Teksty wielojęzyczne w edytorze GNU Emacs

Janusz S. Bień
Zakład Zastosowań Informatycznych
Instytut Orientalistyczny Uniwersytetu Warszawskiego
jsbien@mail.uw.edu.pl

Streszczenie

Artykuł przedstawia wielojęzyczne możliwości edytora GNU Emacs (w wersji 20 i 21) na przykładzie tekstów polsko-japońskich. W zwięzłej formie podaje on informacje niedostępne lub rozproszone w oryginalnej dokumentacji, np. tabele konwersji *romaji-kana*. Oprócz wprowadzania tekstów omówione są również operacje wyszukiwania przyrostowego i zastępowania wykorzystujące tzw. metody wejściowe do wprowadzenia znaków języków orientalnych.

Wprowadzenie

Edytor GNU Emacs jest przykładem oprogrowamowania swobodnego, które jest nie tylko bezpłatne, ale może być również rozpowszechniane — w formie oryginalnej lub ze zmianami — pod mało kłopotliwymi warunkami.

Edytor ten jest dostępny na wielu platformach, w szczególności dla systemu operacyjnego Linux, MS Windows, a w ograniczonej wersji również dla DOS. Począwszy od wersji 20 (z 15 września 1997 r.) edytor GNU Emacs (`http://pl.gnu.org/software/emacs/emacs.html`) może pracować w tzw. trybie wielobajtowym, pozwalającym bez trudu łączyć w jednym tekście różne alfabety i systemy pisma ([3]); możliwości te są obecnie dostępne również w wersji edytora dla MS Windows 95 i nowszych. Stało się to możliwe dzięki zintegrowaniu z edytorem GNU Emacs jego rozszerzenia dostępnego wcześniej jako osobny program MULE (*MULti-lingual Enhancement to GNU Emacs*; obecnie skrót MULE jest rozwijany jako *MULtilingual Environment*). Możliwości edycji tekstów wielojęzycznych demonstruje — o ile są dostępne odpowiednie fonty — komenda `view-hello-file` (`C-h h`)[1].

Możliwości edytora GNU Emacs są bardzo pożyteczne dla filologów i lingwistów oraz osób uczących się języków obcych, zwłaszcza orientalnych. Pracę z tekstami wielojęzycznymi zilustruję przykładami polsko-japońskimi. Wiąże się to z faktem, że kiedyś podjąłem próbę poznania języka japońskiego; wprawdzie zakończyła się ona niepowodzeniem, ale pozostały mi pewne informacje o japońskich systemach pisma.

Chciałbym tutaj skorzystać z okazji, aby serdecznie podziękować Panu Profesorowi Wiesławowi Kotańskiemu, twórcy warszawskiej japonistyki (i kawalerowi Orderu Wschodzącego Słońca nadanego przez cesarza Japonii), za jego dawne próby nauczenia mnie (i kilku moich kolegów z Instytutu Informatyki UW) języka japońskiego. Dziękuję również Michałowi Piskorskiemu za pomoc w interpretacji przykładów japońskich i składzie niniejszego tekstu.

Pismo japońskie

Próbkę tekstu japońskiego można uzyskać wykonując komendę `help-with-tutorial` z argumentem `japanese` (w skrócie `C-u C-h t j TAB`). Jego drugi wiersz ma postać

> あなたが現在見ているのはEmacs 入門ガイドです。

Przez porównanie z angielską lub inną wersją tego tekstu możemy się domyślić, że zdanie to znaczy w przybliżeniu *Patrzysz teraz na tutorial Emacsa*. Jak widać, obce nazwy własne, jak Emacs, są przytaczane w oryginalnym alfabecie łacińskim.

Najbardziej skomplikowane znaki, takie jak 現, 在, 見 czy 門, są pochodzenia chińskiego. Powstały one w czasach dynastii Han, panującej w Chinach przez przeszło cztery wieki od 206.r. p.n.e. ([16], s. 19). Znaki te znalazły zastosowanie nie tylko w języku japońskim, ale także w koreańskim i wietnamskim, przy czym w każdym z tych języków przybrały trochę inne kształty (w języku wietnamskim praktycznie wyszły z użycia, w koreańskim są w zaniku). Zamiast stosować niewygodne określenia

[1] Nazwa Emacs jest żartobliwie rozwiązywana jako *Escape Meta Alt Control Shift*, ponieważ edytor ten intensywnie wykorzystuje wymienione klawisze. Aby wykonać komendę o znanej nazwie, trzeba nacisnąć jednocześnie klawisz `Alt` i `x` (można również najpierw nacisnąć `Escape`, a potem `x`), następnie wpisać jej nazwę i nacisnąć `Enter`; zapisujemy to skrótowo np. `M-x view-hello-file`. Zapis `C-h` oznacza jednoczesne naciśnięcie `Control` i `h`.

opisowe (np. „znaki japońskie pochodzenia chińskiego"), będziemy mówić na nie krótko „znaki hanowskie". Dla pełności obrazu warto dodać, że najbardziej podstawowe z nich mogą mieć całkiem proste kształty, jak np. występujący wyżej znak 入.

Pozostałe znaki należą do dwóch sylabariuszy, nazywanych hiragana i katakana. W przeciwieństwie do znaków hanowskich, reprezentujących pewne pojęcia (i wymawianych w całkowicie odmienny sposób w zależności od kontekstu), znaki hiragany i katakany reprezentują poszczególne sylaby występujące w języku japońskim, chociaż i w tym wypadku wymowa niektórych znaków zmienia się w zależności od kontekstu. Hiragana służy do zapisywania wyrazów i morfemów gramatycznych, np. あなたが, ているの は czy です w cytowanym zdaniu. Katakana służy niemal wyłącznie do zapisywania nazwisk i innych wyrazów zapożyczonych, np. ガイド (patrz niżej).

Pierwsze próby zapisu nazwisk, nazw i tekstów japońskich za pomocą alfabetu łacińskiego pojawiły się wraz z pierwszymi kontaktami europejskich, a później i amerykańskich misjonarzy z Japonią. Obecnie jest w użyciu kilka konwencji tego typu, przy czym największą popularnością — również w Polsce — cieszy się zmodyfikowany system Hepburna, biorący swoją nazwę od autora XIX-wiecznego słownika japońsko-angielskiego[2]. W systemie tym samogłoski wymawiane są tak, jak w łacinie, a spółgłoski w sposób zbliżony do angielskiego. Oto przykłady tej transkrypcji: japońska nazwa systemu Hepburna zapisuje się jako *hebon shiki* (dla Polaka wierniejszym zapisem wymowy byłoby *chebon siki*), znaki hanowskie to *kanji* (wymawiane *kandzi*), zapis języka japońskiego alfabetem łacińskim to *romaji*, hiragana i katakana traktowane łącznie to po prostu *kana*, normalny zapis tekstu japońskiego — tj. z wykorzystaniem znaków hanowskich, hiragany i ewentualnie katakany — to *kanamajiri*. Cytowany wcześniej — zapisany katakaną, a więc zapożyczony — wyraz ガイド w *hebon shiki* przybiera postać *gaido*; z pewnym trudem możemy rozpoznać w nim angielskie słowo *guide*.

