JANUSZ S. BIEŃ
Uniwersytet Warszawski

Standard Unicode i język polski

Słowa kluczowe: *edytorstwo naukowe, dawny język polski, znaki historyczne, kodowanie fontów, standard Unicode*

1. Wstęp

Coraz mniej osób pamięta chaos towarzyszący początkom przetwarzania tekstów polskich przez komputer – istniało wiele sprzecznych ze sobą propozycji kodowania polskich znaków (wydaje się, że większość z nich nie została udokumentowana).

Problem ten został rozwiązany przez omawiany dalej standard Unicode (*http://www.unicode.org*), ale tylko dla polskich tekstów dostatecznie zgodnych ze współczesną pisownią. W przypadku edycji – z musu komputerowej – polskich tekstów dawnych mamy niewątpliwy regres.

W czasach składu gorącego luminarze polskiej polonistyki – Konrad Górski, Władysław Kuraszkiewicz, Franciszek Pepłowski, Stefan Saski, Witold Taszycki, Stanisław Urbańczyk, Stefan Wierczyński, Jerzy Woronczak – pod redakcją naukową Marii Renaty Mayenowej przy współudziale Zofii Florczak – opracowali *Zasady wydawania tekstów staropolskich* (Górski 1955), które – jak się wydaje – były rzeczywiście stosowane w praktyce (ale patrz także Kierkowicz 2010). Z punktu widzenia autora i redaktora krytycznej edycji stosowanie tych zasad było łatwe i wygodne: w wydawnictwie tekst trafiał w ręce kompetentnego redaktora merytorycznego, który kierował tekst do druku do drukarni naukowej dysponującej odpowiednimi czcionkami. Istniały też specjalistyczne zakłady dorabiające potrzebne czcionki na zamówienie i dobra drukarnia czy wydawnictwo wiedziały, do kogo należy się w takiej sprawie zwrócić.

Obecnie autorzy prawie bez wyjątku składają teksty samodzielnie i są zdani tylko na siebie. Daje to efekty widoczne np. w opracowanej pod patronatem Biblioteki Narodowej edycji krytycznej *Kazań świętokrzyskich*, o której pisałem w artykule (Bień 2015, s. 57):

> Ta sama litera rękopisu, por. rys. 1, jest tam transkrybowana na dwa sposoby: jako **φ** i jako **ø**, por. rys. 2. Ponieważ fakt ten nie jest nigdzie skomentowany, nieodparcie nasuwa się podejrzenie, że to skutek „nowoczesnej" techniki redakcyjnej polegającej na mechanicznym sklejaniu plików otrzymanych od autorów.

Znany jest mi tylko jeden wyjątek, mianowicie stworzenie fontu (dokładniej fontów, ale cytuję tutaj tylko informacje dostępne publicznie) na potrzeby sejmowej

edycji dzieł wszystkich Jana Kochanowskiego (Hojdis 2016), ukazującej się sukcesywnie od 1982 r. na mocy uchwały sejmu z 1978 r., a która ma być ukończona w 2023 r. (ostatnio finansowana przez granty nr 11H 12 011181 i 1H 18 0246 86 Narodowego Programu Rozwoju Humanistyki realizowane w Instytucie Badań Literackich PAN). Nie można jednak tego przypadku uznać za wzór z powodów przedstawionych dalej.

Warto podkreślić, że nawet jeśli ktoś chciałby teraz stosować się do *Zasad…* (Górski 1955), np. do reguły na s. 28 *Różne postaci o przekreślonego […] sprowadzamy do znormalizowanej postaci…*, to napotka na poważny problem techniczny – patrz il. 1.

Nie zachowujemy następujących wariantów graficznych oryginałów: Różne postaci *o* przekreślonego (ꝋ ꝋ ꝋ ꝋ ꝋ) sprowadzamy do znormalizowanej postaci ꝋ. Różne postaci *a* przekreślonego (ꝛ ꝛ ꝛ)

Ilustracja 1. Zasady wydawania tekstów staropolskich (Górski 1955, s. 28)

Cytowana na ilustracji reguła stosowania „**o** rogatego" jest praktycznie niewykonalna, ponieważ znak taki nie występuje we współczesnych fontach – mówiąc ściślej, występuje w fontach TEX Gyre, ale niemal nikt o tym nie wie, np. nie wspomina się o tym w artykule (Jackowski, Pianowski i Strzelczyk 2018), a jego użycie jest kłopotliwe.

W artykule przedstawiam standard Unicode z perspektywy wykorzystania go do kodowania polskich tekstów dawnych, jest to więc w pewnym stopniu uzupełnienie artykułu (Strzelczyk 2013), a w pewnym stopniu również niejawna polemika z niektórymi jego tezami. Jest to również uzupełnienie artykułu (Bień 2019), gdzie pisałem:

> Zasadniczym celem artykułu jest prezentacja założeń przygotowanej przeze mnie eksperymentalnej edycji elektronicznej, ma on jednak również ważny cel poboczny, mianowicie zainteresowanie Czytelników funkcjonowaniem konsorcjum Unicode, które ustala sposoby reprezentowania znaków w komputerach; dodanie znaków przydatnych dla dawnej polszczyzny do standardu jest trudne, ale nie niemożliwe – wymaga jednak nie tylko wiedzy specjalistycznej, ale i orientacji w organizacji działalności normalizacyjnej.

Prezentacja standardu z konieczności będzie mocno uproszczona – (Korpela 2006, s. 198), podstawowe zasady i struktura standardu są proste, ale Unicode jako całość jest skomplikowany i zawiera trudne pojęcia, definicje i algorytmy.

Warto dodać, że konsorcjum Unicode utrzymuje również „wspólne repozytorium danych lokalizacyjnych" (*Common Locale Data Repository*) zawierające informacje specyficzne dla konkretnych języków, jak repertuar znaków, kolejność alfabetyczna itp. Wydaje się celowe stworzenie odpowiedniego zestawu informacji dla dawnej polszczyzny, ale to jest odrębny temat.

2. Geneza standardu Unicode

Pierwsze komputery posiadały jako urządzenia wejścia/wyjścia sprzęt korzystający z już istniejących nośników, jak taśma perforowana (dalekopisy) lub

karty perforowane, i stosowały zestawy znaków zaprojektowane dla tych nośników, początkowo 6- i 7-bitowe, potem także 8-bitowe. W epoce komputerów osobistych dominowały już kody 8-bitowe. Były one ograniczone do 256 znaków, co okazywało się często niewystarczające, zwłaszcza dla tekstów wielojęzycznych. W związku z tym opracowano koncepcję „rozszerzania kodu", a ściślej jego zmiany ograniczonej do wskazanego fragmentu tekstu; koncepcja została sformalizowana w obszernym i skomplikowanym dokumencie przygotowanym przez European Computer Manufacturers Association jako standard ECMA-35, a lepiej znanym jako norma ISO/IEC 2022 (Wikipedia contributors 2019a).

Jest to „kodowanie stanowe" (*stateful encoding*) – aby zinterpretować fragment tekstu niezbędne jest przeczytanie go od samego początku w celu odnalezienia kodów sygnalizujących właściwy sposób interpretacji.

Było oczywiste, że należy zwiększyć liczbę bitów przeznaczonych na reprezentację znaku. Radykalna propozycja wykorzystania 4 bajtów na 1 znak została sformułowana pod egidą Międzynarodowej Organizacji Standaryzacyjnej ISO (razem z IEC – Międzynarodowa Komisją Elektrotechniczną – firmującej m.in. wspomnianą normę ISO/IEC 2022), zrzeszającej krajowe organizacje normalizacyjne z prawie wszystkich państw świata. Według (Pike i Thompson 1993) pula dostępnych kodów miała być po prostu rozdzielona między kraje członkowskie do wykorzystania według własnego uznania. W dodatku nie zrezygnowano całkowicie z kodowania stanowego (patrz także: Wikipedia contributors 2019c).

Było to rozwiązanie wysoce niepraktyczne z punktu widzenia producentów oprogramowania, którzy przejęli inicjatywę i przygotowali projekt standardu Unicode, przeznaczającego 16 bitów na znak reprezentowany zawsze bezstanowo – tekst można poprawnie interpretować rozpoczynając czytanie w dowolnym miejscu. ISO ustąpiło i ustanowiona w 1993 r. norma ISO/IEC 10646-1 *Information technology – Universal Coded Character Set (UCS)* miała repertuar znaków i podstawowe zasady zgodne ze standardem Unicode 1.1, w dodatku postanowiono repertuar znaków synchronizować systematycznie (zdarza się jednak, że znaki wprowadzone do repertuaru przez ISO są przez Unicode niezalecane). Dwa bajty na znak okazały się szybko niewystarczające, począwszy od wersji 2.0 Unicode wykorzystuje jednostki 21-bitowe (patrz np. Blewitt 2016).

Warto pamiętać, że w szczegółach norma ISO (aktualnie ISO/IEC 10646: 2017 z późniejszymi dodatkami) cały czas różni się od standardu Unicode (aktualnie wersja 12.1 z 7 maja 2019 r. definiująca 137 929 znaków), nie będziemy jednak się tym tutaj zajmować.

Aby ułatwić powszechną akceptację standardu Unicode, jego twórcy zapewnili pełną zgodność repertuaru z pewną liczbą popularnych zestawów znaków; te odziedziczone po wcześniejszych standardach znaki są do dzisiaj źródłem różnych niekonsekwencji.

Podstawowym założeniem standardu jest reprezentowanie abstrakcyjnych znaków, a nie ich kształtów czyli glifów. Abstrakcyjny znak może mieć wiele kształtów, co jest raczej oczywiste, ale przyjmuje się również, że konkretny glif może reprezentować różne abstrakcyjne znaki, co jest już w dużym stopniu kwestią umowną.

