Numero 7
Aprile 2009

ArsTEXnica

Rivista italiana di TEX e LATEX

GUIT

http://www.guit.sssup.it/arstexnica

GUIT – Gruppo Utilizzatori Italiani di TEX

ArsTEXnica è la pubblicazione ufficiale del GUIT

ArsTEXnica è la prima rivista italiana dedicata a TEX, a LATEX ed alla tipografia digitale. Lo scopo che la rivista si prefigge è quello di diventare uno dei principali canali italiani di diffusione di informazioni e conoscenze sul programma ideato quasi trent'anni fa da Donald Knuth.

Le uscite avranno, almeno inizialmente, cadenza semestrale e verranno pubblicate nei mesi di Aprile e Ottobre. In particolare, la seconda uscita dell'anno conterrà gli Atti del Convegno Annuale del GUIT, che si tiene in quel periodo.

La rivista è aperta al contributo di tutti coloro che vogliano partecipare con un proprio articolo. Questo dovrà essere inviato alla redazione di ArsTEXnica, per essere sottoposto alla valutazione di recensori. È necessario che gli autori utilizzino la classe di documento ufficiale della rivista; l'autore troverà raccomandazioni e istruzioni più dettagliate all'interno del file di esempio (`.tex`). Tutto il materiale è reperibile all'indirizzo web della rivista.

Gli articoli potranno trattare di qualsiasi argomento inerente al mondo di TEX e LATEX e non dovranno necessariamente essere indirizzati ad un pubblico esperto. In particolare tutorials, rassegne e analisi comparate di pacchetti di uso comune, studi di applicazioni reali, saranno bene accetti, così come articoli riguardanti l'interazione con altre tecnologie correlate.

Di volta in volta verrà fissato, e reso pubblico sulla pagina web, un termine di scadenza per la presentazione degli articoli da pubblicare nel numero in preparazione della rivista. Tuttavia gli articoli potranno essere inviati in qualsiasi momento e troveranno collocazione, eventualmente, nei numeri seguenti.

Chiunque, poi, volesse collaborare con la rivista a qualsiasi titolo (recensore, revisore di bozze, grafico, etc.) può contattare la redazione all'indirizzo:

`arstexnica@sssup.it`.

Nota sul Copyright

Associarsi a GUIT

Fornire il tuo contributo a quest'iniziativa come membro, e non solo come semplice utente, è un presupposto fondamentale per aiutare la diffusione di TEX e LATEX anche nel nostro paese. L'adesione al Gruppo prevede una quota di iscrizione annuale diversificata: 30,00 € soci ordinari, 20,00 (12,00) € studenti (junior), 75,00 € Enti e Istituzioni.

Indirizzi

Gruppo Utilizzatori Italiani di TEX:
c/o Ufficio Statistica
Scuola Superiore Sant'Anna
Piazza Martiri della Libertà 33
56127 Pisa, Italia.
`http://www.guit.sssup.it`
`guit@sssup.it`

Redazione ArsTEXnica:
`http://www.guit.sssup.it/arstexnica/`
`arstexnica@sssup.it`

Codice ISSN 1828-2369

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 7, Aprile 2009

Gruppo Utilizzatori Italiani di TEX

Editoriale

Gianluca Pignalberi

Il numero sette di ArsTEXnica, quello che tenete tra le mani, sembrerebbe essere il numero della stasi. No, non parlo dei fantasmi (in questo caso la polizia segreta dell'ex DDR), ma di una specie di limbo di inattività. Cercherò, a costo di arrampicarmi sugli specchi, di dimostrare che non è così.

Per prima cosa le tecnicaglie della rivista: non ho fatto niente, se non correggere dei piccoli ma fastidiosi bug che affliggevano gli script, e scoperti solo nel cambio di sistema operativo (sono passato da OpenSuse 10.3 a 32 bit a Fedora 10 a 64 bit, passando per Ubuntu 8.10 e OpenSuse 11.1 con alterne soddisfazioni e problemi). Ho anche cercato di razionalizzare la struttura dei file che gli autori dovrebbero fornire. L'ho fatto nell'ottica di automatizzare il più possibile la gestione degli articoli, e la loro collezione per comporre il numero della rivista. Forse non avevo le idee ben chiare, e senz'altro ho saputo comunicarle male, ma l'esperimento è riuscito solo a metà, per ora. Ho comunque ricevuto preziosi suggerimenti per migliorare questo aspetto, e spero di farne tesoro per il numero relativo al GUIT *meeting 2009*.

A fine febbraio si è tenuta la prima riunione del costituendo GrUT (la sezione romana del GUIT): pochissime persone, con una conoscenza di cui ho invidia. Voglio conoscere ancora più persone di questa caratura.

È ora di dare il giusto merito a chi ha lavorato veramente: gli autori. Questo numero si apre con un'intervista a Simone Guagnelli. È questi uno dei fondatori di *eSamizdat*, una favolosa rivista di slavistica da sempre composta con LaTeX. L'intervista, nata dall'idea di, e realizzata da Massimiliano Dominici, vede il mio modesto contributo (grazie Max per aver pensato a me), e mostra un risultato pregevolissimo ottenuto da letterati e umanisti che prima di allora non sapevano neanche cosa fosse LaTeX. *Chapeau*.

Si prosegue con un articolo di Giovanni Maschio su LaTeX per ciechi e ipovedenti, con cui rilancia alcune ottime osservazioni per definire uno standard di lavoro per le case editrici impegnate a realizzare libri di matematica per tali categorie di utenti. Queste case editrici si troverebbero ad abbattere notevoli costi di lavorazione, elevando la qualità dei risultati, e standardizzando anche i risultati stessi.

Claudio Beccari ci fa rimanere, con le sue consuete precisione e conoscenza, sugli standard: questa volta quelli relativi alla conservazione digitale di documenti. Il suo discettare sulle versioni di Pdf e le richieste dall'ISO ci fanno capire quanto lavoro è stato fatto, e quanto ne rimane da fare. Potete poi approfondire alcuni aspetti legali della questione su GGN 2009.

Di nuovo Claudio Beccari, questa volta con George Kamel, ci guida nel mondo dei testi copti con l'articolo dal titolo *Typesetting Coptic Liturgy in Bohairic*. Non vi anticipo altro: rimarrete folgorati dalla bellezza delle loro produzioni.

Questo numero si chiude con un articolo vasto e di grande interesse dedicato al PostScript in LaTeX. Siamo chiari subito: di questo articolo mi sono innamorato immediatamente, non appena l'ho visto nella sua forma embrionale. Non so se fosse vero amore o solo sindrome di Pigmalione, fatto sta che l'ho fatto un po' mio e ho costretto l'autore (Riccardo Nisi) ad accettare modifiche su cui abbiamo discusso duramente. Alla fine, a malincuore, le ha accettate tutte, e io gli sono estremamente grato. Spero abbia smesso di volermene per aver perpetrato un vero e proprio *coup* (o golpe, se preferite) ai suoi danni: se l'ho fatto è anche, e soprattutto, nell'interesse dei lettori, in special modo quelli più preparati su algoritmi e programmazione.

Un grazie speciale a Carlo Marmo: per questo numero è stato il correttore di bozze ufficiale. Il suo operato mi ha fatto notare dei dettagli che, come diceva Totò, erano dettagli importanti.

Detto ciò, vi auguro buona lettura.

Riferimenti bibliografici

GGN 2009 (11 febbraio 2009). «Dalla carta al digitale». In *Guida Giuridico Normativa*, a cura di A. Lisi, Italia Oggi, 90007.

▷ Gianluca Pignalberi
`g dot pignalberi at freesoftwaremagazine dot com`

Intervista a Simone Guagnelli, curatore della rivista *eSamizdat*

Simone Guagnelli, Gianluca Pignalberi, Massimiliano Dominici

Sommario

TEX è solitamente associato alla composizione di testi di carattere scientifico, dove si fa un uso rilevante di formule o di notazione matematica in generale. Tuttavia può dimostrarsi uno strumento utile anche in altri ambiti di pubblicazione, come mostra questa intervista a Simone Guagnelli, curatore e fondatore, insieme a Alessandro Catalano, della rivista *eSamizdat*.

Abstract

Usually TEX is linked to the typesetting of scientific texts, where a considerable use of formulae, and mathematical notation in general, is required. Yet it may prove a useful tool also in different kinds of publications, as shown in this interview with Simone Guagnelli, editor and founder, together with Alessandro Catalano, of *eSamizdat*.

4rsTEXnica: *Cominciamo, com'è d'obbligo, con le presentazioni. Cos'è, e come nasce,* eSamizdat*?*

Simone Guagnelli: *eSamizdat* è una rivista elettronica quadrimestrale che, come recita il sottotitolo, si occupa delle culture dei paesi slavi. Nasce dall'amicizia fra me e Alessandro Catalano e dal desiderio, coltivato negli anni, di lavorare insieme. Tutto si concretizza in una notte di primavera del 2003 al termine di una sbronza in un pub del quartiere romano di S. Lorenzo. Parte sia dalla convinzione che esistesse un enorme spazio editoriale e culturale fra slavistica accademica e divulgazione giornalistica, sia dalla consapevolezza che stesse crescendo una generazione in grado di occupare quello spazio. La nostra è stata cronologicamente la prima rivista elettronica dedicata agli studi delle culture slave.

4rsTEXnica: *Come e quando avete conosciuto LATEX? Cosa vi ha convinto ad usarlo per comporre* eSamizdat *in luogo di un prodotto visuale (e commerciale)?*

Simone Guagnelli: La scelta di usare LATEX per l'impaginazione di *eSamizdat* è stata solo una delle tante risposte che abbiamo dovuto escogitare quando abbiamo deciso di creare questa rivista. All'epoca eravamo due dottorandi di slavistica, quindi di una disciplina di nicchia del settore umanistico. Non solo non avevamo nessun finanziamento e nessuna esperienza editoriale, ma nessuno di noi due aveva particolari conoscenze informatiche: sapevamo ovviamente utilizzare i programmi di videoscrittura più comuni per scrivere un articolo, gestire la posta elettronica e navigare in rete, ma nient'altro. Eppure volevamo fare tutto da soli e offrire gratuitamente un prodotto culturale assolutamente innovativo nel nostro campo di studi: una rivista elettronica di qualità che potesse essere facilmente letta e consultata da un pubblico più ampio possibile (occupare quello spazio vuoto tra accademia e giornalismo, come ci piace sempre ricordare). Questo anche per coerenza con il nome che avevamo scelto per la rivista. Come è facile supporre infatti il titolo della testata significa "*samizdat* elettronico", e *samizdat* è una parola russa che significa "pubblicato da sé" e definisce quel fenomeno di editoria indipendente, "artigianale" e clandestina diffuso in tutti i paesi del Patto di Varsavia fino al 1989, in contrapposizione alla censura dei vari regimi. Dopo aver scelto insieme la linea editoriale e il tipo di rivista che avremmo voluto realizzare, io mi sono dovuto occupare di tutta la parte informatica del progetto. Questo per me ha significato mettermi a studiare da zero come funzionano i programmi di grafica, l'HTML e tutto il resto. Per quanto riguarda l'impaginazione mi sono rivolto per consigli ad amici e parenti. In particolare alla mia compagna, che è ricercatore al dipartimento di Informatica della Sapienza, e a mio fratello che è, tra le tante cose, fisico e programmatore. Entrambi, conoscendo perfettamente la mia scarsità di conoscenze e possibilità, mi hanno suggerito LATEX che utilizzavano quotidianamente per i loro articoli nel mondo scientifico. La sfida era allettante, in rete c'era una discreta mole di documentazione, il software era gratuito, di grande portabilità e, dato che avrei dovuto comunque imparare qualcosa di nuovo, l'ho accettata.

4rsTEXnica: *LATEX è diffuso soprattutto in ambito scientifico, dove non è insolito che a chi presenta un articolo sia richiesta la conoscenza di questo programma. Immagino, però, che non sia questa la situazione in ambito umanistico. Che tipo di problemi avete incontrato, e quali soluzioni avete adottato?*

Simone Guagnelli: All'inizio non è stato facilissimo. Rispetto al mondo scientifico, infatti, in quello umanistico quasi nessuno usa LATEX, almeno in Italia, e i contributi ci arrivavano (e ci arrivano

tuttora) in formato `.doc`. Per i primi numeri ho praticamente riscritto in file `.tex` tutti i documenti, articolo per articolo, lettera per lettera, `\footnote` per `\footnote`. Poi ho cominciato a usare delle macro (grazie a una mia amica e collega, Ombretta Gorini) e infine, negli ultimi due anni, ho utilizzato dei programmi di conversione word2tex che mi hanno risolto molti problemi, almeno i più comuni. Da un punto di vista tipografico, inoltre, comporre una rivista di slavistica significa avere a che fare con caratteri e alfabeti particolari (non solo caratteri con diacritici o il cirillico, ma, come è capitato, il greco classico o l'antico slavo ecclesiastico). O con vari tipi di fotografie e immagini. O con poesie e traduzioni ognuna con esigenze particolari di impaginazione dei versi o dei paragrafi. Ma ogni volta che si è presentato un problema, c'è sempre stata la possibilità di reperire informazioni su qualche pacchetto in grado di fare al caso nostro. Ad esempio, una delle più grandi soddisfazioni che ci ha dato LaTeX, è stata impaginare una traduzione parziale di un libro (*Europeana* dell'autore ceco Patrik Ouredník) che successivamente è stato pubblicato integralmente dalla Duepunti edizioni di Palermo e ha avuto un notevole successo. Bene, questo romanzo presenta la particolarità di alcuni titoletti a margine del testo vero e proprio. Questo per una rivista che ha anche le pagine a colonna doppia sembrava una sfida persa in partenza. Eppure, grazie soprattutto al comando `\marginpar`, alle minipage e a qualche altro accorgimento, siamo riusciti ad ottenere un risultato che lo stesso Ouredník ha definito tra i migliori delle sue varie edizioni mondiali. Senza contare il font. Alessandro si era fissato per il Garamond e anche a me sarebbe piaciuto. A questo proposito c'è un aneddoto che mi pare divertente e sintomatico dell'entusiasmo che all'inizio ci accompagnava. Io e la mia compagna, Novella, ci siamo messi a installare Garamond, senza grandi successi per un paio di giorni. Poi una sera, verso mezzanotte, io mi sono arreso e sono andato a dormire, lasciando lei al PC che si riprometteva un ultimo tentativo. Ad un certo punto sono stato svegliato da Novella che mi annunciava di essere finalmente riuscita ad installare Garamond. Ho guardato l'orologio: erano le 5 e 42...

ArsTEXnica: *Indubbiamente LaTeX può essere fonte di grandi soddisfazioni, ma anche di qualche frustrazione. In base alla vostra esperienza, quali sono i punti di forza e quali le debolezze di LaTeX per la composizione di una rivista come la vostra?*

Simone Guagnelli: Premesso che la nostra conoscenza del LaTeX rimane tuttora limitata, direi che le debolezze maggiori sono sicuramente legate ai font e alla necessità di alternare diversi sistemi alfabetici. In questo senso una vera faticaccia, durata settimane, è stata l'impaginazione di un articolo di Eleonora Gallucci in cui erano presenti, e si alternavano continuamente, greco classico e slavo antico. Il risultato è stato garantito, ma è stato la conseguenza di una lunga e paziente trasformazione dei caratteri originali. Ecco come, ed è solo un piccolo esempio, l'inizio di pag. 99 compare nel file `.tex` (il risultato è riportato nella figura 1):

Preslavismi presenti in UE:

власти (gr. καταχυριεύω); вѣрѫвати (gr. πιστεύειν); жизнь, житиє (gr. ζωή); коньць (gr. τέλος, συντέλεια); послоушьствовати (gr. μαρτυρέω); плема (gr. φυλή); съборъ e съборище (per il gr. συνέδριον e συναγωγή).

FIGURA 1: Estratto dall'articolo di Eleonora Gallucci apparso sul numero 2 di *eSamizdat*.

```
Preslavismi presenti in UE:

{\kliment vlasti}
(gr. \g{katakurie{\ua}w}
\selectlanguage{italian});
\selectlanguage{italian}
{\kliment v5r4vati}
(gr. \g{pi-ste{\ua}ein}
\selectlanguage{italian});
\selectlanguage{italian}
{\kliment 'i\symbol{149}n6},
{\kliment 'itie} (gr. \g{zw{\ha}}
\selectlanguage{italian});
\selectlanguage{italian}
{\kliment kon6c6}
(gr. \g{t{\ea}loc}
\selectlanguage{italian},
\selectlanguage{italian}
\g{sunt{\ea}leia}
\selectlanguage{italian});
\selectlanguage{italian}
{\kliment poslou[6stvovati}
(gr. \g{martur{\ea}w}
\selectlanguage{italian});
\selectlanguage{italian}
{\kliment plem3} (gr. \g{ful{\ha}}
\selectlanguage{italian});
\selectlanguage{italian}
{\kliment s7bor7} e
{\kliment s7bori]e}
(per il gr. \g{sun{\ea}drion}
\selectlanguage{italian} e
\g{sunagwg{\ha}}
\selectlanguage{italian}).
```

Un punto innegabilmente a favore di LaTeX, oltre a quelli già fatti presenti nelle risposte precedenti o relativi al facile controllo di margini e spazi, è sicuramente quello di garantire una qualità di stampa eccellente. Noi peraltro all'inizio non avevamo previsto una versione cartacea di *eSamizdat*. Quando però siamo stati invitati da Rita Giuliani a presentare in ambito universitario la rivista, è

FIGURA 2: La redazione di *eSamizdat* durante il Seminario di studi culturali russi «Manifesti in mostra», a Gargnano del Garda, nel 2004.

sorta l'esigenza di avere qualcosa da far vedere e toccare. Abbiamo così provveduto a stampare tre esemplari. Abbiamo provveduto personalmente anche a quelle tre copie, scegliendo la carta, facendo stampare da un amico le copertine, stampando le pagine, e portando il tutto a rilegare. Il successo è stato talmente clamoroso che, fino al 2007, quando abbiamo firmato un accordo con la casa editrice Aracne, abbiamo dovuto continuare ad occuparci anche delle copie a stampa (stiamo parlando, per carità, di cifre bassissime, una ventina per numero) della rivista per quanti ce le chiedevano.

ArsTEXnica: *Da quanto ci dici e ci mostri, il risultato dell'articolo di Gallucci è stato il frutto stupendo di un lavoro improbo. Da allora LaTeX ha beneficiato dell'introduzione di Unicode. Ne ha giovato anche il vostro lavoro, risultandone più semplice e veloce?*

Simone Guagnelli: Sicuramente l'introduzione di Unicode ha semplificato ulteriormente il lavoro richiesto usando il LaTeX, però, un po' per pigrizia, un po' per abitudine, confesso che finora ho sempre continuato a lavorare grossomodo nella stessa maniera, ripromettendomi in futuro di approfondire aspetti e migliorie del LaTeX che mi sono ancora sconosciuti.

