BOGUSŁAW JACKOWSKI, PIOTR PIANOWSKI, PIOTR STRZELCZYK
Polska Grupa Użytkowników Systemu TEX GUST

Kolekcja fontów TEX Gyre: kolejna odsłona*

Słowa kluczowe: *font, struktura fontów, repertuar znaków, OpenType, Metapost, FontForge*

1. Wstęp

Kolekcja fontów TEX Gyre (TG) została przygotowana przez zespół fontowy Polskiej Grupy Użytkowników Systemu TEX GUST w latach 2006 – 2009 jako dostępny nieodpłatnie zamiennik standardowej, komercyjnej kolekcji 35 fontów PostScript-owych (w formacie Type 1) firmy Adobe Systems Inc. Zestaw PostScript-owych fontów TEX Gyre został uzupełniony o fonty w formacie OpenType (OTF, [9]), utworzone z fontów PostScript-owych za pomocą narzędzia udostępnianego nieodpłatnie przez firmę Adobe Systems Inc., a obecnie dostępnego na licencji otwartej – *Adobe Font Development Kit for OpenType*, AFDKO [2].

Materiałem wyjściowym projektu TG była kolekcja fontów Type 1 firmy URW++ rozpowszechniana wraz z programem Ghostscript. Kolekcja ta była dostępna początkowo (od roku 1996) na zasadach *GNU General Public License* (GPL) i *Aladdin Free Public License* (AFPL), a potem (od roku 2009) także na zasadach *LATEX Project Public License* (LPPL).

Głównym celem projektu TG było uzyskanie metryk zgodnych z metrykami fontów firmy Adobe Systems Inc. [1] oraz rozszerzenie repertuaru znaków o znaki diakrytyczne wszystkich języków europejskich [6], [7]. Fonty TEX Gyre są udostępniane na specjalnej licencji GUST-u (*GUST Font License*, GFL [4]), będącej w istocie wariantem licencji LPPL. Po zrealizowaniu tego projektu zespół fontowy GUST-u rozpoczął prace nad przygotowaniem fontów matematycznych w formacie OpenType odpowiadających fontom szeryfowym kolekcji TEX Gyre. Dodatkowo został przygotowany wariant matematyczny fontu DejaVu, formalnie nienależącego do kolekcji TEX Gyre. Prace nad fontami matematycznymi zostały zakończone w roku 2016 [8].

Fonty matematyczne, oprócz symboli *stricte* matematycznych, niezbędnych przy składaniu formuł matematycznych (należą do nich np. warianty znaków o różnych rozmiarach czy znaki zestawiane z innych znaków fontu), zawierają bogaty repertuar „zwykłych" symboli – matematycznych i geometrycznych, przydatnych przy składaniu tekstów technicznych. Hans Hagen, autor opartego na TEX-u systemu ConTEXt, niestrudzenie podsuwający środowisku TEX-owemu inspirujące po-

* Niniejsza publikacja omawia zakończony w roku 2018 pierwszy etap najnowszego projektu fontowego Polskiej Grupy Użytkowników Systemu TEX GUST.

mysły (projekty TEX Gyre i TEX Gyre Math zostały w istocie przezeń zainicjowane), zaproponował, by symbole te przenieść do fontów tekstowych kolekcji TEX Gyre. Z pewną obawą, ale też z ochotą, podjęliśmy się tego zadania.

Pierwszym, niezbędnym krokiem było ustalenie, o jakie znaki należy rozszerzyć repertuar fontów tekstowych. Oczywiście niewielka początkowo lista kandydatów wciąż rosła. Ostatecznie zostało wytypowanych około 1000 znaków, głównie symboli matematycznych i geometrycznych. Na potrzeby fontów matematycznych przygotowane zostały symbole dla fontów szeryfowych prostych, niepogrubionych. Teraz trzeba się było zmierzyć z przygotowaniem dodatkowych znaków również dla fontów pogrubionych i bezszeryfowych (kolekcja TEX Gyre zawiera dwa fonty bezszeryfowe: TG Adventor i TG Heros – zamienniki odpowiednio fontów Avant Garde i Helvetica). Do kursyw zostały dodane znaki proste – nie widzieliśmy powodu by pochylać kwadraty, strzałki czy operatory matematyczne.

Przygotowanie niebagatelnej liczby dodatkowych znaków dla innych odmian pisma oznaczało solidne przetestowanie systemu MetaType1, używanego przez nas od lat do generowania fontów, początkowo w formacie PostScript-owym Type 1 na potrzeby użytkowników TEX-a [5]. Naszym zdaniem test wypadł pomyślnie.

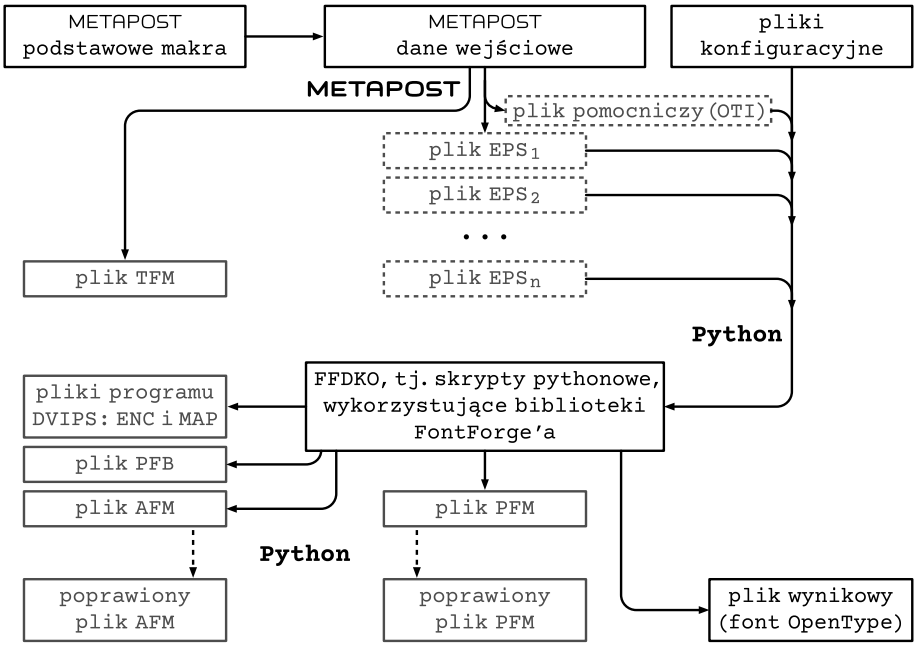
Projekt rozpoczęliśmy od rozszerzenia repertuaru fontów TG Adventor oraz TG Pagella (zamiennika fontu Palatino). Nowe fonty można pobrać z witryny internetowej GUST-u [6]. Zadanie, włącznie z modyfikacją systemu MetaType 1, zajęło nam niespełna rok. Doświadczenie nabyte w trakcie prac pozwala mieć nadzieję, że wzbogacenie pozostałych fontów kolekcji TEX Gyre zabierze znacznie mniej czasu (p. punkt 5).