Jednym z twórców standardu był Joseph D. Becker, który w swoim artykule *Multilingual Word Processing* z 1984 r. rozróżnił wprowadzanie (*input*), reprezentowanie (*reprezentation*) i zobrazowanie (*rendering*) tekstu (Becker 1984).

Problem w tym, że te 3 aspekty są do dziś przez wielu traktowane jako jedyne możliwe sukcesywne etapy przetwarzania tekstu – konsekwencją tego jest silna tendencja do minimalizacji liczby znaków i zakładania, że obsługa wariantów graficznych jest zadaniem autorów fontów. Ta postawa była zrozumiała ćwierć wieków temu, ale w czasach masowej dygitalizacji potrzebujemy często reprezentować wskanowany tekst jako ciąg znaków zanim dokonamy jego ostatecznej interpretacji.

Istnieją też zastosowania, takie jak badanie proweniencji starych druków, dla których właśnie glify są najbardziej istotne. Ciekawą inicjatywą wykorzystania do tego celu metod komputerowych był projekt *Cassetin* (André 2003) i późniejsze prace, (np. André i Jimenes 2013), które chyba nie są jednak kontynuowane. Również trenowanie programów automatycznego rozpoznawania znaków ma podobne potrzeby i wymaga specjalistycznych narzędzi. Odpowiedzialne za standard konsorcjum Unicode ciągle ignoruje te potrzeby, ale mimo tego dla standardu Unicode nie ma alternatywy.

3. Znaki Unicode

Jak było wspomniane, standard wykorzystuje jednostki 21-bitowe. Wszystkie liczby dające się w ten sposób reprezentować stanowią **przestrzeń kodową**, postrzeganą jako 17 płaszczyzn, z których każda ma 256 wierszy w 256 kolumnach. Konkretna liczba to **punkt kodowy** (stosowałem również termin **współrzędna kodowa**), w skrócie **kod** – tego użycia nie należy mylić ze znaczeniem konkretnego zestawu znaków kodowych. Punkty kodowe zapisujemy w postaci liczby heksadecymalnej poprzedzonej prefiksem U+, np. U+754C. Interesujące nas punkty kodowe są również nazywane **skalarami**[1].

Standard wyróżnia 7 równorzędnych typów punktów kodowych, które tutaj połączymy w bardziej intuicyjne grupy stosując własną terminologię;

- znaki ustalone (*Graphic, Format, Control*),
- znaki prywatne (*Private-use*),
- znaki specjalne (*Surrogate, Noncharacter*),
- znaki rezerwowe (*Reserved*).

Znaki ustalone są zdefiniowane w standardzie w sposób stabilny – ich podstawowe własności nigdy nie będą zmienione. Znaki prywatne, jak sama nazwa wskazuje, mają znaczenie i własności uzgodnione przez zainteresowanych użytkowników w sposób niezależny od standardu.

Z wcześniejszych standardów kodowania tekstów (por. np. Bień 1999) standard Unicode przejął definicję znaku (abstrakcyjnego), której tłumaczenie cytuję tutaj za (Bień 2004): *Jednostka informacji stosowana do organizacji danych tekstowych, sterowania nimi lub ich reprezentacji*.

[1] Skalarami nie są m.in. wspomniane niżej surogaty, którymi nie będziemy się w ogóle zajmować.

Oczywiście definicja taka jest pozbawiona praktycznego znaczenia, dlatego wolę inną definicję sformułowaną w cytowanym wyżej artykule (tutaj przytaczam ją nieznacznie zmodyfikowaną): *znak Unicodu to pojęcie pierwotne, zdefiniowane przez wyliczenie*.

Wyliczenie to polega na mniej lub bardziej jawnym wyliczeniu własności przypisanych do danego punktu kodowego. Najważniejszą własnością jest umowna nazwa, np. LATIN SMALL LETTER A WITH OGONEK; standard gwarantuje, że nigdy ona nie ulegnie zmianie, nawet gdy jest w jakimś sensie niepoprawna; punktowi kodowemu może być jednak przypisany w razie potrzeby **alias**.

Drugą ważną własnością znaków jest ich **kategoria** (ang. *general category*), np. **Lc** oznacza literę w kaszcie górnej. Wszystkich możliwych wartości jest 29.

Prawdopodobnie pierwotne ograniczenie do 16 bitów leży u podstaw zasady, że diakryty są w zasadzie kodowane jako oddzielne **znaki dostawne** (ang. *combining characters*) dodawane do **znaków bazowych** (ang. *base characters*). Wszystkie znaki dostawne mają kategorię **Mn** (ang. *Mark, non-spacing*). Sposób, w jaki znak dostawny łączy się ze znakiem bazowym, jest określony przez własność **łączliwość** (ang. *Combine*) – na przykład znak COMBINING OGONEK ma łączliwość równą 202, co oznacza *Attached Below*.

Znaki dostawne są zapisywane zawsze po znaku bazowym, ich kolejność ma znaczenie tylko wtedy, gdy na siebie oddziałują graficznie – znak występujący wcześniej powinien być zobrazowany bliżej znaku bazowego niż znak występujący później. Standard nie nakłada żadnych ograniczeń na liczbę znaków dostawnych odnoszących się do jednego znaku bazowego, co pozwala dla zabawy tworzyć tzw. teksty Zalgo[2].

Sekwencję znaków zapisujemy w nawiasach kątowych oddzielając je przecinkami, np. ⟨U+012F, U+0307, U+0301⟩; sekwencje mogą mieć swoje nazwy, np. przytoczona sekwencja nazywa się LATIN SMALL LETTER I WITH OGONEK AND DOT ABOVE AND ACUTE (podobno stosowana w języku litewskim w specjalistycznych publikacjach).

Na potrzeby segmentacji tekstu na słowa i na „znaki postrzegalne" (ang. *user-perceived characters*) w wersji 3 standardu pojawiło się pojęcie **zbitki grafemicznej** (*graphemic cluster*), a potem pojęcie **rozszerzonej zbitki grafemicznej** (*extended graphemic cluster*). Przykłady ilustrujące te pojęcia w normatywnym aneksie do standardu nr 29 pochodzą głównie z pisma koreańskiego, tajskiego i dewanagari; z pisma łacińskiego pochodzi przykład sekwencji **ch**, której w niektórych językach odpowiada w wymowie jedna spółgłoska – w konsekwencji zapis ten może być uznany za jeden grafem, ale jest moim zdaniem dyskusyjne, czy jest to jeden „znak postrzegalny". Nie ulega jednak wątpliwości, że dla zwykłego użytkownika pojęcie rozszerzonej zbitki grafemicznej jest bliższe intuicyjnemu rozumieniu znaku piśmiennego, niż pojęcie abstrakcyjnego znaku zdefiniowane w standardzie.

[2] Więcej na ten temat pod adresem: https://zalgo.org/ [dostęp 02.10.2019].

Zasada reprezentowania tylko abstrakcyjnych znaków, a nie ich kształtów, została naruszona przez wprowadzenie w wersji 3.2 selektorów wariantów (*variant selector*). Teoretycznie ich zakres użycia jest bardzo szeroki – por. stwierdzenie[3]:

> Dla historycznych systemów pisma sekwencje wariacyjne są pożytecznym narzędziem, ponieważ pozwalają zakodować błędne lub rzadkie (ang. *nonce*) czcionki (ang. *glyphs*) w sposób ukazujący ich związek ze znakiem podstawowym.

Praktyka jest jednak bardzo restryktywna – (por. np. Wikipedia contributors 2019e, Wikipedia contributors 2019d); co najmniej jedna propozycja użycia selektorów wariantów została odrzucona z nieznanych mi powodów[4].

Aktualny wykaz zatwierdzonych wariantów znajduje się w pliku *StandardizedVariants.txt*; ponad 1000 pozycji dotyczy sinogramów (ang. i jap. *kanji* – znaków powstałych na początku naszej ery w Chinach za panowania dynastii Han), ponad 200 dotyczy tzw. emoji, około 30 znaków matematycznych, pozostałe dotyczą języka mongolskiego. Ze względu na szerokie zastosowania sinogramów, używanych w Chinach i Japonii, a wcześniej również w Korei i Wietnamie, i wynikające z tego bardzo różnorodne potrzeby użytkowników, konsorcjum Unicode utrzymuje dodatkowo bazę *Ideographic Variation Database*, gdzie można rejestrować zgodnie z odpowiednią procedurą alternatywne kolekcje wariantów sinogramów (aktualnie baza liczy około 40 tysięcy pozycji). Obsługa selektorów wariantów jest uwzględniona w specyfikacji OpenType w postaci osobnej podtablicy tablicy odwzorowania *cmap*.

W zasadzie diakryty (znaki dostawne) odnoszą się zawsze do jednego znaku bazowego, wyjątkowo do dwóch, np. COMBINING DOUBLE TILDE. Zasada ta w pewnym sensie została naruszona przez wprowadzenie znaków COMBINING CYRILLIC TITLO LEFT HALF i COMBINING CYRILLIC TITLO RIGHT HALF (Andreev, Shardt i Simmons 2013)[5].

Przestawione zasady powodują występowanie zapisów równoważnych. Typowy przykład to znak samodzielny, odziedziczony po jakimś wcześniejszym kodzie jak LATIN SMALL LETTER A WITH OGONEK, i zapis w formie sekwencji zestawionej ze znaków LATIN SMALL LETTER A i COMBINING OGONEK. Jeśli znaki diakrytyczne nie oddziałują na siebie (np. jeden jest nad znakiem, a drugi pod nim), to ich kolejność w sekwencji zestawionej jest obojętna. W rezultacie sprawdzenie, czy dwa napisy – czyli sekwencje punktów kodowych – są równe, wymaga ich **normalizacji**, co jest dla programisty błędogenną komplikacją. W dodatku dwa typy normalizacji zdefiniowane w standardzie – kanoniczna i dostosowawcza (ang. *compatibility*) – nie zawsze odpowiadają potrzebom użytkownika, co zilustrujemy później.