ArsTEXnica: *La collaborazione con la casa editrice Aracne vi ha in qualche modo costretti a rivedere qualcosa, nel vostro metodo di lavoro, o hanno tranquillamente accettato che la rivista continuasse ad essere composta con LaTeX?*

Simone Guagnelli: Assolutamente no, il PDF che noi mettiamo a disposizione della casa editrice è esattamente lo stesso che può essere scaricato direttamente dal sito di *eSamizdat*. Non sono nemmeno sicuro che loro sappiano che noi utilizziamo il LaTeX...

ArsTEXnica: *Un'ultima curiosità. L'abbondanza di* `\selectlanguage{italian}`*, nel codice presentato, farebbe pensare ad un codice generato automaticamente. È così? Se sì, ce ne parli?*

Simone Guagnelli: Ora come ora non posso verificare in quanto, rispetto al 2004, ho cambiato piattaforma (da windows a mac) e non ho attualmente installati i font di greco e slavo antico, però, se ricordo bene, l'abbondanza di `\selectlanguage{italian}` non era dovuta a un codice generato automaticamente, ma a un qualche bug che causava altrimenti un conflitto nell'alternanza tra i due alfabeti.

ArsTEXnica: *Al momento in cui scriviamo l'intervista (fine febbraio 2009) sul vostro sito compare un ultimo Call for Papers dal contenuto tutt'altro che rassicurante. Qual è, a breve e medio termine, il futuro di* eSamizdat*?*

Simone Guagnelli: Dunque, questa è una domanda che mi aspettavo e che nello stesso tempo temevo. Per rispondere devo prima fare una premessa. In un certo senso, per caratteristiche proprie dovute al modo in cui è nata ed è strutturata, *eSamizdat* è una rivista che ha sempre vissuto in uno stato di crisi permanente. All'inizio non pensavamo affatto che saremmo durati così a lungo e più volte in questi anni abbiamo annunciato una chiusura che poi siamo sempre riusciti ad allontanare, grazie soprattutto all'incitamento di tanti amici e lettori. *eSamizdat* è una occupazione che si regge sul solo entusiasmo, ma nello stesso momento è soprattutto un impegno a tempo pieno; avrebbe bisogno quindi di un finanziamento vero o almeno che fosse assicurata e salda la posizione lavorativa di chi la cura. Purtroppo, anche per ragioni difficili da spiegare e non sempre cristalline, i nostri tentativi di risolvere alla base queste problematiche non hanno finora avuto particolare successo. Probabilmente peccherò un po' di presunzione, ma a volte mi sono sorpreso a pensare che la forza di *eSamizdat*, cioè la capacità di essere o anche solo di sembrare un prodotto "professionale", sia stata paradossalmente anche il motivo di una certà difficoltà a crescere e a trovare sostegni di vario genere. Ovviamente è una banalità, ma mi riferisco ad esempio alle numerose richieste di lavoro che abbiamo ricevuto da lettori e esperti di ogni genere nel corso di questi anni con tanto di invio di curriculum vitæ e lettere di accompagnamento... In questo senso il LaTeX ha sicuramente contribuito a dare l'impressione di una organizzazione professionale.

Va detta poi un'altra cosa, strettamente legata alle precedenti. Alcuni risultati non proprio trascurabili li abbiamo del resto raggiunti: quella generazione di giovani studiosi che ci ha accompagnati e sostenuti sin dal primo numero è ormai complessivamente parte integrante del sistema universitario; il riconoscimento internazionale c'è stato e quella che è probabilmente la più autorevole rivista russa del nostro settore nel 2006 ha concluso una

recensione dedicata a *eSamizdat* con queste parole: «Oltre a un indubbio valore scientifico, *eSamizdat* rappresenta un'esperienza sociale molto importante. Questo progetto non ha sponsor, né finanziamenti esterni, né sostegni istituzionali; è apparso grazie all'energia e alla passione professionale di due giovani e gode ora di larghissima richiesta. Nell'internet russo da molto tempo si discute sulla possibilità di creare un'edizione on line indipendente dedicata agli studi letterari russi. Quello che noi progettavamo, loro l'hanno realizzato. E per questo vanno ringraziati molto.» Anche il riconoscimento nazionale si può dire ci sia stato, anche se in modo meno istituzionalizzato e pur con una serie di riserve, polemiche sotterranee, incomprensioni che sono anche difficili da riassumere, ma che hanno dato luogo anche a qualche colpo basso.

Insomma, dal 2003 a oggi sono cambiate molte condizioni e in noi, alla generale soddisfazione, si accompagna anche una certa stanchezza e frustrazione. *eSamizdat* sta chiudendo, il prossimo numero in uscita a fine primavera, sarà probabilmente l'ultimo, anche se stiamo in questi giorni ragionando sulla possibilità di trasformarlo in penultimo... Vedremo. Una cosa, anche per tornare alla questione di fondo di questa intervista, è però certa: se *eSamizdat* proseguirà, lo farà continuando ad utilizzare solo ed esclusivamente il LaTeX. In caso spero di poter contare a questo punto anche sui vostri consigli di esperti in materia.

ArsTEXnica: *E noi saremo ben lieti, se ce ne sarà bisogno, di dare una mano. Fornire un sostegno "TEXnico" a quelle realtà che usano LaTeX rientra sicuramente nelle finalità della nostra associazione e negli intenti dei singoli.*

Chiudiamo questa intervista ringraziando Simone Guagnelli per la disponibilità e la cordialità con cui si è prestato a soddisfare le nostre curiosità e gli rivolgiamo i nostri migliori auguri per il futuro suo personale e delle iniziative intraprese. E ovviamente associamo ai ringraziamenti e agli auguri anche Alessandro Catalano, l'altro curatore di eSamizdat.

▷ Simone Guagnelli
eSamizdat
`www.esamizdat.it`

▷ Gianluca Pignalberi
ArsTEXnica
`g.pignalberi@alice.it`

▷ Massimiliano Dominici
ArsTEXnica
`mlgdominici@interfree.it`

TEX per i ciechi e per gli ipovedenti: Inglese, Esperanto o Romanesco?

Giovanni Maschio

Sommario

Vengono esposte un certo numero di ragioni per scegliere un modo già esistente di codificare la matematica, fruibile anche dai ciechi, senza costruire un nuovo modo di procedere.

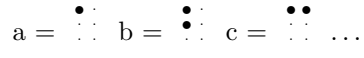
Abstract

We present here a number of reasons to choose an existing way to code mathematics, suitable for blind people, avoiding to build a new way for it.

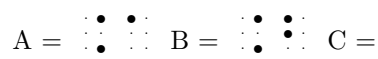
1 Introduzione

L'invenzione del codice braille nel 1829 da parte di Louis Braille ha permesso ai ciechi di leggere, scrivere e confrontarsi con i vedenti. Questo codice si avvale di matrici 2×3 di puntini in rilievo su carta, ottenendo 64 combinazioni che permettono la rappresentazione di tutte le lettere degli alfabeti latini più altri segni che vengono utilizzati in vari modi.

Come esempio[1] le prime lettere dell'alfabeto sono:



per indicare una lettera maiuscola si usa lo stesso segno della minuscola preceduto dal segno 



se è tutta una frase che va resa in maiuscolo, allora si fa precedere e seguire la frase dal segno . Per esempio ANTONIO diventa

Lo stesso metodo è usato per rappresentare i numeri o lettere greche. Nel primo caso si usano le lettere dell'alfabeto dalla a alla j precedute dal segno  , nel secondo opportune lettere dell'alfabeto precedute da un'indicazione che il testo seguente è in greco.

Con la diffusione dei calcolatori è divenuta accessibile quasi a tutti i ciechi la "barra braille", un dispositivo con matrici 2×4 di pistoncini che si sollevano, in corrispondenza di ogni singola lettera che compare sullo schermo, in modo da essere "leggibili" dalle dita di un cieco.

I due punti in più, rispetto al braille convenzionale, permettono la rappresentazione di tutti i 256 caratteri della codifica ASCII. Tuttavia, mentre per la codifica a sei punti dal 1978 si è raggiunto un accordo internazionale sul significato delle 64 combinazioni del braille tradizionale, sulla codifica a otto punti si è ancora in alto mare: ogni paese e ogni utilizzatore usa una codifica fatta, in genere, a suo uso e consumo.

Per quanto riguarda, in particolare, la codifica della matematica, la situazione è addirittura peggiore. Infatti per rappresentare i vari simboli matematici, ogni struttura dedicata alla produzione di libri o supporti informatici per i ciechi adotta un suo dialetto che, in generale, è diverso da quello di altri centri di produzione. Questo comporta grandi difficoltà per i ciechi che si avvicinano alla matematica.

Si sta verificando un processo analogo a quanto è accaduto, in campo scientifico, per trasmettere le proprie conoscenze agli altri.

Fino alla metà dell'Ottocento la lingua "scientifica", cioè la lingua usata dagli scienziati per trasmettere agli altri le proprie scoperte, era il latino. Poi si sono usate le lingue locali, il latino infatti era poco adattabile ai nuovi concetti e, in ogni caso, era una lingua artificiale e quindi difficile da imparare e usare nel modo corretto.

Lo scrivere nella propria lingua, però, ha comportato una serie di problemi nel diffondere i nuovi concetti. Basti un esempio: durante la guerra fredda gli scienziati sovietici scrivevano in russo e quindi, per conoscere il loro pensiero o si imparava il russo oppure, come è accaduto, bisognava utilizzare una struttura di "traduzione" dal russo in un'altra lingua maggiormente nota.

Quello che si è invece propagato naturalmente nel mondo scientifico, per diverse ragioni, è un linguaggio universale, derivato dall'inglese americano con l'aggiunta di moltissimi neologismi importati anche da altre lingue; se oggi un qualunque scienziato, di qualsiasi paese, vuole che le sue ricerche vengano conosciute dagli altri *deve* scrivere un articolo in tale lingua. Le riviste specializzate, infatti, accettano solo articoli scritti in inglese. Ne sono rimaste solo alcune piccole, a diffusione limitata che accettano articoli nella lingua locale.

1. Utilizzando il pacchetto LaTeX `braille.sty` si possono rappresentare in nero le codifiche braille.

Si possono trovare mille critiche a questo modo di procedere, dicendo che tale lingua è inadeguata, che un'altra è migliore, che l'Esperanto (lingua artificiale) è stata costruita in modo scientifico per coprire tutte le esigenze di espressione comprese quelle scientifiche ecc. ma ciò non scalfisce il *dato di fatto* che la lingua scientifica attuale è l'**inglese**.

2 La scrittura della matematica

Scrivere testi contenenti formule matematiche è sempre stato un grosso problema per le tipografie. L'esigenza di avere caratteri con segni strani, di altezze diverse da quelle degli altri caratteri, con la necessità di contornarli con altri caratteri o simboli, ha sempre creato difficoltà notevoli. In effetti il costo di composizione di una pagina contenente matematica è sempre stato di molto superiore a quello di una di solo testo sia con i caratteri mobili, sia con il piombo, sia con l'offset.

Negli ultimi decenni, con l'avvento dei calcolatori e, in particolare, dei personal computer, si sono immediatamente sviluppati numerosi software di scrittura per sostituire la macchina da scrivere e i sistemi di composizione tipografica.

Tra di essi, alcuni erano così macchinosi e lenti che furono immediatamente abbandonati, altri sopravvissero per alcuni anni per venire poi assorbiti o integrati da software di più ampia diffusione.

Attualmente, per quanto riguarda la scrittura di testi normali, il software principe di scrittura è Word in quanto legato al sistema operativo di Microsoft che copre la quasi totalità dei PC.

Altri software sono invece utilizzati, a livello professionale, per la tipografia in quanto più raffinati (QuarkXpress, InDesign, PageMaker...) ma più complessi da utilizzare. Così come i sistemi operativi, Macintosh, Unix, Linux... più duttili per ottenere risultati migliori ma meno diffusi di Windows.

Anche per la scrittura della matematica vi è stata un'evoluzione analoga. Negli anni '70-'80 del Novecento fiorirono numerosi sistemi di scrittura più o meno complicati e che ebbero vita più o meno lunga.

La selezione naturale, così come per la lingua scientifica, ha fatto sì che attualmente siano sopravvissuti solo alcuni di essi. Nel campo commerciale, associato a Word ci sono, oltre Equation editor che fa parte di esso, altri pacchetti che permettono scritture migliori. In ogni caso sono software dove le formule vengono costruite in parte scrivendo da tastiera e in parte selezionando simboli che vengono posizionati nel punto desiderato della formula che si sta scrivendo.

Nel campo scientifico un solo sistema di scrittura si è imposto quasi universalmente: il TEX. Le ragioni di questa rapidissima e totale diffusione sono diverse:

- il suo inventore, Donald Knuth, lo ha *donato* alla comunità scientifica;
- il testo sorgente è un file che può essere scritto anche con i soli caratteri ASCII a 7 bit;
- i simboli matematici e la formattazione delle formule sono ottenuti con comandi che sono solo stringhe di caratteri alfanumerici, facilmente memorizzabili perché, in genere, costituite da abbreviazioni delle parole (inglesi) che individuano i simboli medesimi;
- esistono numerose implementazioni del TEX per le diverse piattaforme, alcune commerciali ma molte fruibili gratuitamente e scaricabili legalmente dai siti opportuni;
- è sorta spontaneamente in tutto il mondo la necessità di scambio di informazioni sul TEX provocando la costituzione di numerose comunità di utilizzatori del TEX (TUG) i cui membri sono sempre molto disponibili per la risoluzione di problemi. Vi sono inoltre diversi siti (specchio l'uno dell'altro) costantemente aggiornati e disponibili per scaricare software vari (per esempio: `www.dante.de`).

Nel campo professionale della tipografia o si importano le formule come immagini o si usano software ad hoc (per esempio sotto Quark ci sono PowerMath, Mathsetter, l'unico che si avvale del TEX, Math Magic, che funziona anche sotto InDesign...).

3 La scrittura della matematica per i ciechi e per gli ipovedenti

È necessario, come prima cosa, distinguere il problema della scrittura della matematica a seconda del destinatario:

1. scrittura per i *ciechi* (braille o formato elettronico vocalizzabile);
2. scrittura per gli *ipovedenti* (stampa di testi correnti con caratteri ingranditi).

Nel primo caso, se si opta per un supporto cartaceo, si ottengono libri molto voluminosi; la pagina standard per scrivere in braille è un foglio di 30×30 cm di carta spessa (per poter punzonare sulle due facce senza bucare il foglio) e quindi anche un libro che nella stampa normale non sia troppo lungo, codificato in braille si traduce in diversi volumi piuttosto ingombranti. Per questa ragione si va diffondendo sempre di più l'uso dei vocalizzatori o della barra braille. Questi due ultimi metodi hanno bisogno di un file "leggibile" da parte dei due sistemi. L'*optimum* sarebbe un file unico per le tre destinazioni: stampa, vocalizzatore, barra braille.

Mentre questa meta è stata più o meno raggiunta per i testi di narrativa o comunque che non contengano matematica o alfabeti diversi (tipo il greco

o il cirillico per non parlare di lingue geroglifiche), per questi ultimi si è ancora in alto mare.

In particolare il problema della codifica della matematica è stato affrontato in maniera locale (il dialetto romanesco) cercando di costruire un linguaggio simbolico ad hoc (esperanto/romanesco)[2] che tenesse conto dei limiti del linguaggio corrente per i ciechi. In questo modo si è ottenuta una messe di testi scritti con linguaggi a volte incompatibili, che in ogni caso hanno bisogno di essere imparati dagli utenti. Un cieco che abbia imparato un linguaggio simbolico e che dovesse usare un libro scritto in un altro linguaggio, dovrebbe imparare una nuova lingua. Se poi, per caso, dovesse arrivare a studi di matematica più avanzati (università o ricerca) si troverebbe a dover in ogni caso imparare il TEX perché gli articoli scientifici (vedi paragrafo precedente) sono ormai scritti in tale linguaggio.

Il secondo problema (ipovedenti) è stato affrontato riscrivendo, con word processor commerciali o con traduttori opportuni (sempre però con metodi locali, cioè in romanesco), il libro a caratteri ingranditi nella misura richiesta dall'utente. In generale con la formula

$$1 \text{ libro} \leftrightarrow 1 \text{ utente}$$

essendo poco probabile che lo stesso libro possa poi essere usato da un altro utente con il medesimo ingrandimento (salvo qualche rara eccezione).

In ambedue i casi (ciechi e ipovedenti) le soluzioni attuali portano a un grande spreco di risorse umane ed economiche, soprattutto perché la soluzione (inglese) già *esiste* e, nel futuro, sarà sempre più economica e veloce.

Tale soluzione è il TEX, usato direttamente o per il tramite dei "dialetti" da esso derivati (come corposi pacchetti di macroistruzioni TEX), citando i più diffusi: il LATEX o l'$\mathcal{AMS}$-TEX.

Quanto segue si propone di illustrare molto schematicamente i vantaggi e i problemi posti dall'uso del TEX.

3.1 TEX per i ciechi

Il TEX, come già accennato, è un linguaggio di programmazione di alto livello che utilizza comandi che sono stringhe di caratteri alfanumerici. Vi sono alcuni caratteri che hanno un ruolo speciale (`# $ % ^ _ & { } \`) i più importanti sono:

- `\` che indica l'inizio di una stringa di comando (per esempio `\sqrt` indica la radice quadrata [**sq**uare**r**oo**t**]),
- le parentesi graffe `{}` che indicano l'inizio e la fine di un gruppo,
- il segno del dollaro `$` che indica l'inizio e la fine di una formula matematica. Se si vuole una formula in linea si usa singolarmente, se invece la si vuole al centro della pagina, va usato doppio.

Per esempio la formula

$$\int_a^{+\infty} f(x)dx = \lim_{\epsilon\to\infty} \int_a^{\epsilon} f(x)dx$$

è ottenuta con i comandi

```
$$
\int_a^{+\infty}f(x)dx=
\lim_{\epsilon\to\infty}
\int_a^{\epsilon}f(x)dx
$$
```

mentre $\int_a^{+\infty} f(x)dx = \lim_{\epsilon\to\infty} \int_a^{\epsilon} f(x)dx$ si ottiene con i comandi

```
$
\int_a^{+\infty}f(x)dx=
\lim_{\epsilon\to\infty}
\int_a^{\epsilon}f(x)dx
$
```

dove:

`\int`	uguale a segno di integrale
`^`	uguale a esponente o apice
`_`	uguale a deponente o pedice
`\infty`	uguale a simbolo di infinito
`\epsilon`	uguale a lettera greca epsilon

Salta subito agli occhi il vantaggio di utilizzare il TEX per codificare la matematica: *tutti i comandi e le macro sono direttamente codificabili in Braille essendo stringhe di caratteri alfanumerici* (a meno di alcuni caratteri speciali che andrebbero codificati a parte).