W dalszej części artykułu opisujemy zasadnicze aspekty zrealizowanego projektu, zwracając uwagę na problemy, które okazały dla nas w tej fazie najtrudniejsze, a tym samym – najbardziej interesujące. Siłą rzeczy przedstawiamy punkt widzenia użytkowników systemu TEX, aczkolwiek główne zagadnienia omawiane w artykule – dotyczące repertuaru znaków w fontach i struktury fontów – mają charakter ogólny.

2. Nowy system MetaType1

Jednym z ważnych kroków przygotowawczych była modyfikacja systemu MetaType1 mająca na celu usprawnienie rozszerzenia serii fontów o tak dużą liczbę znaków. Schemat działania naszego nowego oprogramowania do generowania fontów przedstawia ilustracja 1.

Podstawową zmianą wprowadzoną w systemie MetaType 1 było zastąpienie komponentów różnego rodzaju (AWK, PERL, T1utils) Python-em, a dokładniej zestawem skryptów Python-owych wykorzystujących biblioteki programu FontForge. Pozwoliliśmy sobie nazwać tę część systemu *FontForge Development Kit for OpenType*, FFDKO – na wzór AFDKO [2]. Biblioteki Python-owe programu FontForge, dostępne zarówno dla systemu Linux, jak i systemu Windows, oferują szerokie możliwości, ale niestety nie pozwalają na pełną kontrolę zawartości tworzonych plików AFM i PFM. Dodatkowe modyfikacje, tworzące pliki zgodne z przyjętą ongiś przez nas specyfikacją, przeprowadzane są w osobnym kroku, zaznaczonym na ilustracji 1 strzałkami przerywanymi.



Ilustr. 1. Nowy system MetaType 1: schemat działania

W skład systemu MetaType 1, jako osobny moduł (nieuwidoczniony na ilustracji 1), wchodzi konwerter zamieniający PostScript-owe pliki fontowe w formacie Type 1 na dane wejściowe dla systemu MetaType 1. Moduł ten napisany jest w AWK-u i wykorzystuje deasembler (z zestawu programów T1utils) fontów Type 1. Planujemy zastąpienie go modułem Python-owym, rozszerzonym o możliwość konwersji fontów TrueType i OpenType.

Oczywiście rdzeniem całego systemu MetaType 1 pozostaje program Metapost – jest to nasze podstawowe narzędzie do tworzenia obrysów znaków. Istotną zmianą sposobu używania Metapost-a jest zrezygnowanie z zapisywania w sposób rozproszony dodatkowych informacji, które dotychczas były zapisywane i w plikach pomocniczych, i w komentarzach w plikach EPS. W zmodyfikowanej wersji systemu MetaType 1 tworzony jest – oprócz plików EPS – jeden plik pomocniczy, zawierający dodatkowe informacje niezbędne dla utworzenia fontów OpenType i Type 1. Nazwaliśmy ten plik, zgodnie z informatycznym obyczajem, z angielska: *Olio*** *Typographic Information file* (OTI). Plik OTI istotnie jest mieszanką „różnych różności"; poniżej zamieszczony jest przykładowy fragment pliku OTI dla odmiany prostej zwykłej fontu TG Pagella.

```
  FNT FAMILY_NAME TeX Gyre Pagella
1. FNT HEADER_BYTE49 TeX Gyre Pagella
2. FNT GROUP_NAME TeX Gyre Pagella
3. FNT STYLE_NAME Regular
```

** Olio to tradycyjna angielska nazwa potrawki po francusku, występująca np. w *Odzie do kaszanki* (*Address to a Haggis*) Roberta Burnsa: „French ragout or olio".

```
   . . .
4. FNT WEIGHT Regular
5. FNT ITALIC_ANGLE 0
   . . .
6. GLY A CODE 65
7. GLY A EPS 165
8. GLY A ANCHOR INBAS ALT.ogonek 623 -143
9. GLY A ANCHOR INBAS BOT_MAIN 392 -143
10. GLY A ANCHOR INBAS TOP_MAIN 392 819
11. GLY A WD 778 HT 692 DP 0 IC 6 GA 392
12. GLY A HSBW 778
13. GLY A BBX 15 -3 756 700
   . . .
14. FNT FONT_DIMEN7 0.83
15. FNT DIMEN_NAME7 (extra space)
16. FNT FONT_DIMEN22 2.5
17. FNT DIMEN_NAME22 (math axis)
18. FNT HEADER_BYTE72 234
```

Każdy wiersz pliku OTI zawiera informację bądź globalną, dotyczącą całego fontu (wiersze 1 – 5 i 14 – 18), bądź lokalną, dotyczącą konkretnego znaku (wiersze 6 – 13); w tym drugim przypadku nazwa znaku występuje bezpośrednio po znaczniku GLY.

