Pemrosesan Teks Berbasiskan Standar *Unicode* Aksara Bali

Imam Habibi

Departemen Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

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

Abstrak

Pada dasarnya, komputer hanya mengenal angka sebagai representasi sebuah karakter. Sebelum *Unicode* diperkenalkan, terdapat ratusan sistem *encoding* yang berbeda untuk alokasi angka tersebut. Tidak ada sistem *encoding* yang bisa mencakup semua karakter. Di Eropa, berbagai sistem *encoding* dibutuhkan untuk semua bahasanya.

Sebagai solusinya, diperkenalkan sebuah sistem *encoding* yang mampu menyediakan id yang unik bagi setiap karakter yang berbeda tanpa bergantung pada *platform*, program, dan bahasa yang digunakan. Standar *Unicode* telah digunakan oleh berbagai pusat industri seperti Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, dan Unisys. Lagipula, standar bahasa dan pertukaran informasi modern seperti XML, Java, ECMA Script (JavaScript), LDAP, CORBA 3.0, dan WML telah memanfaatkan *Unicode* sebagai sarana resmi untuk mengimplementasikan ISO/IEC 10646.

Ada empat hal penting yang harus dikerjakan menyangkut aksara Bali yaitu algoritma untuk *transliteration*, *searching*, *sorting*, dan *word boundary analysis* (*spell checking*).

Untuk menguji kebenaran algoritma yang dirancang, dibangun beberapa aplikasi yang berjalan dalam *platform* Linux/Windows OS dengan menggunakan *library* J2SDK 1.5 dan J2ME WTK2. *Input* dari algoritma/aplikasi yang dibuat adalah sekuens karakter yang bisa diperoleh dari ketikan *keyboard* dan *file* eksternal. *Output*nya dalam bentuk modul atau *library* yang dapat melakukan pemrosesan teks aksara Bali sesuai standar *Unicode*.

Kata kunci: *Unicode*, *transliteration*, *searching*, *sorting*, *word boundary analysis*, *canonical combining class*, normalisasi, dan *Unicode* Elemen *collation*.

1. Pendahuluan

Bahasa dan aksara daerah adalah kekayaan budaya yang patut dilestarikan. Aksara Bali yang digunakan untuk menuliskan bahasa Bali sekarang semakin jarang digunakan dan lingkup pemakaiannya semakin sempit. Usaha-usaha untuk melestarikannya sudah ada namun mengalami kendala, salah satunya adalah kurangnya alat bantu untuk mengakomodasi pemikiran-pemikiran yang menggunakan aksara Bali. Pada dasarnya, semakin canggih alat bantu tersebut maka semakin terjamin pula kelancaran dalam proses pendidikan dan pendalaman budaya Bali di masa depan. Alat bantu yang dimaksud adalah komputer karena komputer merupakan alat yang dapat dengan mudah melakukan rekayasa sedemikian rupa sehingga dapat menghasilkan dan memproses aksara Bali secara cepat, indah dan baku [YBG05].

Upaya untuk melakukan komputerisasi aksara Bali sedang dilakukan oleh Yayasan Bali Galang. Langkah pertama dilakukan dengan memasukkan aksara Bali dalam standar karakter Unicode. Unicode Consortium[1] dan komite ISO/IEC JTC1/SC2/WG2[2] telah menyetujui masuknya aksara Bali seperti yang dinyatakan dalam proposal formal yang ditulis oleh Michael Everson dan I Made Suatjana dalam standar yang mereka kelola [EVE05]. Proposal yang bernomor N2908 tersebut dipresentasikan pada pertemuan WG2 yang ke-42 di Xiamen, Cina pada bulan Januari 2005. Informasi dalam proposal tersebut sudah lengkap namun finalisasinya diputuskan pada pertemuan WG2 selanjutnya di Sophia-Antipolis, Perancis pada bulan September 2005.

Pemrosesan aksara Bali tidak dapat dilakukan hanya dengan menerapkan metode konvensional untuk aksara Latin karena terdapat sedikitnya tiga perbedaan untuk algoritma pemrosesan teks aksara Bali [EVE05], yaitu:

1. Algoritma pencarian harus bekerja pada karakter-karakter baik sebelum maupun sesudah digabung. Pencarian untuk U+1B12 BALINESE LETTER OKARA TEDUNG ᬑᬾ harus ekivalen dengan pencarian untuk U+1B11 BALINESE LETTER OKARA ᬑ dan U+1B35 BALINESE VOWEL SIGN TEDUNG ◌ᬵ.
2. Algoritma pengurutan tidak boleh hanya berbasiskan *character code point*. Vokal harus diabaikan ketika membandingkan konsonan, tetapi vokal baru diperhitungkan ketika konsonannya sudah dinyatakan identik. Terlebih lagi, terdapat 2 metoda pengurutan yang berbeda, yaitu pengurutan aksara Bali tradisional HANACARAKA dan pengurutan Sanskrit. *Sorting* yang dilakukan bukan berdasarkan *character code point* tetapi sesuai dengan standar *Unicode* aksara Bali.
3. Teks aksara Bali tidak menggunakan spasi sebagai pembatas kata. Karena itu, algoritma pengecekan ejaan harus dapat dilakukan berbasiskan kamus untuk menentukan potongan-potongan kata serta validasi ejaan teks.

2. Pemrosesan Teks di Komputer

Informasi yang mengalir dari dan menuju komputer umumnya berupa dokumen teks, gambar, audio, video dan kombinasi diantaranya. Teks digunakan untuk menyampaikan informasi dalam bahasa dan ditulis dengan aksara-aksara yang dapat dimengerti oleh manusia sebagai subyek sekaligus obyek dari informasi tersebut. Agar dapat diproses di komputer, aksara-aksara tersebut perlu dikodekan dalam angka karena komputer hanya dapat bekerja dalam angka. Angka yang dimaksud adalah bilangan berbasis dua yang terdiri atas angka 0 dan 1 yang biasanya disebut dengan *bit*.

Pada kenyataannya, pemrosesan *bit* ini dilakukan dalam *octet* (dari kata Latin, *octo* yang berarti delapan), yaitu rangkaian bit sebanyak 8 atau disebut juga *byte*. Berbagai macam konvensi dibuat untuk mencari solusi bagaimana menginterpretasikan *octet* atau rangkaian *octet* untuk menyajikan data. Sebagai contoh, rangkaian empat *octet*

[1] http://www.unicode.org
[2] http://anubis.dkuug.dk/JTC1/SC2/WG2

digunakan untuk menginterpretasikan bilangan riil. Dalam bahasan makalah ini, rangkaian *octet* ini digunakan untuk menginterpretasikan *string* atau rangkaian karakter. Cara paling sederhana yang masih digunakan secara luas sampai sekarang untuk menginterpretasikan karakter adalah dengan memetakan satu *octet* dengan satu karakter berdasarkan tabel pemetaan. Dengan cara ini kita dapat merepresentasikan 256 (2^8=256) karakter. Jumlah ini melebihi jumlah karakter dalam *character set*[3] aksara Latin yang digunakan untuk menulis berbagai bahasa di dunia termasuk bahasa Inggris dan bahasa Indonesia. Cara ini pula yang digunakan oleh standar karakter ASCII (*American Standard Character for Information Interchange*) yang dirancang pada era 1960 dan masih digunakan sampai sekarang.

Secara umum, pemrosesan teks di komputer bermula dari *input user* melalui *keyboard* dan kode yang ditekan dikirimkan ke *keyboard driver*. *Keyboard driver* dapat melakukan transformasi dari rangkaian kode *keyboard* menjadi rangkaian kode karakter yang benar. Kode karakter ini dimanipulasi di dalam *text processor* yang memungkinkan dilakukannya *searching, copy-paste, sorting,* penghitungan kata, penghitungan baris, *line breaking,* dan lain-lain. Kode rangkaian karakter ini juga yang akan disimpan di dalam memori atau media penyimpanan lain. Untuk menampilkan rangkaian kode karakter menjadi *glyph* yang sesuai ke *output device,* seperti monitor dan printer, dilakukan *rendering*.

