sequence of Latin script
 Output: character sequence of Balinese script

1. Perform a block validation of Latin script in Unicode.
2. If there is a separator character, then the split characters are combined.
3. Perform iteration from the first character to the last one.
4. Verify whether the character has a special case as mentioned before, e.g. the vowel.
5. Verify whether the input is a balanced word (symmetric word) in an appropriate position.
6. Validate whether the output contains character virama and rerekan.
7. Then, perform character matching using mapping table in appropriate form. The provided forms are normal form, appended form, case 1...8 forms, and UNKNOWN_TRANSLATE.
8. Finally, verify whether the output is included into the consonant of Balinese script using the data structure of inversion list. If the output is a consonant, then the character virama is appended into the output.

Algorithm 13-2. Inverse Transliteration of Balinese script

14. Searching of Balinese Script

Searching of Balinese script has some challenges, i.e. that there are some characters which are made of other characters and they are even possible to be combined into a single

character. Therefore, it should work on both pre-composed and decomposed strings.

Searching is performed by validating equivalent forms of Balinese script (see table 14-1), e.g.  'daar' (eat) and  'daara' (eaten),  'baang' (give) dan  'baanga' (given).

In searching algorithm, the most dominant functions are normalization, *sorting key* creation according to UCA, and binary comparison functions (algorithm 14-1). Normalization function receives string input and an array of integer. Given string input s with $n = \text{length}(s)$, then perform iteration for n characters. This function consists of two processes:

1. Canonical decomposition, i.e. recursive function transforming Balinese script into atomic form. In the best case, character input has already had an atomic form. Otherwise, in the worst case, character input is processed recursively until the character gets an atomic form. For the case of Balinese script, the recursion depth is one.
2. Reordering canonical combining class is a function that reorders Balinese script character classes already processed in canonical decomposition. In the worst case, iteration is performed until the recursion depth minus one.

Given p is the average recursion depth, the normalization theoretically has a complexity of $O(p)$.

In average, element collation function performs iteration for n times, so the array of integer is produced with the size of n .

Therefore, this function has a complexity of $O(n)$.

Binary comparison function performs comparison of element collation values. In the worst case, the comparison is performed for n times.

Theoretically, searching algorithm has a complexity as the following:

$$\begin{aligned} T(n) &= O(n*(O(p)) + n*(O(n)) + n) \\ &= O(n*p + n^2 + n) \quad \rightarrow \end{aligned}$$

pick the maximum value among the three complexity values ($p < n$)

$$= O(n^2)$$

Therefore, the searching algorithm complexity is $O(n^2)$.

Table 14-1. The Equivalent Form of Balinese Script

| | | |
|-----------------------------------|---|---|
| 1. U+1B03 BALINESE SIGN SURANG |  | ≡ |
| U+1B2D BALINESE LETTER RA |  | |
| 2. U+1B02 BALINESE SIGN CECEK |  | ≡ |
| U+1B17 BALINESE LETTER NGA |  | ≡ |
| U+1B01 BALINESE SIGN ULU CHANDRA |  | |
| 3. U+1B04 BALINESE SIGN BISAH |  | ≡ |
| U+1B33 BALINESE LETTER HA |  | |
| 4. U+1B00 BALINESE SIGN ULU RICEM |  | |
| | ≡ U+1B2B BALINESE LETTER MA |  |

Input: character sequence of Balinese script

Output: list of character sequence of Balinese script

1. Perform a block validation of Balinese script in Unicode.
2. Transform the input into the equivalent form of Balinese script according to table 14-1.
3. Normalize the Balinese script character.

4. Build sorting key according to UCA.
5. Utilize the data structure *ListSorting* and select the appropriate sorting method used, either HANACARAKA or SANSKRIT.
6. Perform binary comparison of the sorting keys.
7. If the values are equal, put the values into the list output.

Algorithm 14-1. Searching of Balinese Script

15. Sorting of Balinese Script

Sorting of Balinese Script is performed by two different ways depending of which sorting method is used: Sanskrit or HANACARAKA.