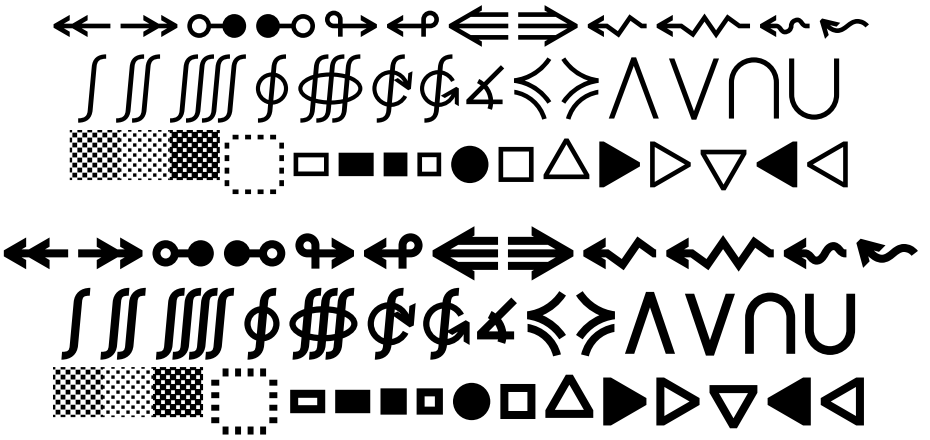
Nie będziemy się zagłębiać w szczegóły specyfikacji pliku OTI – i tak omawiamy wystarczająco dużo kwestii technicznych jak na artykuł o charakterze przeglądowym. Odnotujemy jedynie, że informacje dotyczące całego fontu zawierają zarówno dane przeznaczone do tworzenia plików OpenType i Type 1, jak i dane przeznaczone do tworzenia TEX-owego pliku metrycznego TFM. Informacje dotyczące poszczególnych znaków zawierają informacje metryczne (wiersze 11 – 13) oraz informacje związane ze strukturą plików OpenType (wiersze 8 – 10; w tym przypadku są to informacje dotyczące tzw. kotwic – p. punkt 4), a ponadto informacje uzupełniające (wiersze 6 – 7).

3. Repertuar znaków

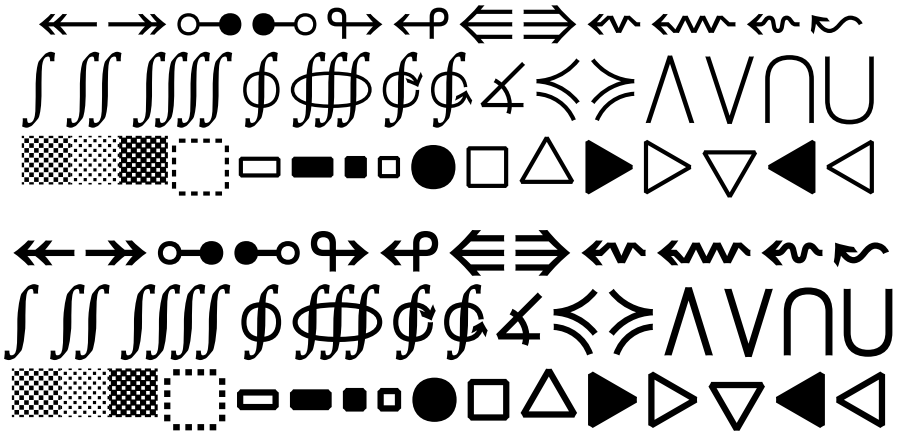
Jak wspomnieliśmy, głównym bodźcem do podjęcia prac nad rozszerzeniem repertuaru znaków w fontach z kolekcji TEX Gyre były rezultaty naszych prac nad fontami matematycznymi: wiele symboli może być stosowanych w fontach „niematematycznych", tj. w fontach niezawierających specjalnej OpenType-owej tablicy MATH. Ponadto, co jest istotne, znaki takie jak na przykład symbole matematyczne (operatory, znaki relacji), strzałki czy symbole geometryczne mogą być przydatne do składania tekstów technicznych. Ilustracje 2 i 3 przedstawiają niektóre spośród dodanych znaków.

Liczba znaków w fontach wzrosła od około 700 do około 1600. Nie jest wykluczone, że liczba dodanych znaków jeszcze się zwiększy, aczkolwiek nie jesteśmy entuzjastami tej idei (p. punkt 5).

Przy okazji rozszerzenia repertuaru znaków niektóre z nich, już obecne w fontach, zostały poprawione, dzięki między innymi zastosowaniu programu FontForge, który automatycznie dokonuje starannego sprawdzenia fontu. Na przykład tylda w foncie TG Adventor została narysowana od nowa, osie optyczne niektórych znaków zostały skorygowane itp. Znaki przeznaczone zasadniczo do skła-



Ilustr. 2. Przykłady dodanych znaków: font TG Adventor zwykły (góra) i pogrubiony (dół)



Ilustr. 3. Przykłady dodanych znaków: font TG Pagella zwykły (góra) i pogrubiony (dół)

du matematyki zostały nieco zmodyfikowane tak, by lepiej wyglądały w formułach matematycznych – p. ilustracja 4. Znaki obecne uprzednio w foncie zostały zachowane. Na dostęp do nich pozwala mechanizm podmiany znaków oferowany przez format OpenType. Fonty OpenType-owe zawierają specjalne tablice (przypomnijmy, że tablice są podstawową strukturą fontów OpenType-owych), o dość niefortunnej nazwie *feature* (ang. *aspekt*, *cecha*, *właściwość*), których interpretowanie programy korzystające z fontów mogą włączać i wyłączać na życzenie użytkownika. W polskim języku używane bywa niemające nic wspólnego ze słowem *feature* określenie *funkcja zecerska*; lepszym

$$f(x) = 1/x \qquad f(x) = 1/x$$
$$(x+1)(x-3) \geq 0 \qquad (x+1)(x-3) \geq 0$$

Ilustr. 4. Domyślne i stare formy znaków fontu TG Pagella, przeznaczonych do składania matematyki (odpowiednio po lewej i po prawej stronie; te drugie uzyskuje się przez włączenie interpretowania tablicy `feature ss10`)

określeniem wydaje się *wytyczna*, gdyż zachowanie programów używających tablic *feature* nie jest precyzyjnie określone, specyfikacja (p. [3], [9], [11]) podaje jedynie wytyczne, jak program powinien się zachowywać, o ile umie zinterpretować daną tablicę (w przeciwnym wypadku powinien ją zignorować). Dostęp do poprzednich form znaków w fontach TEX Gyre uzyskuje się za pomocą włączenia interpretowania tablicy `feature ss10` (dalej w skrócie: `ss10`) z grupy *wytycznych stylistycznych* (*stylistic set*, tablice `ss01`–`ss20`).