3. Standar Pengkodean Karakter *Unicode*

Unicode adalah metode standar pengkodean karakter untuk merepresentasikan bahasa tulis dalam komputer. *Unicode* bukan standar pengkodean karakter yang diciptakan pertama kalinya, melainkan jawaban atas permasalahan-permasalahan yang muncul dalam pengkodean karakter-karakter di dunia selama puluhan tahun [UNI03]. Oleh karena itu *Unicode* sangat erat kaitannya dengan standar pengkodean karakter yang telah ada sebelumnya. Ketika standar *Unicode* versi 1.0 dikeluarkan pada tahun 1991, ASCII dan ISO-8859 adalah standar yang paling banyak digunakan.

Perancangan model karakter standar *Unicode* menganut 10 prinsip dasar yang dinyatakan di bawah ini [UNI03]. Tidak semua prinsip dasar ini dipenuhi semuanya. Pemeliharaan konsistensi dapat dikorbankan demi menjaga kesederhanaan, efisiensi dan kesesuaian dengan standar-standar sebelumnya. Kesepuluh prinsip dasar tersebut adalah:

1. Universalitas
2. Efisiensi
3. Karakter, bukan *glyph*
4. Semantik
5. *Plain text*
6. *Logical order*
7. Unifikasi
8. Komposisi dinamis
9. Rangkaian sepadan
10. *Convertibility*

4. Aksara Bali

Aksara Bali digunakan untuk menuliskan bahasa Bali yang merupakan bahasa penduduk Bali asli. Aksara Bali merupakan turunan dari aksara Brahmi di India dan memiliki banyak kemiripan dengan aksara modern yang digunakan di Asia Selatan dan Asia Tenggara karena mereka berasal dari induk aksara yang sama. Aksara Bali juga digunakan untuk

[3] Himpunan karakter dalam suatu aksara dan belum dikaitkan dengan representasi kodenya (misalnya alfabet bahasa Indonesia beserta tanda baca.

menuliskan bahasa Sasak yang digunakan di Lombok dan Bali bagian timur.

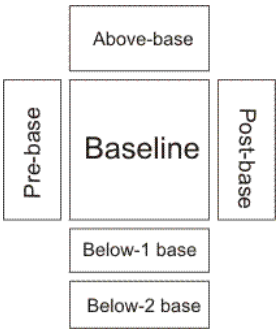
Sistem penulisan aksara Bali sangat kompleks bila dibandingkan dengan aksara Latin. Abjad aksara Bali terdiri atas suku kata. Setiap abjad suku kata tersebut berakhiran dengan bunyi vokal /a/.

Kelompok konsonan (*consonant clusters)* adalah sekelompok konsonan pada suatu suku kata yang muncul tanpa adanya vokal. Pada aksara Bali, kosonan secara alami mendapat akhiran bunyi vokal /a/. Secara umum terdapat dua cara untuk menghilangkan bunyi vokal alami tersebut, yaitu:

1. memakai konsonan bentuk *gantungan* atau *gempelan* pada konsonan berikutnya. Konsonan bentuk *gantungan* atau *gempelan* ini digunakan untuk menghilangkan vokal pada konsonan sebelah kirinya, bukan menghilangkan vokal pada dirinya sendiri. Contohnya adalah kata ᬩᬓᭂ ‘bakta’ (*bring*).
2. memakai *adeg-adeg*, U+1B44 BALINESE ADEG-ADEG. Contohnya adalah kata ᬓᬤᭂᬥ᭄ ‘kadep’ (*sold*).

Pengaturan posisi dalam penulisan aksara Bali secara logik dibagi dalam beberapa area (lihat gambar 4-1), yaitu:

1. *Baseline area*: basis penulisan aksara Bali. Konsonan ditulis dalam area ini.
2. Area sebelah kiri atau *pre-base marks*(*prem*) dan kanan atau *post-base marks* (*pstm*) *baseline*: digunakan untuk menuliskan *dependent vowel* dan *gempelan*.
3. Area sebelah atas atau *above-base marks* (*abvm*) dan bawah atau *below-1 base marks* (*blw1m*) dan *below-2 base marks*(*blw2m*) *baseline*: digunakan untuk menuliskan *gantungan*, *pengangge suara*.

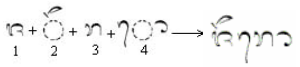


Gambar 4-1. Posisi penulisan aksara Bali

5. *Reordering* dan *Split Vowel*

Dependent vowel pada aksara Bali memodifikasi suku kata konsonan dasarnya dengan berbagai bentuk. Sebuah konsonan atau kelompok konsonan dapat memiliki *dependent vowel* untuk mengubah bunyi vokal akhir konsonan atau kelompok konsonan tersebut. Aksara Bali memiliki berbagai macam bentuk *dependent vowel* baik *nonspacing* maupun *spacing* yang dituliskan pada posisi sebelum, sesudah, di atas, dan di bawah karakter dasar, atau kombinasi di antaranya.

Standar *Unicode* menyatakan bahwa *combining character* dikodekan setelah karakter dasarnya, sehingga apabila rangkaian karakter *Unicode* mengandung karakter-karakter *dependent vowel*, perlu dilakukan *reordering* di dalam memori komputer terlebih dahulu pada waktu ditampilkan ke layar. *Reordering* ini berfungsi untuk melakukan perubahan urutan *glyph* agar komponen *glyph dependent vowel* dapat tergambar dengan benar (lihat gambar 5-1).



Gambar 5-1. Reordering

Split vowel adalah vokal yang komponen-komponennya muncul pada dua sisi yang berbeda pada konsonan dasarnya. Kemunculan komponen ini dapat muncul di sisi atas dan kanan, atau di sisi kiri dan kanan konsonan dasar.

Glyph karakter-karakter vokal yang digambarkan di sisi bawah konsonan dasar perlu mendapat perlakuan khusus karena pemilihan *glyph* tergantung pada konteks kemunculan konsonan atau kelompok konsonan sebelumnya. Karakter-karakter tersebut adalah 1B38 BALINESE VOWEL SIGN SUKU (u) dan 1B39 BALINESE VOWEL SIGN SUKU ILUT (uu). Kedua karakter ini memiliki dua macam bentuk *glyph*, yaitu bentuk yang dilekatkan pada konsonan dasar bentuk biasa dan konsonan dasar bentuk gantungan (Pengangge Aksara).

6. *Ligatures*

Sebuah *glyph* yang dapat merepresentasikan lebih dari satu karakter disebut *ligature*. Dalam hal ini, bila aksara Bali dituliskan di atas kertas dengan pena, maka lebih dari satu karakter dapat dituliskan dengan satu goresan tanpa mengangkat pena tersebut. Beberapa karakter aksara Bali yang muncul berdampingan membentuk *ligature*. Oleh karena itu, karakter-karakter tersebut bila ditampilkan di layar komputer atau media lain seakan-akan tampak seperti sebuah *glyph*. Contohnya adalah karakter U+1B35 BALINESE VOWEL SIGN TEDUNG (aa) yang membentuk *ligature* bila menempel pada suku kata.