Based on the Devanagari and Latin script sorted by *character code point*, Balinese script is processed with the following order:

1. Punctuations consist of seven characters, i.e.: ‘carik’, ‘carik pareren’, ‘panten’, ‘pasalinan’, ‘pamada’, ‘carik agung’, and ‘carik pamungkah’ (U+1B5A – U+1B60).
2. Numbers consist of ten characters (U+1B50 – U+1B59).
3. Sorting methods, i.e.: HANACARAKA (table 15-1) and SANSKRIT (table 15-2).
4. Other symbols (U+1B61 – U+1B7C).

Therefore, sorting algorithm of Balinese script does not depend on the character code pint. Besides, the vowels must be ignored when comparing the consonants. The vowels are taken into account when the consonants are identical. For example,

ꦏꦫꦩꦂ 'krama' (member) appears before ꦕꦏꦫꦂ 'cakra' (disc) in Sanskrit sorting method. On the other hand, 'cakra' appears before 'krama' in HANACARAKA sorting method.

Table 15-1. HANACARAKA Sorting

```
A > Ā > Ę > Ö > I > Ī > U > Ū > E
  > AI > O > AU >
ha (bisah) > ha-rerekan > na > nna
  > ca > cha > ra ([ra-repa = rē],
  surang) > ka > ka-rerekan > kaf-
  sasak > khot-sasak > kha >
da > da-rerekan > dha > dda > ddha
  > ta > tzir-sasak > tha > tta >
  ttha >
sa > zal-sasak > asyura-sasak >
sha > ssa > wa > wa-rerekan > ve-
sasak > la ([la-lenga = lē]) >
ma (uli ricem) > ga > ga-rerekan >
gha > ba > bha > nga (cecek, ulu
  candra) > nga-rerekan
pa > pa-rerekan > ef-sasak > pha
> ja > ja-rerekan > jha > ya > nya
```

Table 15-2. SANSKRIT Sorting

```
A > Ā > Ę > Ö > I > Ī > U > Ū > RĘ
> RÖ > LĘ > LÖ > E > AI > O > AU >
ka > ka-rerekan > kaf-sasak >
khot-sasak > kha > ga > ga-rerekan
> gha > nga (cecek, ulu candra) >
  nga-rerekan >
ca > cha > ja > ja-rerekan > jha >
  nya > tta > ttha > dda > ddha >
  nna >
ta > tzir-sasak > tha > da > da-
  rerekan > dha > na > pa > pa-
  rerekan > ef-sasak > pha > ba >
  bha > ma (uli ricem) >
ya > ra (surang) > la > wa > wa-
  rerekan > ve-sasak >
sha > ssa > sa > zal-sasak >
asyura-sasak > ha (bisah) > ha-
rerekan
```

In sorting algorithm, the most dominant functions are normalization, *sorting key* creation according to UCA, and binary comparison functions (algorithm 15-1).

| |
|---|
| <p>Input: pair of character sequence of Balinese script</p> <p>Output: sequence character order (-1=smaller, 0=equal, 1=bigger)</p> <ol style="list-style-type: none"> 1. Perform a block validation of Balinese script in Unicode. 2. Normalize the Balinese script character. 3. Build sorting key according to UCA. 4. Utilize the data structure <i>ListSorting</i> and select the appropriate sorting method used, either HANACARAKA (see table 15-1) or SANSKRIT (see table 15-2). 5. Perform binary comparison of the sorting keys. |
|---|

Algorithm 15-1. Sorting of Balinese Script

Normalization function receives string input and an array of integer. Given string input s with $n = \text{length}(s)$, then perform iteration for n characters. This function consists of two processes:

1. Canonical decomposition, i.e. recursive function transforming Balinese script into atomic form. In the best case, character input has already had an atomic form. Otherwise, in the worst case, character input is processed recursively until the character gets an atomic form. For the case of Balinese script, the recursion depth is one.
2. Reordering canonical combining class is a function that reorders Balinese

script character classes already processed in canonical decomposition. In the worst case, iteration is performed until the recursion depth minus one.

Given p is the average recursion depth, the normalization theoretically has a complexity of $O(p)$.

In average, element collation function performs iteration for n times, so the array of integer is produced with the size of n . Therefore, this function has a complexity of $O(n)$.

Binary comparison function performs comparison of element collation values. In the worst case, the comparison is performed for n times.