[3] https://www.unicode.org/faq/vs.html#18 [dostęp 10.05.2019], także: bit.ly/2CjcsVA-vsUnicode [dostęp 10.05.2019].

[4] Por.: https://www.unicode.org/mail-arch/unicode-ml/y2018-m07/0120.html [dostęp 02.10.2019].

[5] Dla pełności obrazu można tez wspomnieć o niezalecanych znakach adnotacji interlinearnych (U+FFF9–U+FFFB).

4. Forma standardu Unicode

Podstawową formą standardu jest *Baza Znaków Unicode* (ang. *Unicode Character Database*, w skrócie UCD). Baza dostępna jest w wersji XML, ale bardziej praktyczna jest wersja czysto tekstowa ze względu na mniejszą objętość. Składa się z pewnej liczby plików, z których najważniejszy jest *UnicodeData.txt*. Oto przykład jednego wiersza tego pliku (dla przejrzystości podzielonego tutaj na fragmenty):

```
0105;LATIN SMALL LETTER A WITH OGONEK;
Ll;0;L;0061 0328;;;;N;
LATIN SMALL LETTER A OGONEK;;
0104;;0104
```

Na początku wiersza mamy punkt kodowy, potem oddzielone średnikami m.in. następujące własności: nazwa, kategoria (**Ll** – litera z dolnej kaszty), rozkładalność na sekwencję ⟨U+0061,U+0328⟩, przestarzała nazwa stosowana w Unicode 1.0, odpowiednik w kaszcie górnej i tzw. tytułowej (U+0104).

Inny przykład, to fragment pliku *NamedSequences.txt*:

```
# Additions for Lithuanian.
# Provisional 2006-05-18, Approved 2007-10-19
LATIN CAPITAL LETTER A WITH OGONEK AND ACUTE;0104 0301
```

Oczywiście baza ta nie jest przeznaczona do bezpośredniego wykorzystania przez użytkownika. Do przeglądania jej zawartości w wygodny sposób zostały stworzone różne narzędzia. Nie jest to właściwe miejsce na dokonanie ich przeglądu, dlatego wspomnę tylko o kilku. Witryna Unicode oferuje wyszukiwarkę różnych własności i pokaz działania wybranych algorytmów pod adresem *https://unicode.org/cldr/utility*. Do lokalnej instalacji przeznaczony jest program Unibook™ Character Browser (*https://unicode.org/unibook/help/unibook.htm*), dostępny niestety tylko dla MS Windows, wprawdzie bezpłatny, ale o zamkniętym kodzie – firmuje go Asmus Freytag – jeden z twórców standardu. Z narzędzi online godna polecenia jest witryna *https://codepoints.net/*, do której jeszcze wrócimy.

Druga część standardu to tabele kodów z przykładowymi glifami, pochodzącymi z różnych także komercyjnych fontów. Są one przygotowywane za pomocą wspomnianego wyżej programu Unibook na podstawie pliku *NamesList.txt*, który również należy do UCD. Oto przykładowy fragment:

```
0104 LATIN CAPITAL LETTER A WITH OGONEK
: 0041 0328
0105 LATIN SMALL LETTER A WITH OGONEK
* Polish, Lithuanian, ...
: 0061 0328
```

Fragment ten zawiera informacje o dekompozycji kanonicznej (objaśnionej dalej) i niekonsekwentne komentarze (prawdopodobnie odziedziczone po standardach ISO), plik uwzględnia też inne informacje umieszczane w tabeli kodów.

Trzecia część to opis standardu w języku angielskim w formie plików PDF. Opis zawsze dotyczy aktualnej wersji, z nielicznymi wyjątkami nie zawiera infor-

macji historycznych, w szczególności nie podaje informacji o motywach uwzględnienia w standardzie konkretnych znaków – informacji takich należy szukać za pośrednictwem *Unicode® Technical Committee Document Registry* (*https://unicode.org/L2/*). Zarówno tabele, jak i opis są dostępne bezpłatnie na witrynie konsorcjum Unicode; wersje papierowe mogą być do nabycia na zasadzie druku na żądanie.

Postać stosowanych w standardzie tabel kodów jest wygodna i przejrzysta, dlatego Ievgenii Meshcheriakov stworzył program *fntsample*, który w tym formacie produkuje tabele dla wskazanego fontu lub jego fragmentu. Program, na otwartej licencji, jest aktualnie dostępny pod adresem *https://github.com/eugmes/fntsample*. W 2013 r. jeden z moich studentów, Paweł Parafiński, stworzył na moją prośbę rozszerzenie tego programu upodabniające wyniki jeszcze bardziej do tabel standardu przez uzupełnienie tabel fontu o komentarze pochodzące z oryginalnego pliku *NamesList.txt* lub z własnego pliku przygotowanego w tym formacie. Rozszerzenie to znajduje się obecnie w dwóch repozytoriach *https://bitbucket.org/jsbien/unicode-ucd-parser* oraz *https://bitbucket.org/jsbien/fntsample-fork-with-ucd-comments*[6].

Twórcą i administratorem wspomnianej wcześniej witryny *https://codepoints.net/* jest Manuel Strehl. Witryna wróżnia się od innych, podobnych, m.in. tym, że w pewnym sensie je integruje: strona dla konkretnego znaku, np. *https://codepoints.net/U+A76B?lang=en*, zawiera linki między innymi do stron *http://unicode.org/cldr/utility/character.jsp?a=A76B*; *http://fileformat.info/info/unicode/char/A76B/index.htm*; *https://graphemica.com/*ꝫ (ꝫ = LATIN SMALL LETTER ET U+A76A, font np. NotoSansMono-Regular, *https://www.google.com/get/noto/)*; *scriptsource.org/char/U00A76B* i *http://www.isthisthingon.org/unicode/index.phtml?glyph = A76B* (Unisearcher). Dodatkowy link pokazuje glif znaku wycięty z oryginalnej tabeli standardu[7].

Głównym atutem tej witryny są dla mnie linki do odpowiednich stron Wikipedii. Choć w angielskiej wersji zdarzają się błędy (por. np. mój wpis w dyskusji hasła *https://en.wikipedia.org/wiki/UTF-16*), czasami zawierają one trudno dostępne informacje, a mianowicie linki do dokumentów zawierających uzasadnienie włączenia danych znaków do standardu. Dokumenty te są publicznie dostępne za pośrednictwem już wspomnianego *Unicode® Technical Committee Document Registry*, ale odnalezienie interesującego pliku bywa kłopotliwe. Wprawdzie hasła Wikipedii nie zawsze podają odpowiednie linki w optymalny sposób, wydaje się jednak, że wykorzystywanie jej do tego celu i popieranie tej inicjatywy jest lepsze niż tworzenie jakiegoś alternatywnego indeksu. Można w szczególności rozważyć stworzenie specjalnego portalu (por. *https://pl.wikipedia.org/wiki/Wikipedia:Portal*) lub projektu (por. *https://pl.wikipedia.org/wiki/Wikipedia:Wikiprojekt*).

Źródła witryny *https://codepoints.net/* są dostępne na swobodnej licencji i utrzymywane za pomocą systemu zarządzania wersjami – patrz *https://github.com/Codepoints/codepoints.net*. Język interfejsu jest modyfikowalny, ponieważ wszyst-

[6] Meshcheriakov rozważał włączenie tych funkcji do swojego programu, ale dotąd to nie nastąpiło – por. https://github.com/eugmes/fntsample/issues/7/.

[7] Haralambous (2002) nazywa go glifem uprzywilejowanym, ja wolałbym nazywać go glifem demonstracyjnym (ang. *demonstrative*).

kie napisy są zapisane zgodnie z konwencją *gettext* (por. np. *https://pl.wikipedia.org/wiki/GNU_gettext*). W dodatku autor zapewnił dostęp do witryny wspomagającej tłumaczenie (*https://crowdin.com/project/codepoints*).

Wszystko to sprawiło, że w 2016 r. podjąłem próbę polonizacji tej witryny (pomogła mi w tym Joanna Bilińska). Wynik jest dostępny, ale nie jest to wersja kompletna i ostateczna – prace nad polonizacją z różnych względów zaniechałem, ale byłoby dobrze, gdyby ktoś je kontynuował.

Jeśli tylko jest to możliwe, polskojęzyczna wersja witryny kieruje – co jest w zasadzie słuszne – do polskojęzycznej wersji Wikipedii. Niestety polskie hasła dotyczące znaków standardu są nie tylko mniej liczne i uboższe treściowo, ale i mniej wiarygodne. Edycja haseł Wikipedii wymaga nie tylko wiedzy, ale i talentów dyplomatycznych oraz cierpliwości w dyskusjach z innymi wikipedystami. Pomimo tego warto podejmować próby doprowadzenia polskich haseł dotyczących standardu do satysfakcjonującego poziomu.

5. Znaki prywatne

Na znaki prywatne przeznaczone są trzy obszary: 6 400 punktów kodowych na Podstawowej Płaszczyźnie Wielojęzycznej (BMP) i praktycznie całe płaszczyzny nr 15 i 16 po 65 534 punktów każda (ze względu na wysokie wartości kodów korzystanie z tych płaszczyzn jest mniej wygodne niż z BMP).

Ponieważ nie ma obowiązku rejestrowania porozumień dotyczących wykorzystania znaków prywatnych, sporządzenie ich pełnego wykazu jest niemożliwe. Pewną liczbę przykładów podaje artykuł (Wikipedia contributors 2019b). Najczęstszym powodem korzystania ze znaków prywatnych jest niedostępność w standardzie potrzebnego znaku, odpowiedniej sekwencji dostawnej lub zbitki grafemicznej. Są też powody czysto techniczne używania znaków prywatnych, które dają się reprezentować w standardzie jako sekwencje dostawne – pozwala to tworzyć fonty, w których kształt i położenie diakrytów są dopracowane indywidualnie dla konkretnego znaku, a nie są konsekwencją ogólnych algorytmów.