Ja wpomniałem wyżej, wymowa znaków kany zależy niekiedy od kontekstu, w szczególności różne znaki kany w odpowiednich kontekstach są wymawiane identycznie. System Hepburna oddaje wtedy ich wymowę, gubiąc informację o wyglądzie oryginalnego napisu japońskiego nawet wtedy, gdy jest on zapisany bez użycia znaków hanowskich; podobną własność mają niestety również inne popularne systemy transkrypcji. W związku z tym do wprowadzania do komputera tekstów w kanie muszą być stosowane inne, w pełni jednoznaczne konwencje. Opracowana przez mnie tabela 1 (patrz s. 12) przedstawia pełny repertuar znaków hiragany i katakany oraz ich reprezentacje łacińskie w edytorze GNU Emacs. Tabela ta wymaga kilku komentarzy.

W języku japońskim występuje rozróżnienie samogłosek długich i krótkich, które w transkrypcji Hepburna oddaje się poziomą kreską nad literą. W hiraganie długość samogłoski w sylabie oznacza się dopisując do niej odpowiednią samogłoskę, w katakanie natomiast stosuje się specjalny znak, mianowicie ー (nie należy go mylić ze znakiem hanowskim 一, oznaczającym m.in. „jeden").

Układ tabeli jest oparty na tradycyjnym japońskim sposobie prezentowania znaków kany nazywanym *gojūon hyō* czyli pięćdziesiąt dźwięków. Jest to tabela o 10 wierszach i 5 kolumnach (lub odwrotnie); krawędź krótsza jest oznaczona 5 samogłoskami, zaś krawędź dłuższa 9 spółgłoskami, od których zaczynają się odpowiednie sylaby, a jedna dodatkowa pozycja odpowiada sylabom zawierającym same samogłoski. Jak to bywa z tradycyjnymi określeniami, nie jest ono w pełni adekwatne do sytuacji, ponieważ nie wszystkie pola tej tabeli są wypełnione, nie mieszczą się też w niej wszystkie znaki kany.

Poza podstawową tabelą jest przede wszystkim znak ん(*n*, w pewnych kontekstach transkrybowany jako *m*), jedyny symbol kany oznaczający pojedynczą spółgłoskę (która może jednak pełnić funkcję sylaby). Jego istnienie powoduje dwuznaczność napisów typu *ni*, które mogą oznaczać sylabę (に, ニ) lub spółgłoskę z samogłoską (んい, ンイ). W edytorze GNU w takich dwuznacznych kontekstach znak *n* wprowadzamy jako `n'`, w pozostałych po prostu jako `n`.

Znaki kany nie uwzględnione w tabeli *gojūon hyō* to przede wszystkim znaki traktowane jako warianty znaków podstawowych, utworzone za pomocą znaków diakrytycznych. Diakryty są dwóch rodzajów: *dakuten* czyli *nigori* w postaci dwóch ukośnych kresek w prawym górnym rogu znaku (゛, por. て i で), oraz *handakuten* czyli *maru* w formie kółeczka (゜, por. へ i ぺ). Niektóre znaki występują również w formie zmniejszonej — służą one do modyfikowania w pewien sposób wymowy poprzedzającej sylaby. Repertuar znaków katakany jest nieco większy, dla umożliwienia zapisywania sylab nie występujących w rodzimych słowach japońskich.

Podstawowym celem tabeli 1 jest pokazanie pełnego repertuaru znaków kany. W związku z tym nie uwzględnia ona tych dodatkowych możliwości ich wprowadzania, które są przedstawione osobno w tabeli 2 (patrz s. 13) — np. sekwencję きゃ wygodniej

[2] Por. `http://www.kufs.ac.jp/toshokan/50/wei.htm`.

jest wprowadzić pisząc `kya` (tak zresztą jest zapisywana w transkrypcji Hepburna) zamiast `kixya`, jak wynikałoby z samej tabeli 1. Choć układ tej tabeli jest również inspirowany przez *gojūon hyō*, w niektórych przypadkach trzeba było dokonać rozstrzygnięć czysto arbitralnych.

Kodowe zestawy znaków

Kiedy po wprowadzeniu w opisany niżej sposób tekstu polsko–japońskiego spróbujemy go zachować (komendą `C-x C-s`), Emacs zaproponuje nam przytłaczającą listę możliwych sposobów zakodowania tego tekstu w pliku (łącznie z tak zaskakującymi, jak `hebrew-iso-8bit-with-esc`). Jeśli jedynym celem wprowadzenia tekstu jest jego wydrukowanie, wtedy najlepszym rozwiązaniem jest zapisanie pliku w sposób specyficzny dla edytora Emacs, tj. w formacie `emacs-mule`.

Niezawodną metodą drukowania tekstów wielojęzycznych w edytorze Emacs jest utworzenie pliku w formacie PostScript, zachowanego na dysku (np. `C-u M-x ps-print-buffer`) lub przesłanego bezpośrednio do drukarki (np. `M-x ps-print-buffer`). Znacznie lepszą jakość wydruku można uzyskać stosując LaTeX 2$_\varepsilon$ z pakietem CJK TeX (patrz [17]) dostępnym m.in. na płycie TeX Live — ten właśnie sposób został wykorzystany do przygotowania niniejszego artykułu.

Jeśli przeznaczeniem tekstu jest jego import do jakiejś aplikacji japońskojęzycznej, wtedy właściwy może być wybór jednego z japońskich kodów znaków lub jednej z metod kodowania opartych na normie ISO/IEC 2022 [4], posiadającej również polski odpowiednik [19]; norma ta jest w praktyce równoważna dostępnej bezpłatnie normie ECMA-35 [2]. Nie będziemy tutaj wnikać w subtelne różnice między kodem a metodą kodowania, odsyłając zainteresowanych do książki [18]. Bliższe informacje o konkretnych sposobach kodowania dostępnych w edytorze możemy uzyskać komendą `C-h C` — chodzi o jednoczesne naciśnięcie klawiszy `Control` i `h`, a następnie `Shift` i `c` (można również użyć `M-x describe-coding-system`). Podobnie informacje o dostępnych kodach znaków możemy uzyskać komendą `M-x describe-character-set`; w wersji 21 edytora możemy dodatkowo otrzymać tabelę kodu za pomocą komendy `M-x list-charset-chars`.