Naturalmente, se si dovesse scegliere questo modo di codificare la matematica, i problemi di passaggio dai metodi fin qui seguiti ai nuovi non sarebbero né semplici né di breve durata. Ne elenco solo alcuni di più immediata necessità di soluzione:

1. Decidere se usare il Braille a 6 punti (64 caratteri e quindi difficoltà di resa per i caratteri speciali) o a 8 punti (256 caratteri, l'intera tavola ASCII, ma difficoltà di lettura per i bambini più piccoli in quanto la superficie usata per una codifica a 8 punti è troppo grande per la parte sensibile del polpastrello di un bambino che frequenti le elementari).

2. Per esempio la Biblioteca Italiana per i ciechi "Regina Margherita" - ONLUS ha cercato di uniformare la produzione braille consigliando l'uso di un codice comune (Braille) e attualmente sta cercando un ulteriore adattamento per tener conto delle nuove realtà informatiche.

   Esiste anche un altro progetto (esperanto): LAMBDA (Linear Access to Mathematics for Braille Device and Audiosynthesis ARCHIBRAILLE Association) che si è sviluppato all'interno di un progetto europeo triennale. È un sistema di scrittura che usa MathML versione 2.0 per essere convertito nei più diffusi formati di scrittura matematica. Non è gratuito.

2. Semplificazione del TEX per i ciechi. Molte delle istruzioni presenti in un libro scritto in TEX servono solo per la formattazione del testo (bilanciamento dei bianchi, titolazioni, fonti ecc.) che non sono necessarie a un cieco per la comprensione del testo. Altre macroistruzioni possono invece essere lasciate come sono (per esempio `\footnote{Questa \‘e una nota}` che produce una nota[3] può esser lasciata così com'è magari avendo l'accortezza di mettere un ritorno carrello all'inizio e uno alla fine della nota, in modo che per il cieco sia ben chiaro qual è il gruppo racchiuso dalla nota[4].

3. Manuali per gli insegnanti e per gli utenti, graduati per difficoltà crescente e quindi includenti via via sempre più stringhe di controllo.

4. Istruzione del personale addetto alla trascrizione dei testi di matematica nella scrittura semplificata in TEX.

5. Scrittura diversificata oppure unitaria per ottenere il Braille o il formato elettronico per la vocalizzazione.

6. Utilizzazione del materiale scritto con altri metodi di codifica.

7. Difficoltà nell'ottenere i file sorgenti dalle Case Editrici che sono molto restie a consegnarli perché hanno paura di una loro distribuzione non autorizzata da parte degli utenti finali.

Di alcuni di questi problemi (1-4) le soluzioni sono già *in itinere*, infatti la Biblioteca Regina Margherita sta cercando di uniformare la codifica Braille (per il momento si è adottata la scelta dei 6 punti) decidendo come codificare i caratteri speciali. Inoltre ha deciso di adottare il LATEX come metodo di scrittura per la matematica. Ci sono, infatti, già diversi software di ausilio; uno molto interessante per la lettura di testi in LATEX (TESI, Zattera (1997)) fa sì che il vocalizzatore traduca in testo corrente le istruzioni LATEX. Per esempio l'istruzione `\overline{x}` viene letta "x soprassegnato". Un altro software molto promettente stabilisce un modo interattivo per risolvere esercizi di matematica (C.I.S.A.D.), inoltre è un ausilio all'insegnante perché in contemporanea mostra sullo schermo la formula in chiaro (per l'insegnante) e in braille sulla barra braille (per il cieco). Quest'ultimo esegue la risoluzione in braille e questa viene tradotta sullo schermo per l'insegnante. Ambedue sono basati sul LATEX e quindi possono essere utilizzati su testi scritti in tal modo.

3. Questa è una nota

4. Infatti se all'interno della nota ci sono altri gruppi potrebbero sorgere delle difficoltà per capire i vari livelli di parentesi; lasciare un ritorno carrello dopo la parentesi graffa finale della nota fa capire senza ombra di dubbio il suo significato.

Per il punto 5 bisognerà decidere se trovare una sola semplificazione del TEX in modo da poter utilizzare il medesimo file sia sulla barra braille sia per la vocalizzazione (magari utilizzando anche TESI), oppure dare vita a due file diversi uno per il Braille, l'altro per la vocalizzazione.

A questo proposito basta citare, come esempio, modi diversi per descrivere matrici o tabelle.

3.2 Matrici e tabelle

Vi sono molteplici comandi TEX o LATEX per ottenere tabelle, matrici e oggetti similari. Adattarli ai ciechi o agli ipovedenti potrebbe comportare delle scelte diverse. Io penso che per un testo in braille il modo migliore sia quello di avere una tabella braille con i simboli distribuiti regolarmente sulla pagina in righe e colonne come nella forma stampata in nero. Invece, per la forma elettronica (sia per il vocalizzatore sia per la barra braille) potrebbe essere migliore la forma contenente i comandi TEX. Così, per esempio, i comandi TEX per ottenere

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

sono

```
$$
\left(\matrix{a_{11}&a_{12}&\dots&a_{1n}\cr
a_{21}&a_{22}&\dots&a_{2n}\cr
\vdots&\vdots&\dots&\vdots\cr
a_{n1}&a_{n2}&\dots&a_{nn}}\right)
$$
```

nella forma elettronica si possono lasciare così come sono, invece in braille sarebbe meglio trasformarli in

```
$$
\left(\matrix{
a_{11} a_{12} ... a_{1n}
a_{21} a_{22} ... a_{2n}
   .      .          .
   .      .   ...    .
   .      .          .
a_{n1} a_{n2} ... a_{nn}
}\right)
$$
```

in questo modo il cieco avrebbe sulla pagina una costruzione regolare della matrice, più facile da recepire rispetto a quella, in un certo senso disordinata, che è data dai comandi TEX come sono scritti.

In modo analogo si dovrebbe procedere per le tabelle.

3.3 TEX per gli ipovedenti

Come già accennato il problema maggiore rappresentato dai testi per gli ipovedenti è che si deve costruire un testo *ad personam*; l'uso del LATEX comporta allora due tipi di approccio:

- Il testo che si riceve dall'editore è in TEX.
- Il testo che si riceve dall'editore è un testo non-TEX.

Nel primo caso il problema è minimo, infatti la realizzazione di una copia a una particolare grandezza dei caratteri a partire dai file comporta pochi comandi all'inizio del file per ottenere le dimensioni corrette, serve poi solo un minimo di tempo per l'impaginazione. Inoltre è sufficiente costruire una macro-istruzione per far sì che nella testatina o al piede compaia anche il numero di pagina dell'edizione in commercio in modo che l'ipovedente non abbia alcuna difficoltà a seguire le istruzioni dell'insegnante[5].

Nel secondo caso si tratta di decidere se conviene trasformarlo in TEX oppure usare un metodo più tradizionale. Fattori di decisione possono venire da eventuali accordi con l'Editore (se l'Editore è disposto a caricarsi una parte delle spese di traduzione potrebbe ottenere in cambio i file TEX da dare agli autori per facilitare eventuali riedizioni).

In ogni caso la scelta del LATEX per realizzare testi per gli ipovedenti è senz'altro vincente e per il risparmio e per la velocità di realizzazione dei testi. Attualmente, infatti, per realizzare un testo occorre, mediamente, un tempo di 2-3 mesi, avere il testo in LATEX comporterebbe un tempo di ore anziché mesi.

4 Conclusioni

Credo di aver reso ben chiaro come avere un file scritto in LATEX dia luogo a un risparmio notevole sia per avere un file per i ciechi (per il Braille o per la vocalizzazione) sia per stampare testi a caratteri ingranditi. Le risorse così risparmiate potrebbero essere utilizzate per produrre una maggiore mole di testi fruibili.

Inoltre, la prospettiva per il futuro è che l'uso del LATEX da parte degli autori sarà sempre più diffuso. Infatti è ormai entrato nell'uso comune, in quasi tutte le facoltà scientifiche, quello di fornire agli studenti una classe LATEX per scrivere la tesi. Quindi i nuovi autori, di solito, forniscono alle case editrici dei testi che sono già scritti in TEX o in LATEX.

Le Case Editrici si stanno adeguando, passando da sistemi di stampa tradizionali (tipo Quark) a testi formattati direttamente con classi LATEX costruite in maniera opportuna in modo da ottenere una resa tipografica paragonabile a quella ottenibile con gli altri software di impaginazione.

In quanto alla paura che le Case Editrici hanno di una distribuzione non autorizzata dei file da parte degli utenti finali, non ci sarebbero rischi. Infatti i file per i ciechi sarebbero scritti in un TEX semplificato e non sarebbero adeguati per originare un'uscita utile per la stampa[6].

Se qualcuno volesse utilizzarli in questo senso, la fatica per rendere utilizzabili questi file sarebbe maggiore di quella di riscrivere il libro ex-novo.

Se invece i file fossero utilizzati solo per testi per gli ipovedenti, poiché la realizzazione di tali testi sarebbe affidata a strutture convenzionate con la Biblioteca Italiana, non ci dovrebbero essere paure di "fughe" indesiderate di file sorgenti a danno dei diritti degli autori o delle Case Editrici.

Riferimenti bibliografici

ARCHIBRAILLE Association. «Lambda project». URL `http://www.lambdaproject.org`.

Braille (1998). *Codice braille italiano*. Biblioteca italiana per i ciechi "Regina Margherita" Onlus, Monza.

C.I.S.A.D. «Braillemat». URL `http://www.cisad.it/matematica/braillemat`.

Zattera, M. (1997). TESI, *Ausilio informatico per non vedenti per la manipolazione di espressioni matematiche in TEX*. Tesi di Laurea, Padova. URL `www.math.unipd.it/~artico/TeSI.html`.

▷ Giovanni Maschio
Dipartimento di
Metodi e Modelli Matematici
per le Scienze Applicate
Sapienza, Università di Roma
`giovanni.maschio@uniroma1.it`

5. Per esempio istruzioni del tipo: fare l'esercizio x a pagina y.

6. I file di questo tipo sono paragonabili a quelli che si possono ottenere con una scansione passata poi a un qualunque OCR, oppure estraibili da un file pdf.

Il formato archiviabile dei file PDF

Claudio Beccari

Sommario

Il problema dell'archiviazione elettronica richiede un formato particolare definito PDF/A-1 dalla norma ISO 19005-1. In questo articolo si mostra come ottenere questo formato con i programmi principali o accessori del sistema TEX. Si metteranno in luce anche le difficoltà che si incontrano sul cammino della produzione di un documento archiviabile e si proporranno alcune soluzioni.

Abstract

Archiving electronic documents requires a special format called PDF/A-1 by the ISO regulation 19005-1. This paper shows how to obtain this result with the main and subsidiary programs of the TEX system. Some difficulties that are encountered in this process will be also highlighted; some solutions to the above problems will be suggested.

1 Introduzione

1.1 L'archiviazione dei documenti elettronici

Ho cominciato ad interessarmi dell'archiviazione dei documenti elettronici quando la mia università mi ha chiesto di affrontare il problema; nelle università esiste certamente il problema dell'archiviazione dei documenti, ma è particolarmente gravoso quello dell'archiviazione delle tesi di laurea, delle tesi magistrali, delle tesi di dottorato: volumi abbastanza ingombranti, spesso con copertine rigide, altrettanto spesso con allegati in formati strani e comunque 'grandi', che mettono a dura prova i sistemi di archiviazione 'cartacea' tradizionali.

Ho poi rilevato che il problema dell'archiviazione riguarda tutte le amministrazioni pubbliche, gli uffici degli enti e delle compagnie private, gli studi professionali, con particolare riguardo a quelli di tipo legale. Il problema è quindi di vastissima portata.

La soluzione di questo problema è stato affrontato dall'International Standards Organisation con l'emissione della norma ISO 19005-1; ma al di là della norma esso è stato affrontato anche dalle aziende produttrici di software al fine di mettere a disposizione dell'utenza programmi affidabili per poter produrre documenti che soddisfino alla norma suddetta. Il sistema TEX non può essere estraneo a questo processo; infatti è recentissima la pubblicazione sugli archivi CTAN dei pacchetti necessari per produrre documenti in questo formato.

1.2 La norma ISO 19005-1

La norma ISO 19005-1 specifica un particolare formato per l'archiviazione dei documenti elettronici; essa specifica che il documento sia sostanzialmente salvato in formato PDF 1.4 ma con alcune varianti che rappresentano la sostanza della norma. Questo vuol dire che qualunque documento salvato in formato PDF in versione 1.4 *non* è di per sé un documento archiviabile, se non soddisfa alle ulteriori specificazioni. Va da sé anche che file salvati in versioni PDF precedenti alla 1.4 devono essere convertiti alle specifiche della versione 1.4, che è quella che si produce, in sostanza, con il programma Adobe Acrobat 5. Allo stesso modo documenti salvati in formato PDF di versione successiva alla 1.4 devono ugualmente venire convertiti alla versione 1.4.

Il programma eseguibile pdftex gestisce la composizione diretta dei file sorgente che abitualmente hanno il mark-up di LATEX, e, se è sufficientemente recente, produce la sua uscita in formato PDF versione 1.4; il suo autore Hàn Thế Thành continua ad aggiungere funzionalità al suo magnifico programma; fra l'altro sembra che egli intenda produrre versioni finali del formato che stiano a giorno con gli standard fissati dalla Adobe.

Le specifiche ulteriori per il formato archiviabile danno luogo a due diverse sottoversioni del formato PDFA, precisamente:

PDF/A-1a Implica che, oltre alla indispensabile fedeltà di visualizzazione e di riproduzione, il formato contenga la mappatura completa dei font alla codifica UNICODE e contenga tutto il necessario per la conservazione della struttura logica del documento;

PDF/A-1b implica solo l'indispensabile fedeltà di visualizzazione e di riproduzione.

Non è il caso di entrare nei dettagli delle varie specifiche, ma è necessario vedere quali programmi siano in grado di produrre direttamente il formato PDFA, anche i programmi che non fanno parte del sistema TEX.

Successivamente si esamineranno le possibilità offerte dai programmi del sistema TEX, entrando nei dettagli, visto che finora nessuna documentazione esplicita diceva come fare; i pochi documenti che riguardano questa problematica sono abbastanza fumosi per la persona inesperta; anch'io, che sono

decisamente inesperto di linguaggi di scripting e di linguaggio PostScript, ho avuto le mie difficoltà per ottenere risultati di qualche utilità. Anche dopo la pubblicazione su CTAN del pacchetto pdfx ho avuto le mie difficoltà, che fortunatamente sono riuscito a superare e vorrei condividere con i lettori le soluzioni che ho trovato.

Le difficoltà che si incontrano, e che verranno esposte nel seguito, richiedono di essere piuttosto esperti del sistema TEX per porvi rimedio; insomma, produrre documenti in formato PDFA è piuttosto difficile e non sempre ci si riesce.

2 I programmi che convertono e/o che esportano i loro file in formato PDFA

A tutt'oggi non sono molti i programmi che producono file di uscita in formato PDFA.

Da una parte c'è il programma "principe" della Adobe, il suo Acrobat Professional versione 8 (forse anche dalla versione 7, ma non ho avuto modo di verificare) permette la migliore conversione dei file in formato PDF nel formato PDFA. Esso contiene anche una funzione dei suoi menù, denominata `Preflight`, che permette di analizzare un dato file PDF in modo da sapere se esso è già conforme a una delle due versioni del formato PDFA; eventualmente un semplice click del mouse permette di eseguire la conversione. Naturalmente si tratta di un prodotto commerciale che ha un costo in generale non indifferente, sebbene sia anche offerto alle istituzioni formative a prezzi scontati. Bisogna subito dire che non tutti i file PDF prodotti con altri sistemi possono venire convertiti nel formato PDFA con Acrobat Professional e nel seguito si discuteranno in parte i problemi che si possono incontrare.

Parlando di programmi commerciali la rete offre diverse possibilità con programmi che consentono di elaborare in vari modi i file PDF, e includono spesso l'opzione per la conversione in formato PDFA. Non se ne elenca nessuno; chiunque può cercare su Internet la parola PDFA oppure PDF/A e avrà solo l'imbarazzo della scelta.

Fra i word processor ci sono almeno due soluzioni che devono essere citate, una commerciale e una gratuita: si tratta della suite Microsoft Office 2007, e quella di OpenOffice.org successiva alla versione 2.4.

La suite della Microsoft ha introdotto con la versione 2007 numerose modifiche ai suoi programmi, alcune delle quali riguardano i formati con markup esteso (XML), ma, per quel che ci interessa qui, ha introdotto la possibilità di esportare i suoi file in formato PDF e PDFA. Il cambiamento dei suoi formati può forse essere anche legato al fatto che l'Unione Europea ha stabilito delle norme sugli "open format document" richiesti dalla grande mole di documenti prodotti nei vari uffici della Unione Europea.

Il programma gratuito (e open source) OpenOffice.org fin dal suo inizio ha un suo formato aperto, sostanzialmente XML, anche se può salvare i suoi documenti nei formati compatibili con quelli della suite di Microsoft; dalla versione 3.0 esso può aprire e salvare anche i formati di MS Office 2007. Tuttavia esso ha anche sempre avuto la possibilità di esportare i suoi file in formato PDF; a partire dalla versione 2.4.1 esso ha esteso la sua funzionalità all'esportazione dei suoi file in formato PDFA.

La versione di OpenOffice.org che si chiama NeoOffice ed è disponibile solo per le piattaforme Macintosh, non è ancora all'altezza della versione coeva di OpenOffice.org; esso può esportare il suoi file nel formato "Tagged PDF", ma questo è solo una parte delle ulteriori specifiche non tanto del formato PDFA, quanto del formato PDFX, un altro standard fissato dall'ISO per la compatibilità di stampa dei formati PDF.

Tra i programmi della distribuzione del sistema TEX, con qualche "incertezza", esistono due possibilità di eseguire la creazione diretta oppure la conversione di un file PDF in formato PDFA. Le incertezze sono dovute alla mancanza di documentazione o alla documentazione scritta essenzialmente per gli addetti ai lavori.

3 Conversione di un file PDF in un file PDFA mediante ghostscript

La documentazione del programma ghostscript contiene vari file, nessuno che riporti nel suo nome la stringa 'pdfa'; cercando meglio si trova che il file `ps2pdf.html` contiene anche alcune informazioni per la trasformazione di un file in formato PS, *oppure in formato PDF*, nei formati PDFA e PDFX.