Ponadto cały alfabet grecki w foncie TG Pagella został zamieniony, wzorem fontu matematycznego, na alfabet zaczerpnięty ze znakomicie przygotowanego fontu Mathpazo, będącego, podobnie jak TG Pagella, zamiennikiem fontu Palatino. Diego Puza, autor fontu Mathpazo, uprzejmie zgodził się na użycie fragmentu jego fontu na licencji GFL [4]. Również niektóre znaki alfabetu greckiego w foncie TG Adventor (wstępnie zaprogramowane przez Janusza M. Nowackiego) zostały skorygowane.

Poprawki te, aczkolwiek uzasadnione, wprowadziły istotne zmiany w metrykach znaków. Zmiany metryk tolerujemy niechętnie – takie modyfikacje, nawet drobne, mogą spowodować zupełnie inne przełamanie tekstu po zmianie wersji fontu. Jednakże „idąc za ciosem" zdecydowaliśmy się zrezygnować z pełnej zgodności z metrykami 35 standardowych fontów Type 1 firmy Adobe Systems Inc. [1] z dwóch powodów:

1. Dane metryczne tych fontów, jak podkreślaliśmy w dokumentacji fontów TEX Gyre [6], są niespójne w kilku przypadkach.
2. Zachowanie pełnej zgodności miałoby sens jedynie wówczas, gdyby odpowiednie fonty PostScript-owe były używane do podglądu dokumentów przeznaczonych do drukowania na drukarkach z wbudowanymi fontami firmy Adobe Systems Inc. – fonty TEX Gyre w zasadzie nadawały się do tego celu. Jednakże w dwóch najważniejszych pakietach rozpowszechnianych na otwartych licencjach (Ghostscript i TEX Live), wykorzystujących fonty Type 1, używane są fonty firmy URW++.

Z tych względów wprowadziliśmy kilka drobnych zmian w danych metrycznych, mając nadzieję, że w przyszłości uda się nam uniknąć tego rodzaju poprawek.

4. Struktura fontu

Oprócz rozszerzenia struktury fontów TEX Gyre o omówioną w punkcie poprzednim tablicę `ss10`, rozbudowaliśmy fonty o dodatkowe tablice (również typu *feature*) związane z tzw. *kotwicami* (ang. *anchors*), służące do tworzenia znaków złożonych, np. znaków akcentowanych. Wyraźnie projektanci tego udogodnienianie nie byli zdecydowani, jeśli chodzi o nazewnictwo – w opisie składni języka do definiowania tablic *feature* [3] występuje nazwa *anchor*, a w dokumentacji zarejestrowanych tablic *feature* [11] odwołujących się do *kotwic* używane jest wieloznaczne określenie *mark* (*znak*, *znaczek*, *kreska*, *oznaczenie*, *ocena* i in.). Nam się ten mechanizm bardziej kojarzy z przypinaniem elementów odzieży za pomocą zatrzasków niż z zarzucaniem kotwicy czy stawianiem znaczków. Nie wdając się w spory nomenklaturowe, w dalszym ciągu będziemy używali określenia *kotwica*.

Specyfikacja *Unicode Standard* zaleca, by procesor tekstów przetwarzając strumień znaków o przypisanych numerach unikodowych, po których występuje seria akcentów, nakładał na dany znak te akcenty, o ile stosowna informacja o pozycjonowaniu (kotwice) znajduje się w foncie [13].

Zdawać by się mogło, że operacja umieszczenia akcentu nad czy pod literą jest zabiegiem prostym. Tak w każdym razie nam się wydawało, gdy rozpoczynaliśmy pracę nad implementacją kotwic. Kotwice to w istocie pary liczb rzeczywistych, czyli punkty na płaszczyźnie, przypisane znakom. Tworzenie znaków złożonych polega na umieszczeniu znaków składowych (akcentu i znaku akcentowanego lub akcentu i akcentu) w taki sposób, by odpowiednie kotwice nałożyły się na siebie. Ku naszemu zdziwieniu zagadnienie okazało się stosunkowo złożone, a zaimplementowanie kotwic – pracochłonne, mimo iż przedmiotem naszych zainteresowań były jedynie znaki o proweniencji łacińskiej. Na dobitkę mechanizm kotwic, mimo skomplikowania, okazał się, naszym zdaniem, niewystarczająco uniwersalny.

Do pozycjonowania akcentów za pomocą kotwic w językach posługujących się alfabetem łacińskim służą trzy tablice typu *feature*, opisane w dokumentacjach [3] i [11] (niestety niezbyt jasno):

1. `ccmp` – *glyph composition / decomposition*, tablica służąca do kompozycji i dekompozycji znaków;
2. `mark` – *mark positioning*, a dokładniej *accent-to-base positioning*, tablica służąca do pozycjonowania akcentów nad lub pod literami alfabetu, zwanymi znakami bazowymi;
3. `mkmk` – *mark-to-mark positioning*, tablica służąca do pozycjonowania akcentów nad lub pod akcentami.

Interpretowanie tych tablic jest włączane w stosownym momencie przez program składający tekst danym fontem. Znaki ze strumienia wejściowego przetwarzane są kolejno i po natrafieniu na akcent (umieszczany za znakiem akcentowanym – p. niżej) program powinien rozpocząć procedurę akcentowania znaku poprzedzającego akcent. W trakcie tej procedury używane są właśnie tablice `ccmp`, `mark` i `mkmk`.

W fontach TEX Gyre, zgodnie z powszechnie przyjętą praktyką, spośród akcentów z przypisanymi numerami unikodowymi w kotwice wyposażyliśmy jedynie tzw. *combining accents* (*akcenty przyłączane*), stanowiące podzbiór przyłączanych znaków diakrytycznych [12]. W dalszym ciągu mówiąc o akcentach będziemy mieli na myśli akcenty przyłączane. Znaki przyłączane mają zerową szerokość i są wysunięte całkowicie w lewą stronę względem metrycznego początku znaku. Akcenty przyłączane niewyposażone w kotwice jedynie w bardzo ograniczonym stopniu mogą służyć do budowania znaków akcentowanych poprzez umieszczenie znaku akcentu za znakiem akcentowanym. Kotwice poprawiają jakość akcentowania w sposób istotny, ponadto umożliwiają budowanie akcentów piętrowych.