Stosowane w edytorze Emacs zestawy znaków (*charset*) nie zawsze są w pełni zgodne z oficjalnymi standardami. Warto w związku z tym wiedzieć, że wykaz i tabele najważniejszych kodów są także dostępne w rejestrze kodowych zestawów znaków i odpowiadających im tzw. sekwencji rozszerzających, prowadzonym na zlecenie ISO[3] i IEC[4] pod adresem `http://www.itscj.ipsj.or.jp/ISO-IR/`.

Obecnie do celów wymiany informacji najlepiej służy obszerny zestaw znaków UNICODE. Jego pełna specyfikacja wraz z różnymi dodatkowymi materiałami informacyjnymi jest dostępna bezpłatnie na stronach konsorcjum UNICODE (`www.unicode.org`). UNICODE jest w dużym stopniu zgodny z Uniwersalnym Zestawem Znaków (*Universal Character Code*) zdefiniowanym w normie ISO/IEC 10646 (drugie wydanie ukazało się w r. 2000). Zarówno UNICODE jak i ISO/IEC 10646 przewidują kilka alternatywnych reprezentacji tekstów nazywanych formatami transformacyjnymi (*UTF — Unicode/UCS Transformation Format*), jak np. UTF-8. Sposób i zakres korzystania z Uniwersalnego Zestawu Znaków w edytorze GNU Emacs zależy w istotny sposób od jego wersji — jest wskazane, a dla wersji 20 niezbędne, użycie pakietu *Mule-UCS* (`ftp://ftp.m17n.org/pub/mule/Mule-UCS/`). Niekiedy celowe może być także użycie dodatkowego pakietu *Emacs-UTF* (`ftp://ftp.cs.ust.hk/pub/ipe/oc-unicode-0.72.2.tar.gz`)

Wyboru sposobu kodowania tekstu warto dokonać jak najwcześniej, wykonując komendę `M-x set-buffer-file-coding-system` (`C-x RET f`); w linii statusu bufora wyświetli się wtedy odpowiedni symbol (np. `=` dla `emacs-mule`). Warto tę decyzję utrwalić w pliku, wpisując na jego końcu tzw. blok zmiennych lokalnych z odpowiednim podstawieniem na pseudo-zmienną `coding` (istnieje alternatywna możliwość zapisania tej informacji w pierwszym wierszu pliku, ale w ogólnym przypadku jest ona mniej wygodna). Jeśli plik jest przeznaczony do składu za pomocą systemu LaTeX 2$_\varepsilon$ z pakietem CJK, blok zmiennych lokalnych może mieć następującą postać:

```
%%% Local Variables:
%%% mode: latex
%%% coding: emacs-mule
%%% End:
```

Postać taka jest jednak właściwa tylko wtedy, jeśli korzystamy z preprocesora — dostępnego w pliku `cjk-enc.el` wchodzącym w skład pakietu CJK — wywoływanego komendą `M-x cjk-write-file` (lub `M-x cjk-write-all-files`), który tworzy pliki o rozszerzeniu `*.cjk`, przeznaczone do bezpośredniego przetworzenia przez LaTeX2e.

[3] International Organization for Standarization.
[4] International Electrotechnical Committee.

Wprowadzanie tekstów polskich

W przypadku tekstów wielojęzycznych wprowadzanie tekstów polskich wygląda nieco inaczej, niż w przypadku tekstów czysto polskich. W tej drugiej sytuacji racjonalną metodą jest wybranie już na samym początku odpowiedniego środowiska językowego (`M-x set-language-environment`), co w dobrze skonfigurowanym systemie powinno spowodować, że znaki wprowadzane z klawiatury będą interpretowane zgodnie z oczekiwaniami użytkownika. W przypadku tekstów wielojęzycznych podstawowym środowiskiem językowym może być środowisko obcojęzyczne, sprawę klawiatury należy więc potraktować indywidualnie, wykonując komendę `M-x set-keyboard-coding-system` z odpowiednim argumentem: dla spolonizowanego systemu Unix lub GNU/Linux będzie to `iso-latin-2`, dla systemu MS Windows z włączoną obsługą polskiej klawiatury — `cp1250`.

Jeśli z dowolnych powodów wprowadzanie polskich znaków bezpośrednio z klawiatury sprawia jakieś problemy, możemy zawsze posłużyć się jedną z *metod wejściowych* dostępnych w edytorze Emacs, które wybieramy komendą `M-x set-input-method` (`C-x RET C-\`). Do przełaczania się między kilkoma metodami wejściowymi lepiej użyć `C-u C-\` (`M-x toggle-input-method`), która wprawdzie też wymaga argumentu, ale z reguły jego wartość domyślna jest akceptowalna.

Do sporadycznego użytku najwygodniejsza wydaje się metoda `latin-2-postfix` (symbol `2<` w linii statusu bufora), ponieważ po wprowadzeniu znaku, który może być opatrzony diakrytem, ukazuje nam się wiersz pomocy z diakrytami do wyboru; są one oznaczane dość umownie, bo ogonek wprowadzamy znakiem przecinka, a kreskę dla *ł* — znakiem ukośnika. W tej metodzie kłopotliwe jest wprowadzanie prawdziwego przecinka po słowach kończących się na *a* lub *e* — trzeba albo wprowadzić np. spację i ją skasować, ale włączyć i wyłączyć metodę wejściową (`C-u C-\`). Wady tej nie ma metoda `latin-2-alt-postfix`, dostępna jednak dopiero w wersji 21.

Do systematycznego użytku lepiej nadaje się metoda `latin-2-prefix` (symbol `2>` w linii statusu bufora). Tutaj trzeba z góry wiedzieć jak oznacza się diakryty. Znak `'` zgodnie ze swoją genezą i wyglądem na typowej klawiaturze oznacza kreskę nad *c*, *n*, *o*, *s* i *z*. Znak `` ` `` służy zbiorczo do wprowadzania pozostałych diakrytów: ogonków dla *ą* i *ę*, kreski dla *ł*, kropki dla *ż*.

Jest również możliwe stosowanie metody wejściowej realizującej „reprezentację ciachową" polskich liter, w której wszystkie polskie diakryty reprezentowane są przez ukośnik przed odpowiednią literą (np. *żółć* — `/z/o/l/c`). Pierwszeństwo ma tutaj Juliusz Chroboczek, który pod koniec 1999 r. zaanonsował za pomocą wiadomości sieciowych (w grupie `pl.comp.ogonki`) dostępność opracowanej przez siebie metody. W wersji 21 edytora dostępna jest analogiczna metoda o nazwie `polish-slash` (symbol `PL>` w linii statusu bufora), autorstwa Włodka Bzyla; jest ona domyślną metodą wejściową w polskim środowisku językowym.