Ho letto e riletto la documentazione e ho scoperto che disgraziatamente l'interfaccia ps2pdf ha troppe limitazioni per eseguire la conversione; di fatto bisogna ricorrere all'eseguibile di ghostscript. Sulle macchine UNIX, comprese le macchine Linux e i Macintosh con sistema operativo Mac OS X, questo eseguibile si chiama gs; invece sulle macchine Windows esso si chiama generalmente gswin32c. Ma anche l'esempio riportato sulla documentazione `file:///usr/local/share/ghostscript/8.63/doc/Ps2pdf.htm` così com'è non funziona.

Il comando che bisogna dare (su una riga sola, senza andare a capo come si fa qui per necessità di giustezza) è invece:

```
gs -dPDFA -dBATCH -dNOPAUSE -dNOOUTERSAVE
-dUseCIEColor -sDEVICE=pdfwrite
-sProcessColorModel=DeviceCMYK
-sOutputFile=out-a.pdf PDFA_def.ps
input.ps
```

dove `out-a.pdf` rappresenta il nome del file trasformato e `input.ps` indica il nome del file da trasformare; questo file, invece dell'estensione `.ps`, potrebbe avere l'estensione `.pdf` perché ghostscript è in grado di ricevere in input anche i file in formato PDF. La parte importante che manca nella documentazione è proprio la specifica che indica il modello di colore; nell'esempio sopra riportato il comando specifica di usare il `DeviceCMYK`; l'alternativa sarebbe il `DeviceGray`; sembrerebbe che sia vietato usare il `DeviceRGB`, ma non sono sicuro che sia una specifica del formato PDFA o se sia un vincolo imposto da ghostscript; avendo acquisito maggiore familiarità con questa problematica, ritengo che l'esclusione del modello di colore RGB, che si fa nella documentazione di ghostscript, dipenda dal fatto che essa riguarda essenzialmente il formato PDFX, dove effettivamente sono ammessi solo il modello CMYK e il modello del grigio; per il formato PDFA si può usare anche il modello di colore RGB.

Non solo, ma il comando fa riferimento ad un file `PDFA_def.ps` che va cercato fra i "ColorSpace' delle 'Resource' di ghostscript; questo file va configurato per rispettare la situazione esistente sulla macchina specifica sulla quale si sta operando. Il punto delicato è che esso deve fare riferimento a un modello di colore conforme al 'Device' specificato nella linea di comando. Nell'esempio sopra riportato, che fa riferimento a un `DeviceCMYK`, bisogna specificare un file `.icc` che si riferisca alla quadricromia e che sia disponibile sulla specifica macchina. Sul mio MacBookPro ho listato tutti i file con quella estensione, e ho capito che sono relativi alle varie stampanti e ai vari schermi che si possono usare per rendere visibile ogni file; ho scelto un file abbastanza generale: `/System/Library/ColorSync/Profiles/Generic CMYK Profile.icc` e l'ho usato per configurare il file `PDFA_def.ps`; il risultato è mostrato nella figura 1.

Con il modello di colore CMYK, che indica la stampa in quadricromia, si possono ottenere quattro differenti lastre con la separazione dei colori, ma per i documenti in bianco e nero ne serve una sola; questo è un problema minore, perché nella tipografia dove avviene la stampa si può scegliere se produrre tutte e quattro le lastre o se produrre solo quella per il colore nero; come si vede questi requisiti sono importanti per il formato PDFX, ma sono abbastanza irrilevanti per il formato PDFA. Si veda più avanti la problematica relativa ai modelli di colore e ai file che li descrivono.

Come si vede nella figura 1, i punti da personalizzare sarebbero tre: (*a*) bisogna specificare il nome del file `.icc` relativo al modello di colore; (*b*) bisognerebbe specificare il titolo del documento contenuto nel file PDFA; e, infine, (*c*) bisognerebbe specificare il valore del comando `/OutputConditionIdentifier`. Quest'ultimo punto non è critico e forse è importante solo per il formato PDFX, che qui viene nominato solo perché sotto certi aspetti esso è simile al formato PDFA e spesso è trattato congiuntamente; sapendo che cosa indica quel comando, si potrebbe specificare un valore sensato e adatto allo scopo, ma nelle prove che ho condotto, non ho mai cambiato questo valore. Per quanto riguarda il titolo sembrerebbe che quelle due righe potrebbero essere eliminate del tutto se si avesse l'abitudine di specificare tutti i necessari parametri del file PDF; questa è un'operazione che viene compiuta raramente e il compilatore pdflatex non se ne lamenta, ma così facendo si perdono alcune possibilità offerte dal formato PDF. Per il formato PDFA è invece obbligatorio specificare almeno il titolo e diventa necessario specificare queste informazioni nel file `PDFA_def.ps`, perché, se anche fossero presenti nel file PDF da trasformare, ho scoperto con mia grande sorpresa che non vengono trasferite direttamente nel file PDFA.

L'esperienza mi ha dimostrato due cose:

1. disponendo di un MacBookPro, per il quale il formato PDF è quello di default per tutti i documenti, non ho mai usato pdflatex per creare un file DVI, per poi trasformarlo in file PS attraverso dvips; quindi non ho mai verificato che cosa succede ad eseguire la conversione in formato PDFA partendo da un file sorgente in formato PS. Tuttavia ho provato ad usare il programma pdf2ps; il risultato della conversione del file sorgente PDF in formato PS ha avuto luogo in modo soddisfacente e in pochi istanti; tuttavia credo che questo programma di conversione non gestisca bene i font di tipo vettoriale, e usi una qualche sostituzione con i font bitmapped. Il risultato della successiva trasformazione in un file PDFA è insoddisfacente perché quasi tutti i font del documento sono stati sostituiti con font bitmapped e la visualizzazione sullo schermo lo denota chiaramente, con una visione non nitida come avverrebbe con i font vettoriali. Non sono in grado di dire se il passaggio DVI $\longrightarrow$ PS $\longrightarrow$ PDFA dia risultati soddisfacenti, mentre sono in grado di dire che il passaggio PDF $\longrightarrow$ PS $\longrightarrow$ PDFA dà risultati insoddisfacenti.

2. La trasformazione diretta PDF $\longrightarrow$ PDFA attraverso il comando esposto sopra dà risultati soddisfacenti sia per quel che riguarda i font vettoriali, sia per quel che riguarda la velocità della trasformazione, ma non è scevra da problemi.

```
%!
% $Id: PDFA_def.ps 8284 2007-10-10 17:40:38Z giles $
% This is a sample prefix file for creating a PDF/A document.
% Feel free to modify entries marked with "Customize".

% This assumes an ICC profile to reside in the file (ISO Coated sb.icc),
% unless the user modifies the corresponding line below.


systemdict /ProcessColorModel known {
  systemdict /ProcessColorModel get dup
     /DeviceGray ne exch /DeviceCMYK ne and
} {
  true
} ifelse
{ (ERROR: ProcessColorModel must be /DeviceGray or DeviceCMYK.)=
  /ProcessColorModel cvx /rangecheck signalerror
} if


% Define entries to the document Info dictionary :

%/ICCProfile (ISO Coated sb.icc)   % Customize.
/ICCProfile (/System/Library/ColorSync/Profiles/Generic CMYK Profile.icc)
def

[ /Title (Title)                  % Customize.
  /DOCINFO pdfmark

% Define an ICC profile :

[/_objdef {icc_PDFA} /type /stream /OBJ pdfmark
[{icc_PDFA} <</N systemdict /ProcessColorModel get
   /DeviceCMYK eq {1} {4} ifelse >> /PUT pdfmark
[{icc_PDFA} ICCProfile (r) file /PUT pdfmark

% Define the output intent dictionary :

[/_objdef {OutputIntent_PDFA} /type /dict /OBJ pdfmark
[{OutputIntent_PDFA} <<
  /Type /OutputIntent             % Must be so (the standard requires).
  /S /GTS_PDFA1                   % Must be so (the standard requires).
  /DestOutputProfile {icc_PDFA}            % Must be so (see above).
  /OutputConditionIdentifier (CGATS TR001)      % Customize
>> /PUT pdfmark
[{Catalog} <</OutputIntents [ {OutputIntent_PDFA} ]>> /PUT pdfmark
```

FIGURA 1: La mia configurazione del file `PDFA_def.ps`

4 La generazione diretta di un file PDFA mediante pdflatex

Ho contattato Hàn Thế Thành per posta elettronica e mi ha risposto dicendo che il suo programma è già in grado di produrre i file in formato PDFA; la cosa non è assolutamente descritta nel file di documentazione del programma pdftex fornito insieme ad ogni distribuzione del sistema TEX; ma Hàn Thế Thành mi ha segnalato un link, Hàn Thế Thành (2008b), aggiungendo che in questo link espone quanto egli sia riuscito a fare in questa direzione: "As it can be seen in the examples, creating a pdf/a from a typical latex doc is already possible. Creating tagged pdf requires more work than pdf/a, and certain require rewriting latex macros for headings and environments."[1]

In effetti, andando a vedere quel link si ricava che Hàn Thế Thành sta lavorando nella direzione di aggiungere la possibilità di produrre con il suo programma i file PDF con la caratteristica di "Tagged PDF". Si rileva anche che la cosa è molto sperimentale e tutt'altro che risolta; tuttavia i *patch* che egli ha predisposto e che si possono scaricare e installare permettono di ottenere direttamente file PDF/A-1a (sic), certificati come tali mediante il programma Acrobat Professional 8. Dunque il problema sarebbe risolto.

Successivamente Hàn Thế Thành mi ha informato che in un altro sito, Hàn Thế Thành (2008a), aveva inserito una specie di guida per ottenere dal suo programma sia il formato PDF/A-1a sia quello PDF/A-1b; ancora più importante, in questo sito egli elencava un certo numero di problemi e alcune possibili soluzioni.

In questo sito egli indicava anche che gli stessi risultati si sarebbero potuti ottenere applicando i 'patch' descritti nel sito Hàn Thế Thành (2008b). Questi 'patch' erano applicabili se e solo se si disponeva di una macchina di tipo UNIX; per fortuna il mio sistema operativo Max OS X lo è, e quindi mi sono azzardato a eseguire questi 'patch', a compilare gli eseguibili, a rimetterli nell'albero principale del mio sistema TEX, e a rigenerare tutti i file di formato; non mi è riuscita l'operazione al primo colpo, ma infine le operazioni da eseguire erano proprio quelle indicate sopra, né più né meno, salvo che non erano specificate da nessuna parte. Ero abbastanza tranquillo, perché in caso di insuccesso completo avrei potuto reinstallare MacTEX senza perdere nulla (salvo un po' di tempo).

Nonostante ciò, pur disponendo del file eseguibile aggiornato con i 'patch' di Hàn Thế Thành, e dei file aggiuntivi da lui predisposti, riuscivo a ottenere file PDF perfettamente composti, ma non confacenti al formato PDFA; Acrobat Professional non riusciva nemmeno a convertirli in formato PDFA.

Ero arrivato quasi al punto di gettare la spugna, quando finalmente su CTAN (Radhakrishnan e Hàn Thế Thành (2008)) è apparsa la distribuzione non più sperimentale e adeguatamente completata dei file necessari per compilare un file in formato PDFA direttamente con pdflatex. Il programma funziona bene ma la documentazione è scarna e comunque sussistono alcuni problemini di cui si parlerà nel seguito.

5 La versione quasi definitiva di pdfx

Per usare la versione (quasi) definitiva di pdfx bisogna scaricare il pacchetto da CTAN e installare i vari file nelle cartelle giuste; visto che sono consentite alcune personalizzazioni e la distribuzione sicuramente verrà aggiornata, io suggerirei di installare questi file nell'albero locale e nelle cartelle giuste perchè il sistema TEX possa trovarli; personalmente ho installato il file di documentazione `pdfx.pdf` in `$HOME/texmf/doc/pdfx`[2]; la sottocartella `src` contenuta nel pacchetto ZIP scaricato dalla rete è stata direttamente innestata in `$HOME/texmf/`; infine gli altri file contenuti nel pacchetto compresso sono stati salvati in `$HOME/texmf/tex/latex/pdfx/`. Bisogna anche scaricare il pacchetto xmpincl installandone i pochi file in `$HOME/texmf/tex/latex/xmpincl/`.

Bisogna però scaricare dalla rete un file ICC corrispondente al modello di colore desiderato; la documentazione del pacchetto pdfx suggerisce di usare un file con un vecchio formato, `sRGBIEC1966-2.1.icm`; sul mio Mac io avrei potuto scegliere fra un'altra miriade di file ICC, ma ho preferito scaricare dal sito dell'European Color Initiative `www.eci.org` il file `ECI-RGB.V1.0.icc`, sia perché corrisponde ad un formato più recente, sia perché è scaricabile e usabile da chiunque; non ho abbastanza esperienza per dire se i colori resi attraverso questo file siano migliori o peggiori di altri, ma ritengo che l'ente che li mette a disposizione sia sufficientemente affidabile. Anche questo file va inserito in `$HOME/tex/latex/pdfx/`.

Mancano ancora i due file `glyphtounicode.tex` e `glyphtouunicode-cmr.tex` da inserire entrambi nella solita cartella `$HOME/tex/latex/pdfx/`. Il primo file dovrebbe già essere compreso nella distribuzione del sistema TEX aggiornata del 2008; invece il secondo file è distribuito insieme a pdfx.

Per produrre un file in formato PDFA bisogna fare due cose:

1. Come si può vedere negli esempi, creare un file PDF/A da un tipico documento LATEX è già possibile. Creare un file di tipo "Tagged PDF" richiede più lavoro che per il formato PDF/A e alcuni richiedono che si riscrivano certe macro LATEX per le intestazioni e per gli ambienti.

2. `$HOME` nei sistemi UNIX corrisponde alla cartella radice di ogni utente e viene solitamente abbreviata con il simbolo `~`. Nei sistemi Windows dal 2000 in poi essa corrisponde alla sottocartella contenuta in `C:\Document and Settings\` con il nome di ogni utente.

1. poco prima di `\begin{document}` eseguire la chiamata del pacchetto:

   ```
   \usepackage[a-1b]{pdfx}
   ```

   La necessità di ritardare questa chiamata dipende dal fatto che questo stesso pacchetto invoca direttamente hyperref con l'opzione `pdfa`. Perciò solo dopo questa chiamata è possibile configurare hyperref con i parametri che si desiderano; io, per esempio, generalmente specifico la seguente configurazione:

   ```
   \hypersetup{%
       pdfpagemode={UseOutlines},
       bookmarksopen,
       pdfstartview={FitH},
       colorlinks,
       linkcolor={blue},
       citecolor={green},
       urlcolor={blue}
   }
   ```

2. Come per la conversione eseguita mediante il programma ghostscript, è necessario predisporre un file che l'estensione pdfx richiamerà mediante l'inclusione del file `\jobname.xmpdata`. Si ricorda che `\jobname` è il nome privo di estensione del file principale che governa la composizione del documento. Quindi se si sta componendo il documento lanciando pdflatex sul file `pippo.tex`, sarà necessario avere predisposto un file `pippo.xmpdata` contenente le informazioni sul documento, i *metadata*, che servono per completare la composizione del file PDFA. Le varie informazioni che si possono inserire vengono attribuite a comandi interni mediante una sequenza di comandi esterni (accessibili al compositore) che sono descritti nel file `pdfx.pdf`. A titolo di esempio il file che ho usato per comporre il manualetto di *technical writing* `tw.tex` si chiama `tw.xmpdata` e contiene quanto segue

   ```
   \Title{Saper comunicare --%
       Cenni di scrittura t
       ecnico-scientifica}
   \Author{Claudio Beccari,
           Flavio Canavero, %
           Ugo Rossetti,
           Paolo Valabrega}
   \Keywords{pdfTeX</rdf:li>
   <rdf:li>Comunicazione
                   scritta</rdf:li>
   <rdf:li>Norme ISO e UNI</rdf:li>
   <rdf:li>Curriculum vitae</rdf:li>
   <rdf:li>Lettere</rdf:li>
   <rdf:li>Collaudi</rdf:li>
   <rdf:li>Perizie</rdf:li>
   <rdf:li>Tesi}
   \Org{Politecnico di Torino}
   ```

   La prima e l'ultima parola chiave nella lista delle 'Keywords' sono delimitate solo da un lato, perché i delimitatori mancanti sono già previsti nella lista che le accoglierà; le parole chiave intermedie, invece, sono delimitate da entrambi i lati dai comandi specifici XML da usarsi nei descrittori dei *metadata*. Chi volesse approfondire l'argomento può trovare abbondanti informazioni nel sito `www.pdfa.org` oppure nel libro Drümmer *et al.* (2007).

Va notato che la predisposizione della lista di *metadata* obbligatori o facoltativi da inserire in un file PDFA è una procedura piuttosto complessa, che richiede di rispettare scrupolosamente una grammatica fissata da regole molto precise; la Adobe mette a disposizione gratuitamente un pacchetto (Adobe, 2006) da compilare; se ne ottiene una adeguata libreria di routine da associare ad un 'main program' adeguato alla circostanza per scrivere correttamente il file `.xmp` che contiene i *metadata*; questo deve poi venire incluso nel file PDF affinché questo possa 'candidarsi' alla qualifica di file PDFA. Per fortuna ora il meccanismo prodotto mediante l'uso attento del pacchetto pdfx e dei suoi accessori evita all'utente comune di doversi studiare le finezze della grammatica XMP[3].

Il pacchetto pdfx si avvale di adeguati file 'maschera', (per i file PDFA si avvale del file `pdfa-1b.xmp` scritti in un misto di XML e LaTeX; è assolutamente vietato modificarli, non solo perché così è scritto nell'intestazione dei file, ma anche perché si tratta di cose talmente delicate che qualunque anche piccola modifica potrebbe renderli inservibili.

6 Prima conclusione

Per ora il programma 'accessorio' del sistema TeX che consente di ottenere con certezza in uscita file nel formato PDFA è solamente ghostscript configurato adeguatamente attraverso file di configurazione dipendenti dalla particolare piattaforma e dal sistema operativo. Anche il programma pdftex consente di compilare direttamente un file in formato PDFA, ma entrambe le soluzioni non danno la certezza del risultato, sebbene con un po' di ingegno si possano risolvere alcuni dei problemi che possono manifestarsi.

7 I problemi della trasformazione

Fin qui la descrizione che ho fatto indica che la configurazione adeguata di ghostscript e l'esecuzione di ghostscript su un file prodotto da pdflatex produce

3. Quando stavo per gettare la spugna, uno dei motivi era che non riuscivo a impostare il file `xmp` con la grammatica giusta, e non sono mai riuscito a trovare l'errore; eppure confrontando i miei tentativi con i *metadata* contenuti negli esempi di Hàn Thế Thành non sembrava che ci fosse nessuna differenza; evidentemente invece c'era!

senza incertezze un file PDFA. Anche l'esecuzione di pdflatex con l'ausilio dell'estensione pdfx produce senza incertezze un file PDFA. È vero? Talvolta sì, talvolta no.