Jednym z przykładów wykorzystania znaków prywatnych są ustalenia inicjatywy stworzenia fontów unikodowych dla tekstów średniowiecznych, np. *Medieval Unicode Font Initiative*, w skrócie MUFI – patrz (Haugen 2013) i *https://skaldic.abdn.ac.uk/m.php?p=mufi*.

Od 2001 r. ukazały się 4 wersje rekomendacji, ostatnia w 2015 r. (Haugen 2015); inicjatywa ta wydaje się, niestety, martwa – chyba uznano, że cele zostały osiągnięte. Wersja 4 uwzględnia postulowaną przeze mnie ligaturę długiego **s** i małego **ł** występującą w korpusie IMPACT (Bień 2014) – aktualnie używany jest kod przyjęty w stworzonej ad hoc na potrzeby projektu modyfikacji fontu DejaVu Sans – por. *http://dl.psnc.pl/activities/projekty/impact/results/*. Uwzględnienie znaku w specyfikacji MUFI daje nadzieję, że znak ten pojawi się również w innych fontach, choć na dzisiaj jedynym fontem zgodnym z najnowszą wersją specyfikacji jest Junicode w wersji 1 lub wyższej (*http://junicode.sourceforge.net/*), dostępny na otwartej licencji.

Wszystkie wersje specyfikacji mają postać ręcznie opracowanych tekstów w formacie PDF, co nie ułatwia automatycznego przetwarzania omawianych w nich znaków. Wyszukiwarka dostępna na witrynie MUFI nie obsługuje wersji 4, w dodatku ani oprogramowanie, ani wykorzystywane dane nie są udostępnione w wersji źródłowej. W tej sytuacji na duże uznanie zasługuje Rebecca G. Bettencourt, która we własnym zakresie stworzyła pliki w formacie *UnicodeData.txt* m.in. dla znaków MUFI (*http://www.kreativekorp.com/charset/PUADATA/*).

W artykule (Bień 2016) zaproponowałem, aby kody znaków prywatnych zgodnych z rekomendacją MUFI poprzedzać prefiksem M+ zamiast U+, na przykład kod wspomnianego na s. 00 znaku SMALL VIRGULA możemy zapisać M+F1F7.

Dla pełności obrazu chciałbym również wspomnieć, że cenna moim zdaniem inicjatywa koordynacji wykorzystania PUA do szeroko rozumianych celów lingwistycznych, z którą w 2012 r. wystąpił Andreas Stötzner, pozostała praktycznie bez echa – informacje o niej są obecnie dostępne tylko w archiwum Internetu (*http://bit.ly/2XVTzRL-LINCUA*).

Wykorzystanie znaków prywatnych do składu dawnych tekstów polskich jest, moim zdaniem, nieuniknione.

6. Tekstele

Od dawna uważam, że aparat pojęciowy standardu Unicode jest niewystarczający. Powstają w związku z tym dwa pytania: jakie dodatkowe pojęcia zdefiniować i jak je nazwać.

Najbardziej potrzebne jest pojęcie, które odnosiłoby się do elementu tekstu niezależnie od sposobu jego zapisu, np. obejmujące łącznie LATIN SMALL LETTER A WITH OGONEK i sekwencję LATIN SMALL LETTER A i COMBINING OGONEK. W standardzie jest zdefiniowane pojęcie kanonicznej dekompozycji, oznaczane myląco znakiem równoważności (≡) choć nie jest to relacja symetryczna; np. sekwencja LATIN SMALL LETTER A i COMBINING OGONEK jest kanoniczną dekompozycją LATIN SMALL LETTER A WITH OGONEK, ale oczywiście nie odwrotnie. Jeśli dwa elementy tekstu mają taką samą kanoniczną dekompozycję to są kanonicznie równoważne (ta relacja nie ma swojego oznaczenia, choć jest równoważnością w matematycznym sensie). Pozornie proponowane pojęcie mogłoby być zdefiniowane jako klasa abstrakcji dla relacji kanonicznej równoważności, ale twórcy standardu skomplikowali sprawę uznając, że relacja ta zachodzi między niektórymi pojedynczymi znakami, np. OHM SIGN i GREEK CAPITAL LETTER OMEGA. Znaki te wyglądają identycznie, ale niosą inną informację, z której użytkownik może nie chcieć rezygnować.

Bardzo przydatną metodą analizy tekstu jest sporządzenie histogramu (wykazu częstości) znaków. Dla tekstów zapisanych zgodnie ze standardem Unicode do dzisiaj jedynym łatwo dostępnym (np. w dystrybucjach systemu Linux) narzędziem jest program unihist, wchodzący w skład zestawu narzędzi, których autorem jest Bill Poser (patrz *https://billposer.org/Software/unidesc*); znaki dostawne są zliczane osobno, co moim zdaniem, czyni program mało przydatnym. W 2008 r. mój stu-

dent Piotr Findeisen przygotował według mojej specyfikacji program unihistext, który jest pozbawiony tej wady – zlicza obiekty o tej samej nazwie, a sekwencje mają nadane nazwy za pomocą dostarczonego przez użytkownika pliku w formacie *NamedSequenced.txt*. Taki plik został przygotowany m.in. dla prywatnych znaków MUFI. Obecnie program jest dostępny w repozytorium *https://bitbucket.org/jsbien/unihistext/*, ale jest to niestety program osierocony.

Obiekty, które zlicza program unihistext, są przykładem obiektów, które proponuję nazywać **tekstelami** – powinny one być definiowane przez wyliczenie, ponieważ trzeba nadać im nazwy (niekoniecznie jawnie, mogą być tworzone za pomocą odpowiednich reguł mniej lub bardziej mechanicznie). Tekstele mają też inne własności, ale wynikają one w oczywisty sposób z własności ich znaków składowych.

Na składowe teksteli możemy mówić **tekstony**, aby jawnie odróżnić znak o kodzie U+0105 od tekstela o nazwie LATIN SMALL LETTER A WITH OGONEK (można pomyśleć o jakiejś odrębnej konwencji zapisywania nazw teksteli, ale na tym etapie to chyba zbędna komplikacja).

Standard definiuje również odwzorowanie dostosowawcze, oznaczane znakiem ≋. Informuje ono, że znaki lub ich sekwencje są w jakimś sensie podobne. Na przykład ligatury są rozpisywane na składowe, np. znakowi LATIN SMALL LIGATURE FI odpowiada sekwencja LATIN SMALL LETTER F i LATIN SMALL LETTER I, co jest sensowne. Jednak niektóre inne decyzje szczegółowe są kontrowersyjne, np. MICRO SIGN ma odwzorowanie dostosowawcze na GREEK SMALL LETTER MU (dlaczego nie kanoniczne?), a LATIN SMALL LETTER LONG S na LATIN SMALL LETTER S. Przygotowując przeszukiwalną digitalizację *Słownika polszczyzny XVI wieku* (Bień 2010) chcieliśmy rozpisać ligatury (użytkownik pisząc kwerendę nie powinien zajmować się kwestiami czysto typograficznymi), ale zachować długie s, które jest istotnym elementem tekstów z tego okresu (w kwerendzie w razie potrzeby można utożsamić długie i krótkie s stosując wyrażenia regularne), w konsekwencji nie mogliśmy użyć normalizacji dostosowaczej. Jest to więc też zastosowanie pojęcia **tekstela**, choć tym razem niejawne.

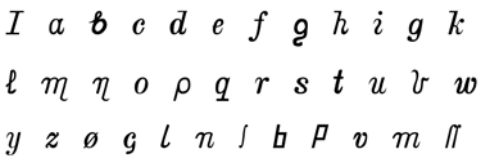
Stosunkowo mało znany i względnie nowy język programowania Swift definiuje znakowy typ danych następująco: *Every instance of Swift's Character type represents a single extended grapheme cluster*[8]. Inny słowy podstawowym znakowym typem danych jest swoisty „znak postrzegalny", który moim zdaniem jednak zdecydowanie lepiej nazywać tekstelem.

Na potrzeby systemu składania tekstów Omega[9] zostało wprowadzone pojęcie tekstemu (Haralambous 2007, s. 313) oznaczającego strukturę danych zawierająca jednocześnie; znak (Unicode'u), jeden lub więcej glifów, i potencjalnie zestaw par klucz–wartość nazywanych własnościami (m.in. kolor, font, język). System Omega nie spełnił oczekiwań i jego rozwój został zaniechany, ale pojęcie „tekstemu" jako obiektu zawierającego kod i konkretny glif z konkretnego fontu oraz należącego do konkretnego języka może w przyszłości okazać się pożyteczne.

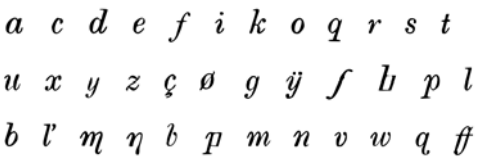
[8] https://docs.swift.org/swift-book/LanguageGuide/StringsAndCharacters.html.

[9] https://en.wikipedia.org/wiki/Omega_(TeX).

7. Analiza przypadku – traktat Parkosza



Ilustracja 2. Repertuar fontu Parkosz (autor Maciej Strzelczyk)



Ilustracja 3. Repertuar fontu Parkosz1907 (autor Janusz S. Bień)

Piętnastowieczny rękopiśmienny łaciński traktat Jakuba Parkosza (Parkoszowica) jest pierwszym tekstem poświęconym pisowni polskiej, z tego powodu znanym i często cytowanym – także dlatego, ze proponował wprowadzenie nowych specyficznych liter. Na potrzeby krytycznej edycji opublikowanej w 1907 r. najwyraźniej odlano specjalne czcionki. Na potrzeby krytycznej edycji opublikowanej w 1985 r. nietypowe znaki zostały narysowane przez grafika, a następnie prawie cała redakcja (?) PWN brała udział we wklejaniu około dwóch tysięcy wycinków we właściwe miejsca tekstu. W czasach edycji elektronicznych sytuacja wygląda lepiej tylko teoretycznie, jak opisałem to szczegółowo w artykule (Bień 2019).