Pełną listę wszystkich metod wejściowych możemy uzyskać komendą `list-input-methods`. Jeśli chcemy sprawdzić dostępnośc konkretnej metody, której nazwę znamy lub się jej domyślamy, lepiej użyć `C-h I` (`M-x describe-input-method`) i wykorzystać technikę tzw. dopełniania (*completion*) wartości argumentu: naciśnięcie od razu klawisza tabulacji daje nam pełną listę możliwości, naciśnięcie klawisza tabulacji po wprowadzeniu jednego lub więcej znaków daje nam listę możliwości zaczynających się w ten właśnie sposób (jeśli w odpowiednim oknie widać tylko fragment listy, przewijamy ją kolejnymi naciśnięciami klawisza tabulacji).

Wprowadzanie tekstów japońskich

Metody wejściowe dla tekstów japońskich są przeznaczone przede wszystkim dla rodzimych użytkowników tego języka, dlatego zakładają one znajomość wymowy wprowadzanego tekstu. Dla osób uczących się dopiero języka japońskiego oznacza to, że wprowadzenie nieznanych im znaków hanowskich wymaga korzystania z tradycyjnego lub elektronicznego słownika, pozwalającego odszukać znak na podstawie jego kształtu — jest to osobny temat, którym nie będziemy się zajmować w niniejszym artykule.

Wrócimy obecnie do naszego przykładu:

あなたが現在見ているのはEmacs 入門ガイドです。

Aby wprowadzić go do komputera, wybieramy metodę wejściową `japanese` (w linii statusu bufora pojawi się jej symbol Aあ; ponieważ metoda ta ma kilka odmian, będzie on ulegał zmianom w trakcie pracy). Następnie — korzystając w razie potrzeby z tabeli 1 — piszemy `anataga`, które jest sylaba po sylabie przekształcone na あなたなが. Napis ten jest podkreślony, co oznacza, że może on podlegać jeszcze dalszej konwersji — my akceptujemy jego obecną postać naciskając klawisz `Enter`, po czym podkreślenie znika i napis あなたなが staje się częścią zawartości bufora.

Następnie — korzystając z wiedzy o wymowie (i ewentualnie znaczeniu) znaków hanowskich — piszemy `genzai`, przekształcone na げんざい, po czym naciskamy spację, w wyniku czego podkreślony napis げんざい zamieni się na napis 原罪 (na zielonym tle, co jest odpowiednikiem podkreślenia), kolejne naciśnięcie spacji zmienia ten napis na 現在. Jednocześnie symbol metody wejściowej w linii statusu zmienia się na 漢, pokazując nam, że znajdujemy się w trybie KKC (*Kana to Kanji Conversion*). Ponieważ napis 現在 jest tym, o co nam chodzi, akceptujemy go klawiszem `Enter` i wracamy do poprzedniego trybu.

Jak zobaczymy niżej, w ogólnym wypadku różnych możliwości konwersji na znaki hanowskie może być więcej, stąd kolejność ich prezentowania jest istotna. Nie jest ona stała, lecz uwzględnia wcześniejsze decyzje użytkownika, zapisane — razem z innymi pożytecznymi informacjami — w pliku `.kkcrc`. Jeśli ten plik istnieje, to opisywany eksperyment może mieć w szczegółach nieco inny przebieg.

Wprowadźmy obecnie `mi` (み) i naciśnijmy spację. Taką wymowę ma aż 17 znaków: 見, 身, 未, 味, 美, 実, 三, 巳, 箕, 魅, 己, 御, 稔, 弥, 靡, 深i 壬. Przeglądanie tych możliwości po jednej za pomocą spacji byłoby niewygodne, dlatego jest możliwe przeglądanie ich wiekszymi porcjami — maksymalnie po 10 znaków. Przejście w ten tryb przeglądania odbywa się automatycznie po naciśnięciu kilku spacji, może być też dokonane jawnie przez naciśnięcie `l` lub `L`: w minibuforze ukaże się wtedy rodzaj menu przedstawionego na ilustracji 1 na s. 8.

Kolejne naciśnięcia `l` lub `L` powodują przewijanie listy propozycji w jedną lub drugą stronę, a z bieżącej listy możemy wybrać znak wpisując odpowiednią cyfrę; cyfra wskazująca na aktualnie wybrany znak jest ujęta w nawiasy kątowe. Po wybraniu interesującego nas znaku (czyli 見) akceptujemy konwersję klawiszem `Enter`.

Możemy również od razu napisać `genzaimi` (げんざいみ). Po naciśnięciu spacji otrzymamy[5] napis 現在み, przy czym 現在 będzie na żółtym tle, a み będzie podkreślone. W takiej sytuacji naciśnięcie `Enter` jest błędem — powoduje ono wprawdzie akceptację znaków hanowskich, ale podkreślona część napisu zostaje zignorowana i trzeba ją wprowadzać na nowo. Właściwą komendą jest tutaj `C-f` czyli `kkc-next-phrase`. Spowoduje ona zatwierdzenie 現在 i przejście do konwersji み.

Efektywność wprowadzania tekstu japońskiego zależy w dużym stopniu od tego, jak dzielimy go na fragmenty poddawane konwersji na znaki hanowskie. Dla napisu `genzai` (げんざい) mamy tylko dwie propozycje: 現在 i 原罪. Dla samego `gen` (げん) mamy natomiast aż 31 propozycji: 現, 限, 言, 元, 原, 源, 減, 玄, 絃, 弦, 諺, 舷, 幻, 厳, 訝, 嫌, 監, 修, 儼, 呟, 巖, 广, 愿, 痃, 眩, 芫, 衒, 軒, 験, 彦 i 顔, zaś dla samego `zai` (ざい) — 11 propozycji: 在, 財, 材, 剤, 罪, 済, 才, 西, 斉, 劑, 濟 i 戝.

Nie oznacza to jednak, że im dłuższy napis, to tym lepszy wynik konwersji. Oto przykład: wprowadzamy `genzaimitei` (げんざいみてい), po naciśnięciu spacji otrzymujemy 現在みてい, przechodzimy do konwersji następnej frazy za pomocą `C-f` i otrzymujemy 現在未定. Nie jest to ten wynik, o który nam nachodzi, naciskamy więc spację, aby zobaczyć następną propozycję, ale zamiast niej otrzymujemy ponownie pierwotny zapis w hiraganie, a po naciśnięciu `l` lub `L` otrzymujemy komunikat `No alternative`. Należy wówczas skrócić frazę podlegającą konwersji za pomocą klawisza `C-i` (komenda `kkc-shorter`, jej przeciwieństwem jest `kkc-longer` dostępna jako `C-o`); otrzymamy wówczas oczekiwane 現在見てい.

Aby wprowadzić w japońskim zdaniu słowo `Emacs` możemy po prostu wyłączyć stosowanie metod wejściowych za pomocą `C-\`, możemy również napisać `qq`[6] w celu włączenia (a później wyłączenia) metody wejściowej `japanese-ascii` (symbol `Aa` w linii statusu bufora). Sekwencję 入門 wprowadzamy bez problemu podając jej wymowę `nyuumon` (*nyūmon*, にゅうもん). Sekwencję w katakanie ガイド wprowadzamy pisząc `gaido`, które jest przekształcone na podkreślony napis w hiraganie がいど; naciśnięcie `K` (komenda `quail-japanese-toggle-kana`) zamienia go na jego odpowiednik w katakanie, który zatwierdzamy np. przez `Enter`. Na zakończenie piszemy `desu.` i zatwierdzamy otrzymaną postać (です。).