Jeżeli chodzi o znaki akcentowane, to – dla zredukowania objętości danych opisujących kotwice – w kotwice zostały wyposażone jedynie litery alfabetu łacińskiego bez znaków diakrytycznych, litery zawierające obwiednie połączone z niektórymi akcentami (*cedilla*, *horn*, *ogonek*), a ponadto litery *ĺ*, *Ľ*, *ł*, *Ł*, *ø* oraz *Ø*.

Gdy zachodzi potrzeba dołożenia akcentu do znaku już zawierającego akcenty, najpierw ten znak jest zamieniany na elementy składowe (akcenty i znak bazowy), które są włączane do strumienia wejściowego i dalej są przetwarzane zgodnie z regułami akcentowania za pomocą kotwic. Do tego między innymi służy tablica `ccmp`, która pozwala na tworzenie znaków złożonych z kilku znaków następujących po sobie, a także na zastępowanie pojedynczego znaku serią znaków.

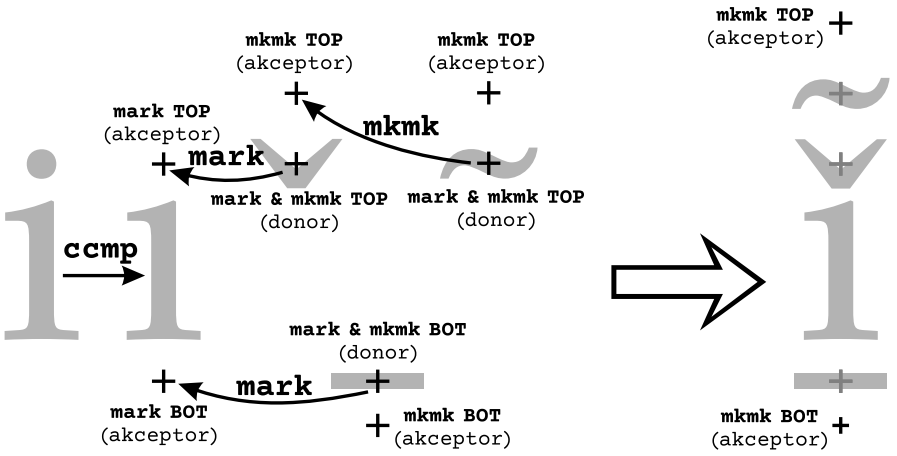
Nie wszystkie znaki akcentowane w fontach TEX Gyre są zamieniane na elementy składowe w trakcie akcentowania. Nie podlegają rozłożeniu znaki zawierające akcenty *cedilla*, *horn* i *ogonek*. Tablica `ccmp` w fontach TEX Gyre służy także do łączenia w pojedynczy znak liter z występującym po niej jednym z akcentów *cedilla*, *horn* i *ogonek* (o ile znak ten jest obecny w foncie; w przeciwnym razie używane są kotwice). Ponadto tablica ta używana jest do zastępowania liter *i* oraz *j*, po których występują akcenty górne, formami bezkropkowymi (ang. *dotless*) – *ɪ* oraz *ȷ*, zaś akcenty górne występujące po dużych literach zamieniane są na formy spłaszczone, które bardziej pasują do dużych liter.

Odpowiednie zamiany powinny być w zasadzie stosowane jedynie wówczas, gdy są zdefiniowane w foncie, ale niektóre programy korzystające z fontów „wiedzą lepiej" i stosują takie zamiany nawet wówczas, gdy font nie zawiera stosownych informacji. Na przykład Microsoft Word literę *i* (U+0069) wraz z następującym po niej akcentem górnym, niech to będzie przykładowo *caroncomb* (U+030C), zastąpi pojedynczym znakiem, w tym przypadku znakiem *icaron* (U+01D0), o ile znak ten jest obecny w foncie; żadna dodatkowa informacja, w szczególności tablica `ccmp`, nie jest potrzebna. Podobnie działa XƎTEX, odmiana TEX-a przeznaczona do składania wielojęzycznych tekstów. Opisanego sposobu działania nie da się wyłączyć.

Nakładanie akcentów z użyciem kotwic, pozornie trywialne – jako się rzekło – zadanie, wymaga w istocie serii nie całkiem oczywistych operacji ze strony programu korzystającego z fontu. Typowy przykład zastosowania mechanizmu kotwic z użyciem tablic `ccmp`, `mark` oraz `mkmk` przedstawia ilustracja 5.

Na rysunku krzyżykami oznaczone są kotwice, etykiety `ccmp`, `mark` oraz `mkmk` zapisane mniejszymi literami oznaczają nazwy tablic, które odwołują się do danej kotwicy, zaś te same etykiety zapisane większymi literami oznaczają operacje, jakie z użyciem danych tablic powinien wykonać program składający tekst danym fontem. Etykiety TOP and BOT są przykładowymi nazwami nadanymi przez twórcę fontu (ang. *top* – górny i *bottom* – dolny), wykorzystywanymi w procesie akcentowania (p. niżej). Każda kotwica jest scharakteryzowana nieformalnie za pomocą podanych w nawiasach określeń *donor* i *akceptor*, zaczerpniętych z chemii fizycznej. W foncie kotwice donorowe i akceptorowe są odróżniane – tylko znak z kotwicą donorową może być pozycjonowany na znaku z kotwicą akceptorową o tej samej etykiecie.