Theoretically, sorting algorithm has a complexity as the following:

$$\begin{aligned} T(n) &= O(n*(O(p)) + n*(O(n)) + n) \\ &= O(n*p + n^2 + n) \quad \rightarrow \end{aligned}$$

pick the maximum value among the three complexity values ($p < n$)

$$= O(n^2)$$

Therefore, the sorting algorithm complexity is $O(n^2)$.

16. Word Boundary of Balinese Script

In computer terminology, word boundary or spell checker is a software facility designed for verifying the correctness of the spelling in a document in order to help user obtain the correct spelling [WIK05].

To handle Balinese script, word boundary breaks down the sentence structure into its building words. As mentioned before, in Balinese script there is no usage of spaces as separator between two adjacent words, so a dictionary-based lookup is required.

Word boundary algorithm is performed by separating character sequences of Balinese script into base character cluster (including

consonant and consonant cluster). This algorithm uses data structure of stack to retrieve valid character cluster and then process it. The validity of a piece of the character sequences can be inferred through comparison between the sequences and the data stored in dictionary. There are three possible outputs resulted from the word boundary algorithm, i.e.:

1. The character sequences are successfully separated completely and correctly (the best case).
2. The character sequences are separated, but only partially completed because it is possible that the input is invalid or the data in the dictionary are not complete. However, the algorithm returns the best word boundary obtained according to the statistical principle (the endeavor that produces the maximum number of Balinese script words).
3. No output is returned (the worst case).

For example,  'pakraman paksu'

[(U+1B27,U+1B13,U+1B44,U+1B2D,U+1B2B,U+1B26,U+1B44)(U+1B27,U+1B13,U+1B44,U+1B32)].

In word boundary algorithm, the most dominant function is word searching in I/O file according to Balinese script comparison algorithm (see algorithm 16-1). The complexity of this function is the same as that of searching algorithm of Balinese script, i.e. $O(n^2)$. For the most outer iteration, word boundary breaks down the sentence into the building words. Given a sentence input s with $m = \text{length}(s)$, iterate for m/k words with $k = m$ in the best case and $k = 1$ in the worst case. Therefore, the

complexity of the word boundary algorithm is $O(m*n^2)$.

Input: Balinese script character sequence
 Output: stack of Balinese script character sequence

1. Perform a block validation of Balinese script in Unicode.
2. Break down the character sequence and push the pieces into a stack according to the base character. Then, perform validation based on the dictionary.
3. Perform the algorithm 15-1.
4. If the output is invalid, pop the topmost element from the stack.
5. There are three possible outputs as explained above.

Algorithm 16-1. Balinese Script Word Boundary

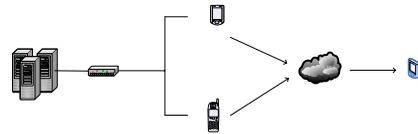
17. Software Specification

Applications made related to text processing based on Unicode standard are B-Linguist, B-Unicode, and desktop application. These applications require JVM (Java Virtual Machine) with the following specification: B-Linguist and B-Unicode is developed on J2ME WTK2 whereas desktop application on J2SDK 1.5.

B-Linguist is a mobile dictionary application which has a function as a translator from Balinese to Indonesian and Balinese to English. B-Unicode is a mobile text processing application of Balinese script according to Unicode standard. This application covers the four main functions, i.e.: transliteration, searching, sorting, and word boundary. The desktop application serves as the dictionary and transliteration generator which is required

by both B-Linguist and B-Unicode. Especially for desktop application, quick sort algorithm is implemented to build index file and dictionary data structure that later are used by both mobile and desktop applications.

All applications utilize a single library named balinese. The balinese library has universal characteristics in order to run on several platforms, i.e.: the platform of J2SDK and J2ME.



Picture 17-1. Software Architecture

In addition, the functions of B-Unicode application are also implemented to desktop application through some simple drivers in order to test and verify the truth of designed algorithm and library. Those drivers are run on TUI (Text User Interface) mode.

In conclusion, picture 17-1 shows that this software involves mobile device network. The hardware specifications used in this research are:

1. HP Sony Ericsson K750i, as well as an active SIM card. The HP serves as a tool running B-Linguist and B-Unicode. It also sends feedback to the developer through SMS technology.
2. A PC serves as a tool running desktop application to generate files required by both B-Linguist and B-Unicode.
3. Data cable or Bluetooth connection serves as a connector between HP and computer.