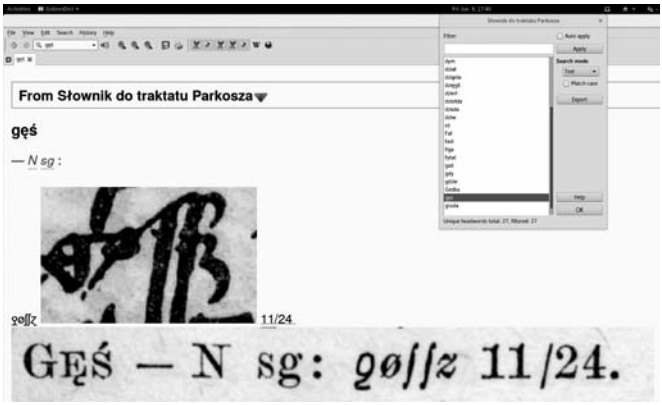
W 2013 r. próbę stworzenia fontu dla traktatu Parkosza podjął mój student Maciej Strzelczyk[10] (por. il. 2), który wzorował się na kształtach liter z edycji z 1985 r. Później samodzielnie spróbowałem stworzyć font wzorowany na czcionkach z edycji z 1907 r.[11] (por. il. 3). Obecnie uważam to za błąd – ponieważ chodzi o rękopis, czcionki powinny być kursywne, jak w edycji z 1985 r. Niestety, żaden z tych tworzonych nieprofesjonalnie fontów nie może być zalecany do użytku ze względu na niesatysfakcjonującą jakość.

Nowym elementem w sprawie jest udostępnienie w Internecie dobrej jakości skanu oryginału – patrz *https://jsbien.github.io/Parkosz4IIIF/*. Istnieje też niedokończony elektroniczny indeks wystąpień poszczególnych polskich słów w rękopisie, dostępny w dwóch formach (por. il. 4 i il. 5) *https://bitbucket.org/jsbien/parkosz-traktat/*. Można więc przeanalizować kształt liter i przemyśleć, jak oddać je w foncie.

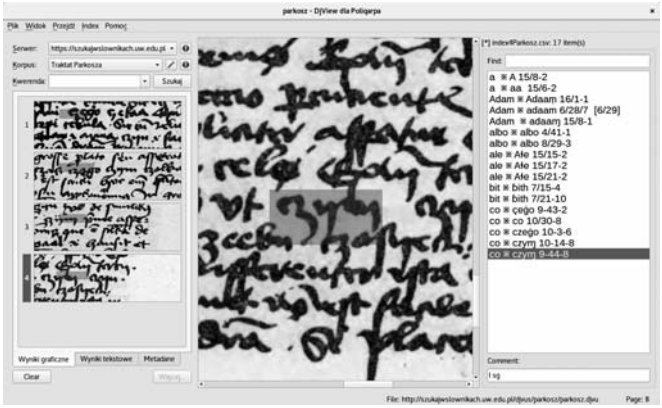
Ze względu na brak odpowiedniego fontu, w eksperymentalnym wydaniu elektronicznym zastosowałem transliterację szczegółowo opisaną we wspomnianym artykule (Bień 2019). W stosunku do poprzednich edycji wprowadziłem pewną innowację: litery, które mają standardowy kształt, ale u Parkosza wymawiają się inaczej, są również transkrybowane – na przykład zamiast zwykłego *m* (które u Parkosza oznacza zawsze spółgłoskę miękką) stosuję *ṃ* (LATIN SMALL LETTER M WITH DOT BELOW). Obecnie zastosowałbym inna konwencję, mniej lub

[**10**] https://bitbucket.org/jsbien/parkosz-font-old [dostęp 10.05.2019].

[**11**] https://bitbucket.org/jsbien/parkosz-font/ [dostęp 10.05.2019].



Ilustracja 4. Indeks wyrazów jako słownik elektroniczny

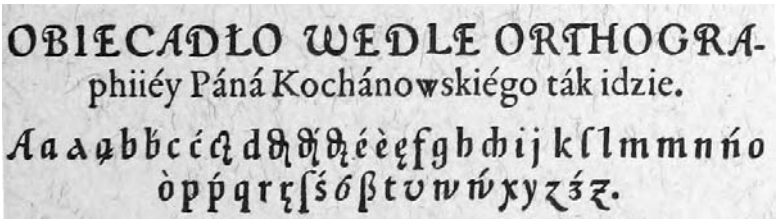


Ilustracja 5. Indeks wyrazów – program djview4poliqarp

bardziej inspirowaną niekiedy ekscentrycznymi propozycjami, które na liście dyskusyjnej standardu Unicode (*http://www.unicode.org/mailman/listinfo/unicode*) zgłasza William Overington (np. *https://forum.high-logic.com/viewtopic.php?f=10&t=7831*). Mianowicie każdy znak z transkrybowanego alfabetu Parkosza – niezależnie od tego, czy jest stosowany współcześnie czy nie – łączyłbym ze znakiem COMBINING TRIPLE UNDERDOT, np. m̟. Powód, dlaczego nie zrobiłem tego wcześniej jest prosty – dopiero niedawno zauważyłem, że taki znak istnieje. Dałoby to wyrazistą transkrypcję alfabetu Parkosza, a po stworzeniu odpowiedniego fontu można by wykorzystać mechanizm ligatur do zastępowania tych sekwencji przez właściwe glify.

8. Analiza przypadku – *Nowy Karakter Polski*

Opublikowany w 1594 r. Jana Januszowskiego traktat *Nowy Karakter Polski* porównuje trzy propozycje pisowni polskiej: własną, Jana Kochanowskiego i Łukasza Górnickiego. W publikacji użyto przy tym zaprojektowanego przez Januszowskiego kroju czcionek. Ich skany są obecnie dostępne w wielu bibliotekach cyfrowych, najwygodniej udostępnia je moim zdaniem Dolnośląska Biblioteka Cyfrowa (*https://www.dbc.wroc.pl/publication/4239*).


OBIECADŁO WEDLE ORTHOGRA-
phiiéy Páná Kochánowskiégo ták idzie.
Aa ą b b c ć q d đ đ ę è ę f g h ch i j k l l m m n ń o
ò p ṕ q r ę ſ ś ó ß t v w ẃ x y z ź ż.

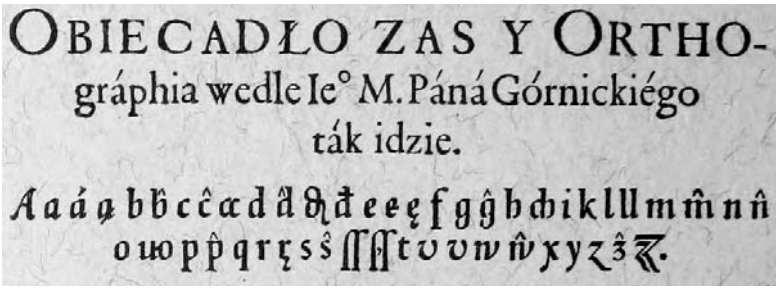
Ilustracja 6. Pisownia Jana Kochanowskiego (s. G3)

Digitalizacja kroju pisma Nowy Karakter Polski stanowi duże wyzwania ze względu na dużą liczbę znaków, które nie są współcześnie używane. Stwarza to szereg problemów, z których najbardziej rzuca się w oczy potrzeba odpowiedniego fontu do prezentacji tych znaków.

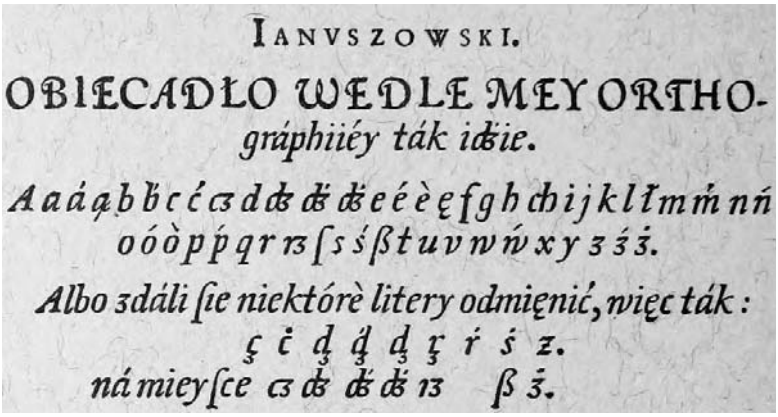
Pierwsza znana digitalizacja czcionek Januszowskiego to font FA Karakter Artura Frankowskiego stworzony w 1995 r. Na pytanie skierowane do autora o dostępność fontu otrzymałem w 2010 r. odpowiedź *Obecnie dostępna jest jedna odmiana kroju FA Karakter. Jeden font (odmiana) kosztuje 90 zł netto. Powyższe cena dotyczy licencji na jedno stanowisko komputerowe dla jednego nabywcy*. Na pytanie, jaki jest repertuar znaków i jaki format, nie otrzymałem nigdy odpowiedzi.

Wiadomo też, że font zdigitalizował Paweł Osial (Wieluńska 2018a, s. 39) na potrzeby swojej pracy dyplomowej; repertuar znaków nie jest mi znany.