Vediamo meglio il problema. Per sapere se un file è completamente conforme al formato PDF/A-1a oppure PDF/A-1b, bisogna ricorrere alla funzione di verifica espletata da `Preflight` di Acrobat Professional (versione 8 o successive). Alternativamente si potrebbe usare `Preflight` per trasformare in PDFA un file PDF comunque prodotto.

In entrambi i casi `Preflight` produce una diagnostica nel caso che il file presunto PDFA in effetti non lo sia, oppure che il file PDF da trasformare non sia trasformabile.

Ho riscontrato diversi problemi e anche alcuni presunti file PDFA prodotti da ghostscript o da pdflatex o da OpenOffice.org talvolta non sono conformi alle specifiche PDFA; analogamente ho incontrato diversi file PDF prodotti da altri programmi che non sono risultati trasformabili in file PDFA.

Si tratta in generale di problemi legati ai link ipertestuali, alle figure incluse nel file PDF, o nei font usati, ahimè, specialmente con i font tipici del sistema TEX.

Qui mi limiterò a descrivere le soluzioni che ho trovato per aggirare i problemi e arrivare finalmente ad un file che passa i test di verifica eseguiti con `Preflight` e in particolare mi limiterò alla creazione dei file PDF ottenibili con i programmi principali o accessori del sistema TEX, in concreto con pdflatex e/o con ghostscript.

7.1 Il modello di colore

Nella descrizione fatta per la conversione con ghostscript ho indicato come preferibile il modello di colore disponibile attraverso il file `Generic CMYK Profile.icc`; nella descrizione della compilazione diretta mediante pdflatex ho indicato il modello di colore descritto mediante il file `ECI-RGB.V1.0.icc`. È importante scegliere il modello di colore conformemente al tipo di colori usati nel file PDF; bisogna stare molto attenti a quanto si è detto e a quanto si dirà a proposito delle immagini importate e dei disegni creati internamente mediante le varie estensioni grafiche disponibili normalmente. Spesso non si ottiene la validazione del formato PDFA proprio a causa del modello di colore non conforme ai modelli di colore assunti nelle immagini importate o nei grafici prodotti internamente. Bisogna allora cambiare la specificazione del modello di colore dentro ai file `PDFA_def.ps` o `pdfx.sty` o, alternativamente, di convertire le immagini cambiando loro la codifica del colore.

7.2 Gli hyperlink

Gli hyperlink si possono evitare facilmente con pdflatex basta *non* fare uso del pacchetto hyperref.sty e dei comandi che esso definisce; per altro il pacchetto hyperref nella versione distribuita con TEXlive 2008 ammette una nuova opzione pdfa che modifica alcuni comandi interni ed elimina la definizione di alcuni altri in modo che non esistano problemi di compatibilità. Si tratta dunque di usare questa opzione se si vogliono conservare gli hyperlink interni che facilitano grandemente la consultazione di un documento *on line*.

7.3 Le figure

Le figure da inserire da file esterni possono avere il modello di colore RGB, invece del modello CMYK; si tenga presente che quasi tutte le immagini inseribili nel file PDF sono immagini configurate con il modello RGB, sia le figure in formato PNG (con compressione senza perdite, particolarmente indicato per i disegni al tratto, i disegni tecnici, e simili) sia le fotografie in formato JPEG (con compressione di ridondanza). Per i file in formato PDFA si suppone che la resa sia perfetta sullo schermo; per questo bisogna avere un modello di colore adeguato tenendo conto che le areole luminescenti (i 'fosfori') dello schermo emettono luce colorata, cosicché il modello di colore *deve* essere RGB.

Le illustrazioni prodotte internamente con i colori specificati mediante i pacchetti color.sty oppure xcolor.sty o, indirettamente, mediante l'uso di pacchetti di disegno come PSTricks oppure pgf e/o l'environment tikz, devono fare esplicitamente riferimento al modello di colore RGB.

Il problema delle figure nasce quando si suppone di poter usare il modello del grigio, visto che le figure inserite nel documento appaiono in bianco e nero; no, come detto sopra, le figure prodotte internamente con i programmi indicati o importate da file PNG e JPEG (gli unici, oltre ai formati PDF e MPS[4], che possono venire importati con pdflatex) sono sempre in realtà dei file composti con il modello di colore RGB[5].

7.4 Le immagini importate che contengono font

Un altro problema consiste nel fatto che spesso le figure create esternamente al file PDF o PDFA e importate mediante il pacchetto graphicx e il suo comando `\includegraphics` possono fare riferimento a font che però non vengono inclusi nel file che contiene l'immagine; finché si visualizza l'immagine con il programma che l'ha creata e/o sulla macchina usata per crearla, la figura appare completa e i font fanno bella mostra di sé.

4. Questo formato è il risultato della compilazione di un disegno descritto mediante il linguaggio METAPOST.

5. Per le fotografie, specialmente se un po' datate, possono venire usati anche altri formati e altri modelli di colore; Per queste foto sarebbe preferibile usare un programma di fotoritocco, come ad esempio GIMP, disponibile gratuitamente per i tre sistemi operativi, al fine di cambiarne il formato e il modello di colore.

Nel momento in cui questa figura viene esportata su un'altra macchina non dotata della stessa collezione di font della macchina 'madre', anche in formato PDF l'immagine appare mal composta e male etichettata, perché i font mancanti sono stati sostituiti con altri font. Per i file PDFA tutti, assolutamente tutti i font usati devono essere contenuti nel file, anche quelli usati nelle immagini.

Talvolta non è facile modificare il programma con cui si è creata ed etichettata l'immagine affinché incorpori tutti i font usati; talvolta non si dispone dell'accesso alla macchina 'madre'. La soluzione di questo problema consiste nell'elaborare l'immagine con il programma inkscape salvando il risultato sempre in formato PDF; i font mancanti vengono sostituiti con i loro disegni vettoriali, che sono la specialità di inkscape. La soluzione non sarà perfetta, ma è certamente accettabile ed elimina ogni problema di mancanza di font che impedirebbe la creazione e la certificazione dei file PDFA.

7.5 I font

I font usati normalmente dai programmi del sistema TEX sono: la collezione sviluppata da D.E. Knuth dei font Computer Modern (CM), la collezione European[6] Computer Modern (EC); la collezione dei font CM-super che contiene sia i font EC sia i font cirillici; la collezione Latin Modern (LM) sviluppata più recentemente ed estesa al formato OpenType; la collezione Concrete Computer Modern (CC); la collezione dei font cirillici CMCYR creata con i parametri dei font CM; eccetera.

Ogni distribuzione del sistema TEX comprende sia altre collezioni di font, sia i file di estensione per usarli.

Mi rincresce constatarlo, ma tutti i font tradizionali CM, EC, LM, CM-super danno problemi con la trasformazione dal formato PDF al formato PDFA. Oserei dire che i font CM sono i più critici, come lo sono tutti i font "derivati" che ne conservano le caratteristiche; forse i meno critici sono gli EC veri (degli EC contenuti nei CM-super non sono in grado di dire nulla, perché non li ho controllati). Tuttavia i font EC (e probabilmente anche quelli contenuti nei CM-super) sembra che producano meno problemi perché sono font usati solamente per comporre le parti testuali.

Invece la collezione CM serve anche per comporre la matematica; anche i font testuali corrispondenti alla forma tonda entrano a far parte della matematica perché, oltre a contenere un buon numero di simboli comuni, costituiscono l'alfabeto tondo necessario per la composizione dei nomi degli operatori matematici come 'sin', 'log', eccetera. Se anche si specifica la codifica `T1` per la composizione del documento, i font CM testuali (con la codifica OT1) entrano comunque nella composizione della matematica.

Se allora si produce un file PDFA, con qualunque codifica per il font da usare in uscita, o direttamente con pdflatex e il pacchetto pdfx, oppure attraverso la trasformazione in PDFA con ghostscript come descritto nelle sezioni precedenti, e poi lo si sottopone alla verifica mediante `Preflight`, le probabilità che il file non passi il test di conformità sono altissime. Gli errori che vengono segnalati in relazione ai font sono di due tipi: (*a*) mancanza di informazioni di larghezza per alcuni segni, e (*b*) informazioni dimensionali inconsistenti.

Nel sito di Hàn Thế Thành (2008a) egli segnala gli stessi problemi e suggerisce alcuni procedimenti per superarli. Io ho trovato altre soluzioni, ma purtroppo questo aspetto lascia un po' di amaro in bocca, visto che con il sistema TEX tutto dovrebbe filare liscio e meglio che con i soliti programmi WYSIWYG.

Sia ben chiara la differenza: la mancanza dell'informazione della larghezza dipende da un artificio per comporre segni matematici composti mediante la sovrapposizione o la giustapposizione di diversi elementi. Non è che i file metrici dei font manchino della larghezza, ma questa ha un valore nullo che per lo standard del linguaggio PDF equivale alla mancanza dell'informazione.

L'inconsistenza delle dimensioni dipende invece dal fatto che l'informazione contenuta nei file metrici differisce da quella contenuta nei file dei font scalabili (Type 1, essenzialmente). Per questo problema Hàn Thế Thành suggerisce di predisporre degli 'script' per esaminare la consistenza dei file metrici con i file PostScript Type 1 e porvi rimedio. La cosa migliore da fare, evidentemente, è quella di usare font che non presentino queste inconsistenze, ma questo lo si può sapere solo a posteriori.

L'applicativo `Preflight` permette di esaminare ciascun errore, per vedere in ogni pagina a quale punto della pagina si riferisca; lo si può ottenere sia eseguendo un doppio click sulla riga di segnalazione dell'errore che porta la schermata principale a mostrare la pagina contenente l'errore con adeguati marker rossi in corrispondenza del punto critico, oppure si può aprire una piccola finestra che mostra solo l'elemento errato.

Personalmente ho risolto il problema della mancanza di informazione sulla larghezza in due modi diversi.

Una prima soluzione è stata quella di evitare di usare i font CM e LM al fine di eliminare gli errori collegati all'inconsistenza delle informazioni metriche; ho usato il pacchetto mathdesign.sty

6. La prima 'E' viene anche interpretata come 'Extended'; la 'C' viene interpretata anche come 'Cork'; e il nome per esteso viene anche indicato con la locuzione 'Extended Cork encoding'. In realtà l'encoding dei font EC è davvero stato deciso dalla conferenza di Cork del 1990, ma il primo nome di quei font fu 'DC' e cambiò in 'EC' solo quando la versione beta fu rilasciata da Knappen come versione definitiva.

specificando di volta in volta per i font testuali ciascuna delle tre collezioni di font che il pacchetto fornisce: i Bitstream Charter, gli Adobe Utopia, oppure gli URW Garamond. Ho accompagnato questa scelta con gli Helvetica e i Courier, per le famiglie di font senza grazie e per quelli a spaziatura fissa; onestamente, secondo il mio gusto, i Courier sono un po' troppo chiari e un po' troppo larghi, ma la scelta è coerente con gli altri font. Nello stesso tempo questi font commerciali, disponibili gratuitamente per la comunità degli utenti del sistema TEX, sono disponibili solo nel corpo di 10 pt, per cui gli ingrandimenti e le riduzioni di corpo avvengono a partire dal solo corpo disponibile; le note vengono quindi un pochino troppo chiare e i titoli e titolini un pochino troppo scuri, rispetto a quello che si potrebbe ottenere con i font CM, EC, LM, (CM-super), che invece hanno diversi corpi iniziali con cui procedere per la composizione e per gli eventuali ingrandimenti e riduzioni.

Il pregio di usare il pacchetto mathdesign.sty è che i font matematici sono ottenuti in modo da andare d'accordo con la famiglia testuale con grazie specificata come opzione; per esempio, volendo usare i font Bitstream Charter, insieme agli Helvetica e ai Courier, basta inserire nel preambolo le dichiarazioni:

```
\usepackage[bitstream-charter]{mathdesign}
\usepackage[scaled=0.92]{helvet}
\usepackage{courier}
```

Il risultato della composizione è ineccepibile, anche se, abituati come siamo ai font tradizionali del sistema TEX, ci sembra tutto un po' strano, un po' differente (come per altro deve essere, altrimenti perché cambieremmo font?).

Tuttavia anche così facendo la trasformazione nel formato PDFA non va a buon fine; resta un solo errore relativo alla mancanza di informazione di larghezza per un segno[7]; esaminando in dettaglio e con l'aiuto di `Preflight` il file convertito, è risultato che l'errore era associato alla barra usata per comporre il segno ≠, che è ottenuto sovrapponendo al segno = la barra della negazione; per ottenere questa sovrapposizione il meccanismo più semplice escogitato da Knuth è stato quello di disporre di una forma particolare di barra priva di dimensioni orizzontali ottenibile con il comando `\not`; le dimensioni orizzontali *nulle* sono la causa dell'errore.

Riparare a questo errore richiede capacità che il neofita del sistema TEX non ha; e non le ha nemmeno il veterano del sistema TEX che non sia versato nella programmazione in linguaggi di programmazione non molto diffusi o nella manipolazione dei font con adeguati font editor. Infatti si tratta di modificare il comando `\not` o il file metrico del font insieme al file binario che contiene la descrizione PostScript del segno. Queste operazioni vanno ripetute per ogni segno privo di dimensioni orizzontali, e probabilmente per ogni segno delle collezioni CM e compagni le cui informazioni metriche risultano nulle (o inconsistenti). *Un lavoro non da poco, che dovrebbe essere curato dai responsabili delle varie collezioni di font distribuite insieme al sistema TEX.*

Qui comunque si mostrerà che cosa si è fatto per correggere il comando `\not` e i font che ne producono il segno. Questa strada è sostanzialmente quella indicata da Hàn Thế Thành, ma per mia esperienza non funziona sempre.

7.6 Come apportare qualche correzione ai font

Innanzi tutto dobbiamo renderci conto che dobbiamo lavorare con font PostScript di tipo 1 o font TrueType, o font OpenType. Abbiamo quindi bisogno di un programma di editing per font di questo tipo; esiste il programma FontForge, Open Source e gratuito, liberamente scaricabile dalla rete. Sulle macchine Windows è necessario disporre di un emulatore dell'ambiente UNIX, disponibile attraverso il programma gratuito CygWin liberamente scaricabile dalla rete; per le macchine UNIX/Linux/Mac OS X non è necessario null'altro che FontForge, senonché sulle macchine Mac è necessario disporre dell'ambiente grafico X11[8]. Se non si vuole o non si può installare il programma FontForge, il sistema TEX dispone di altri programmi privi di interfaccia grafica per svolgere le correzioni che servono. Per le altre operazioni il sistema TEX è autosufficiente, ma si badi che tutti i programmi da usare, non solo quelli di editing dei font, non hanno una interfaccia grafica e sono usabili solo dalla linea di comando.

1. Con il programma FontForge si apre il font dei simboli matematici relativi alla collezione che si sta usando; per l'opzione Charter di mathdesign.sty bisogna aprire il file `md-chr7y.pfb`; nella finestra che si apre bisogna eseguire un doppio click sul glifo della barra senza larghezza nella casella con indirizzo decimale 54 (esadecimale 36, ottale 66) e nella sotto finestra relativa al glifo bisogna cliccare la voce del menu 'Metrics' scegliendo 'Set width' e specificando il valore 1; bisogna salvare il font correttamente nel suo formato `.pfb` sostituendolo al file precedentemente esistente. Se si usano i programmi della distribuzione del sistema TEX, si userà il comando

   ```
   t1disasm md-chr7y.pfb md-chr7y.txt
   ```

7. Se nel file sorgente si fosse usato anche il comando matematico `\mapsto` gli errori sarebbero stati due.

8. Per le versioni del sistema operativo precedenti a quella denominata Leopard, questo ambiente X11 non è installato di default, ma è presente sul disco di installazione del sistema operativo e fa parte degli accessori standard forniti insieme al sistema.

per trasformare il file binario del font in un file testuale; con un editor testuale si cerchino le stringhe ␣0␣hsbw per sostituirle con le stringhe ␣1␣hsbw. Poi si riconverta il file testuale in file binario mediante il comando

```
t1asm md-chr7y.txt md-chr7y.pfb
```

Su quasi tutti i sistemi bisogna eseguire tutto ciò come amministratore o come 'root' o come 'superuser'.

2. Per modificare il file `.tfm` del nostro font si faccia riferimento al file `mdbchr7y.tfm` e dalla linea di comando lo si trasformi dal linguaggio macchina al formato ASCII in modo che si possano leggere le proprietà metriche del font; il comando da dare è

   ```
   tftopl mdbchr7y.tfm mdbchr7y.pl
   ```

   Si apra poi il file `mdbchr7y.pl` con un editor ASCII, per esempio `notepad` sulle macchine Windows, o `vim` sulle macchine UNIX e affini; in realtà la cosa migliore è quella di usare lo stesso editor che si usa abitualmente per creare il file `.tex` sorgente dei documenti da comporre. Si vada alla linea 433 dove sono definite le proprietà metriche del nostro carattere e si modifichi la riga successiva

   ```
   (CHARWD R 0.0)
   ```

   in

   ```
   (CHARWD R 0.001)
   ```

   Il numero 'magico' 0.001, non è altro che il risultato della divisione fra l'unità PostScript, precedentemente assegnata al segno, diviso per le 1000 unità che rappresentano la dimensione PostScript normalizzata del 'design size'. Si proceda ora alla conversione del file delle proprietà metriche appena modificato nel file `.tfm` mediante il comando:

   ```
   pltotf mdbchr7y.pl mdbchr7y.tfm
   ```

   e lo si salvi al posto del vecchio file, servendosi delle prerogative dell'amministratore, o dell'utente 'root' o del 'superuser', se fosse necessario.

3. Non è necessario ridefinire il comando `\not`, perché la dimensione assegnata al segno è talmente piccola che in stampa non si riesce a notare nessuna differenza rispetto alla larghezza nulla assunta nella operatività del comando `\not`.

È ovvio che tutto questo lavoro dovrebbe essere svolto dai curatori dei vari pacchetti e dei vari font, senza che venga ripetuto per ogni segno problematico da ogni possibile utente del sistema TEX! Tuttavia, in attesa che la grande macchina gestita e curata dal LATEX 3 Team produca i suoi risultati, bisogna fare di necessità virtù.

Questo modo di procedere non mi è andato a buon fine con i font della famiglia CM né con i font delle famiglie derivate. Non sono riuscito a capire il perché, ma tant'è. Può darsi che la cosa sia andata a buon fine con i font Bitstream Charter perché essi fanno uso di font virtuali che a loro volta rimandano a font reali in cui tutti i segni hanno le loro dimensioni; ma se devo essere onesto questa spiegazione non mi convince. Resta il fatto che con i normali font CM non sono riuscito a procedere per questa via e allora ne ho escogitata un'altra che mi pare più 'pulita', almeno nei casi in cui la si può percorrere.