Reasumując, zdanie

あなたが現在見ているのは Emacs 入門ガイドです。

wprowadziliśmy pisząc (z dokładnością do spacji i dużych liter)

Anataga genzai miteirunoha Emacs nyuumon gaido desu.

Przykład powyższy zilustrował najważniejsze, ale nie wszystkie aspekty wprowadzania tekstów japońskich. Ze względów objętościowych pomijamy w szczególności kwestię interpunkcji, wprowadzania

[5] W zależności od zawartości pliku `.kkcrc` wynik może być nieco inny.

[6] Jest to komenda `quail-japanese-switch-package`.

<1>見 2 身 3 未 4 味 5 美 6 実 7 三 8 巳 9 箕 0 魅

Figure 1: Menu wyboru znaków hanowskich.

symboli takich jak 【, 】 czy ♂, oraz kwestię stosowania „zwykłych" znaków ASCII (*hankaku*, np. „1", „a") oraz ich „pełnogabarytowych" odpowiedników (*zenkaku*, np. „１", „ａ"). W wersji 21 informacje na ten temat można znaleźć w opisie metody wejściowej `japanese` i pochodnych, w wersji 20 trzeba niestety stosować zasadę *Use the force, read the source* i po pełną informację sięgać do plików źródłowych (`lisp/international/quail.el`, `leim/quail/japanese.el`).

Inne pisma i języki

Ponieważ Emacs, jak każde oprogramowanie swobodne, jest rozwijany przez wolontariuszy, stopień i jakość obsługi poszczególnych systemów pisma i języków zależy od tego, w jakim stopniu były one potrzebne użytkownikom chcącym i będącym w stanie wnieść wkład w rozwój edytora. Najbardziej zaawansowaną formą obsługi pewnego języka w Emacsie jest stworzenie dla niego osobnego środowiska językowego (jak zobaczymy, nie zawsze chodzi tu o język naturalny w ścisłym znaczeniu), najmniej zaawansowaną — obsługa odpowiedniego kodu znaków. Poniżej dokonamy pobieżnego przeglądu.

angielski — środowisko językowe;

arabski — kody znaków. m.in. ISO/IEC 8859-6;

ASCII — środowisko językowe, kod ASCII;

brytyjski — metoda wejściowa;

chiński — 3 środowiska językowe, 24 metody wejściowe, kilka zestawów znaków;

cyrylica — 2 środowiska językowe, 9 metod wejściowych (m.in. ukraińska i białoruska), kod ISO/IEC 8859-5;

czeski — środowisko językowe i 5 metod wejściowych;

dewanagari — środowisko językowe oraz 4 metody wejściowe, kody znaków;

duński — środowisko językowe i 3 metody wejściowe (Emacs 21);

esperanto — 3 metody wejściowe;

etiopski — środowisko językowe, metoda wejściowa, kod znaków (patrz [23]);

fiński — 3 metody wejściowe;

francuski — 5 metod wejściowych;

grecki — środowisko językowe, 4 metody wejściowe, kod znaków ISO 8859-7;

hebrajski — środowisko językowe oraz kod znaków ISO/IEC 8859-8;

hiszpański — środowisko językowe i 4 metody wejściowe (Emacs 21);

IPA (*International Phonetic Alphabet*— środowisko językowe, metoda wejściowa, kod znaków;

irlandzki — metoda wejściowa;

islandzki — 3 metody wejściowe;

ISO/IEC 8859-1 — kod znaków (ECMA-94), p. Latin-1;

ISO/IEC 8859-2 — kod znaków (ECMA-94), p. Latin-2;

ISO/IEC 8859-3 — kod znaków (ECMA-94), p. Latin-3;

ISO/IEC 8859-4 — kod znaków (ECMA-94), p. Latin-4;

ISO/IEC 8859-5 — kod znaków (ECMA-113), p. cyrylica;

ISO/IEC 8859-6 — kod znaków (ECMA-114), p. arabski;

ISO/IEC 8859-7 — kod znaków (ECMA-118), p. grecki;

ISO/IEC 8859-8 — kod znaków (ECMA-121), p. hebrajski;

ISO/IEC 8859-9 — kod znaków (ECMA-128), p. Latin-5;

ISO/IEC 8859-14 — kod znaków, p. Latin-8;

ISO/IEC 8859-15 — kod znaków, p. Latin-9;

japoński — środowisko językowe, 6 metod wejściowych, kody znaków;

kataloński — metoda wejściowa;

koreański — środowisko językowe, 6 metod wejściowych, kod znaków;

laotański — środowisko językowe, 2 metody wejściowe, kod znaków;

Latin-1 — środowisko językowe (Europa zachodnia), 3 metody wejściowe, kod ISO/IEC 8859-1;

Latin-2 — środowisko językowe (Europa wschodnia), 3 metody wejściowe, kod ISO/IEC 8859-2;

Latin-3 — środowisko językowe (Europa południowa), 3 metody wejściowe, kod ISO/IEC 8859-3;

Latin-4 — środowisko językowe (Europa północna), 2 metody wejściowe, kod ISO/IEC 8859-4;

Latin-5 — środowisko językowe (Europa zachodnia i turecki) i 2 metody wejściowe;

Latin-8 — metoda wejściowa (Europa zachodnia i języki celtyckie), kod ISO/IEC 8859-14 (Emacs 21);
Latin-9 — metoda wejściowa (Europa zachodnia, dodatkowe litery francuskie i znak euro), kod ISO/IEC 8859-15 (Emacs 21);
niemiecki — 4 metody wejściowe;
norweski — 3 metody wejściowe;
polski — środowisko językowe i 1 metoda wejściowa (Emacs 21);
portugalski — metoda wejściowa;
rumuński — środowisko językowe i 2 metody wejściowe;
skandynawski — 2 metody wejściowe;
słowacki — środowisko językowe i 4 metody wejściowe;;
słoweński — środowisko językowe i 4 metody wejściowe;
szwedzki — 3 metody wejściowe;
tajski — środowisko językowe, 2 metody wejściowe (patrz [21], [22]);
tybetański — środowisko językowe;
turecki — środowisko językowe, 4 metody wejściowe;
wietnamski — środowisko językowe i metoda wejściowa;
włoski — 3 metody wejściowe;

Niestety, Emacs nie obsługuje obecnie języków pisanych od prawej do lewej, w związku z tym jego przydatność dla języków arabskiego i hebrajskiego ogranicza się w praktyce do wyświetlania poszczególnych znaków.