O wiele lepiej udokumentowana jest rodzina czterech fontów, które stworzyła Anna Wieluńska. Proces ich tworzenia został opisany artykule (Wieluńska


OBIECADŁO ZAS Y ORTHO-
gráphia wedle Ie° M. Páná Górnickiégo
ták idzie.
Aa á ą b b̆ c ĉ ć d đ đ̆ e e ę f g ĝ h ch i k l ll m m̂ n n̂
o ꝏ p p̂ q r ę s ŝ ſſ ſſ t v v w ŵ x y z ẑ ʒ.

Ilustracja 7. Pisownia Łukasza Górnickiego (s. G3v)


IANVSZOWSKI.
OBIECADŁO WEDLE MEY ORTHO-
gráphiiéy ták idzie.
Aa á ą b b̆ c ć cz d đ đ̆ e é è ę f g h ch i j k l ł m ḿ n ń
o ó ò p ṕ q r rz ſ s ś ß t u v w ẃ x y z ź ż.
Albo zdáli ſie niektórè litery odmięnić, więc ták:
ę č ḍ ą́ ḍ ę ŕ ś z.
ná mieyſce cz đ đ̆ đ rz ß ż.

Ilustracja 8. Pisownia Jana Januszowskiego (s. H3v)

2018a) będącym omówieniem jej pracy magisterskiej, opublikowanej również w wersji rozszerzonej (Wieluńska 2018b). Rodzinie fontów, nazwanej Lazarus, poświęcona jest witryna *www.lazarus.wtf*.

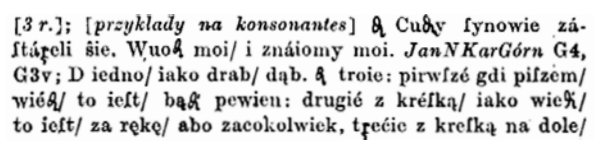
W pracy (Wieluńska 2018b, s. 52) znajdujemy stwierdzenie: *Projekt i rozbudowa historycznych odmian odlanych przez Januszowskiego (italic, upright italic) zostały dofinansowane w ramach stypendium Ministra Kultury i Dziedzictwa Narodowego i zostaną udostępnione na otwartej licencji*. Aktualnie są one jednak dostępne – o ile mi wiadomo – tylko na stronie *https://capitalics.wtf/pl/font/lazarus* na zasadach *Licencji Darmowej* (*https://capitalics.wtf/pl/licence*), która licencją otwartą nie jest. Pozostałe dwa fonty są dostępne w cenie 123 zł każdy (cały komplet kosztuje 153,75 zł).

Repertuar znaków został uwspółcześniony. Na stronach 58–59, 66–67 i 74––75 znajdują się porównania stron oryginału z tekstem złożonym fontami z rodziny Lazarus. Najbardziej rzuca się w oczy brak znaków wymaganych przez pisownię Kochanowskiego. Inne różnice to pewnie też wynik świadomych decyzji projektowych: współczesny kształt liter **k** i **r**, nosówek **ą** i **ę** (w oryginale przekreślenia zamiast ogonków), współczesny dywiz używany do dzielenia również słów w gotyku (zamiast DOUBLE OBLIQUE HYPHEN), współczesny ukośnik zamiast SHORT VIRGULA (M+F1F1), odróżnianie **u** i **v**. Niektóre różnice to po prostu przeoczenia autorki – tak chyba trzeba zaklasyfikować np. brak diakrytu nad **e** w trzecim wierszu od góry na s. 58.

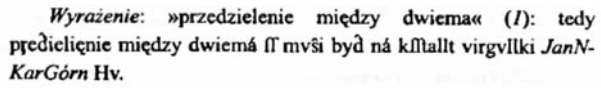
O wspomnianej we wstępie digitalizacji fontów z *Nowego Karakteru Polskiego* uwzględniającej m.in. znaki pisowni proponowanej przez Jana Kochanowskiego wiadomo niestety bardzo mało: została wykonana pod kierunkiem Bogdana Hojdisa, a jej autorami są Monika Marek-Łucka i Damian Langosz. Nie wiadomo, czy font jest dostępny publicznie, a jeśli tak, to na jakiej licencji. Niewielka próbka dostępna jest w artykule (Hojdis 2016, s. 9), nieco większa na slajdzie cytowanym w sprawozdaniu z konferencji *Tekstologia i typografia w przestrzeni cyfrowej* (Toruń 2015), gdzie artykuł ten był referowany – patrz (Pest 2015, s. 73). Sprawozdanie cytuje też bardzo trafne stwierdzenie referenta Bodana Hojdisa: *najtrudniejszym problemem do rozwiązania pozostaje jednak samo kodowanie staropolskich znaków, dla których nie przewidziano miejsca w międzynarodowym standardzie Unicode* (Pest 2015, s. 72).

Instytut Badań Literackich PAN, który jest odpowiedzialny za wydanie sejmowe dzieł wszystkich Jana Kochanowskiego, od 1948 r. prowadzi również prace nad *Słownikiem polszczyzny XVI wieku*. Dotychczas ukazało się 37 tomów (łącznie około 20 tysięcy stron). *Nowy Karakter Polski* jest cytowany w treści słownika, a nietypowe litery są oddawane w sposób zależny od zastosowanej technologi składu – jak widać, wierne cytowanie oryginału jest istotne przynajmniej dla niektórych polonistów.

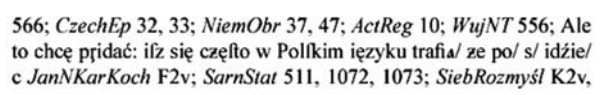
Redakcja słownika w ramach grantu NPRH nr 0138/FNiTP/H11/80/2011 *Korpus polszczyzny XVI wieku. Etap I: digitalizacja źródeł oraz stworzenie narzędzi informatycznych i udostępnienie materiałów testowych korpusu* realizowanego w latach 2012–2017 opublikowała w wersji elektronicznej kilka pozycji serii *Biblioteka*


[3 r.]; [przykłady na konsonantes] ą Cuby ſynowie zá-
ſtąpeli śie. Wuoą moi/ i znáiomy moi. *JanNKarGórn* G4,
G3v; D iedno/ iako drab/ dąb. ą troie: pirwſzé gdi piſzem/
wiąą/ to ieſt/ bąą pewien: drugié z kréſką/ iako wieą/
to ieſt/ za rękę/ abo zacokolwiek, tręćie z kreſką na dole/

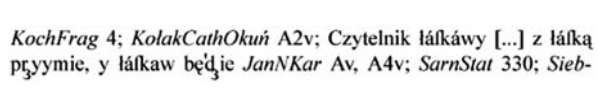
Ilustracja 9.
SpXVIw t. VI (1972) s. 303
– skład monotypowy


Wyrażenie: »przedzielenie między dwiema« (*1*): tedy
prędzieląnie między dwiemá ſi mvſi być ná kſtałt virgvlki *JanN-*
KarGórn Hv.

Ilustracja 10.
SpXVIw t. XXXI (2003) s. 221
– skład komputerowy (Kombi)


566; *CzechEp* 32, 33; *NiemObr* 37, 47; *ActReg* 10; *WujNT* 556; Ale
to chcę prędać: iſz się częſto w Polſkim ięzyku trafia/ ze po/ s/ idźie/
c *JanNKarKoch* F2v; *SarnStat* 511, 1072, 1073; *SiebRozmyśl* K2v,

Ilustracja 11.
SpXVIw t. XXXIII (2009) s. 172
– skład komputerowy (Kombi)

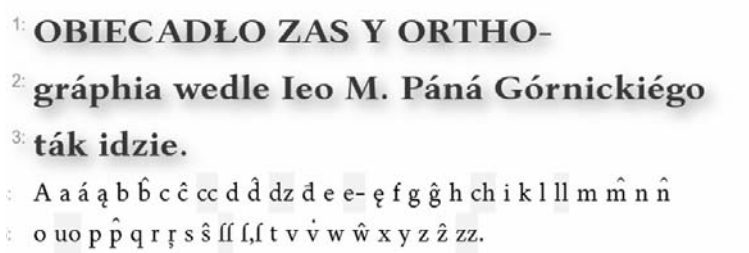

KochFrag 4; *KołakCathOkuń* A2v; Czytelnik łáſkáwy [...] z łáſką
prʒyymie, y łáſkaw będʒie *JanNKar* Av, A4v; *SarnStat* 330; *Sieb-*

Ilustracja 12.
SpXVIw t. XXXIII (2009) s. 261
– skład komputerowy (Kombi)

źródeł Słownika polszczyzny XVI wieku. Repozytorium cyfrowe tekstów szesnastowiecznych (w jednolitej transliteracji zgodnej z „Zasadami wydawania tekstów staropolskich (projekt)”, Wrocław 1955[12]) pod redakcją Patrycji Potoniec i Krzysztofa Opalińskiego (ISBN 978-83-65832-90-0), w szczególności kilka pozycji podserii


15: OBIECADŁO WEDLE ORTHOGRA-
16: phiiéy Páná Kochánowſkiégo ták idzie.
17: A a ą b b́ c ć cz d dzdźdz é è ę f g h ch i j k l ł m ḿ n ń o
18: ò p ṕ q r ŗ ſ ś σ ſz t v w ẃ x y z ź z.

Ilustracja 13. Transkrypcja – HTML: pisownia Jana Kochanowskiego (s. G3)


1: OBIECADŁO ZAS Y ORTHO-
2: gráphia wedle Ieo M. Páná Górnickiégo
3: ták idzie.
A a á ą b b̂ c ĉ cc d d̂ dz đ e e- ę f g ĝ h ch i k l ll m m̂ n n̂
o uo p p̂ q r ŗ s ŝ ſſ ſ,ſ t v v̇ w ŵ x y z ẑ zz.