Ho ridefinito il comando `\not` mediante l'uso di segni correttamente dimensionati, ma affidando l'annullamento delle dimensioni apparenti a comandi TEX di basso livello;

```
\renewcommand*\not{\mathrel{\mathchoice
  {\rlap{$\displaystyle
       \mkern2.5mu\mathnormal{/}$}}%
  {\rlap{$\textstyle
       \mkern2.5mu\mathnormal{/}$}}%
  {\rlap{$\scriptstyle
       \mkern2.5mu\mathnormal{/}$}}%
  {\rlap{$\scriptscriptstyle
       \mkern2.5mu\mathnormal{/}$}}%
}}
```

La barra inclinata è presa dal font matematico che fornisce anche il corsivo matematico; siccome questo segno è leggermente più verticale del segno usato normalmente per il comando `\not`, è necessario spostarlo un pochino a destra di 2,5 unità matematiche[9] per centrarlo sopra il segno da 'negare'. `\rlap` serve per produrre una scatola di larghezza nulla il cui contenuto sporge a destra; `\mathchoice` serve all'interprete TEX per scegliere il corpo giusto a seconda del modo di composizione della matematica in cui si trova ad operare; `\mathrel` serve invece per attribuire le caratteristiche di 'operatore di relazione' al comando appena definito, in modo che l'interprete possa attribuirgli la spaziatura giusta a seconda del segno da cui è seguito (in genere un altro operatore).

Il vantaggio di questa seconda soluzione, quando la si può usare, risiede nel fatto che non si debbono ritoccare i font, che nel caso della collezione CM

9. Una unità matematica è 1/18 del 'design size' nel font corrente; quindi lo stesso valore numerico produce uno spostamento relativo 'costante' indipendentemente dal fatto che il font sia quello della matematica in display, di quella in linea con il testo, o degli indici di primo o di secondo livello.

sarebbero cmsy5, cmsy6, cmsy7, cmsy8, cmsy9 e cmsy10. A suo tempo, agli albori di TEX questa soluzione era considerata poco attraente perché richiedeva un maggior tempo di elaborazione e un maggiore ingombro della memoria; se si tiene conto della lentezza dei 'clock' e delle dimensioni minime dei dischi e delle memorie RAM dei PC di 30 anni fa, ci si rende conto benissimo del perché Knuth abbia scelto la strada delle dimensioni nulle, invece di questa soluzione o di soluzioni simili.

8 Conclusioni

Devo dire che con il pacchetto pdfx il problema delle dimensioni orizzontali nulle è l'unico che si presentava; mentre con la trasformazione mediante ghostscript ho incontrato spesso anche il problema delle dimensioni inconsistenti, che sono riuscito a risolvere solo usando font alternativi.

Con il sistema TEX e con il programma accessorio ghostscript è possibile creare dei documenti conformi alle norme ISO19005-1 per il formato PDF/A-1b, ma l'operazione non è completamente automatica; bisogna fare attenzione a non pochi dettagli e talvolta è necessario modificare comandi interni di LATEX insieme alle prerogative dei font PostScript o TrueType o OpenType che si usano; queste operazioni sono difficili e non sono alla portata di ogni utente, se non dopo un notevole lavoro di apprendimento e una dose infinita di attenzione.

Mi auguro che il problema venga sentito anche dal LATEX-3 Team e che questo attivi i vari curatori dei font e dei pacchetti affinché i problemi oggi esistenti vengano rimossi. Nel frattempo ogni utente deve arrangiarsi da solo senza scoraggiarsi. L'utente sappia che il gioco vale la candela e che il risultato positivo viene sempre raggiunto, mentre con gli altri programmi il risultato positivo non è garantito e, più che altro, non è facile scoprire dove bisogna agire per superare i problemi che la diagnosi di `Preflight` mette in evidenza. Sarebbe anche molto utile se esistesse un programma gratuito per la verifica della conformità alle norme ISO di un file PDF presunto archiviabile.

Quello che ho potuto osservare è che non esiste, o almeno io non sono riuscito a trovarlo, un software gratuito che permetta di verificare la conformità di un file allo standard ISO 19005 relativo all'archiviabilità dei documenti. Il documento di Martini (2007) discute le performance di alcuni di questi software commerciali. Tuttavia finora non ho trovato nulla di altrettanto buono del plug-in Preflight di Acrobat Professional, che per altro è un software commerciale; forse l'Adobe potrebbe prendere in considerazione l'inserimento di una versione di Preflight, ridotta alla sola verifica di conformità, in Adobe Reader, il visualizzatore gratuito più usato e disponibile per tutte le piattaforme.

Ringraziamenti

Sono ben contento di esprimere il mio grazie più sentito a Gianluca Pignalberi che mi ha ben puntualizzato la differenza fra i vari tipi di software gratuito, libero, o freeware. A Luigi Scarso che mi ha messo a disposizione le presentazioni esposte dai congressisti al primo Congresso sull'archiviazione dei documenti svoltosi ad Amsterdam nel 2008. Al recensore delle varie versioni di questo articolo le cui osservazioni e i cui suggerimenti mi sono stati particolarmente utili. A Hàn Thế Thành che ha avuto la pazienza di rispondere ai miei messaggi dandomi preziose indicazioni, e, soprattutto, per aver fatto insieme a Radhakrishnan tutto quello che era necessario per rendere accessibile il formato PDFA attraverso pdflatex.

Riferimenti bibliografici

Adobe (2006). «Adobe's free xmp toolkit for reading/writing xmp». `http://www.adobe.com/devnet/xmp/sdk/eula.html`. In basso a sinistra nella schermata puntata dall'URL c'è l'ancora con il link per il download.

Drümmer, O., von Seggern, D. e Oettler, A. (2007). *PDF/A in a Nutshell: Long term archiving*. Callas Software GmbH. ISBN: 978-398116481-7.

Hàn Thế Thành (2008a). «Generating PDFA compliant PDFs from pdftex». `http://support.river-valley.com/wiki/index.php?title=Generating_PDF/A_compliant_PDFs_from_pdftex`.

— (2008b). «Web message». `http://sarovar.org/tracker/index.php?func=detail&aid=945&group_id=106&atid=495`.

Hàn Thế Thành, Rahtz, S., Hagen, H., Henkel, H., Jackowski, P. e Schröder, M. (2007). «The pdfTEX user manual». `http://www.ctan.org/get/systems/pdftex/pdftex-a.pdf`.

Martini, E. (2007). «Lo standard PDF/A: Sperimentazione di software per la verifica di conformità allo standard ISO 19005:2005». `http://www.cnipa.gov.it/site/_files/ref02_RS_3.1_martini.pdf`.

PDF/A (2005–2008). «Technical notes». `http://www.pdfa.org/`. Il sito contiene i testi delle annotazioni tecniche rappresentati dai file `tn0001_pdfa-1_and_namespaces_2008-03-19.pdf`, `tn0003_metadata_in_pdfa-1_2008-03-18.pdf`, `tn0006_digital_signatures_in_pdfa-1_2008-03-14.pdf`, `tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf`, `tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf`.

Radhakrishnan, C. e Hàn Thế Thành (2008). «Generation of PDF/X-1a and PDF/A-1b compliant PDF's with PDFTEX — `pdfx.sty`». `http://tug.ctan.org/tex-archive/macros/latex/contrib/pdfx/`.

▷ Claudio Beccari
`claudio dot beccari at gmail dot com`

Typesetting Coptic Liturgy in Bohairic

Claudio Beccari, George Kamel

Abstract

This paper describes what the authors have done in order to typeset some Coptic texts with LaTeX mainly in the Bohairic variant used in liturgy. This implied the creation of suitable fonts, the macros for typesetting special liturgical symbols, the hyphenation patterns necessary to typeset with the Coptic alphabet and the rules used by the Bohairic variant.

Sommario

In questo rapporto si espone quello che gli autori hanno fatto per poter comporre con LaTeX alcuni testi copti, specialmente nella variante bohairica usata nella liturgia. Questo ha comportato la creazione di font adatti allo scopo, le macro per comporre i simboli liturgici, i pattern di sillabazione necessari per usare i nuovi alfabeti e le regole della variante linguistica bohairica.

1 Introduction

Claudio Beccari (CB in the following) a few years ago developed a Coptic font on behalf of Cristiano Pulone, a philologist who was studying the Coptic courses written in a time span of several centuries from the IV century onwards.

The font was the one that many years earlier Karl Berry had used for testing a sort of tracing program in order to transform some outline fonts into METAFONT sources; Karl's sources were available on the net, and a certain font encoding was sort of crystallized within Karl's arrangement of the main glyphs; several other glyphs had to be designed in order to enrich the font facilities, and many macros had to be set up in order to typeset the philological notation of Coptic critical editions.

The font table is shown in figure 2. This particular version of the Coptic font was suitable for typesetting the alto medieval Sahidic variant of the language, although it also contained some glyphs suitable for other variants. More details may be read on the paper Beccari e Pulone (2006) that was published on TUGboat.

In 2008 George Kamel (GK in the following) wrote to CB in order to cooperate so as to set up another font suitable for typesetting liturgical Coptic.

The "modern" Coptic language (in all its variants) derives from the language spoken at the times of the Pharaohs, while its writing was done with hieroglyphs, or in a simplified form called hieratic, and eventually it was written in the demotic script; this script was very practical, but its signs did not have the elegance of hieroglyphs and was never used in formal inscriptions.

The Coptic Orthodox Church originated in the Hellenized area of northern area of Egypt, in the delta of the Nile river, the region of Alexandria, after the Apostle St. Mark announced the Evangel. Eventually, in the second century, a new script was developed by St. Pantaenus in order to transmit the "Good News" in the language of the local people but such as to be read by the missionaries, who had little experience with the demotic script; it had to be suitable also to represent the Hellenized words and personal names that appear in the Gospel. In practice the script was a slightly simplified version of the Greek alphabet, with a peculiar stylistic look, to which eight new glyphs were added to the twenty four Greek ones, so as to represent the sounds of the spoken language that were missing from the Greek language. The script of figure 2 is an example.

In northern Egypt the Bohairic variant of the Coptic language was used and this variant became the official language of the Coptic Church. Still today the Bible and other liturgical books are written in Bohairic Coptic, and even prayer books and icons are written with this variant of the language. The icon shown in figure 1, painted by Fadi Mikhail and reproduced with his permission, shows the name of the Apostle Mark written out in vertical; it reads: **ⲘⲀⲢⲔⲞⲤ ⲡⲓⲁⲡⲟⲥⲧⲟⲗⲟⲥ**, "Mark the apostle".

The spelling of the Bohairic liturgical texts is enriched with accents: the vowels may receive a grave accent for stress; the consonants, almost all of them, may receive a grave accent that indicates a vocalic indistinct utterance supporting the consonant (a schwa or an /e/) since very often the Bohairic words contain sequences of consonants that would be difficult to pronounce, at least for people not used to the Semitic languages where vowels are seldom written down.

GK desired a traditional looking font, not so old looking as the one represented in figure 2, but enriched with the necessary commands for accenting the Bohairic words without impairing the hyphenation algorithm and with some more symbols necessary for liturgical texts. He found a version of the Athanasius TrueType font that had the necessary look; he discovered also an extended version of the Athanasius TrueType font developed by the

Figure 1: An icon of St. Mark
(Courtesy of Fadi Mikhail, Fadi Mikhail (2008))

Coptic Standard Project, and Michael Sleman, one of the project leaders, kindly gave us permission to use and modify their font, Coptic Standard Project (2005).

Unfortunately, it was not possible to preserve the Coptic Standard Encoding provided by the Coptic Standard Project that Michael Sleman is leading; that encoding is oriented to the existing WYSIWYG word processors, while LaTeX, besides not being WYSIWYG, is a text processor with essentially 7-bit input characters; moreover there already existed a Coptic encoding for LaTeX, and it appeared reasonable to maintain that encoding. In order to facilitate the input of text already encoded with the Coptic Standard, a Python 2.x script was developed to convert text encoded in the Coptic Standard to the Coptic encoding for LaTeX. May be in the future, when LaTeX will become more UNICODE oriented, we will exploit the UNICODE encoding whose Coptic "page" came into existence very recently[1].

Since the *CS New Athanasius.ttf* has its own encoding and lacks the properties requested by GK, it was necessary to re-encode it and add some fifty glyphs in order to use the new font with pdfLaTeX and with all the LaTeX machinery for typesetting documents in two column bilingual texts. The final outcome of our efforts was the font represented in figure 3.

During the development of the work, we discovered the existence of a UNICODE encoded TrueType font that contained also the Coptic page, FreeSerifAvvaShenouda.ttf, WAZU JAPAN (2006); we decided that the Coptic page, adequately extracted and re-encoded, would have been a good lighter variant than the Athanasius font, and we decided to use both; the latter for boldface and the former for medium weight.

2 The Type 1 Athanasius and CBbohairicmedium fonts

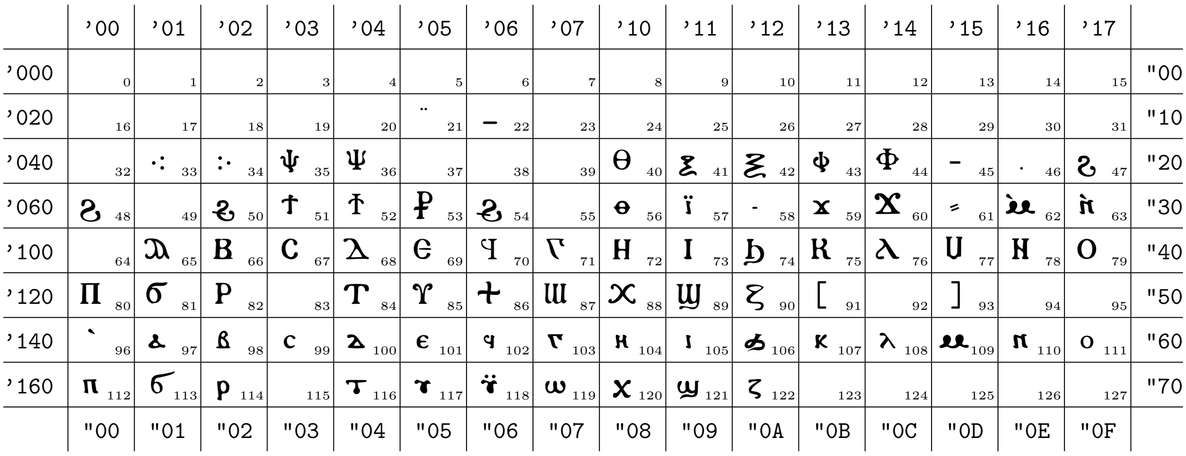
In order to transform the TrueType fonts, Athanasius.ttf and FreeSerifAvvaShenouda.ttf[2] we used the program FontForge, Williams (2008); this is a freeware program for creating and editing a variety of outline fonts; its performance is remarkable, although it's necessary to get experience with it.

The first step was to copy in the proper position the glyphs of the TrueType font into a new font we called `athanasius3`. We added the missing glyphs and in particular we added the accented ones, both upper and lower case, since all of them may take the grave accent; we decided to map the grave accent on the apostrophe, so we are sure that this sign is present on all Latin keyboards; in fact to our knowledge, there is not a standard Coptic keyboard, in spite of the efforts of the Coptic Standard Project. It is therefore necessary to input all characters by means of a Latin keyboard trying to maintain a certain association between the Coptic glyphs and the Latin ones, either by form or by sound; of course it's impossible to map the 26 letters of the Latin keyboard to the 32 or 34 Coptic letters. On our side there was the necessity to maintain the initial encoding established by the Coptic font developed for the Sahidic variant.

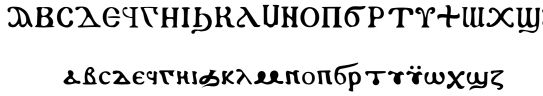
Some glyphs, we discovered, were originally made up of several superimposed strokes; these glyphs should have been classified as Type 3 fonts, but, evidently, TrueType fonts don't mind these inconsistencies. We had to patch such glyphs with single contour counterparts, and eventually we generated the Type 1 version of the font.

1. Although, the existing UNICODE Coptic page does not contain synthetic accented characters so that even with XeLaTeX (the pdfLaTeX version that can deal with scalable UNICODE fonts) the hyphenation process gets impaired.

2. Since the Type 1 version of the Athanasius font contains essentially the same glyphs as the original TrueType font, we decided to maintain the name with the addition of a digit; on the opposite the Type 1 version of the FreeSerifAvvaSenouda contains only a minimal part of the UNICODE encoded TrueType font, so we decided to change the name into CBbohairicmedium, so that this subset is easily distinguishable from the original complete set.



Font: copto.



Font parameters

slant per pt	0.0pt
inter word space	3.99998pt
inter word stretch	1.99998pt
inter word shrink	1.99998pt
x-height	5.59999pt
quad width	10.0pt
extra space	1.0pt

FIGURE 2: The Coptic font suitable for typesetting the Sahidic variant of the language

Later on we had to change the file we were working on because there was an important mismatch between the declared design size and the real one; we therefore scaled the FontForge glyphs, corrected the `tfm` files and eventually we ended up with the Type 1 modified Athanasius font named `athanasius6.pfb` represented in figure 3.

We performed the same operations in the CB-bohairicmedium font, although there were fewer glyphs to correct; the most important one was the accent glyph, because it was too thin. We decided to add the specific liturgical symbols identical, even in weight, to those of the athanasius6 font; these symbols, in fact, according to our perception of the typesetting activity with Coptic liturgy, appear mostly either in normal text, or as isolated instances around pictures or in some versicle beginnings.