4. Pocket PC serves as an SMS receiver from mobile application user.

18. Testing

Testing is performed on a PC with following specifications:

1. Processor : AMD Athlon XP 1600+
2. Memory : 1 GB
3. Hard disk : 20 GB

Testing is divided into two parts, i.e. software testing (see table 18-1) and input keyboard testing (see table 18-2).

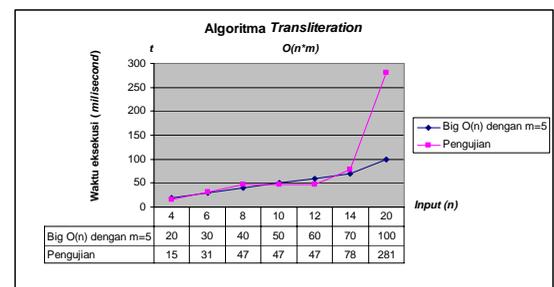
Table 18-1. Software Testing

| No. | Name | Description |
|-------------------|-----------------------------------|---------------------|
| B-Linguist | | |
| 1 | Balinese to English | Successfully tested |
| 2 | Balinese to Indonesian | Successfully tested |
| 3 | Balinese Index to English | Successfully tested |
| 4 | Balinese Index to Indonesian | Successfully tested |
| 5 | Change Language | Successfully tested |
| 6 | Send SMS | Successfully tested |
| B-Unicode | | |
| 7 | Balinese Transliteration to Latin | Successfully tested |
| 8 | Latin Transliteration to Balinese | Successfully tested |
| 9 | Searching | Successfully tested |
| 10 | HANACARAKA Sorting | Successfully tested |
| 11 | SANSKRIT Sorting | Successfully tested |
| 12 | Word Boundary | Successfully tested |

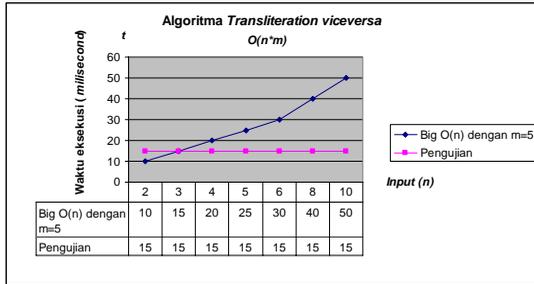
| | | |
|----------------------------|---|---------------------|
| 13 | Change Language | Successfully tested |
| 14 | Send SMS | Successfully tested |
| Desktop Application | | |
| 15 | Balinese Transliteration Generator to Latin | Successfully tested |
| 16 | Latin Transliteration Generator to Balinese | Successfully tested |
| 17 | Unicode Code Generator | Successfully tested |
| 18 | HANACARAKA Generator | Successfully tested |
| 19 | SANSKRIT Generator | Successfully tested |

Table 18-2. Input Keyboard Testing

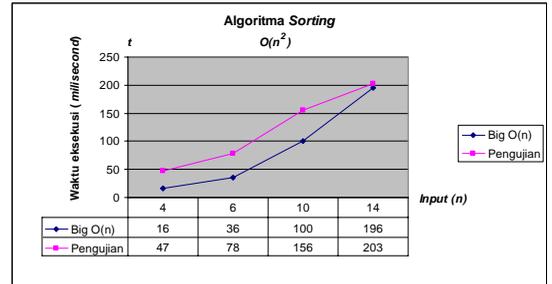
| No. | Input keyboard | Description |
|-----|--|---------------------|
| 1 | NULL | Successfully tested |
| 2 | Only consonants | Successfully tested |
| 3 | Both consonants and vowels | Successfully tested |
| 4 | Both consonant clusters and vowels | Successfully tested |
| 5 | All special cases (e.g., in searching testing) | Successfully tested |



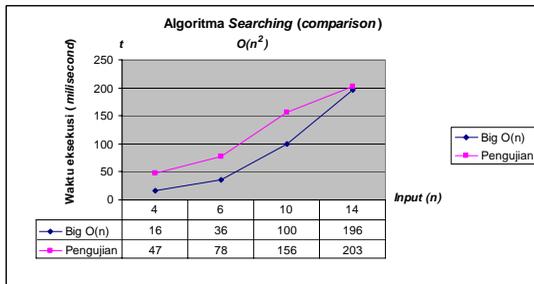
Picture 18-1. Big $O(n)$ Testing of Transliteration Algorithm