Ilustracja 14. Transkrypcja – HTML: pisownia Łukasza Górnickiego (s. G3v)

Słowniki i gramatyki w opracowaniu Anny Nath-Dokurno, Patrycji Potoniec i Piotra Małka. W podserii tej ukazał się również *Nowy karakter Polski* (ISBN 978-83-65832-09-2) dostępny w wersji PDF w Repozytorium Cyfrowym Instytutów Naukowych *http://rcin.org.pl/publication/82568*, a w formie HTML pod adresem *http://spxvi.edu.pl/korpus/teksty/JanNKar/*. Teksty zakodowane są zgodnie ze stan-

[**12**] Stwierdzenie to nie jest w pełni prawdziwe.

1: IANVSZOWSKI.
2: OBIECADŁO WEDLE MEY ORTHO-
3: gráphiiéy ták idźie.
4: A a á ą b b́ c ć cz d dzdźdż e é è ę f g h ch i j k l ł m ḿ n ń
5: o ó ò p ṕ q r rz ſ s ś ſz t u v w ẃ x y z ź ż.
6: Albo zdáli ſie niektórè litery odmięnić, więc ták:
7: ç ċ ḍ ḍ́ ḍ̣ ṛ ṛ́ ṡ z.
8: ná mieyſce czdzdźdżrzſz ż.

Ilustracja 15. Transkrypcja – HTML: pisownia Jana Januszowskiego (s. H3v)

dardem Unicode z wykorzystaniem znaków prywatnych z rekomendacji MUFI. Wykorzystywany jest font Palemonas MUFI, który można pobrać – w kilku odmianach – z witryny MUFI *https://folk.uib.no/hnooh/mufi/fonts*; nie należy go mylić z fontem Palemonas (*https://klevas.mif.vu.lt/~vladas/Palemonas.htm*), przeznaczonym specjalnie dla języka litewskiego. Problem niedostępnych znaków rozwiązano stosując transkrypcję, przy czym transkrybowane fragmenty są wyróżnione kolorem – patrz ilustracje 13, 14, 15, 16, 17, 18.

15: OBIECADŁO WEDLE ORTHOGRA-
16: phiiéy Páná Kochánowſkiégo ták idzie.
17: A a ɑ ą b b́ c ć cz d dzdźdz é è ę f g h ch i j k l ł m ḿ n ń o
18: ò p ṕ q r ṛ ſ ś σ ſz t v w ẃ x y z ź ʒ.

Ilustracja 16. Transkrypcja – PDF: pisownia Jana Kochanowskiego (s. G3)

1: OBIECADŁO ZAS Y ORTHO-
2: gráphia wedle Ieº M. Páná Górnickiégo
3: ták idzie.
4: A a á ą b b̂ c ĉ cc d d̂ dz đ e e- ę f g ĝ h ch i k l ll m m̂ n n̂
5: o uo p p̂ q r ṛ s ŝ ſſ ſ,ſ t v v̇ w ŵ x y z ẑ zz.

Ilustracja 17. Transkrypcja – PDF: pisownia Łukasza Górnickiego (s. G3v)

1: IANVSZOWSKI.

2: OBIECADŁO WEDLE MEY ORTHO-

3: gráphiiéy ták idźie.

4: A a á ą b b́ c ć cz d dzdźdż e é è ę f g h ch i j k l ł m ḿ n ń

5: o ó ò p ṕ q r rz ſ s ś ſz t u v w ẃ x y z ź ż.

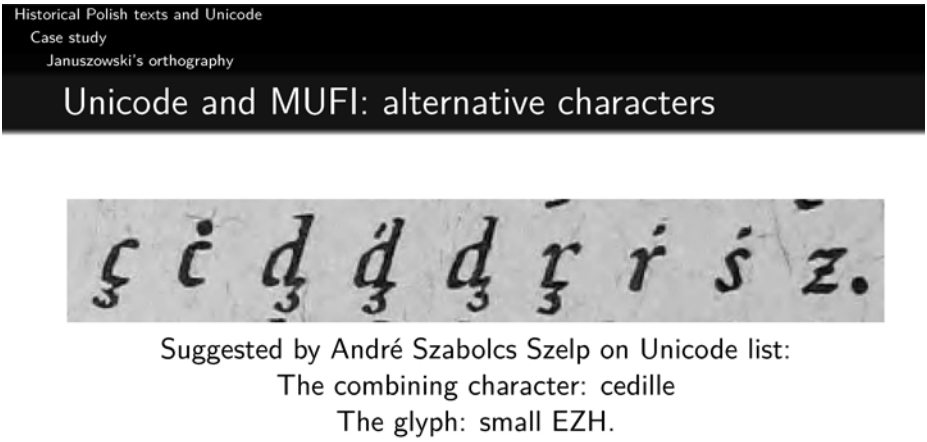
6: Albo zdáli ſie niektórè litery odmięnić, więc ták:

7: ç ċ ḑ ḑ́ ḑ̧ ŗ ṙ ṡ ʒ.

8: ná mieyſce czdzdźdżrzſz ż.

Ilustracja 18. Transkrypcja – PDF: pisownia Jana Januszowskiego (s. H3v)

Nowy Karakter Polski intrygował mnie od dawna jako swoiste wyzwanie dla informatyka i lingwisty. W 2010 r. przygotowałem propozycję kodowania w formie slajdów pod tytułem *Historical Polish texts and Unicode* i udostępniłem w nieistniejącej już Bibliotece Cyfrowej Katedry Lingwistyki Formalnej UW (obecnie opracowanie jest dostępne pod adresem *https://core.ac.uk/display/11337645*). Informację o tym wysłałem na listę użytkowników standardu Unicode inicjując wątek *preparing a PUA specification (for historical Polish texts)* (*https://www.unicode.org/mail-arch/unicode-ml/y2010-m04/0024.html*). Odzew był niewielki, ale pojawiły się jednak pewne pożyteczne uwagi, które uwzględniłem w kolejnej wersji tekstu, por. il. 19. (znaki alternatywnej pisowni Januszowskiego).



Ilustracja 19. Janusz S. Bień *Historical Polish texts and Unicode* (2011) s. 65

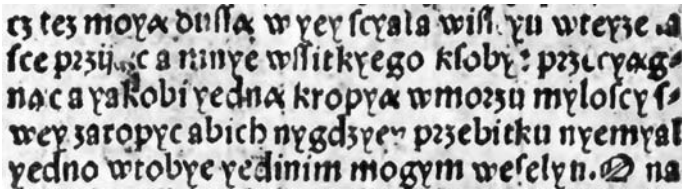
9. Inne przykłady

Nowy Karakter Polski nie jest jedynym tekstem źródłowym, który sprawia problemy techniczne redakcji słownika polszczyzny XVI wieku. Pewne pojęcie daje o nich artykuł (Opaliński 2007), elektroniczna wersja instrukcji redakcyjnej (*http://spxvi.edu.pl/instrukcja/*), a także wewnętrzna instrukcja *Podstawy edycji haseł w programie KOMBI 6* (Toruń 2003) autorstwa Krzysztofa Opalinskiego, która niestety nie jest dostępna publicznie.

Jednym z takich problematycznych utworów jest *Raj duszny* (*http://www.wbc.poznan.pl/publication/418100*) z 1513 r. uważany za pierwszą polską książkę drukowaną, por. il. 20 i 21.

Zwrot: »przebytek mieć [*w kim*]« (*1*): racz tez moyα duſſα w yey ſcyala wiſſcyu wteyze laſce przijąc a mnye wſſitkyego kſobye przecyągnąc a yakobi yednα kroplα wmorzu myloſcy ſwey zatopyc abich nygdzyey przebitku nyemyal yedno wtobye [*Boże*] yedinim mogym weſelym. *BierRaj* 17v.

Ilustracja 20. SpXVIw t. 31 s. 42 – *Raj Duszny* s. 17v



IIlustracja 21. *Raj Duszny* s. 17v

W tekście widzimy m.in. literę alfa zwykłą i przekreśloną. Jak wspominałem o tym na liście dyskusyjnej standardu (*http://unicode.org/mail-arch/unicode-ml/y2010-m08/0197.html*), interpretowałbym ją jako LATIN SMALL LETTER ALPHA (U+0251), choć w większości fontów jej kształt nie jest satysfakcjonujący.

10. Uwagi końcowe

Na witrynie konsorcjum Unicode znajdują się również notatki techniczne (Unicode Technical Notes, *http://www.unicode.org/notes/*), które nie są częścią standardu i mają charakter czysto informacyjny. Autorami większości z nich są współtwórcy standardu, ale nie jest to zasadą. Notatka nr 41 *Church Slavonic Typography in Unicode* nasunęła mi myśl, że można spróbować przygotować i zgłosić notatkę dotycząca polszczyzny. Materiały do takiej notatki zacząłem gromadzić w repozytorium *https://bitbucket.org/jsbien/unicode4polish*, częściowo w formie wiki (*https://bitbucket.org/jsbien/unicode4polish/wiki/Home*). Okazało się to jednak zbyt żmudne. Informacje bibliograficzne przeniosłem do tzw. grupy systemu Zotero (*https://www.zotero.org/groups/2196114/unicode_for_polish*), niektóre inne informacje w miarę wolnego czasu umieszczałem w odpowiednich hasłach Wikipedii, por. np. *https://pl.wikipedia.org/wiki/ƶ*; *https://pl.wikipedia.org/wiki/Ǫ*.

Utworzone przeze mnie repozytoria nie są już rozwijane, ale będę się starał, aby były dostępne w Internecie przez rok od publikacji niniejszego artykułu, potem zostaną zlikwidowane. Zainteresowane osoby proszone są więc o ich skopiowanie

lub zgłoszenie się do mnie w celu przejęcia administracji repozytoriami. Do przejęcia lub współadministrowania dostępna jest również wspomniana wyżej grupa Zotero.