Operacje edycyjne na tekście wielojęzycznym

Jedną z najsilniejszych operacji edytorskich w edytorze Emacs jest wyszukiwanie przyrostowe, dostępne w dwóch formach: jako wyszukiwanie zwykłe i według wyrażeń regularnych. Pierwszej formie odpowiadają komendy `isearch-forward` (szukanie do przodu) i `isearch-backward` (szukanie do tyłu), wywoływane za pomocą `C-s` lub `C-r`. Drugiej formie odpowiadają komendy `isearch-forward-regexp` i `isearch-backward-regexp`, wywoływane za pomocą `M-C-s`, `M-C-r`. Jeśli w momencie wykonywania tych komend jest aktywna jakaś metoda wejściowa, to jest ona w użyciu również w trakcie wprowadzania szukanego napisu lub wyrażenia regularnego. Jeśli żadna metoda wejściowa nie jest aktywna, to domyślną metodę można włączyć, jak zwykle, za pomocą `C-\`; dowolną metodę można włączyć za pomocą `C-^` (w kontekście wyszukiwania przyrostowego jest to więc odpowiednik `C-u C-\`).

W trakcie wyszukiwania wyrażeń regularnych można odwoływać się do tzw. kategorii znaków[7]; służy do tego konstrukcja `\c`C, gdzie C oznacza symbol kategorii, np. `a` dla znaków ASCII, `l` dla znaków pisma łacińskiego. Inne kategorie nie są już tak intuicyjne, np. `j` oznacza dowolny znak pochodący z jakiegoś japońskiego zestawu znaków, które może być również znakiem łacińskim, greckim lub cyrylickim. Tym szczegółowym rozróżnieniom odpowiadają inne kategorie, a mianowicie: `k` i `K` (różne sposoby kodowania katakany), `r` i `A` (różne sposoby kodowania łacińskich znaków alfanumerycznych), `H` (hiragana), `C` (*Chinese* czyli znaki hanowskie), `G` (litery greckie), `Y` (litery cyrylickie). Ich znaczenie najlepiej sprawdzić doświadczalnie, np. na tabelach kodów otrzymanych (w wersji 21) za pomocą `list-charset-chars`.

Bezpośrednie wykorzystanie skomplikowanych metod wejściowych w trakcie wprowadzenia napisu lub wyrażenia do wyszukania wymaga pewnej wprawy. Warto z tego względu pamiętać, że ostatnio umieszczony w pierścieniu schowków (*kill-ring*) napis — również wielojęzyczny — można wkleić jako część wyszukiwanego napisu za pomocą `M-y`. Warto pamiętać, że ten skrót klawiaturowy oznacza tutaj komendę `isearch-yank-kill`, której nie należy mylić ze zwykła komendą `yank`, działającą nieco inaczej — kolejne użycia `yank` zamieniają wstawiony napis na inny, pobrany z następnego elementu pierścienia schowków, zaś kolejne użycia `isearch-yank-kill` ponownie wstawiają ten sam napis.

Inną często stosowaną komendą jest komenda zastępowania, dostępna w edytorze Emacs w kilku odmianach: napis zastępowany może być wskazany bezpośrednio lub za pomocą wyrażenia regularnego, zastępowanie może być bezwarunkowe lub wymagać potwierdzenia. Komendy te wczytują specyfikację napisu zastępowanego i zastępującego z tzw. minibufora. W typowej konfiguracji edytora Emacs w każdym momencie dostępny jest tylko jeden minibufor, co w przypadku tekstów wielojęzycznych okazuje się istotnym ograniczeniem, ponieważ minibufor jest wykorzystywany również przez metody wejściowe. W wersji 20 musimy usunąć to ograniczenie umieszczając w pliku konfiguracyjnym instrukcję

```
(setq enable-recursive-minibuffers t)
```

W wersji 21 zmiana domyślnej wartości tej zmiennej nie jest niezbędna, ale osobiście uważam ją i tak za wskazaną.

[7] Kategorie te są zdefiniowane i skomentowane w pliku `lisp/international/characters.el`.

Aby zatem wszystkie zastąpienia np. frazy あなたが現在 zastąpić innym napisem, trzeba wykonać następujące kroki:

1. wybrać japońską metodę wejściową i wywołać odpowiednią komendę, np. `query-replace` (`M-%`) — kolejność tych dwóch operacji jest obojętna,
2. po pojawieniu się ponaglenia postaci `Query replace` wprowadzić `anataga`, co zostanie przekształcone na あなたが,
3. zatwierdzić wynik konwersji za pomocą `Enter` — znajdujemy się wówczas w minibuforze metody wejściowej,
4. wprowadzić `genzai`, co zostanie przekształcone na げんざい,
5. nacisnąć — odpowiednią liczbę razy[8] — spację w celu przekształcenia げんざい na znaki hanowskie 現在,
6. zatwierdzić wynik konwersji za pomocą `Enter` — znajdujemy się wówczas w minibuforze metody wejściowej,
7. nacisnąć Enter w celu zakończenia wprowadzania napisu zastępowanego w minibuforze komendy,
8. w analogiczny wprowadzić napis zastępujący.

Warto wspomnieć, że niezmiernie pożyteczne funkcje Emacsa dotyczące porównywania plików i buforów zawarte w pakiecie ediff (dostępnym np. z menu `Tools` — `Compare`) działają również na tekstach wielojęzycznych; w trakcie porównywania Emacs okresowo zadaje pytania o system kodowania — wystarczy akceptować wartośc domyślną.

W Emacsie są również dostępne — choć bardzo nieliczne — komendy specyficzne dla konkretnych języków. Ich przykładem jest komenda `M-x japanese-hiragana-region`, która w bieżącym regionie zamienia wszystkie znaki katakany na odpowiadające im znaki hiragany. Inne komendy tego typu można znaleźć w standardowy sposób, korzystając z jednej z wersji komendy `apropos`, np. `C-h a japanese`.

Konfiguracja edytora Emacs

Jedno z powiedzeń na temat edytora Emacs brzmi *You don't have to like Emacs to like it* — za pomocą pliku konfiguracyjnego i własnych rozszerzeń można tak dalece zmienić zachowanie edytora, że w niewielkim stopniu będzie przypominał oryginał. W konsekwencji wszystkie stwierdzenia niniejszego artykułu odnoszą się do edytora skonfigurowanego w określony sposób. Odpowiedni plik konfiguracyjny został opracowany przeze mnie na potrzeby płyty CD zatytułowanej *Wybrane narzędzia przetwarzania tekstów wielojęzycznych* (w skrócie WNPTW). Dwie edycje tej płyty — 1999 (wersja 0.9) i 2000 (wersja 0.91) — zostały już zrealizowane w ramach badań własnych Instytutu Orientalistycznego UW; na konferencji BachoTEX 2001 zostanie zademonstrowana wersja 0.92B.