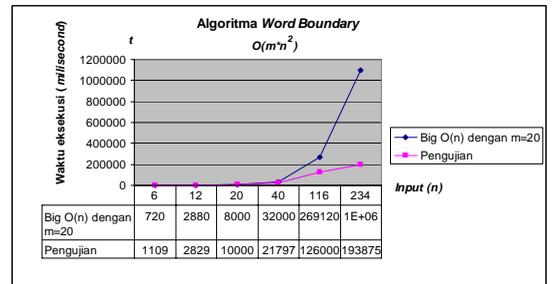
Picture 18-2. Big O(n) Testing of Inverse Transliteration Algorithm



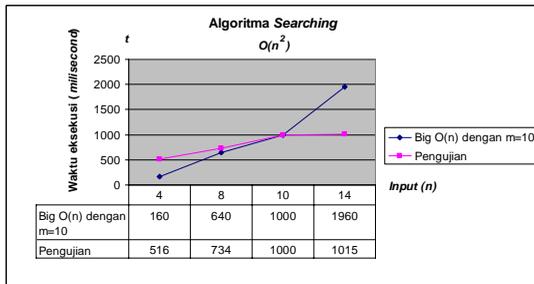
Picture 18-5. Big O(n) Testing of Sorting Algorithm



Picture 18-3. Big O(n) Testing of Searching Algorithm (comparison)



Picture 18-6. Big O(n) Testing of Word Boundary Algorithm



Picture 18-4. Big O(n) Testing of Searching Algorithm

19. Implementation of Application

The result of GUI designing implementation can be seen on picture 19-1, 19-2, and 19-3.





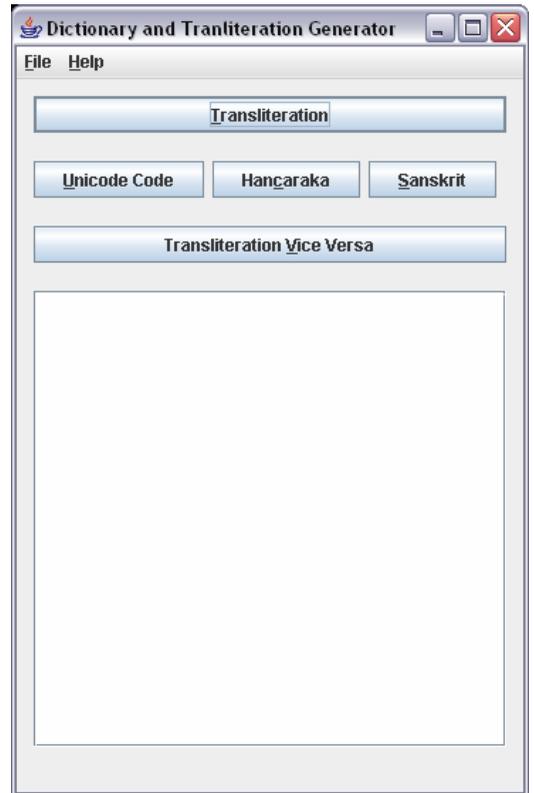
Picture 19-1. User Interface of B-Linguist



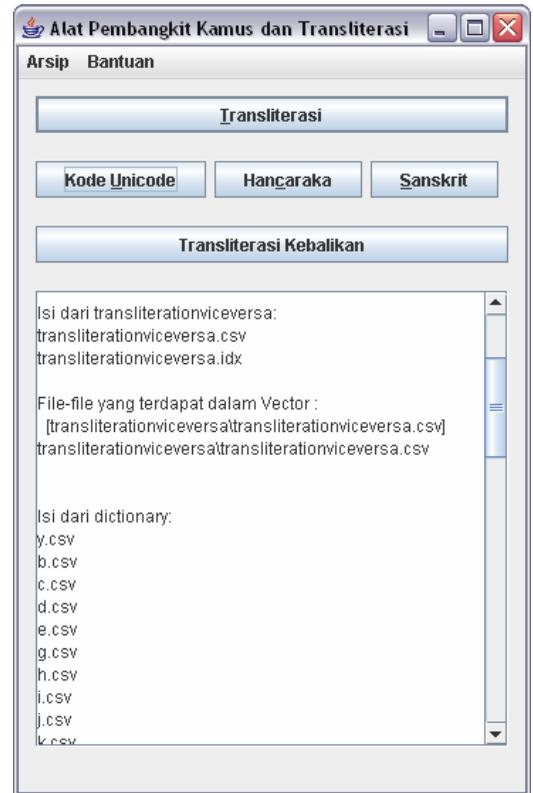
Picture 19-2. User Interface of B-Unicode