7. *Line Breaking*

Meskipun penulisan aksara Bali tanpa spasi antar kata, pemenggalan baris (*line breaking*) tidak dapat dilakukan pada sembarang tempat. Terdapat dua aturan umum dalam pemenggalan baris, yaitu:

1. pemenggalan baris tidak dapat dilakukan di antara suku kata dan semua *combining characters* yang mengikutinya
2. pemenggalan baris tidak dapat dilakukan tepat sebelum tanda baca

8. Karakteristik Aksara Bali

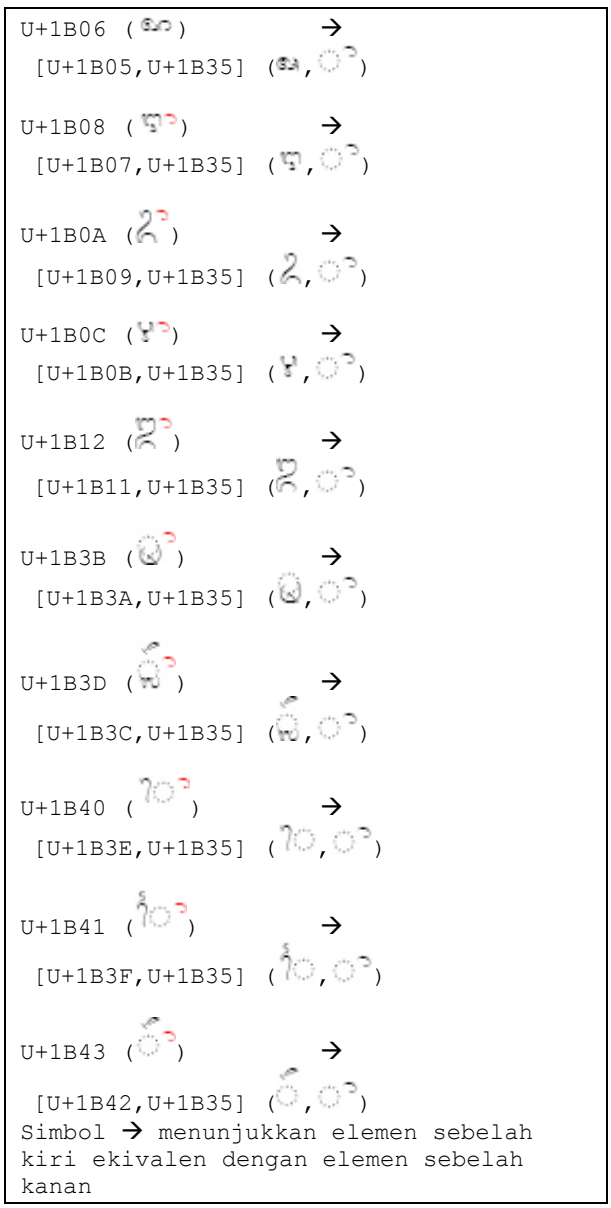
Aksara Bali memiliki karakteristik seperti halnya karakter *Unicode* lainnya (lihat tabel 8-1). Karakteristik ini telah dimuat dalam proposal L2/05-008 yang telah disetujui oleh *Unicode Consortium*. Akan tetapi, *decomposition mapping property* perlu ditambahkan dalam proposal tersebut. Oleh karena itu, sesuai dengan standar *Unicode*, dalam aksara Bali terdapat proses *decomposition mapping* yang dilakukan pada 10 karakter (algoritma 8-1).

Tabel 8-1. Karakteristik karakter Aksara Bali

```
1B00;BALINESE SIGN ULU
RICEM;Mn;230;NSM;;;;;N;;ardhacandra
;;;

1B01;BALINESE SIGN ULU
CANDRA;Mn;230;NSM;;;;;N;;candrabind
u;;;

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;;;
```



Algoritma 8-1. Decomposition mapping karakter Aksara Bali [CON05]

9. *Canonical Combining Class* Aksara Bali

Tujuan dari proses *canonical combining class* adalah untuk mendapatkan/memperoleh kelas-kelas ekivalen yang tepat sesuai dengan standar normalisasi *Unicode* terhadap sekuens-sekuens karakter dan *combining mark*-nya.

Ada dua hal yang menjadi pokok permasalahannya, yaitu:

1. Permasalahan jika terdapat sepasang *combining mark* pada posisi/letak yang sama relatif terhadap karakter dasar. Proses *encoding* yang urutannya berbeda menghasilkan tampilan *mark* dan semantik yang berbeda. Dengan menetapkan bahwa *marks* tersebut berada dalam *canonical combining class* yang sama (nol atau bukan nol), urutan yang berlainan dan bersifat *non canonically equivalent* dapat terpenuhi melalui normalisasi.
2. Permasalahan jika terdapat sepasang *combining mark* pada posisi/letak yang berbeda relatif terhadap karakter dasar. Proses *encoding* yang urutannya berbeda menghasilkan tampilan *mark* dan semantik yang sama. Dengan menetapkan bahwa *marks* tersebut berada dalam *canonical combining class* yang berbeda dan bukan nol, urutan yang berlainan dan bersifat *canonically equivalent* dapat terpenuhi melalui normalisasi.

Dalam *canonical combining class*, terdapat aturan bahwa kelas 0 merupakan kelas khusus yang memiliki karakteristik untuk setiap kelas *n* (*n* bukan nol) yang mengikuti kelas 0. Karakteristik tersebut adalah bahwa kelas 0 memiliki kelas *n* tersebut, sehingga urutan yang berbeda menjadi *non canonically equivalent* melalui normalisasi.

Dalam proposal L2/05-008, hanya terdapat sepasang *combining mark* dengan

urutan yang berbeda yang menghasilkan *canonically equivalent* (tabel 9-1).

Tabel 9-1. Canonical combining class *Aksara Bali pada L2/05-008*

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

Kendala L2/05-008 adalah bahwa penggunaan kelas 0 berhasil memproses *combining mark* yang posisinya sama, tetapi secara visual berbeda menjadi *non canonically equivalent*. Akan tetapi, kelas 0 gagal memproses *combining mark* yang berlainan posisinya menjadi *canonically equivalent*. Sesuai standar *Unicode*, usulan perbaikan yang dilakukan berupa pemberian kelas 220, 224, 226, 230 masing-masing untuk setiap karakter pada posisi bawah, kiri, kanan, dan atas [CON05], sedangkan karakter U+1B34 BALINESE SIGN REREKAN dan U+1B44 BALINESE SIGN VIRAMA tetap memiliki kelas 7 dan 9.

10. *Normalisasi* Aksara Bali

Terdapat empat jenis normalisasi dalam standar *Unicode*, yaitu:

1. Normalisasi bentuk D yang sama dengan proses *canonical decomposition*.
2. Normalisasi bentuk C yang sama dengan proses *canonical decomposition* dan dilanjutkan proses *canonical composition*.
3. Normalisasi bentuk KD yang sama dengan proses *compatibility decomposition*.
4. Normalisasi bentuk KC yang sama dengan proses *compatibility decomposition* dan dilanjutkan proses *canonical composition*.

Berdasarkan pertimbangan efisiensi dan performansi algoritma normalisasi *Unicode*, aksara Bali diproses dalam normalisasi bentuk D. Di samping itu, normalisasi bentuk C juga pada dasarnya harus melalui tahapan yang sama dengan bentuk D.

Algoritma normalisasi bentuk D terdiri dua tahapan. Pertama, pada setiap kata Bali dilakukan proses *decomposition mapping*. Kedua, dilakukan *canonical reordering* berdasarkan *canonical combining class* dari masing-masing karakter. *Decomposition mapping* memiliki sifat rekursif, sehingga didapatkan sekuens-sekuens karakter yang benar (algoritma 10-1). Contoh penggunaan *decomposition mapping* aksara Bali adalah sebagai berikut. U+1B40 menjadi [U+1B3E,U+1B35] sehingga  <U+1B13 KA,U+1B40 VOWEL SIGN TALING TEDUNG> ≡ <U+1B13 KA,U+1B3E VOWEL SIGN TALING,U+1B35 VOWEL SIGN TEDUNG>.