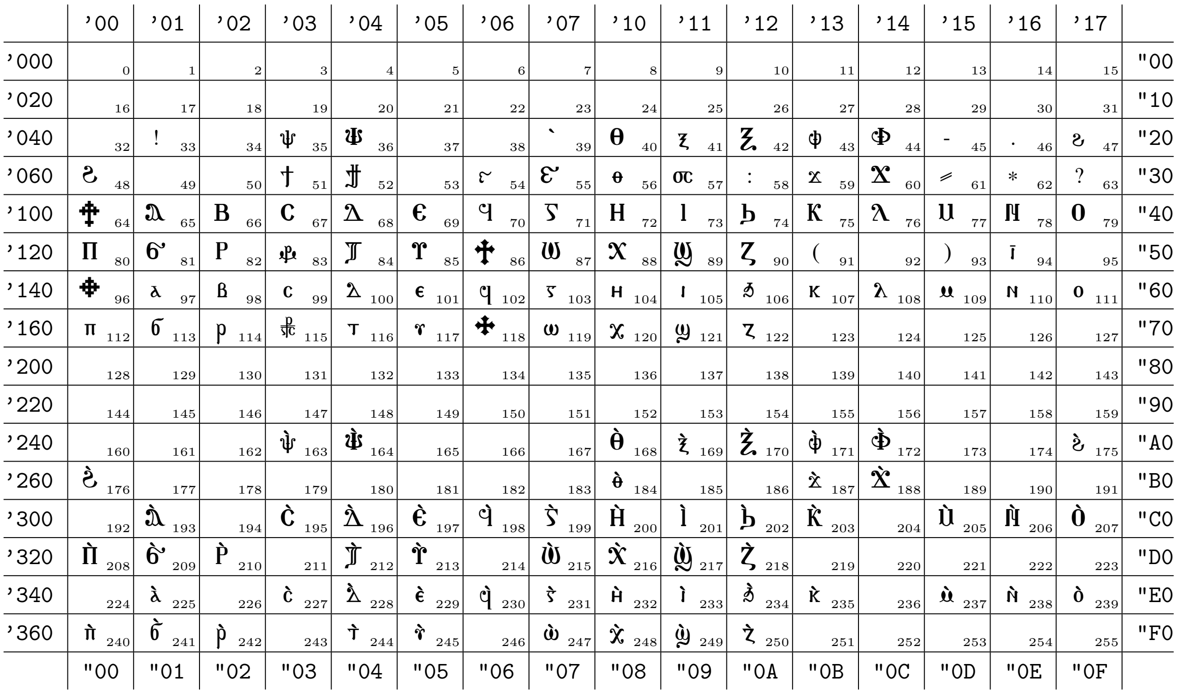
3 The encoding and the TEX metric files

FontForge is capable of outputting both the binary font files for the Type 1 fonts, and the `afm` and `tfm` files. Actually FontForge is capable of setting also the kerns and the ligatures, but CB preferred to work on the TEX side of the problem.

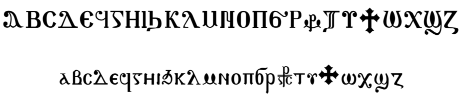
For what concerns the encoding it was not difficult to start with the one output by FontForge and to edit it in order to put the proper names for each glyph. Each Coptic glyph has a Coptic name that we extracted from the UNICODE page; it must be noticed that the Coptic page names are generally different from the ones in the "old" UNICODE where the Greek subset shared the same names as those of the Greek glyphs; the result is that FontForge very often is incapable of labelling correctly the various font boxes in the program's main view. Of course the accented glyph names exist only for the vowels with Greek names. The created encoding was forced onto the generated font so that this side of the problem was solved.

On the side of the `tfm` file, CB used the application `tftopl` to have a readable and editable ASCII file, and added ligatures and kerns "by hand"; after each group of modification he would control with a suitable test file the results and then interact with the property list `.pl` file in order to edit it again or to keep the modifications; at each test, of course, he would use the inverse application `pltotf` in order to create the corresponding TEX metric file so that LATEX could typeset the test file.

The resulting ligature table was very long because of the ligatures between the accent (the Latin



Font: cbbohairicmedium.



Font parameters

slant per pt	0.0pt
inter word space	3.29999pt
inter word stretch	1.65pt
inter word shrink	1.09999pt
x-height	4.79999pt
quad width	10.0pt
extra space	1.09999pt

FIGURE 3: The Coptic font suitable for typesetting the Bohairic variant of the language

apostrophe) and almost all lower and upper case letters. Very few kerns had to be set; may be this question should be further examined and more kerns introduced.

4 The actual mapping of the Coptic fonts to the Latin keyboard

As previously mentioned, to our knowledge there does not exist a (hardware) Coptic keyboard. We had to attack this problem with a reasonable mapping to the Latin keys, either by shape or by pronunciation, and we had to make it up with the fact that the Latin letter keys are just 26, while the Bohairic letters are 34. We had to invade the other Latin keys and we chose the number keys and some punctuation keys so as to avoid conflicts with the keys that input TeX-related special textual symbols; we had to cope with both lower and upper case extra letters, so we needed 14 extra key strokes to access the seven couples of extra Bohairic letters.

We also needed some keys to be assigned to special liturgical symbols, even if these were also accessible through specific macros; we wanted to leave free the addresses from 0 to 31, as well as from 128 to 159 that are usually assigned to control codes; TeX does not care, but if the font is used outside the TeX system it's better if these codes remain available for the operating system.

Therefore, figure 3 shows a font table that is pretty full, with the exception of the first two rows and the corresponding rows in the second part of the 8-bit encoding.

Table 1 displays the mapping of the Latin keys to the Coptic glyphs corresponding to the first half of the font table. The second half contains only the accented glyphs; most of these glyphs may be obtained in one of the following ways:

1. just prefixing the letter to be accented with an apostrophe; the font ligature mechanism simply replaces the sequence of the apostrophe and the letter with the corresponding accented glyph;

2. by using the `\'` accent macro; this macro is redone in a similar way as when the T1 encoding is selected, so that the composite sequence `\'{`⟨*letter*⟩`}` (or simply `\'`⟨*letter*⟩) directly maps the result to the accented letter.

These methods are almost equivalent; where they show some differences is in hyphenation, in particular in the definition of the first or last syllable length; with the first method the ligature mechanism outputs one character, but the input word string contains two characters; with the second method there is one character both in input and output; therefore hyphen points may be different in a given text; at the same time the first input method is simpler than the second one. The choice of which to use is up to the compositor.

5 The hyphenation patterns

Of course, in order to typeset in a certain language it's best to have an efficient and correct hyphenation algorithm.

As everybody knows, the hyphenation algorithm used by all the TEX system typesetting programs is based on patterns. These are letter sequences that may be thought of as complete word building blocks, or as word fragments. Each pattern encodes the possibility of hyphenating between two given letters in the pattern sequence, and priorities are established by means of weights that indicate if hyphenation between those two letters is very important, or less important; there are five levels of importance that may override opposite weights; permission to hyphenate is given by an odd numbered weight in the range from 1 to 9, while prohibition to hyphenate is given by an even numbered weight in the range from 0 to 8. The same pair of letters may appear in different patterns with different weights; if two or more different patterns may be found as word fragments of a certain word, the highest weight between those two letters is the one that eventually applies to the final hyphenation of that word.

The mechanism is simple and very efficient, but the patterns of each language must be preliminarily loaded into the format file of the working version of TEX, so as to be easily processed at typesetting time. This format file also contains the macro definitions translated into the internal TEX language so that they can be processed and executed much faster at typesetting time.

For what concerns the patterns there are two distinct approaches that are applicable for their generation:

Table 1: Mapping of the Latin keys to the Coptic glyphs

Latin key(s)	Coptic glyph
a, A	ⲁ, Ⲁ
b, B	ⲃ, Ⲃ
c, C	ⲥ, Ⲥ
d, D	ⲇ, Ⲇ
e, E	ⲉ, Ⲉ
f, F	ⲫ, Ⲫ
g, G	ⲅ, Ⲅ
h, H	ⲏ, Ⲏ
i, I	ⲓ, Ⲓ
j or hj; J or HJ or Hj	ϫ, Ϫ
k, K	ⲕ, Ⲕ
l, L	ⲗ, Ⲗ
m, M	ⲙ, Ⲙ
n, N	ⲛ, Ⲛ
o, O	ⲟ, Ⲟ
p, P	ⲡ, Ⲡ
q, Q	ⲧ, Ⲧ’
r, R	ⲣ, Ⲣ
t, T	ⲧ, Ⲧ
u, U	ⲩ, Ⲩ
w, W	ⲱ, Ⲱ
x, X	ⲭ, Ⲭ
y, Y	ⲯ, Ⲯ
z, Z	ⲍ, Ⲍ
8; (or 81	ⲑ, Ⲑ
) or ks; * or KS or Ks	ⲝ, Ⲝ
+ or p1; , or P1	ϕ, Ⲫ
# or ps; $ or PS or Ps	ψ, Ⲯ
; or dj or d1; < or DJ or Dj or D1	ϧ, Ϧ
3 or tj; 4 or TJ or Tj	ϯ, Ϯ
/ or h1; 0 or H1	ⲋ, Ⲋ
6, 61	ⳁ, Ⳁ
[,]	(,)
`\Asterisk`, .	*, .
-, =	-, ⸗
9, `\chois`	ⲍ̄, Ⲍ̄
`\estavros`, `\martyros`	⳨, ⳩
`\shortcross`, `\longcross`	✣, ✝
`\varshortcross`, `\varlongcross`	✣, ✝

1. precise and simple hyphenation rules established by the language grammar, or

2. hyphenated word extensive lists from which a suitable program, `patgen`, can create the patterns.

English patterns, for example, are generated through `patgen` (actually `patgen` was conceived for the very purpose of creating the English patterns), while Italian patterns are generated from grammar rules.

For the Sahidic variant of Coptic the patterns were generated by means of grammar rules; for the Bohairic variant of Coptic we followed the same approach, except that the process was much

more complicated due to the presence of accented consonants, to the agglutination of articles and pronouns, to the presence of "syllables" that do not follow the general grammatical rule, and the like. Nevertheless we proceeded by hand, since a hyphenated word list did not exist, and to create one was definitely too complicated and time consuming.

The file `bohahyph.tex` was eventually prepared; we are not satisfied yet, because although we could deal correctly with the accent-letter sequences, we could not attack in the proper way the single glyph patterns, when these were in the second part of the font page. The results are acceptable, but certainly better results might be achieved.

6 The font description file

In order to use (pdf)LaTeX in a proper way a font description file is also necessary; its name is made up with the agglutination of the font encoding (in lowercase letters) and the family name; the extension must be fd; in our case this file name is `lbohcoptic.fd`.

This file contains the declaration of the encoding and the family and the specification of which actual fonts must be loaded for each series, shape and size of every font belonging to that family. In our case, where we have available only scalable fonts, and these are pretty dark, we have only one size to load, two series, and two shapes. We have the upright shape and the slanted one that we used also for the italic declaration. We did not actually create a slanted font; since we plan to use only the pdfLaTeX typesetting program, we exploited its ability to perform some transformation on the fonts themselves; in the map file we need to configure the typesetting program, we declared both the upright and its slant font transformation, both of which actually load the same binary font file.

The only thing we had to create on purpose was the `tfm` file for the slanted version; starting from the property list file, it was pretty simple to edit the single line where the slant is specified and to create the tfm file from this edited property list file.

In practice, the single `athanasius6.pfb` file, that contains in binary form the data concerning the Type 1 scalable font athanasius6, is accompanied by two `.tfm` files, `athanasius6.tfm` and `athanasius6o.tfm`, and two entries into the map file. Similarly the single `cbbohairicmedium.pfb`, that contains in binary form the data concerning the Type 1 scalable font CBbohairicmedium, is accompanied by two `.tfm` files, `cbbohairicmedium.tfm` and `cbbohairicmediumo.tfm` and two entries in the map file.

7 The Bohairic extension package

We thought that it was not possible, at least in this early stages of the project development, to create a real language definition file, but that it was feasible to define some macros capable of switching alphabet and hyphenation rules, besides defining language or encoding specific macros.

We eventually produced the bohairic.sty file where all font encoding and language specific macros are defined; the results are acceptable and the tests we made until now exhibit pretty good typesetting quality.

8 The Bohairic Nicene Creed

As an example in figure 4 there is the trilingual Nicene Creed typeset in three columns with parallel paragraphs.

Those who can read Greek will find no difficulty in fetching the usual personal names that appear in the Creed; they will also recognize some common words of Greek etymology. The rest is understandable only by those who can read Coptic, of course, be they members of the Coptic Church or scholars dedicated to this particular language.

At the same time, from a typographical point of view, while the CMbohairicmedium is just a little darker than the medium Latin roman font, the Athanasius font is much darker; this is why we defined the Athanasius font as the boldface variant of the font. The design size turned out to be well tuned to the standard TeX fonts (here the Latin Modern are used), and the baseline skip, obviously with the same value in the three texts in order to get them in parallel, is well suited for both scripts.

The example of Coptic Bohairic text shown in figure 4 was completely done by accenting the various letters with the simple ligature mechanism of the apostrophe with the following letter; it may be easily recognized from the presence of one letter first syllables in several broken lines. This has been explained in the previous sections.

Definitely some more work must be done, but the start is encouraging; the purpose of this work is the typesetting of devotional writings; the purpose is elevated, let's hope it to be adequate for good work.

References

Beccari, C. e Pulone, C. (2006). «Facilities for typesetting Coptic texts». *TUGboat*, **22** (2), pp. 187–192.

Coptic Standard Project (2005). «CSAthansius.ttf font». `http://copticchurch.net/coptic_fonts/`.

Fadi Mikhail (2008). «St. Mark icon». `http://www.ukcopticicons.com/`.

The Nicene Creed in three columns

Crediamo in un solo Dio Padre, onnipotente, creatore del cielo e della terra, di tutti gli esseri visibili e invisibili.

E in un solo Signore Gesù Cristo, il Figlio di Dio, l'Unigenito, generato dal Padre prima di tutti i secoli, Luce da Luce, Dio vero da Dio vero, generato, non creato, consustanziale al Padre, per mezzo del quale sono state create tutte le cose.

Egli per noi, gli uomini, e per la nostra salvezza è disceso dai cieli, si è incarnato da Spirito Santo e Maria, la Vergine, e si è fatto uomo.

E fu crocifisso per noi sotto Ponzio Pilato, patì e fu sepolto, e risuscitò il terzo giorno secondo le Scritture, e ascese ai cieli, e siede alla destra del Padre, e di nuovo verrà con gloria a giudicare i vivi e i morti; del suo regno non ci sarà fine.

E nello Spirito, che è Santo, che è Signore, che dà la Vita, e procede dal Padre, è adorato e glorificato insieme con il Padre e il Figlio, ha parlato per mezzo dei profeti. In una sola Chiesa, santa, cattolica, apostolica. Confessiamo un solo battesimo in remissione dei peccati, attendiamo la risurrezione dei morti e la vita del tempo futuro. Amen.

We believe in One God, the Almighty God, the Father, maker of heaven and earth, of all things visible and invisible.

We believe in one Lord, Jesus Christ the only begotten Son of God, born of the Father before all ages; light out of light, true God out of true God, begotten not made; consubstantial with the Father, through whom all things came into being.

He descended from heaven for us and for our salvation, and was incarnated from the Holy Spirit and of the Virgin Mary, and became man.

He was crucified for us at the time of Pontius Pilate. He suffered and was buried; arose from the dead on the third day in accordance with the Scriptures; He ascended to the heavens and sat at the right hand of the Father; He shall also come in His glory to judge the living and the dead; of whose kingdom there will be no end.

Truly we believe in the Holy Spirit, the Life-giving Lord, who proceeds from the Father, we worship and glorify Him together with the Father and the Son, who spoke in the prophets. And in one, holy, universal and Apostolic Church. We acknowledge one baptism for the remission of sins. And we look for the resurrection of the dead and the life of the world to come. Amen.

Ⲓⲉⲛⲛⲁⲍⲧ ⲉⲟⲩⲛⲟⲩϯ ⲛ̀ⲟⲩⲱⲧ: Ⲫⲛⲟⲩϯ Ⲫⲓⲱⲧ ⲡⲓⲡⲁⲛⲧⲟⲕⲣⲁⲧⲱⲣ: ⲫⲏⲧⲁϥⲑⲁⲙⲓⲟ̀ ⲛ̀ⲧⲫⲉ ⲛⲉⲙ ⲛ̀ⲕⲁⲍⲓ: ⲛⲏⲧⲟⲩⲛⲁⲩ ⲉ̀ⲣⲱⲟⲩ ⲛⲉⲙ ⲛⲏⲧⲉ ⲛ̀ⲥⲉⲛⲁⲩ ⲉ̀ⲣⲱⲟⲩ ⲁⲛ.

Ⲓⲉⲛⲛⲁⲍⲧ ⲉⲟⲩⲃⲟⲓⲥ ⲛ̀ⲟⲩⲱⲧ Ⲓⲏⲥⲟⲩⲥ Ⲡⲓⲭ̀ⲣⲓⲥⲧⲟⲥ Ⲡ̀ϣⲏⲣⲓ ⲙ̀Ⲫⲛⲟⲩϯ ⲡⲓⲙⲟⲛⲟⲅⲉⲛⲏⲥ: ⲡⲓⲙⲓⲥⲓ ⲉⲃⲟⲗϩⲉⲛ Ⲫⲓⲱⲧ ϩⲁϫⲱⲟⲩ ⲛ̀ⲛⲓⲉ̀ⲱⲛ ⲑⲣⲟⲩ: ⲟⲩⲟⲩⲱⲓⲛⲓ ⲉⲃⲟⲗϩⲉⲛ ⲟⲩⲟⲩⲱⲓⲛⲓ: ⲟⲩⲛⲟⲩϯ ⲛ̀ⲧⲁⲫ̀ⲙⲏⲓ ⲉⲃⲟⲗϩⲉⲛ ⲟⲩⲛⲟⲩϯ ⲛ̀ⲧⲁⲫ̀ⲙⲏⲓ: ⲟⲩⲙⲓⲥⲓ ⲡⲉ ⲟⲩⲑⲁⲙⲓⲟ̀ ⲁⲛ ⲡⲉ: ⲟⲩⲙⲟⲟⲩⲥⲓⲟⲥ ⲡⲉ ⲛⲉⲙ Ⲫⲓⲱⲧ: ⲫⲏⲧⲁ ϩⲱⲃ ⲛⲓⲃⲉⲛ ⲱⲱⲡⲓ ⲉⲃⲟⲗϩⲓⲧⲟⲧⲥ.

Ⲫⲁⲓ ⲉ̀ⲧⲉ ⲉⲑⲃⲏⲧⲉⲛ ⲁ̀ⲛⲟⲛ ϩⲁ ⲛⲓⲣⲱⲙⲓ ⲛⲉⲙ ⲉⲑⲃⲉ ⲡⲉⲛⲟⲩϫⲁⲓ: ⲁⲥⲓ̀ ⲉ̀ⲡⲉⲥⲏⲧ ⲉⲃⲟⲗϩⲉⲛ ⲧ̀ⲫⲉ: ⲁⲥⲃⲓⲥⲁⲣⲝ ⲉ̀ⲃⲟⲗϩⲉⲛ Ⲡⲓⲡ̃ⲛⲉⲩⲙⲁ ⲉⲑⲟⲩⲁⲃ ⲛⲉⲙ ⲉ̀ⲃⲟⲗϩⲉⲛ Ⲙⲁⲣⲓⲁ̀ ⲧⲡⲁⲣⲑⲉⲛⲟⲥ ⲟⲩⲟⲩ ⲁⲥⲉⲣⲣⲱⲙⲓ.