Picture 19-2. User Interface of B-Unicode (continued)



Picture 19-3. User Interface of Desktop



Picture 19-3. User Interface of Desktop
(continued)



20. Conclusion and Suggestion

Based on the result of conducted analysis and research, there are several conclusions taken into account, i.e.:

1. The system has successfully performed Balinese Unicode text processing covering transliteration, searching, sorting, and word boundary functions. Moreover, a general and multiplatform library has been successfully built. This library runs well on both computer and handheld device as long as JVM (*java virtual machine*) is installed on them.

2. The algorithm complexity of transliteration, searching, sorting, and word boundary are $O(n*m)$, $O(n^2)$, $O(n^2)$, $O(m*n^2)$ with $n = \text{length}(\text{string input})$ and $m =$ the number of comparison at lookup table in average, respectively.
 3. It is proved that Unicode has been supported by many operating systems and browsers. The trend of global software technology nowadays involves the usage of Unicode standard and the availability of supporting devices.
 4. Building software capable of processing words containing Balinese characters requires a good cooperation between two fields of study, i.e. computer science and Balinese culture and language science. Without mastering those fields, the software would not handle all possible cases.
 5. When analyzing software requirements, communication with the user must frequently be done so that the software meets the demand of the user.
- Beside of the conclusions, there are several suggestions that can be considered, i.e.:
1. An expert of Balinese script is required in order to help develop the software. Specifically:
 - a. Grouping letters
 - b. Recognizing Balinese character properties.
 - c. Analyzing letter combinations.
 - d. Evaluating the correctness of the execution flow of the software.
 2. Communication with the user can be conducted with an interview or a questionnaire to know specifically the following things:
 - a. Layout keyboard compatibility for Balinese letter.
 - b. The means of communication with software, including the language used when interacting.
 3. SIM card used for updating dictionary should be the prepaid one, so that controlling bill is not required.

21. Bibliography

- [CON05] Constable, Peter(2005). Microsoft. *Comments on Balinese Proposal*, L2/05-008.
- [EVE05] Everson, Michael dan I Made Suatjana(2005). *Proposal for Encoding Balinese Script in the UCS*.
- [GIL03] Gilliam, Richard(2003). *Unicode Demystified: A Practical Programmer's Guide to the Encoding Standard*. Addison-Wesley Professional.
- [MAR91] Martha, IBM Jaya(1991). *Perangkat Lunak Teks Editor Berhuruf Bali*.
- [MED96] Medra, I Nengah, et al (1996). *Pembinaan Bahasa, Aksara, dan Sastra Bali: Pedoman Penulisan Papan Nama dengan Aksara Bali*. Dinas Kebudayaan Prop. Dati I Bali.
- [NIK05] Nikanaya, I Nyoman(2005). *Surat Rekomendasi Nomor 042/84/DISBUD tentang Temu Wicara Aksara Bali* (http://anubis.dkuug.dk/JTC1/SC2/WG2/Bali_Government_n2916.pdf).
- [SUT03] Sutjaja, I Gusti Made(2003). *Kamus Sinonim Bahasa Bali*.
- [UNI03] The Unicode Consortium(2003). *The Unicode Standard*, Version 4.0. Addison-

Wesley Professional.

[YBG05] Galang, Yayasan Bali(2005). *Komputerisasi Aksara Bali*. Maret 2005
(<http://www.babadbali.com/aksarabali>).

[WIK05] Wikipedia(2005). *The Free Encyclopedia*. Desember 2005
(<http://www.en.wikipedia.org>).