If: X → [Y,Z] (terdefinisi) If: Y → [Y_1,Y_2] (terdefinisi) Then: X → [Y_1,Y_2,Z] (kesimpulan) Simbol → menunjukkan elemen sebelah kiri ekivalen dengan elemen sebelah kanan

Algoritma 10-1. Algoritma Decomposition mapping

11. *Unicode* Elemen *collation* (UCA)

Pada dasarnya, algoritma UCA merupakan proses pembentukan elemen-elemen pembanding (*key sorting*) yang

memiliki nilai prioritas tertentu. Suatu *string* diproses terlebih dahulu pada *primary level*. Jika hasil perbandingan masih sama, digunakan perbandingan pada *secondary level*, dan terakhir pada *tertiary level*.

Pada umumnya, yang menjadi urutan/ukuran pembanding UCA adalah:

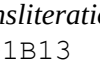
1. Pengurutan abjad/karakter dasar
2. *Diacritic mark*/*accent*
3. *Uppercase* dan *lowercase*

12. Algoritma Perbandingan Aksara Bali

Algoritma Perbandingan untuk aksara Bali terdiri dari 4 proses yaitu:

1. Setiap *input* aksara Bali diproses terlebih dahulu dalam algoritma normalisasi bentuk D.
2. Hasil dari tahap (1) kemudian dilanjutkan dengan algoritma UCA sesuai dengan metoda pengurutan yang diinginkan (HANACARAKA atau SANSKRIT).
3. Selanjutnya adalah pemisahan jenis nilai-nilai elemen *collation* dan penggabungan kembali nilai-nilai tersebut dengan nilai tertentu dari *level separator*.
4. Tahap terakhir adalah perbandingan nilai-nilai yang didapatkan pada tahap (3) menggunakan algoritma *binary comparison*.

13. *Transliteration* Aksara Bali

Transliteration merupakan pemetaan dari sebuah sistem penulisan ke bentuk sistem penulisan yang lain, misalnya dari aksara Bali ke Latin dan sebaliknya dengan memperhatikan aksen dan tatabahasa aksara tersebut [WIK05]. Kriteria yang harus dipenuhi oleh *transliteration* adalah *lossless information*, sehingga *user* seharusnya dapat mengembalikan informasi yang didapatkan ke dalam format aslinya. Oleh karena itu, *transliteration* berbeda dengan *transcription* yang hanya melakukan pemetaan suara dari satu bahasa ke aksara bahasa lainnya. Kegunaan *transliteration* adalah untuk membantu orang yang tidak bisa membaca aksara Bali. Contoh *transliteration* aksara Bali adalah  [U+1B13 KA,U+1B2E LA] menjadi ‘kala’ (*time*).

Transliteration dilakukan dengan cara membuat tabel untuk memetakan karakter-karakter dari aksara Bali ke Latin dan sebaliknya. Untuk mencapai tujuan *transliteration*, diperlukan konversi yang kompleks dalam menangani perubahan bentuk dan kasus khusus huruf-huruf dalam aksara asal. *Input* untuk fungsi *transliteration* adalah 2 digit dari *keyboard* dalam format hexadesimal (0,1,2,…,A,B,C,D,E,F) dengan *padding bit* 0 jika *input* digit *user* tidak sebanyak kelipatan 2. Algoritma *transliteration* memanfaatkan struktur data *inversion list* (termasuk fungsi-fungsi dasar untuk *invert*, *union*, *intersection*, *set difference*, *add*, dan *delete*), sehingga dapat menghemat *storage*/*memory* yang dipakai (lebih *compact*). Performansinya lebih cepat karena adanya *random access* yang sama untuk tiap elemen yang ingin dicari.

Karakter U+1B33 BALINESE LETTER HA  dapat berfungsi sebagai tempat netral untuk karakter vokal, sehingga jika karakter vokal ingin dituliskan pada posisi awal suatu kata, selain digunakan karakter *independent vowel,* juga dapat digunakan karakter U+1B33 yang diikuti dengan karakter *dependent vowel* yang bersangkutan.

Karakter U+1B05 BALINESE LETTER AKARA  dipakai sebagai tempat netral ketika diikuti karakter/tanda yang bersesuaian, sehingga karakter ini dapat ditransliterasi menjadi ‘e’, ‘i’, ‘o’, ‘u’, ‘ī’, ‘ū’, ‘ĕ’, dan ‘ŏ’.

Karakter nania (U+1B2C) digunakan dalam kelompok konsonan di antara kata-kata yang memakai bentuk *appended* ‘ya’, sehingga karakter ini ditransliterasi menjadi ‘ia’. Contohnya adalah pada kata ‘siap’ dan ‘tabia’.

Karakter suku kembung (U+1B2F) digunakan dalam kelompok konsonan di antara kata-kata yang memakai bentuk *appended* ‘wa’, sehingga karakter ini ditransliterasi menjadi ‘ua’.

Algoritma *transliteration* memanggil fungsi *translate string* yang menerima *input string* kata dan *output string* (algoritma 13-1 dan 13-2). Misalkan *input string s* dengan $n = length(s)$, maka lakukan iterasi sebanyak n karakter.

Input : sekuens karakter aksara Bali
Output : sekuens karakter aksara Latin
1. Lakukan validasi blok *Unicode* karakter aksara Bali.
2. Iterasi mulai dari karakter awal sampai akhir.
3. Jika karakter tersebut adalah *base character* maka ke langkah 4. Selain itu ke langkah 7.
4. Jika karakter tersebut menghasilkan *glyph* dalam bentuk *appended form* berarti terdapat pemakaian karakter virama.
5. Periksa apakah karakter yang dimaksud merupakan karakter dengan kemungkinan kasus khusus seperti yang telah disebutkan sebelumnya.
6. Jika karakter merupakan karakter rerekan maka harus diperiksa validasi karakter sebelumnya. Di samping itu, *output* yang sebelumnya juga turut berubah.
7. Jika karakter merupakan karakter vokal *dependant* maka karakter vokal *base character* sebelumnya berubah menjadi karakter yang bersangkutan.
8. Kemudian, lakukan pencocokan karakter dengan tabel *mapping* dalam bentuk yang sesuai. Bentuk yang tersedia adalah *normal form*, *appended form*, *case 1...8 form*, dan *UNKNOWN_TRANSLATE*.

Algoritma 13-1. Transliteration karakter Aksara Bali

Input : sekuens karakter aksara Latin
Output : sekuens karakter aksara Bali
1. Lakukan validasi blok *Unicode* karakter aksara Latin.
2. Jika terdapat karakter pemisah maka gabung karakter-karakter yang terpisah tersebut.
3. Iterasi mulai dari karakter awal sampai akhir.
4. Periksa apakah karakter yang dimaksud merupakan karakter dengan kemungkinan kasus khusus. Contohnya adalah karakter vokal.
5. Periksa apakah *input* merupakan *balanced word* (kata yang simetris) pada posisi karakter yang bersangkutan.
6. Validasi apakah *output* mengandung karakter virama dan rerekan.
7. Kemudian, lakukan pencocokan karakter dengan tabel *mapping* dalam bentuk yang sesuai. Bentuk yang tersedia adalah *normal form*, *appended form*, *case 1...8 form*, dan *UNKNOWN_TRANSLATE*.
8. Terakhir, verifikasi apakah karakter *output* termasuk dalam konsonan aksara Bali memanfaatkan struktur data *InversionList*. Jika konsonan maka tambahkan *output* dengan karakter virama.

Algoritma 13-2. Transliteration kebalikan karakter Aksara Bali

Pada algoritma *transliteration*, fungsi yang paling dominan adalah fungsi pencarian karakter dalam I/O *file*. Perbandingan dilakukan pada blok aksara Bali (U+1B00–U+1B7F). Dalam kasus terburuk, jumlah perbandingan dilakukan sebanyak 128 kali.