Płyta WNPTW ma za zadanie zademonstrować wielojęzyczne możliwości edytora GNU Emacs użytkownikom MS Windows 95, NT i wersji nowszych — po włożeniu płyty do napędu uruchamia się gotowy do użytku edytor. Płyta ma charakter „nieinwazyjny" — jej wykorzystanie nie zmienia w żaden sposób konfiguracji komputera. Z drugiej strony zachowanie się edytora jest niezależne od konfiguracji konkretnego komputera — aby to osiągnąć, edytor korzysta wyłącznie z fontów BDF zapisanych na płycie, a także z zainstalowanego na płycie programu Ghostscipt.

Oprócz wersji edytora dla MS Windows na płycie znajduje się również pełna dystrybucja edytora i wybranych programów pomocniczych dla systemów Linux i DOS oraz inne pożyteczne materiały.

Płyta może być swobodnie kopiowana i rozpowszechniana[9]; aby to ułatwić, na płycie znajdują się również gotowe do wydrukowania wkładki do kasetki jubilerskiej (czyli, mówiąc bardziej potocznie, pudełka na CD ROM).

Zakończenie

Edytor GNU Emacs jest niewątpliwie bardzo pożytecznym narzędziem do edycji tekstów wielojęzycznych. Jego słabą stroną jest bardzo nierówna jakość dokumentacji — te jego możliwości, które są w nim dostępne od dawna, są opisane dokładnie i szczegółowo, te zaś, które są stosunkowo nowe, są udokumentowane słabo, czasami nawet tylko w plikach źródłowych. Tak np. funkcjonujące już w wersji 20 kategorie znaków do wersji 21 nie były w ogóle wspomniane w podręczniku, podobnie jak możliwość zmiany metody wejściowej w trakcie wyszukiwania przyrostowego czy konieczność (w wersji 20) zmiany domyślnej wartości `enable-recursive-minibuffers`. Tabele konwersji *romaji-kana*, które w wersji 20 były dostępne tylko w plikach źródłowych, w wersji 21 są — moim zdaniem — mało przejrzyste, dlatego uznałem za

[8] Jak pisaliśmy wcześniej, po kilku naciśnięciach spacji metoda wejściowa przechodzi w tryb wyświetlania listy możliwości wybieranych za pomocą cyfr i przewijanych w razie potrzeby komendami `l` i `L`.

[9] Pierwsze edycje płyty zawierają program Ghostscript w wersji Alladin (wersja GNU sprawiała pewne kłopoty techniczne), co nakłada niewielkie ograniczenia.

stosowne umieścić opracowane przez siebie tabele w niniejszym artykule pomimo tego, że większość Czytelników prawdopodobnie nie będzie z nich korzystać.

Mam nadzieję, że niniejszy tekst zachęci nie tylko do korzystania z edytora Emacs, ale również do dokumentowania i popularyzacji jego mniej znanych możliwości, jak np. metody wejściowe dla innych języków orientalnych.

Bibliografia

[1] Janusz S. Bień. Kodowanie tekstów polskich w systemach komputerowych. `ftp://ftp.mimuw.edu.pl/pub/users/polszczyzna/ogonki/katow98.*`. Także (wersja skrócona) *Postscriptum* nr 27-29 (jesień 1998 — wiosna 1999), s. 4-27 (ISSN 1427-0501).

[2] ECMA-35. *Character Code Structure and Extension Techniques*. Sixth Edition (December 1994). `http://www.ecma.ch/STAND/ECMA-035.HTM`.

[3] Kenichi Handa, Mikiko Nishikimi, Satoru Tomura, Naoto Takahashi. Unified and Extensible Mechanism for Multilingual Text Processing. *The Fourth Pacific Rim Conference on Artificial Intelligence* PRICAI '96, 26–30 August 1996, Cairns, North Queensland, Australia. `http://www.m17n.org/mule/pricai96/`.

[4] ISO/IEC 2022:1994. *Information technology — Character code structure and extension techniques*.

[5] ISO/IEC 8859-1:1998. *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1.*

[6] ISO/IEC 8859-2:1999. *Information technology — 8-bit single-byte coded graphic character sets — Part 2: Latin alphabet No. 2.*

[7] ISO/IEC 8859-3:1999. *Information technology — 8-bit single-byte coded graphic character sets — Part 3: Latin alphabet No. 3.*

[8] ISO/IEC 8859-3:19??. *Information technology — 8-bit single-byte coded graphic character sets — Part 4: Latin alphabet No. 4.*

[9] ISO/IEC 8859-5:1999. *Information technology — 8-bit single-byte coded graphic character sets — Part 5: Latin/Cyrillic alphabet.*

[10] ISO/IEC 8859-6:1999. *Information technology — 8-bit single-byte coded graphic character sets — Part 6: Latin/Arabic alphabet.*

[11] ISO 8859-7:1987. *Information technology — 8-bit single-byte coded graphic character sets — Part 7: Latin/Greek alphabet.*

[12] ISO/IEC 8859-8:1999. *Information technology — 8-bit single-byte coded graphic character sets — Part 8: Latin/Hebrew alphabet.*

[13] ISO/IEC 8859-9:1999. *Information technology — 8-bit single-byte coded graphic character sets — Part 9: Latin alphabet No. 5.*

[14] ISO/IEC 8859-14:1998. *Information technology — 8-bit single-byte coded graphic character sets — Part 14: Latin alphabet No. 8.*

[15] ISO/IEC 8859-15:1999. *Information technology — 8-bit single-byte coded graphic character sets — Part 15: Latin alphabet No. 9.*

[16] Mieczysław Jerzy Künstler. *Pismo chińskie*. Państwowe Wydawnictwo Naukowe: Warszawa 1970.

[17] Werner Lemberg. The CJK package for LaTeX 2_ε — Multilingual support beyond babel. *TUGboat* Vol. 18(1997) No. 3 — Proceedings of the 1997 Annual Meeting, pp 214–224

[18] Ken Lunde. *CJKV Information Processing. Chinese, Japanese, Korean & Vietnamese Computing*. O'Reilly: Sebastopol 1999.

[19] PN-ISO/IEC 2022:1996. *Technika informatyczna — Struktura kodu znaków i techniki rozszerzania*. Polski Komitet Normalizacyjny, grudzień 1996.

[20] PrPN-ISO/IEC 8859-2. *Technika informatyczna — Zestawy znaków graficznych w jednobajtowym kodzie 8-bitowym — Alfabet łaciński nr 2*. Polski Komitet Normalizacyjny 2000. Norma zastąpi PN-91/T42115.

[21] Naoto Takahashi, Kenichi Handa, Mikiko Nishimi, Satoru Tomura. Thai handling in Mule/Emacs. Orchid Corpus Technical Report. `http://www.links.nectec.or.th/orchid/pub/Paper3.html`.