11. Podziękowania

Niniejszy artykuł został przygotowany – jak wszystkie moje artykuły – za pomocą systemu TEX (konkretnie XƎLATEX). Na życzenie Redakcji został on skonwertowany do formatu Worda (konkretnie do akceptowanego przez Worda formatu ODT) za pomocą programu make4ht. W dokonaniu konwersji pomógł mi istotnie autor programu Michal Hoftich – jestem mu za to bardzo wdzięczny.

Bibliografia

André, Jacques (2003), *The Cassetin Project – Towards an Inventory of Ancient Types and the Relate Standardised Encoding*. „TUGboat" nr 24(3), s. 314–318. URL: http://www.tug.org/TUGboat/tb24-3/andre.pdf.

André, Jacques i Rémi Jimenes (2013), *Transcription et codage des imprimés de la Renaissance*. „Revue des Sciences et Technologies de l'Information – Série Document Numérique" nr 16(3), s. 113–139. DOI: 10.3166/DN.16.3.113-139. URL: https://halshs.archives-ouvertes.fr/ halshs-00983575.

Andreev, Aleksandr, Yuri Shardt i Nikita Simmons (2013), *Proposal to Encode Combining Half Marks Used for Cyrillic Supralineation*. Spraw. tech. L2/13-139. ISO/IEC JTC 1/SC 2/ WG 2. URL: http://std.dkuug.dk/jtc1/sc2/wg2/docs/n4475.pdf (dostęp 10.05.2019).

Becker, Joseph D. (1984), *Multilingual word processing*. „Scientific American" nr 251(1), s. 96–107. URL: http://www.jstor.org/stable/24969416.

Bień, Janusz S. (1999). *Kodowanie tekstów polskich w systemach komputerowych*. „Postscriptum" nr 27-29, s. 4–27. URL: https://core.ac.uk/display/11337736.

Bień, Janusz S. (2004). *Standard Unicode 4.0. Wybrane pojęcia i terminy*. „Biuletyn GUST" nr 20, s. 9–14. URL: https://core.ac.uk/reader/11337594.

Bień, Janusz S. (2010). *Dygitalizacja i komputeryzacja słowników na przykładzie Słownika polszczyzny XVI wieku*, w: *Język polski – wczoraj, dziś, jutro*. Red. Barbara Czopek-Kopciuch i Piotr Żmigrodzki. Kraków: Instytut Języka Polskiego PAN i Wydawnictwo LEXIS, s. 131–138. URL: https://core.ac.uk/display/11337634.

Bień, Janusz S. (2014). *The IMPACT project Polish Ground-Truth texts as a DjVu corpus*. „Cognitive Studies | Études Cognitives" nr 14, s. 75–84. URL: https://ispan.waw.pl/ journals/ index.php/ cs-ec/article/view/cs.2014.008 (dostęp 10.05.2019).

Bień, Janusz S. (2015). *O pojęciach znaku, słowa i fleksemu*. „Poradnik Językowy" nr 2, s. 66–72. URL: https://www.researchgate.net/publication/334646088_O_pojeciach_ znaku_slowa_ i_fleksemu.

Bień, Janusz S. (2016). *Problemy kodowania znaków w korpusach historycznych*, w: *Semantyka a konfrontacja językowa*. Red. Danuta Roszko i Joanna Satoła-Staśkowiak. T. 5. Warszawa: Instytut Slawistyki PAN, s. 67–76. URL: https://core.ac.uk/display/77128852.

Bień, Janusz S. (2019). *Traktat Parkosza. Eksperymentalna edycja elektroniczna*. „Poznańskie Studia Polonistyczne. Seria Językoznawcza" nr 26(1), s. 27–69. DOI: 10.14746/pspsj. 2019.26.1.2. URL: http://pressto.amu.edu.pl/index.php/pspsj/article/view/19852.

Blewitt, Alex (2016). *A brief history of Unicode*, w: *The Docklands. JLC*. London. URL: https://www.infoq.com/presentations/unicode-history/.

Górski, Konrad, red. (1955). *Zasady wydawania tekstów staropolskich: projekt*. Wrocław: Zakład im. Ossolińskich – Wydawn. Polskiej Akademii Nauk.

Haralambous, Yannis (2002). *Unicode et typographie: un amour impossible*. „Document numérique" nr 6(3), s. 105–137. DOI: 10.3166/dn.6.3-4.105-137. URL: https://www.cairn.info/ revue-document-numerique-2002-3-page-105.htm (dostęp 25.08.2019).

Haralambous, Yannis (2007). *Fonts & Encodings. From Advanced Typography to Unicode and Everything in Between*. O'Reilly Media.

Haugen, Odd Einar (2013). *Dealing with glyphs and characters. Challenges in encoding medieval scripts*. „Document numérique" nr 16(3), s. 97–111. DOI: 10.3166/DN.16.3.97-111. URL: https://www.cairn.info/revue-documentnumerique-2013-3-page-97.htm (dostęp 29.08.2019).

Haugen, Odd Einar, red. (2015). *MUFI character recommendation version 4.0.* Medieval Unicode Font Initiative. ISBN: 978-82-8088-411-4. URL: http://hdl.handle.net/1956/10699 (dostęp 10.05.2019).

Hojdis, Bogdan (2016). *Problemy krojów pisma w edycji sejmowej dzieł Jana Kochanowskiego*. „Sztuka Edycji" nr 10(2), s. 7–14. ISSN: 2391-7903. DOI: 10.12775/SE.2016.013. URL: https://apcz.umk.pl/czasopisma/index.php/sztukaedycji/article/view/SE.2016.013 (dostęp 16.06.2019).

Jackowski, Bogusław, Piotr Pianowski i Piotr Strzelczyk (2018). *Kolekcja fontów TeX Gyre: kolejna odsłona*. „Acta Poligraphica" nr 12, s. 17–30. URL: http://www.cobrpp.com.pl/actapoligraphica/uploads/pdf/AP2018_2_Jackowski.pdf.

Kierkowicz, Magdalena (2010). *Zasady wydawania tekstów staropolskich wobec wiedzy o fonetyce historycznej i najnowszej praktyki wydawniczej – rekonesans*. „Podteksty" nr 20(2). ISSN: 1895-4901. URL: http://podteksty.amu.edu.pl/ (dostęp 04.10.2014).

Korpela, Jukka K. (2006). *Unicode Explained*. O'Reilly Media, Inc. ISBN: 978-81-8404-160-6.

Opaliński, Krzysztof (2007). *Problemy kodowania korpusów historycznych (na przykładzie tekstów XVI-wiecznych)*, w: *Z zagadnień leksykologii i leksykografii języków słowiańskich*. Red. Joanna Kamper-Warejko i Iwona Kaproń-Charzyńska. Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika, s. 107–114.

Pest, Monika (2015). *T&T, czyli jak typografia jest interfejsem tekstologii*. „Acta Poligraphica" nr 6, s. 71–84. ISSN: 2299-9981. URL: http://www.cobrpp.com.pl/actapoligraphica/uploads/pdf/AP2015_02_Pest.pdf.

Pike, Rob i Ken Thompson (1993). *Hello World or Καλημέρα κόσμε or こんにちは世界*, w: *Proceedings of the Winter 1993 USENIX Conference*. San Diego, s. 43–50. URL: http://doc.cat-v.org/plan_9/4th_edition/papers/utf.

Strzelczyk, Piotr (2013). *Standard Unicode w typografii*. „Acta Poligraphica" nr 1, s. 41–50. URL: http://www.cobrpp.com.pl/actapoligraphica/uploads/pdf/AP2013_ 01_ Strzelczyk.pdf.

Wieluńska, Anna (2018a). *Digitalizacja karakteru ukośnego Jana Januszowskiego – studium procesu*. „Acta Poligraphica" nr 12, s. 31–50. ISSN: 2299-9981. URL: http://www.cobrpp.com.pl/actapoligraphica/.

– (2018b). *Lazarus: Nowy Karakter Polski. Proces dygitalizacji i specimen*. Prac. dokt. Warszawa: Akademia Sztuk Pięknych. URL: https://lazarus.wtf/wp- content/uploads/2018/12/LAZARUS_BOOK_WEB.pdf.

Wikipedia contributors (2019a). *ISO/IEC 2022* – Wikipedia, The Free Encyclopedia. (Online; dostęp 22.05.2019). URL: https://en.wikipedia.org/w/index.php?title=ISO/IEC_2022&oldid=896826507.

Wikipedia contributors (2019b). *Private Use Areas* – Wikipedia, The Free Encyclopedia. (Online; dostęp 27.05.2019). URL: https://en.wikipedia.org/w/index.php?title=Private_Use_Areas&oldid=887603290.

Wikipedia contributors (2019c). *Universal Coded Character Set* – Wikipedia, The Free Encyclopedia. (Online; dostęp 22.05.2019). URL: https://en.wikipedia.org/w/index.php?title=Universal_Coded_ Character_Set&oldid=896597636.

Wikipedia contributors (2019d). *Variant form (Unicode)* – Wikipedia, The Free Encyclopedia. (Online; dostęp 25.05.2019). URL: https://en.wikipedia.org/w/index.php?title=Variant_form_(Unicode) &oldid=896012681.

Wikipedia contributors (2019e). *Variation Selectors (Unicode block)* – Wikipedia, The Free Encyclopedia. (Online; dostęp 25.05.2019). URL: https://en.wikipedia.org/w/index.php?title=Variation_Selectors_(Unicode_block)&oldid=891132820.

Abstract

The Unicode Standard and Polish language

A short overview of the standard, together with the relevant tools and resources, is presented from the historical Polish language perspective. The notions of code space, code point, character, combining sequence, graphemic cluster, variation selectors etc. are presented or at least mentioned. A relatively new notion of textel is briefly discussed. The 15th and 16th century treatises are used to illustrate some open problems. Various relevant Internet resources are mentioned, including those created by the author.