Jika dinyatakan bahwa m merupakan jumlah perbandingan pada *lookup table* secara rata-rata, secara teoritis algoritma *transliteration* memiliki kompleksitas O(*n***m*).

14. *Searching* Aksara Bali

Searching merupakan proses untuk mencari karakter aksara Bali. Tantangan dari proses ini adalah bahwa terdapat beberapa karakter yang bisa digabung menjadi karakter tunggal atau dipecah menjadi beberapa karakter, sehingga pencarian harus bisa menemukan kombinasi karakter yang digunakan.

Searching dilakukan dengan memvalidasi bentuk-bentuk ekivalensi dalam aksara Bali (tabel 14-1). Contoh penggunaan *searching* aksara Bali adalah  ‘daar’ (*eat*) dan  ‘daara’ (*eaten*),  ‘baang’ (*give*) dan  ‘baanga’ (*given*).

Pada algoritma *searching*, fungsi yang paling dominan adalah fungsi normalisasi karakter aksara Bali, fungsi pembentukan *sorting key* sesuai aturan UCA, dan fungsi perbandingan secara biner (algoritma 14-1). Fungsi normalisasi menerima *input string* kata dan *array of int*. Misalkan *input string s* dengan *n* = *length*(*s*), maka lakukan iterasi sebanyak *n* karakter. Fungsi ini terdiri atas dua proses:

1. *canonical decomposition* merupakan fungsi rekursif yang mengubah karakter aksara Bali dalam bentuk atomik. Dalam kasus terbaik, *input* karakter telah memiliki bentuk atomik. Dalam kasus terburuk, *input* karakter diproses secara rekursif, sehingga karakter tersebut berada dalam bentuk atomik. Untuk kasus aksara Bali, tingkat kedalaman rekursif berjumlah 1 kali.
2. *reordering canonical combining class* merupakan fungsi menyusun kelas-kelas karakter aksara Bali yang telah diproses dalam *canonical decomposition*. Dalam kasus terburuk, iterasi dilakukan sejumlah tingkat kedalaman rekursif - 1.

Jika dinyatakan bahwa *p* merupakan tingkat kedalaman rekursif secara rata-rata, secara teoritis fungsi normalisasi memiliki kompleksitas O(*p*).

Secara rata-rata, fungsi elemen *collation* melakukan iterasi yang berjumlah *n*, sehingga *array of int* dihasilkan dengan ukuran *n*. Oleh karena itu, fungsi ini memiliki kompleksitas O(*n*).

Fungsi *binary comparison* melakukan perbandingan nilai-nilai elemen *collation*. Dalam kasus terburuk, jumlah perbandingan dilakukan sebanyak *n* kali.

Secara teoritis algoritma *searching* memiliki kompleksitas:

$T(n) = O(n*(O(p)) + n*(O(n)) + n)$

$= O(n*p + n^2 + n)$ → ambil nilai maksimum diantara ketiga kompleksitas tersebut ($p<n$)

$= O(n^2)$

Oleh karena itu, kompleksitas algoritma *searching* adalah $O(n^2)$.

Tabel 14-1. Bentuk ekivalen karakter Aksara Bali

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

Input : sekuens karakter aksara Bali
Output : *list of* sekuens karakter aksara Bali
1. Lakukan validasi blok *Unicode* karakter aksara Bali.
2. Ubah *input* ke dalam bentuk ekivalen karakter aksara Bali sesuai dengan tabel 14-1.
3. Lakukan normalisasi karakter aksara Bali.
4. Bentuk *sorting key* berdasarkan aturan elemen *collation*.
5. Gunakan struktur data *ListSorting* dan perhatikan metoda *sorting* yang dipakai apakah HANACARAKA atau SANSKRIT.
6. Lakukan perbandingan *sorting key* secara *binary*.
7. Jika nilainya sama maka masukkan ke dalam list *output*.

Algoritma 14-1. Searching karakter Aksara Bali

15. *Sorting* Aksara Bali

Sorting merupakan proses untuk mengurutkan tulisan aksara Bali dengan menggunakan metoda *sorting* Sanskrit dan aksara Bali tradisional HANACARAKA.

Berdasarkan aksara Devanagari dan Latin yang terurut berdasarkan *character code point*, pengurutan aksara Bali diproses dengan urutan:

1. Tanda Baca sebanyak 7 karakter yaitu ‘carik’, ‘carik pareren’, ‘panten’, ‘pasalinan’, ‘pamada’, ‘carik agung’, dan ‘carik pamungkah’ (U+1B5A – U+1B60).
2. Angka sebanyak 10 karakter (U+1B50 – U+1B59).
3. Metoda pengurutan HANACARAKA (tabel 15-1) dan SANSKRIT (tabel 15-2).
4. Simbol lainnya (U+1B61 – U+1B7C).

Oleh karena itu, algoritma *sorting* aksara Bali tidak berdasarkan *character code point*. Di samping itu, vokal harus diabaikan ketika konsonan dibandingkan. Vokal baru diperhitungkan ketika konsonan sudah dinyatakan identik. Contoh penggunaan *sorting* aksara Bali adalah ᬓᬫ ‘krama’ (*member*) muncul terlebih dahulu dibanding ᬘᬓ ’cakra’ (*disc*) pada metoda *sorting* Sanskrit sedangkan ‘cakra’ muncul terlebih dahulu dibanding ‘krama’ pada metoda *sorting* HANACARAKA.

Tabel 15-1. Pengurutan HANACARAKA

```
A > Ā > Ë > Ö > I > Ī > U > Ū > E > AI
> O > AU >
ha (bisah) > ha-rerekan > na > nna >
ca > cha > ra ([ra-repa = rë], 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 = lë]) >
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
```

Tabel 15-2. Pengurutan SANSKRIT

```
A > Ā > Ë > Ö > I > Ī > U > Ū > RË >
RÖ > LË > 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
```

Pada algoritma *sorting*, fungsi yang paling dominan adalah fungsi normalisasi karakter aksara Bali, fungsi pembentukan *sorting key* sesuai aturan UCA, dan fungsi perbandingan secara biner (algoritma 15-1).

```
Input : pair of sekuens karakter
aksara Bali
Output : urutan sekuens karakter (-
1=lebih kecil, 0=sama, 1=lebih besar)
1. Lakukan validasi blok Unicode
   karakter aksara Bali.
2. Lakukan normalisasi karakter
   aksara Bali.
3. Bentuk sorting key berdasarkan
   aturan elemen collation.
4. Gunakan struktur data
   ListSorting dan perhatikan
   metoda sorting yang dipakai
   apakah HANACARAKA (tabel 15-1)
   atau SANSKRIT(tabel 15-2).
5. Lakukan perbandingan sorting key
   secara binary.
```

Algoritma 15-1. Sorting karakter Aksara Bali

Fungsi normalisasi menerima *input string* kata dan *array of int*. Misalkan *input string s* dengan $n = length(s)$, maka lakukan iterasi sebanyak n karakter. Fungsi ini terdiri atas dua proses:

1. *canonical decomposition* merupakan fungsi rekursif yang mengubah karakter aksara Bali dalam bentuk atomik. Dalam kasus terbaik, *input* karakter telah memiliki bentuk atomik. Dalam kasus terburuk, *input* karakter diproses secara rekursif, sehingga karakter tersebut berada dalam bentuk atomik. Untuk kasus aksara Bali, tingkat kedalaman rekursif berjumlah 1 kali.
2. *reordering canonical combining class* merupakan fungsi menyusun kelas-kelas karakter aksara Bali yang telah diproses dalam *canonical decomposition*. Dalam kasus terburuk, iterasi dilakukan sejumlah tingkat kedalaman rekursif - 1.