[22] Naoto Takahashi. Enhanced Thai handling in Mule-2.3. Orchid Corpus Technical Report. `http://www.links.nectec.or.th/orchid/pub/Paper4.html`.

[23] Naoto Takahashi, Daniel Yacob. Adding Ethiopic Features to Emacs. *First International Conference on Information Technology in Ethiopia and Computational Ethiopics Symposium*, Addis Abeba, 1997. `http://m17n.org/ntakahas/abstracts/mule-paper.ps.gz`.

◇ Janusz S. Bień
`jsbien@mail.uw.edu.pl`

	A	**I**	**U**	**E**	**O**	
	あ ア	い イ	う ウ	え エ	お オ	ん ン
	a	i	u	e	o	n', n
K	か カ	き キ	く ク	け ケ	こ コ	ー
	ka	ki	ku	ke	ko	-
S	さ サ	し シ	す ス	せ セ	そ ソ	
	sa	si	su	se	so	
		shi				
T	た タ	ち チ	つ ツ	て テ	と ト	
	ta	ti	tu	te	to	
		chi	tsu			
N	な ナ	に ニ	ぬ ヌ	ね ネ	の ノ	
	na	ni	nu	ne	no	
H	は ハ	ひ ヒ	ふ フ	へ ヘ	ほ ホ	
	ha	hi	hu	he	ho	
			fu			
M	ま マ	み ミ	む ム	め メ	も モ	
	ma	mi	mu	me	mo	
Y	や ヤ		ゆ ユ		よ ヨ	
	ya		yu		yo	
R	ら ラ	り リ	る ル	れ レ	ろ ロ	
	ra	ri	ru	re	ro	
	la	li	lu	le	lo	
W	わ ワ	ゐ ヰ	(う) (ウ)	ゑ ヱ	を ヲ	
	wa	wi	(wu)	we	wo	
G	が ガ	ぎ ギ	ぐ グ	げ ゲ	ご ゴ	
	ga	gi	gu	ge	go	
Z	ざ ザ	じ ジ	ず ズ	ぜ ゼ	ぞ ゾ	
	za	zi	zu	ze	zo	
		ji				
D	だ ダ	ぢ ヂ	づ ヅ	で デ	ど ド	
	da	di	du	de	do	
B	ば バ	び ビ	ぶ ブ	べ ベ	ぼ ボ	
	ba	bi	bu	be	bo	
P	ぱ パ	ぴ ピ	ぷ プ	ぺ ペ	ぽ ポ	
	pa	pi	pu	pe	po	
V			ヴ			
			vu			
	あ ア	い イ	う ウ	え エ	お オ	
	xa	xi	xu	xe	xo	
k	カ			ケ		
	xka			xke		
t			っ ツ			
			xtu			
y	や ヤ		ゆ ユ		よ ヨ	
	xya		xyu		xyo	
w	わ ワ					
	xwa					

Tabela 1: Podstawowe reguły konwersji *romaji-kana*

	A		I		U		E		O	
Ky	きゃ	キャ			きゅ	キュ	きぇ	キェ	きょ	キョ
	kya				kyu		kye		kyo	
Sy	しゃ	シャ			しゅ	シュ	しぇ	シェ	しょ	ショ
	sya				syu		sye		syo	
	sha				shu		she		sho	
Ty	ちゃ	チャ	てぃ	ティ	ちゅ	チュ	ちぇ	チェ	ちょ	チョ
	tya		tyi		tyu		tye		tyo	
	cha				chu		che		cho	
Ny	にゃ	ニャ			にゅ	ニュ	にぇ	ニェ	にょ	ニョ
	nya				nyu		nye		nyo	
Hy	ひゃ	ヒャ			ひゅ	ヒュ	ひぇ	ヒェ	ひょ	ヒョ
	hya				hyu		hye		hyo	
My	みゃ	ミャ			みゅ	ミュ	みぇ	ミェ	みょ	ミョ
	mya				myu		mye		myo	
Ry	りゃ	リャ			りゅ	リュ	りぇ	リェ	りょ	リョ
	rya				ryu		rye		ryo	
	lya				lyu		lye		lyo	
Gy	ぎゃ	ギャ			ぎゅ	ギュ	ぎぇ	ギェ	ぎょ	ギョ
	gya				gyu		gye		gyo	
Zy	じゃ	ジャ			じゅ	ジュ	じぇ	ジェ	じょ	ジョ
	zya				zyu		zye		zyo	
	ja				ju		je		jo	
	jya				jyu		jye		jyo	
Dy			でぃ	ディ	どぅ	ドゥ	でぇ	デェ	どぉ	ドォ
			dyi		dyu		dye		dyo	
By	びゃ	ビャ			びゅ	ビュ	びぇ	ビェ	びょ	ビョ
	bya				byu		bye		byo	
Py	ぴゃ	ピャ			ぴゅ	ピュ	ぴぇ	ピェ	ぴょ	ピョ
	pya				pyu		pye		pyo	
Kw	くゎ	クヮ	くぃ	クィ			くぇ	クェ	くぉ	クォ
	kwa		kwi				kwe		kwo	
Ts	つぁ	ツァ	つぃ	ツィ			つぇ	ツェ	つぉ	ツォ
	tsa		tsi				tse		tso	
F	ふぁ	ファ	ふぃ	フィ	(ふ)	(フ)	ふぇ	フェ	ふぉ	フォ
	fa		fi		(fu)		fe		fo	
Gw	ぐゎ	グヮ	ぐぃ	グィ			ぐぇ	グェ	ぐぉ	グォ
	gwa		gwi				gwe		gwo	
V		ヴぁ		ヴぃ		(ヴ)		ヴぇ		ヴぉ
	va		vi		(vu)		ve		vo	
			うぃ	ウィ			うぇ	ウェ	うぉ	ウォ
			xwi				xwe		xwo	
			いぇ	イェ						
			ye							

Tabela 2: Dodatkowe reguły konwersji *romaji-kana*

Uwagi do wersji elektronicznej

Niniejszy artykuł ukazał się w *Biuletynie GUST* (ISSN 1230-5560) w zeszycie 16 z r. 2001 na stronach 3–13. Stanowi on tekst referatu wygłoszonego na IX Ogólnopolskiej Konferencji Polskiej Grupy Użytkowników Systemu TEX *BachoTEX 2001* pod tytułem *Współczesne TEXniki publikacyjne*, która odbyła się w Bachotku w dniach od 29 kwietnia do 1 maja 2001 r.

Wersja elektroniczna — w formacie Postscript (`JSB-Bach01.ps`) i PDF (`JSB-Bach01.pdf`) — jest dostępna pod adresem

`http://www.orient.uw.edu.pl/~jsbien/publikacje/JSB-Bach01.*`

oraz na stronach Grupy Użytkowników Systemu TEX (`http://www.gust.org.pl/BachoTeX/2001.html`). Różni się ona od wersji drukowanej tylko drobnymi poprawkami.