W przykładzie zakładamy, że strumień wejściowy składa się z litery, po której występują trzy akcenty: *i*, *macronbelowcomb*, *caroncomb*, *tildecomb* o numerach unikodowych odpowiednio U+0069, U+030C, U+0331, U+0303. Przy tych założeniach akcentowanie przebiega następująco:



Ilustr. 5. Schemat użycia kotwic – przykład (objaśnienie w tekście)

1. najpierw program odwołuje się do tablicy `ccmp`: litera *i*, po której występuje akcent górny, jest zamieniana na znak *dotlessi* (U+0131);
2. następnie wykorzystywana jest tablica `mark`: znak *caroncomb* jest umieszczany nad znakiem *dotlessi* w taki sposób, że jego kotwica z etykietą TOP nakłada się na kotwicę znaku *dotlessi* o takiej samej etykiecie; obie kotwice stają się nieaktywne;
3. następnie ponownie wykorzystywana jest tablica `mark`: znak *macronbelowcomb* jest umieszczony poniżej znaku *dotlessi*, tym razem kotwica BOT akcentu jest nakładana na kotwicę BOT znaku *dotlessi*; obie kotwice stają się nieaktywne;
4. na koniec wykorzystywana jest tablica `mkmk`: akcent *tildecomb* jest umieszczany nad świeżo położonym akcentem *caroncomb* w taki sposób, by kotwice z etykietami TOP nałożyły się na siebie; obie kotwice stają się nieaktywne;
5. ostatecznie powstaje znak złożony mający dwie aktywne (akceptorowe) kotwice – górną TOP i dolną BOT, które mogą być wykorzystane przez tablicę `mkmk`, o ile odpowiednie znaki pojawią się w strumieniu wejściowym (w tym przypadku bezpośrednio po znaku *tildecomb*).

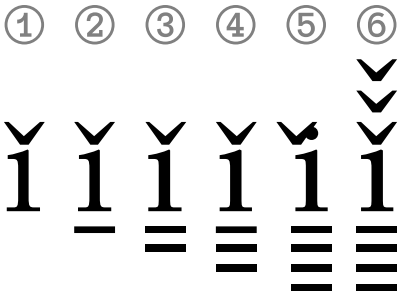
Proces nakładania akcentów jest więc całkiem zawiły, nawet nie tyle sam algorytm jest skomplikowany, co zadanie specyfikacji kotwic (zważywszy że znaki mogą mieć po wiele kotwic donorowych i akceptorowych). Należy jednak przyznać, że mechanizm ten umożliwia obsługę takich nietypowych przypadków, jak akcentowanie liter *l*, *L* czy *J* za pomocą akcentu *caron*, co wymaga zmiany kształtu akcentu i umieszczenia obok znaku akcento-

```
Ľ   L%
    ˇ% U+030C (caroncmb, caroncomb)
ģ   g%
    ,% U+0326 (uni0326, commaaccentcomb)
ỵ   y%
    .% U+0323 (uni0323, dotbelowcomb)
```

Ilustr. 6. Przykłady nietypowego pozycjonowania akcentów: lewa strona – wynik składu, prawa strona – źródło (strumień wejściowy w postaci danych do programu TEX, tekst od znaku % do końca wiersza jest komentarzem)

wanego zamiast ponad znakiem, umieszczenie obróconego akcentu *commaaccent* (akcent przecinkowy) nad literą *g*, normalnie umieszczanego pod literą, czy nietypowe pozycjonowanie kropki dolnej pod literą *y*. Przypadki te zostały przedstawione na ilustracji 6.

Niestety nie wszystkie ważne w praktyce przypadki są obsługiwane w sposób niezawodny. Znamiennym przypadkiem jest zastępowanie znaków *i* oraz *j* formami bezkropkowymi – wynik zależy w znacznym stopniu od kolejności znaków w strumieniu wejściowym. Ilustracja 7 pokazuje taką sytuację. Załóżmy, że mamy sześć ciągów wejściowych różnej długości (*c* oznacza tu *caroncomb*, *m* – *macronbelowcomb*, *i* – literę *i*), mianowicie: 1. *ic*; 2. *imc*; 3. *immc*; 4. *immmc*; 5. *immmmc*; 6. *icccmmmm*. W jednym z przypadków akcent *caron* jest błędnie pozycjonowany. To skutek przyjętych przez nas ograniczeń. Zamiana *i→dotlessi* jest wykonywana jedynie wówczas, gdy akcenty górne nie pojawiają się zbyt daleko za literą *i* w strumieniu wejściowym, a najlepiej – bezpośrednio za nią. Format OpenType dopuszcza kontekstowe zamiany, które w tym przypadku są wykorzystywane, a które pozwalają wyrazić polecenie zamiany litery *i* na formę bezkropkową, jeżeli akcent górny pojawi się nie dalej niż po k o n k r e t n e j liczbie akcentów dolnych. W fontach TEX Gyre ograniczyliśmy liczbę akcentów dolnych do trzech (przypadek 4 na ilustracji 7). Jeżeli między literą a akcentem górnym pojawią się więcej niż trzy akcenty dolne, zamiana na formę bezkropkową nie zostanie dokonana i znaki zostaną po prostu na siebie nałożone, gdyż litera *i* nie jest wyposażona w kotwice (przypadek 5 na ilustracji 7). Niektóre fonty dopuszczają dłuższe ciągi akcentów dolnych (na przykład Charis SIL). Uznaliśmy jednak, że do celów praktycznych wystarczy dopuścić ciągi trzyznakowe. Zainteresowanych szczegółami implementacji odsyłamy do plików `*.fea`, znajdujących się w opublikowanych pakietach fontów TG Adventor i TG Pagella [6]. Pakiety te zawierają użyte w fontach tablice typu *feature* (zapisane w języku wyspecyfikowanym w [3]).



Ilustr. 7. Kłopotliwa zamiana *i→dotlessi* (objaśnienia w tekście)

W celu uniknięcia tego rodzaju kłopotliwych sytuacji chciałoby się zalecić odpowiednią kolejność akcentów w strumieniu: najpierw górne, potem pozostałe (przypadek 6 w powyższym przykładzie). Jest wszakże problem z takim zaleceniem: proponowana kolejność znaków w strumieniu wejściowym może zostać zmieniona przez program przetwarzający strumień, a nawet powinna! Zgodnie bowiem z zaleceniem zawartym w specyfikacji *Unicode Standard* [14] najpierw powinny być przetwarzane akcenty dolne. Ponadto ten kanoniczny porządek nie może zostać zmieniony przez protokoły nadrzędne (*canonical ordering behavior cannot be overridden by higher-level protocols*). Nie ma więc gwarancji, że jakiś program, w którym jest zaimplementowany algorytm „kanonicznego porządkowania" strumienia wejściowego, nie spłata nam figla i nie zepsuje starannie przygotowanej kolejności.