Jika dinyatakan bahwa p merupakan tingkat kedalaman rekursif secara rata-rata, secara teoritis fungsi normalisasi memiliki kompleksitas $O(p)$.

Secara rata-rata, fungsi elemen *collation* melakukan iterasi yang berjumlah *n*, sehingga *array of int* dihasilkan dengan ukuran *n*. Oleh karena itu, fungsi ini memiliki kompleksitas O(*n*).

Fungsi *binary comparison* melakukan perbandingan nilai-nilai elemen *collation*. Dalam kasus terburuk, jumlah perbandingan dilakukan sebanyak *n* kali.

Secara teoritis algoritma *sorting* memiliki kompleksitas:

$T(n) = O(n*(O(p)) + n*(O(n)) + n)$

$= O(n*p + n^2 + n)$ → ambil nilai maksimum diantara ketiga kompleksitas tersebut ($p<n$)

$= O(n^2)$

Oleh karena itu, kompleksitas algoritma *sorting* adalah $O(n^2)$.

16. *Word Boundary* Aksara Bali

Dalam terminologi dunia komputer, *word boundary* atau *spell checker* merupakan suatu fasilitas perangkat lunak yang didesain untuk melakukan verifikasi ejaan kata-kata dalam sebuah dokumen, sehingga dapat membantu *user* mendapatkan ejaan yang benar [WIK05].

Untuk menangani aksara Bali, *word boundary* memecah struktur kalimat menjadi kata-kata pembentuknya. Aksara Bali tidak mengenal adanya pemakaian spasi sebagai pemisah antarkata, sehingga harus digunakan kamus untuk bisa mengenal potongan kata aksara Bali.

Algoritma *word boundary* dilakukan dengan pemisahan sekuens-sekuens karakter aksara Bali ke dalam kelompok karakter dasar (termasuk konsonan dan kelompok konsonan). Algoritma ini menggunakan struktur data *stack* untuk mengambil kelompok karakter dasar yang *valid* dan kemudian diproses. Validitas suatu potongan sekuens-sekuens karakter dapat diketahui melalui perbandingan antara sekuens-sekuens tersebut dengan data yang tersimpan di dalam kamus. Terdapat tiga kemungkinan *output* yang dihasilkan dari algoritma *word boundary*, yaitu:

1. Sekuens-sekuens karakter telah dipisah-pisah sampai selesai dan benar (kasus terbaik).
2. Sekuens-sekuens karakter telah dipisah-pisah, tetapi tidak sampai selesai karena ada kemungkinan *input* tidak *valid* atau data di dalam kamus tidak lengkap. Akan tetapi, algoritma akan tetap mengeluarkan *word boundary* terbaik yang didapatkan berdasarkan prinsip statistik (percobaan yang menghasilkan maksimum kata askara Bali).
3. Tidak ada sekuens-sekuens karakter yang dihasilkan (kasus terburuk).

Contoh penggunaan word boundary aksara Bali adalah  ‘pakraman paksa’ [(U+1B27,U+1B13,U+1B44,U+1B2D,U+1B2B,U+1B26,U+1B44)(U+1B27,U+1B13,U+1B44,U+1B32)].

Pada algoritma *word boundary*, fungsi yang paling dominan adalah fungsi *searching* kata dalam I/O *file* sesuai algoritma perbandingan aksara Bali (algoritma 16-1). Kompleksitas fungsi ini adalah sama dengan algoritma *searching* aksara Bali, yaitu $O(n^2)$. Untuk iterasi terluar, *word boundary* memecah kalimat menjadi kata-kata pembentuknya. Misalkan *input* kalimat *s* dengan *m* = *length*(*s*), lakukan iterasi sebanyak *m*/*k* kata dengan *k* = *m* pada kasus terbaik dan *k* = 1 pada kasus terburuk. Oleh karena itu, kompleksitas algoritma *word boundary* adalah $O(m*n^2)$.

```
   Input   : sekuens karakter aksara
Bali
   Output : stack of sekuens karakter
aksara Bali
1. Lakukan validasi blok Unicode
   karakter aksara Bali.
2. Pecah sekuens karakter ke dalam
   stack (push) berdasarkan base
   character dan lakukan validasi pada
   kamus.
3. Lakukan algoritma 15-1.
4. Jika tidak valid/sama maka pop item
   paling atas stack yang
   bersangkutan.
5. Output yang dihasilkan memiliki 3
   kemungkinan sebagaimana telah
   dijelaskan di atas.
```

Algoritma 16-1. Word boundary karakter Aksara Bali

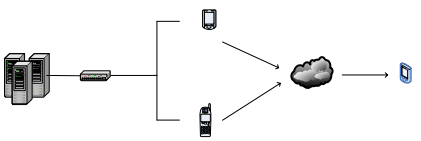
17. *Deskripsi Umum Perangkat Lunak*

Aplikasi yang dibangun berkaitan dengan pemrosesan teks berbasiskan standar *Unicode* aksara Bali adalah B-Linguist, B-Unicode, dan aplikasi *desktop*. Ketiga aplikasi ini membutuhkan JVM (*Java Virtual Machine*) dengan spesifikasi sebagai berikut. B-Linguist dan B-Unicode dibangun di atas J2ME WTK2, sedangkan aplikasi *desktop* dibangun di atas J2SDK 1.5.

B-Linguist merupakan aplikasi *mobile dictionary* yang memiliki fungsi sebagai *translator* dari bahasa Bali ke Indonesia dan Inggris. B-Unicode merupakan aplikasi *mobile* pemrosesan teks aksara Bali sesuai standar *Unicode*. Aplikasi ini mencakup fungsi *transliteration*, *searching*, *sorting*, dan *word boundary*. Aplikasi *desktop* berfungsi sebagai *dictionary* dan *transliteration generator* yang dibutuhkan oleh B-Linguist dan B-Unicode. Khusus untuk aplikasi *desktop*, algoritma *quick sort* digunakan untuk membangun file indeks dan struktur data kamus yang kemudian dipakai baik pada aplikasi *mobile* maupun aplikasi *desktop*.

Keseluruhan aplikasi ini menggunakan *library* tunggal *balinese*. *Library balinese* memiliki karakteristik umum, sehingga dapat berjalan pada *multiplatform*, baik pada J2SDK maupun J2ME.

Sebagai tambahan, fungsi-fungsi pada aplikasi B-Unicode juga diimplementasikan pada aplikasi *desktop* melalui *drivers* sederhana untuk menguji algoritma dan *library* yang dibuat. *Driver* ini dijalankan dalam mode TUI (*text user interface*).



Gambar 17-1. Arsitektur Sistem Perangkat Lunak

Dari gambar 17-1 dapat disimpulkan bahwa perangkat lunak ini adalah sebuah aplikasi yang berbasis komputer, yang melibatkan jaringan (*network*) *mobile device*. Perangkat keras yang digunakan dalam tugas akhir ini adalah :

1. HP Sony Ericsson, dalam hal ini adalah HP Sony Ericsson K750i beserta SIM *card* yang masih aktif. HP ini berfungsi sebagai alat yang menjalankan aplikasi B-Linguist dan B-Unicode dan memberikan umpan balik ke pihak pengembang melalui SMS.
2. Sebuah komputer sebagai alat yang menjalankan aplikasi *desktop* untuk menghasilkan *files* yang dibutuhkan oleh aplikasi *mobile* B-Linguist dan B-Unicode.

3. Kabel data atau koneksi *bluetooth* sebagai penghubung antara HP dengan komputer.
4. Pocket PC berfungsi sebagai alat yang menerima SMS dari *user* aplikasi *mobile*.

18. *Pengujian*

Pengujian sistem dilakukan pada komputer PC dengan spesifikasi sebagai berikut:

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

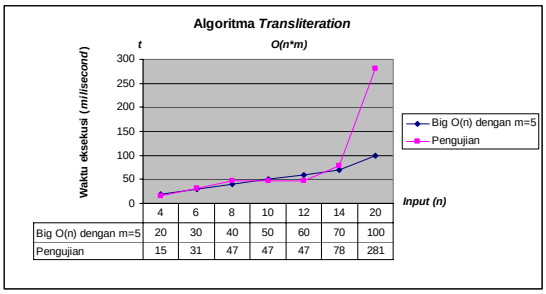
Pengujian sistem dibagi atas 2 bagian yaitu pengujian perangkat lunak (tabel 18-1) dan pegujian *input keyboard* (tabel 18-2).

Tabel 18-1. Pengujian perangkat lunak

No.	Nama	Deskripsi
	B-Linguist	
1	Bali ke Inggris	Berhasil dan lulus verifikasi
2	Bali ke Indonesia	Berhasil dan lulus verifikasi
3	Indeks Bali ke Inggris	Berhasil dan lulus verifikasi
4	Indeks Bali ke Indonesia	Berhasil dan lulus verifikasi
5	Ganti Bahasa	Berhasil dan lulus verifikasi
6	Kirim SMS	Berhasil dan lulus verifikasi
	B-Unicode	
7	*Transliteration* Bali ke Latin	Berhasil dan lulus verifikasi
8	*Transliteration* Latin ke Bali	Berhasil dan lulus verifikasi
9	*Searching*	Berhasil dan lulus verifikasi
10	*Sorting* HANACARAKA	Berhasil dan lulus verifikasi
11	*Sorting* SANSKRIT	Berhasil dan lulus verifikasi
12	*Word Boundary*	Berhasil dan lulus verifikasi
13	Ganti Bahasa	Berhasil dan lulus verifikasi
14	Kirim SMS	Berhasil dan lulus verifikasi
	Aplikasi *Desktop*	
15	*Transliteration Generator* Bali ke Latin	Berhasil dan lulus verifikasi
16	*Transliteration Generator* Latin ke Bali	Berhasil dan lulus verifikasi
17	*Unicode Code Generator*	Berhasil dan lulus verifikasi
18	HANACARAKA *Generator*	Berhasil dan lulus verifikasi
19	SANSKRIT *Generator*	Berhasil dan lulus verifikasi

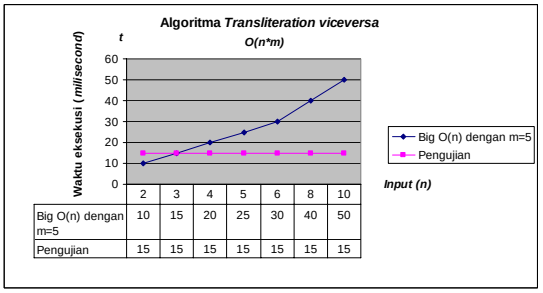
Tabel 18-2. Pengujian input keyboard perangkat lunak

No.	*Input keyboard*	Deskripsi
1	Kosong	Berhasil dan lulus verifikasi
2	Hanya berisi konsonan saja	Berhasil dan lulus verifikasi
3	Berisi konsonan dan vokal	Berhasil dan lulus verifikasi
4	Berisi konsonan *cluster* dan vokal	Berhasil dan lulus verifikasi
5	Berisi kemungkinan kasus khusus (contoh dalam *searching*)	Berhasil dan lulus verifikasi



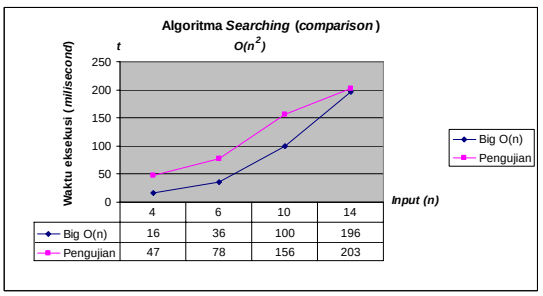
	4	6	8	10	12	14	20
Big O(n) dengan m=5	20	30	40	50	60	70	100
Pengujian	15	31	47	47	47	78	281

Gambar 18-1. Pengujian Big O(n) algoritma transliteration



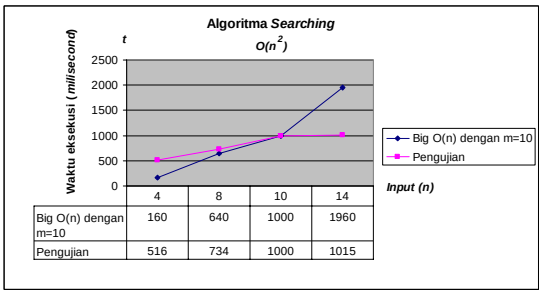
	2	3	4	5	6	8	10
Big O(n) dengan m=5	10	15	20	25	30	40	50
Pengujian	15	15	15	15	15	15	15

Gambar 18-2. Pengujian Big O(n) algoritma transliteration viceversa



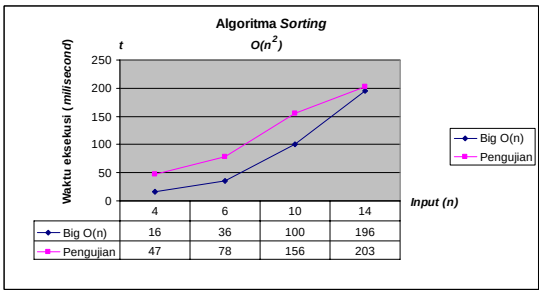
	4	6	10	14
Big O(n)	16	36	100	196
Pengujian	47	78	156	203

Gambar 18-3. Pengujian Big O(n) algoritma searching (comparison)



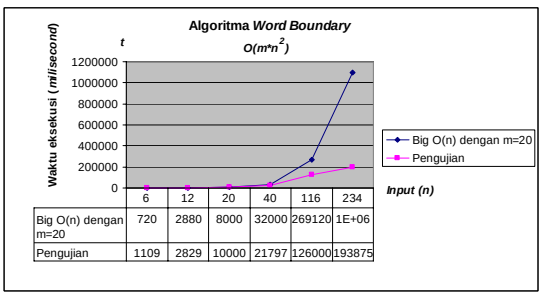
	4	8	10	14
Big O(n) dengan m=10	160	640	1000	1960
Pengujian	516	734	1000	1015

Gambar 18-4. Pengujian Big O(n) algoritma searching



	4	6	10	14
Big O(n)	16	36	100	196
Pengujian	47	78	156	203

Gambar 18-5. Pengujian Big O(n) algoritma sorting



	6	12	20	40	116	234
Big O(n) dengan m=20	720	2880	8000	32000	269120	1E+06
Pengujian	1109	2829	10000	21797	126000	193875

Gambar 18-6. Pengujian Big O(n) algoritma word boundary

19. Implementasi Aplikasi

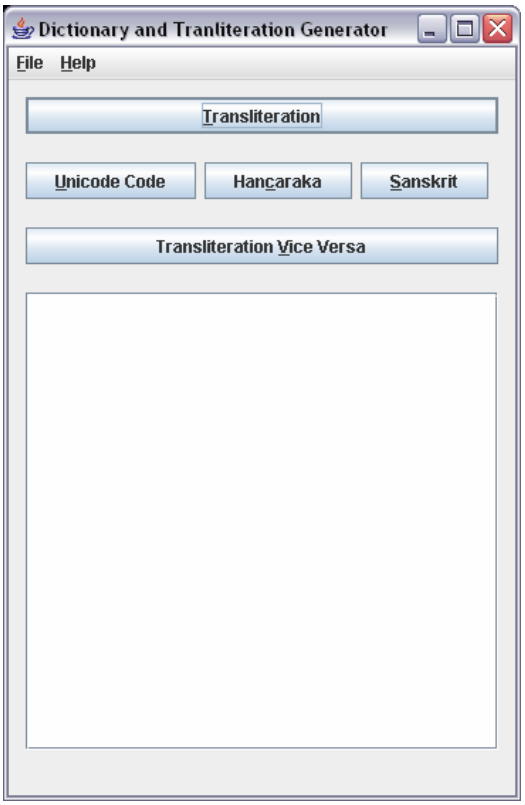
Hasil dari implementasi perancangan antarmuka dapat dilihat pada gambar 19-1, 19-2, dan 19-3.