W nowoczesnych implementacjach systemu TEX, obsługujących fonty OpenType-owe, interpretowanie tablic takich jak `ccmp`, `mark` czy `mkmk` może być włączane i wyłączane przez użytkownika (nawiasem mówiąc, ilustracja 7 została przygotowana za pomocą programu LuaTEX). Jednakże nie wszystkie edytory tekstów oferują tę możliwość. Znamiennym przykładem jest Microsoft Word, w którym interpretowanie tych tablic jest włączone na stałe. Nie jest jasne, czy Microsoft Word korzysta z zalecanego przez specyfikację *Unicode Standard* algorytmu „kanonicznego porządkowania". Dodatkowo użytkownikom może skomplikować życie fakt, że – jak wspomnieliśmy wyżej – niektóre programy (w tej liczbie Microsoft Word i XETEX) dość swobodnie traktują reguły zapisane w tablicach `ccmp`, `mark` i `mkmk`.

Naszym zdaniem kłopoty i komplikacje związane z implementacją i stosowaniem mechanizmu kotwic wynikają ze zbyt prymitywnych środków, jakie oferuje specyfikacja formatu OpenType: jedynymi dopuszczalnymi operacjami są repozycjonowanie i podmiana znaków. Pozostaje to w ścisłym związku z tablicową strukturą fontów OpenType-owych. Otóż podstawowe tablice, w których operacje te są zapisane, to odpowiednio `GPOS` i `GSUB`. Repozycjonowanie jest ograniczone do przesunięcia (obroty, choćby o wielokrotność 90 stopni, czy odbicie lustrzane nie wchodzą w rachubę). Podmiany zaś ograniczone są do zamian jednego znaku na jeden znak, wielu znaków na jeden znak i jednego znaku na wiele znaków (co wyklucza zmianę kolejności). Podmiany mogą być bądź „zwykłe" (tylko znaki biorące udział w podmianie występują w regułach podmian), bądź – jak wspomnieliśmy przy omawianiu kłopotów z akcentami – kontekstowe. Podmiany kontekstowe nie poprawiają w sposób istotny funkcjonalności, a za to skutecznie utrudniają panowanie nad danymi i konstrukcję stosownych tablic.

Format OpenType to w istocie nie język programowania, a język danych. Dlatego nie można się spodziewać po nim zbyt wiele. Chciałoby się jednakże, na przykład, móc używać w tablicy `GSUB` prastarych osiągnięć informatyki, takich jak wyrażenia regułowe (zwane też wyrażeniami regularnymi), bądź w tablicy `GPOS` – przekształceń afinicznych. Niestety, specyfikacja formatu OpenType nie zezwala na takie fanaberie.

5. Plany na przyszłość

Kolejnym krokiem będzie analogiczna modyfikacja pozostałych fontów kolekcji TEX Gyre: zarówno szeryfowych, jak i bezszeryfowych, mianowicie: TG Heros, TG Bonum, TG Cursor, TG Schola, TG Termes. Z projektu wyłączyliśmy fonty TG Cursor i TG Chorus – zamienniki fontów Courier i Zapf Chancery. O ile rozszerzanie kancelareski TG Chorus o symbole matematyczne i geometryczne wydaje się nam nieuzasadnione, o tyle font TG Cursor jest po prostu kłopotliwy ze względu na stałą metrykę, a niektóre symbole geometryczne (np. strzałki) mają nietypową szerokość; być może w przypadku TG Cursora rozwiązaniem jest ograniczenie zestawu dodatkowych znaków.

Rozważamy wprowadzenie nazw dla tablic opisujących odmiany stylistyczne (w fontach TEX Gyre zaimplementowane zostały tablice `ss01`–`ss04` oraz – ostat-

nio wprowadzona – tablica `ss10`, p. punkt 3). Taki krok wymaga namysłu – niewłaściwie dobrane nazwy mogą zamiast porządku wprowadzić zamieszanie.

Po zakończeniu prac nad fontami tekstowymi kolekcji TEX Gyre chcielibyśmy powrócić do fontów matematycznych – chcemy się staranniej przyjrzeć odsadkom, które są bardzo ważnym czynnikiem wpływającym na wygląd formuł, a ponadto zamierzamy wprowadzić specjalne *podcięcia matematyczne* (ang. *math kerns*) poprawiające jakość składu formuł matematycznych, dostępne w fontach zawierających tablicę `MATH` [10].

Nie mamy jasności w kwestii „małych cyfr". W fontach OpenType może się pojawić wiele rodzajów zmniejszonych znaków: frakcje dolne (*subscripts*), frakcje górne (*superscripts*), frakcje dolne naukowe (*scientific inferiors*), liczniki (*numerators*) i mianowniki (*denominators*); każdy z tych rodzajów znaków dostępny jest poprzez tablice typu *feature*, odpowiednio `subs`, `sups`, `sinf`, `numr` i `dnom`. W fontach matematycznych dostępne są specjalne frakcje dolne i górne (zaprojektowane jako znaki normalnej wielkości i zmniejszane przez program składający formuły).

Jeżeli byśmy dodali wymienione znaki, to pojawią się nastepne pytania: czy do kapitalików (tablica `smcp`) potrzebujemy specjalnych zmniejszonych cyfr? (W świecie TEX-owym z kapitalikami tradycyjnie używane są cyfry nautyczne). Czy cyfry zmniejszone powinny, wzorem cyfr normalnej wielkości, mieć odmianę nautyczną (*old-style figures*), wyrównaną (*lining figures*), proporcjonalną (*proportional figures*) i tabelaryczną (*tabular figures*)? Odmiany te są dostępne poprzez odpowiednio tablice `onum`, `lnum`, `pnum` i `tnum`. Powściągliwie odnosimy się do tych pomysłów. Wydaje się nam, że przeładowywanie fontu tego rodzaju znakami jest niezbyt sensowne – jest bowiem, naszym zdaniem, dokładanie kolejnych warstw złożoności (by nie powiedzieć galimatiasu) do skomplikowania wynikającego ze specyfikacji fontów OpenType-owych.

Oczywistym, niejako rutynowym, zadaniem będzie czyszczenie systemu MetaType 1, zarówno źródeł Python-owych, jak i Metapost-owych. Planujemy ponadto, jak wspomnieliśmy w punkcie 2, zaprogramowanie rozszerzenia modułu FFDKO o funkcję zamieniającą fonty Type 1, TrueType i OpenType na dane wejściowe systemu MetaType 1.

6. Podziękowania

Jesteśmy ogromnie wdzięczni wszystkim, którzy wspierali i wspierają nasze przedsięwzięcia fontowe. Niemal wszystkie projekty fontowe GUST-u były wspierane przez: Czeską Grupę Użytkowników TEX-a CS TUG, Niemieckojęzyczną Grupę Użytkowników TEX-a DANTE e.V., Polską Grupę Użytkowników Systemu TEX GUST, Holenderskojęzyczną Grupę Użytkowników TEX-a NTG, TUG India, UK-TUG, oraz TUG – najdłużej działającą grupę TEX-ową z USA. W kilku projektach uzyskaliśmy wsparcie również od Francuskojęzycznej Grupy Użytkowników TEX-a GUTenberg.

Wyjątkowe, osobiste podziękowania kierujemy do przyjaciół, którzy od wielu lat niezmiennie zachęcają nas do kontynuowania prac fontowych – do Hansa Hagena, Johannesa Küstera, Jurka Ludwichowskiego, Volkera RW Schaa, Joli Szelatyńskiej i Ulrika Vietha. Wielkie dzięki!

Wszystkie znaki towarowe zostały użyte w niniejszym artykule jedynie w celach informacyjnych.

Cytowana literatura

1. Adobe Systems Inc., *Adobe Metric Files*; ftp://ftp.adobe.com/pub/adobe/type/win/all/afmfiles/base35/ [dostęp 12.11.2018].
2. Adobe Systems Inc., *Adobe Font Development Kit for OpenType (AFDKO)*; https://pypi.org/project/afdko/ [dostęp 12.11.2018].
3. Adobe Systems Inc., *Feature file syntax*; https://www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html [dostęp 12.11.2018].
4. GUST e-Foundry, *GUST Font License*; http://www.gust.org.pl/projects/e-foundry/licenses [dostęp 12.11.2018].
5. Bogusław Jackowski and Janusz M. Nowacki, *Programming PostScript Type 1 Fonts Using MetaType 1: Auditing, Enhancing, Creating*, Proc. of 14th EuroTEX, June 24th – 27th 2003, Brest, France, p. 151 – 157; https://www.tug.org/TUGboat/tb24-3/jackowski.pdf [dostęp 12.11.2018].
6. Bogusław Jackowski, Janusz M. Nowacki, and Piotr Strzelczyk *TEX Gyre fonts collection*; http://www.gust.org.pl/projects/e-foundry/tex-gyre TG Adventor: http://www.gust.org.pl/projects/e-foundry/tex-gyre/adventor TG Pagella: http://www.gust.org.pl/projects/e-foundry/tex-gyre/pagella [dostęp 12.11.2018].
7. Bogusław Jackowski, Piotr Strzelczyk, and Piotr Pianowski, *GUST e-foundry font projects*, TUGboat, 2016, vol. 37(3), 317–336; http://tug.org/TUGboat/tb37-3/tb117jackowski.pdf [dostęp 12.11.2018].
8. Bogusław Jackowski, Piotr Strzelczyk, and Piotr Pianowski, *TEX Gyre math fonts collection*; http://www.gust.org.pl/projects/e-foundry/tg-math [dostęp 12.11.2018].
9. Microsoft Corp., *OpenType Font Format*, ver. 1.8.3, ISO/IEC 14496-22, 4th edition; https://docs.microsoft.com/en-us/typography/opentype/spec/ [dostęp 12.11.2018].
10. Microsoft Corp., *MATH – The mathematical typesetting table*; https://docs.microsoft.com/en-us/typography/opentype/spec/math [dostęp 12.11.2018].
11. Microsoft Corp., *Registered features*; https://docs.microsoft.com/en-us/typography/opentype/spec/featurelist [dostęp 12.11.2018].
12. Unicode Consortium, *Combining Diacritical Marks*; http://unicode.org/charts/PDF/U0300.pdf [dostęp 12.11.2018].
13. Unicode Consortium, *The Unicode Standard 10.0.0; chapters: 2.3 Compatibility Characters, 2.11 Combining Characters, 2.12 Equivalent Sequences and Normalization*; http://unicode.org/versions/Unicode10.0.0/ch02.pdf [dostęp 12.11.2018].
14. Unicode Consortium, *The Unicode Standard 10.0.0; chapter 3.11 Normalization Forms*; http://unicode.org/versions/Unicode10.0.0/ch03.pdf [dostęp 12.11.2018].

Abstract

The collection of TEX Gyre fonts: next edition

The TEX Gyre free collection of fonts [15] (released under the *GUST Font License* [4]) was created from 2006 to 2009 as a free replacement for the renowned

standard set of the 35 Type 1 fonts by Adobe Systems Inc. The article concerns the first phase of a recent GUST e-foundry project of enhancing the text fonts of the TEX Gyre collection; it describes the project's essential, most difficult and thus, presumably, most interesting aspects.

This "face-lifting" of the text TEX Gyre fonts is mainly rooted in the math part of those fonts. Many symbols contained therein do not need the mathematical extension of the font structure, i.e., the MATH table in OTF files [10], but still prove useful for typesetting technical texts; these, e.g., are mathematical symbols (operators, relational symbols), arrows, geometrical symbols, etc. Such symbols can be used in usual text fonts (without the MATH table).

The main goal was to include selected math-oriented symbols into the TEX Gyre text fonts (circa 1000 characters were selected to be included). Apart from that, the structure of the fonts was enhanced by adding the so called anchor mechanism. So far, two fonts, TEX Gyre Pagella (the replacement for Palatino) and TEX Gyre Adventor (the replacement for Avant Garde), are released [6].

Additionally, for the purpose the project, our MetaType 1 engine for generating fonts was improved – it is now completely based on Metapost and Python with FontForge libraries.