Gambar 19-1. Tampilan aplikasi B-Linguist

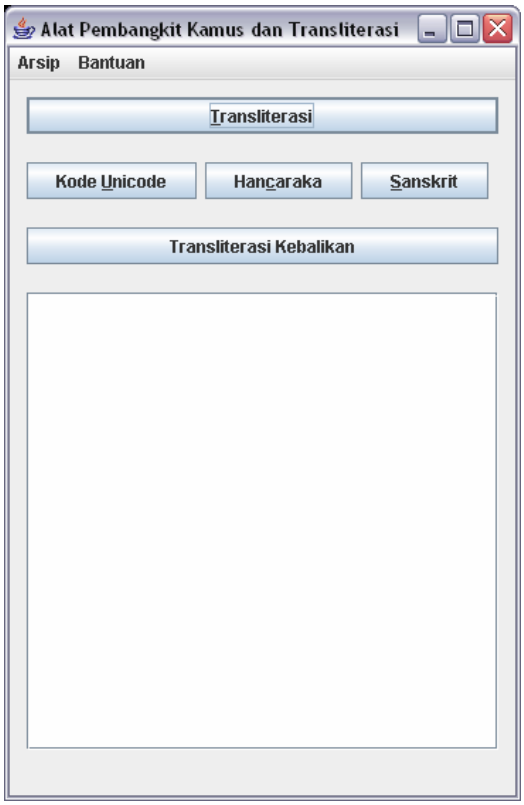
Gambar 19-2. Tampilan aplikasi B-Unicode

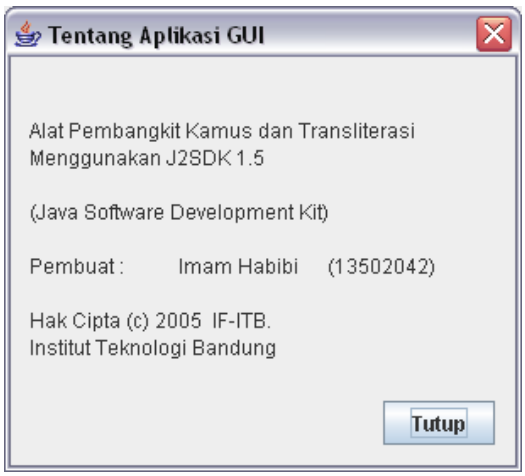
Gambar 19-3. Tampilan aplikasi B-Linguist

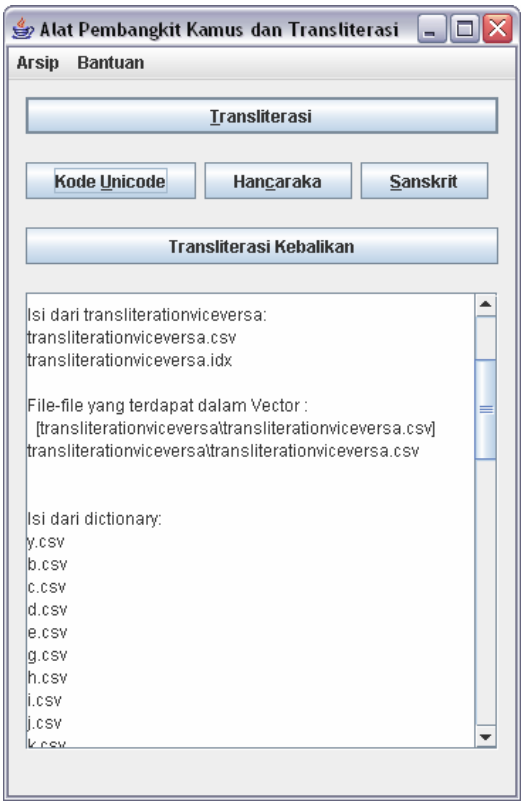
Gambar 19-2. Tampilan aplikasi B-Unicode (lanjutan)



Gambar 19-3. Tampilan aplikasi Desktop







Gambar 19-3. Tampilan aplikasi Desktop (lanjutan)

20. Kesimpulan dan Saran

Berdasarkan hasil analisis dan kajian yang telah dilakukan, beberapa kesimpulan dapat diambil dari pembangunan sistem ini, yaitu:

1. Sistem ini telah berhasil melakukan pemrosesan teks aksara Bali yang meliputi fungsi *transliteration*, *searching*, *sorting*, dan *word boundary*. Terlebih lagi, pembuatan *library* yang bersifat umum dan *multiplatform* telah berhasil dilakukan. *Library balinese* ini dapat berjalan baik pada komputer maupun *handheld device* yang

memiliki JVM (*java virtual machine*).
2. Kompleksitas algoritma *transliteration*, *searching*, *sorting*, dan *word boundary* secara berurutan adalah O(n*m), O(n^2), O(n^2), O(m*n^2) dengan n = *length*(*input string*) dan m = jumlah perbandingan pada *lookup table* secara rata-rata.
3. *Unicode* telah didukung oleh berbagai sistem operasi dan *browser*. Tren teknologi perangkat lunak global saat ini adalah pemakaian standar *Unicode* dan ketersediaan perangkat yang mendukungnya.
4. Untuk membuat suatu perangkat lunak yang mampu mengolah kata yang mengandung huruf Bali diperlukan kerja sama antar dua bidang ilmu, yaitu ilmu yang menyangkut tata tulis huruf Bali dan ilmu informatika. Tanpa adanya penguasaan terhadap bidang ilmu tersebut, perangkat lunak yang dihasilkan tidak akan mencakup seluruh kemungkinan yang ada.
5. Dalam analisa kebutuhan perangkat lunak, komunikasi dengan calon pemakai harus sering dilakukan sehingga perangkat lunak yang dihasilkan mencerminkan kebutuhan dari pemakai.

Di samping kesimpulan di atas, terdapat beberapa saran yang perlu dipertimbangkan lebih lanjut, yaitu:

1. Untuk pengembangan perangkat lunak agar mampu menangani setiap kasus dalam masalah pemrosesan teks dengan huruf Bali dibutuhkan seorang ahli dalam tata tulis huruf Bali yang akan membantu proses pengembangan perangkat lunak dalam hal :
   a. pengelompokan jenis huruf.
   b. pengenalan sifat-sifat huruf Bali.
   c. menganalisa kombinasi huruf.
   d. mengevaluasi kebenaran jalannya proses pada perangkat lunak.
2. Komunikasi dengan calon pemakai bisa dilakukan dengan wawancara atau penyebaran kuisioner untuk mengetahui dengan lebih baik hal-hal seperti :
   a. kesesuaian *layout keyboard* untuk huruf Bali.
   b. cara berkomunikasi dengan perangkat lunak termasuk bahasa yang dipergunakan dalam berinteraksi.
3. Kartu SIM yang dipakai untuk fungsi *update* kamus sebaiknya adalah kartu jenis pascabayar agar tidak perlu dilakukan kontrol isi pulsa.

21. Daftar Pustaka

[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).