Ⲟⲩⲟⲩ ⲁⲩⲉⲣⲥ̀ⲧⲁⲩⲣⲱⲛⲓⲛ ⲙ̀ⲙⲟⲥ ⲉ̀ⲃ̀ⲣⲏⲓ ⲉ̀ϫⲱⲛ ⲛⲁⲍⲣⲉⲛ Ⲡⲟⲛⲧⲓⲟⲥ Ⲡⲓⲗⲁⲧⲟⲥ: ⲁⲥϣⲉⲡⲉⲙⲕⲁⲩ ⲟⲩⲟⲩ ⲁⲩⲕⲟⲥⲥ. **Ⲟⲩⲟⲩ** ⲁⲥⲧⲱⲛⲥ ⲉⲃⲟⲗϩⲉⲛ ⲛⲏⲉⲑⲙⲱⲟⲩⲧ ϩⲉⲛ ⲡⲓⲉ̀ⲃⲟⲟⲩ ⲙ̀ⲙⲁⲍⲱⲟⲩⲧ ⲕⲁⲧⲁ ⲛⲓⲃ̀ⲣⲁⲫⲏ. Ⲁⲥϣⲉⲛⲁⲥ ⲉ̀ⲛ̃ϣⲱⲓ ⲉ̀ⲛⲓⲫⲏⲟⲩⲓ̀: ⲁⲥⲉⲓⲙⲓ ⲥⲁⲟⲩⲓ̀ⲛⲁⲙ ⲙ̀ⲡⲉϥⲓⲱⲧ. **Ⲕⲉ** ⲡⲁⲗⲓⲛ ϥⲛⲏⲟⲩ ϩⲉⲛ ⲡⲉϥⲱⲟⲩ ⲉ̀ⲧⲁⲓ ⲉⲛⲏⲉⲧⲟⲛϩ ⲛⲉⲙ ⲛⲏⲉⲑⲙⲱⲟⲩⲧ: ⲫⲏⲧⲉ ⲧⲉϥⲙⲉⲧⲟⲩⲣⲟ ⲟⲩⲁⲑⲙⲟⲩⲛⲕ ⲧⲉ.

Ⲥⲉ ⲧⲉⲛⲛⲁⲍⲧ ⲉ̀Ⲡⲓⲡ̃ⲛⲉⲩⲙⲁ ⲉⲑⲟⲩⲁⲃ: **Ⲡ̀**ⲃⲟⲓⲥ ⲛ̀ⲣⲉϥⲧ ⲙ̀ⲡ̃ⲱⲛϩ: ϕⲏⲉⲑⲛⲏⲟⲩ ⲉⲃⲟⲗϩⲉⲛ Ⲫⲓⲱⲧ: ⲥⲉⲟⲩⲱⲱⲧ ⲙ̀ⲙⲟⲥ ⲥⲉⲧ̀ⲱⲟⲩ ⲛⲁϥ ⲛⲉⲙ Ⲫⲓⲱⲧ ⲛⲉⲙ Ⲡ̀ϣⲏⲣⲓ: ϕⲏⲉⲧⲁϥⲥⲁϫⲓ ϩⲉⲛ ⲛⲓⲡ̀ⲣⲟⲫⲏⲧⲏⲥ. **Ⲉ**ⲟⲩⲓ̀ ⲛ̀ⲁⲓⲁ̀ ⲛ̀ⲕⲁⲑⲟⲗⲓⲕⲏ ⲛ̀ⲁⲡⲟⲥⲧⲟⲗⲓⲕⲏ ⲛⲉⲕⲕⲗⲏⲥⲓⲁ. Ⲓⲉⲛⲉⲣ̀ⲙⲟⲗⲟⲅⲓⲛ ⲛⲟⲩⲱⲙⲥ ⲛⲟⲩⲱⲧ ⲉ̀ⲛ̃ϫⲱ ⲉ̀ⲃⲟⲗ ⲛ̀ⲧⲉ ⲛⲓⲛⲟⲃⲓ. Ⲓⲉⲛϫⲟⲩⲱⲧ ⲉ̀ⲃⲟⲗ ϩⲁⲧ̀ϩⲏ ⲛ̀ⲧⲁ̀ⲛⲁⲥⲧⲁⲥⲓⲥ ⲛ̀ⲧⲉ ⲛⲓⲣⲉϥⲙⲱⲟⲩⲧ: ⲛⲉⲙ ⲡⲓⲱⲛϩ ⲛ̀ⲧⲉ ⲡⲓⲉ̀ⲱⲛ ⲉⲑⲛⲏⲟⲩ: ⲁ̀ⲙⲏⲛ.

Figure 4: The Nicene Creed typeset in Italian, in English and in Bohairic Coptic

WAZU JAPAN (2006). «Gallery of Unicode Fonts». http://www.wazu.jp/gallery/Fonts_Coptic.html.

Williams, G. (2000-2008). «Fontforge». http://fontforge.sourceforge.net/.

▷ Claudio Beccari
Villarbasse (TO), Italy
claudio dot beccari at gmail dot com

▷ George Kamel
Hampton, United Kingdom
georgekamel at gmail dot com

Il PostScript in LaTeX

Riccardo Nisi

Sommario

L'ambiente PSTricks con i suoi collegamenti al PostScript mi ha dato lo spunto per interessarmi a questo linguaggio cercando ulteriori occasioni per integrarlo con LaTeX, anche accrescendone il campo applicativo.

Abstract

PSTricks, along with his links to PostScript, gave me the chance to take an interest in this language, searching for further occasions to integrate it with LaTeX, and enrich its application scope.

1 Introduzione

I documenti TeX e LaTeX, da quelli semplici a quelli complessi, sono in grado di soddisfare le esigenze più varie e, se si considera la disponibilità dei numerosissimi *package*, anche quelle più particolari. Ad esempio la grafica, specialmente in ambiente scientifico, è ben risolta dai pacchetti PSTricks.

Certo per calcolare il codominio di una funzione reale l'ambiente di calcolo non può essere TeX. LaTeX crea il documento in cui si scrive del contesto della funzione e in cui la si stampa, ma è PSTricks che ricorrendo all'ambiente PostScript calcola le coordinate della funzione e traccia il grafico. LaTeX gestisce tutto così bene che quasi non ci si accorge della sortita. Per riconoscere la diversità dell'ambiente occorre leggere o pensare di scrivere l'equazione che calcola le coordinate della funzione o della curva, soprattutto con una versione non recentissima di PSTricks. In alcuni casi, tuttavia, quando si tratta di effettuare calcoli con delle distanze, è possibile utilizzare direttamente TeX, come negli esempi della punteggiata e della spirale archimedea in cui si confrontano l'uso della macro `\multido` con il calcolo effettuato da TeX e dal PostScript. Ma TeX limita il calcolo numerico agli interi. È proprio questo limite che, a partire dal caso di una elementare tabella che prevedeva oltre alla rappresentazione di dati anche il calcolo di alcune percentuali con cifre decimali, mi ha spinto a guardare al PostScript. In questo caso, come nei due precedenti, l'ho fatto comparando procedimento e risultato del PostScript con quello di TeX. Nei casi delle curve e del calcolo dei coefficienti angolari non c'è alternativa all'uso del PostScript.

Poiché l'approccio al PostScript, pur limitato ad alcuni aspetti semplici, ha significato per un autodidatta come me un notevole anche se stimolante impegno, ho cercato di raccontarlo provando a chiarire, nella misura parziale e provvisoria di cui sono stato capace, alcuni aspetti di quel linguaggio che a mio parere sono essenziali per iniziare a comprenderlo. Questo lo farò nella sezione 2.2 *Esecuzione del codice* e in maniera più particolareggiata nella presentazione degli esempi e nel loro commento.

2 Uno sguardo al PostScript

Il PostScript (Adobe System Inc., 1991; Utting, 1992), un vero e proprio linguaggio di programmazione, nasce per descrivere la pagina di un documento con una resa grafica ottimale. L'interprete PostScript utilizza gli oggetti del linguaggio immettendoli in una memoria particolare, lo stack o pila, in cui l'ultimo elemento immesso è il primo ad essere estratto: gli elementi sono letti ed elaborati nell'ordine inverso a quello in cui sono immessi. Proprio da questa gestione degli oggetti del linguaggio discende la sua caratteristica notazione postfissa in cui l'operatore segue gli operandi: `a b add`. Il primo elemento ad essere immesso nello stack è l'operando `a` (un numero reale), il secondo è il numero `b` e infine l'operatore `add`, che è il più alto dello stack, detto stack degli operandi. La lettura da destra a sinistra dice a noi e all'interprete PostScript che l'operatore ha i due operandi previsti e che l'operazione può essere eseguita: i tre elementi della operazione vengono tolti dallo stack e viene eseguita la somma, inserita a sua volta nello stack.

L'area su cui si appoggiano tutte le comunicazioni, per passare i parametri ed eseguire le operazioni del linguaggio, è lo stack degli operandi. Ogni oggetto su cui si deve intervenire deve essere inserito nello stack. Per oggetto si intende qualsiasi entità del linguaggio PostScript. Un oggetto semplice è contenuto in una determinata dimensione, la stessa per ogni oggetto semplice, che in genere è di 8 byte. Gli oggetti composti come le stringhe e gli array hanno il nome associato ad un puntatore che si riferisce alla locazione del contenuto. Nel caso di duplicazione, mentre un oggetto semplice duplica la coppia (*nome*, *valore*), un oggetto composto duplica solo il nome: un nuovo puntatore che si riferisce allo stesso indirizzo di memoria dell'oggetto originale. Intervenire sugli oggetti dello stack degli operandi è anche il modo più diretto e veloce di farlo. Si vedrà che creare variabili e procedure implica un doppio passaggio tra stack degli operandi e dizionari che rallenta il processo esecutivo.

I programmi PostScript sono scritti in caratteri ASCII e raggruppati in alcuni costrutti sintattici. Quando si lancia un programma, il flusso dei dati viene letto dallo scanner che ha il compito di convertire i costrutti incontrati (esaminati uno alla volta) in oggetti che passa all'interprete. Distingue il nome di una procedura e di una variabile inteso in senso letterale (con lo slash) da un nome eseguibile.

In tabella 1 presentiamo un elenco dei costrutti sintattici.

2.1 Gli operatori utilizzati

Il PostScript ha una collezione molto ampia di operatori che sono raccolti in un dizionario (system dictionary). Tra gli altri esistono operatori matematici, booleani, di controllo del flusso, di manipolazione degli stack, per la grafica, per il testo. I font sono raccolti in uno specifico dizionario.

Aritmetici e matematici

Gli operatori `add`, `sub`, `div` e `mul` prevedono due operandi razionali. In particolare, in `a b div`, `a` è il dividendo e `b` il divisore e in `a b sub`, `a` è il minuendo e `b` il sottraendo. Gli operatori `idiv` e `mod` prevedono due operandi interi; il primo restituisce la parte intera del quoziente il secondo il resto della divisione. `neg`, `abs` e `round` hanno un solo operando razionale; `sqrt` un operando razionale non negativo. Operatori matematici sono `sin`, `cos` e `atan`: l'operando dei primi due è un angolo in gradi sessagesimali, l'operando di `atan` è un razionale, accetta $\frac{n}{0}$, non accetta la frazione $\frac{0}{0}$ e restituisce un angolo in gradi sessagesimali.

Manipolazione dello stack

`pop`, `exch`, `dup` e `roll` sono adibiti a manipolare lo stack.
`pop` elimina l'elemento top dello stack.
Se gli operandi sono gli oggetti `a` e `b`, `exch` li restituisce nell'ordine inverso: `a b exch` restituisce `b a`.
`dup` duplica l'ultimo elemento dello stack degli operandi: il frammento di codice `10 dup mul` restituisce `100`.
L'operatore `roll n m` agisce sugli ultimi `n` elementi dello stack cambiandone `m` volte le posizioni con una sorta di rotazione. Il frammento di codice `a b c roll 3 1` restituisce `c a b`: agisce sugli ultimi 3 elementi dello stack (quelli rappresentati) facendoli scorrere di una posizione verso destra e recuperando a sinistra l'elemento che è dovuto uscire dal gruppo di tre. `a b c roll 3 2` effettua due spostamenti verso destra, restituendo `b c a`. `a b c roll 3 -1` esegue uno spostamento verso sinistra, restituendo `b c a`.

Operatori di controllo

Tra questi operatori, essenzialmente cicli e istruzioni condizionate, troviamo `if`, `ifelse`, `repeat` e `for`.
`bool Procedura if`: se `bool` è vero esegue `Procedura`.
`bool Procedura1 Procedura2 ifelse`: in caso `bool` sia vero esegue la `Procedura1`, altrimenti `Procedura2`.
`n repeat Procedura`: ripete `n` volte `Procedura`.
L'operatore `for` ha quattro argomenti: `iniz`, `incr`, `lim` e `Procedura`. Ad esempio, in `1 2 8 add for`, i primi tre argomenti dicono che a partire da 1 con incremento 2 si creano nuovi elementi senza che superino il valore limite 8. Gli elementi costruiti sono 1, 3, 5, 7. Questi valori sono passati uno alla volta alla procedura che in questo caso li addiziona calcolando la loro somma. Il ciclo restituisce il numero 16.

Operatori su stringhe

Questi sono: `string`, `stringwidth` e `show`.
`n string` crea una stringa vuota di `n` caratteri.
`(Stringa) show` stampa la `Stringa`.
`(Stringa) stringwidth` calcola preventivamente gli spostamenti che subisce il punto corrente per la stampa di `Stringa` nel font corrente. L'ascissa dello spostamento esprime l'ampiezza della stringa. Non c'è spostamento verticale.

Tabella 1: Elenco dei costrutti sintattici di PostScript

Costrutto	Significato
Numero	Interi e reali, rappresentati in base 10 o in altre basi riconosciute dallo scanner. Il numero decimale 100 in base 2 è 2#1100100, 8#144 in base 8, 16#64 in base 16.
Stringa	Sequenza di caratteri ASCII, anche su più righe, racchiusa da parentesi tonde. Ad esempio (sono una stringa) è una stringa.
Commento	Qualsiasi cosa compresa tra il carattere '%' e il fine linea.
Nome	È una stringa senza caratteri speciali (le parentesi). A231M08 è un nome. Una stringa preceduta dallo slash è un nome inteso in senso letterale. Un nome indica una variabile o una procedura ed è il tramite con cui il PostScript ricerca una procedura e/o una variabile nei dizionari. Un nome preceduto dallo slash come /A231M08 dice che il valore cui è associato non è stato ancora interpretato.
Procedura	Una sequenza di elementi (operatori, operandi, variabili) chiusi da graffe.
Array	È una collezione eterogenea di elementi racchiusi da parentesi quadre.

Conversione di operatori

L'operatore `cvs` converte un qualsiasi oggetto in una stringa con un numero di caratteri assegnato.

Operatori del dizionario

Tale operatore è `def`.
`/Nome-chiave Valore def` crea la coppia <`Nome-chiave`, `Valore`> e la immette nel dizionario corrente. Se il codice è `/a 5 def`, nel dizionario il nome `a` è associato al numero 5.

Operatori di stato per grafici

Sono `gsave`, `grestore` e `setgray`.
`gsave` salva una copia dello stato corrente dei grafici.
`grestore` recupera lo stato corrente dei grafici.
`r setgray` stabilisce l'intensità del grigio; $0 \leq$ `r` ≤ 1. `r` $= 0$ equivale a nero.

Sistema di coordinate

Sono `rotate` e `translate`.
`a rotate` ruota il sistema di riferimento in senso antiorario di `a` gradi sessagesimali.
`a b translate` sposta l'origine del sistema di coordinate di `a` unità nella direzione dell'asse x e di `b` unità in quella dell'asse y.

Costruzione e tracciamento di percorsi

Gli operatori sono `newpath`, `currentpoint`, `setlinewidth`, `moveto`, `rmoveto`, `lineto`, `rlineto`, `arc`, `fill`, `stroke` e `newpath`.
`newpath` vuota il percorso corrente e si prepara ad accoglierne uno nuovo.
`currentpoint` restituisce le coordinate del punto corrente.
`r setlinewidth` assegna lo spessore `r` alla linea da tracciare.
`x y moveto` assegna le coordinate `x` e `y` al punto corrente.
`dx dy rmoveto` sposta le coordinate del vigente punto corrente di (`dx`, `dy`).
`x y lineto` collega con un segmento il punto corrente con il punto (`x`, `y`).
`dx dy lineto` collega con un segmento il punto corrente al punto spostato rispetto a questo di (`dx`, `dy`).
`x y r ai af arc` traccia un arco riferito ad una circonferenza di centro (`x`, `y`) e raggio `r`. Le coordinate angolari iniziale e finale sono `ai` e `af`.
`fill` riempie un percorso chiuso con il colore attuale e cancella il punto corrente. Poiché `fill` cancella il punto corrente per riutilizzare il percorso, occorre salvare il punto stesso con `gsave` prima di utilizzare `fill`, per poi recuperarlo con `grestore`.
`stroke` traccia il percorso; `showpage` stampa la pagina corrente.

Operazioni su caratteri e font

Detti operatori sono `findfont`, `scalefont`, `setfont`, `currentfont` e `stringwidth`.
`/Times-Roman findfont` restituisce il dizionario che descrive il font `/Times-Roman`.
`12 scalefont` porta il font `/Times-Roman` descritto dal dizionario nella dimensione 1 pt alla dimensione richiesta di 12 punti.
`setfont` trasforma il font richiesto nel font corrente.
`currentfont` attiva il dizionario del font corrente.

Operazioni sulle procedure

L'operatore `bind` cerca ogni nome eseguibile della procedura nel dizionario; se lo trova, e il suo valore è un operatore, sostituisce il nome con l'operatore vero e proprio: l'interprete non dovrà cercare i nomi nei dizionari ed eseguirà direttamente le operazioni. La ricerca nel dizionario è stata fatta in via preliminare da `bind`.

2.2 Esecuzione del codice

Un codice costituito da operazioni PostScript che rispetti le proprietà degli operatori e il giusto tipo di operando immesso nello stack degli operandi viene direttamente eseguito (Adobe System Inc., 1984, 1988). Dopo l'esecuzione il codice è tolto dallo stack e sostituito dal risultato, se previsto. È l'approccio più diretto, indicato come il migliore da Adobe System Inc. (1988). Se il codice deve essere utilizzato più volte o per evitare una sua eccessiva complessità, si ricorre a definizioni di procedure e variabili. L'operatore da utilizzare è `def`. Le espressioni `/S 5 def` e `/P{5 3 sub 4 mul}def` definiscono la variabile `/S` e la procedura `/P`. Nel primo caso il PostScript, con l'operatore `def`, salva nelle due posizioni più alte del dizionario corrente la coppia (*nome della variabile*, *valore della variabile*). Immette per primo il nome (*key*) `S` e per secondo, nella posizione più alta, il valore `5`. Nome e valore sono due oggetti semplici. Nel caso della procedura, nel dizionario corrente sono salvati la coppia data dal nome `P` e dal valore dato dall'array `{5 3 sub 4 mul}`. Questa volta gli oggetti formano una coppia composta in cui il nome rappresenta un puntatore all'array. Se il programma lancia, ad esempio, il nome `P`, questo verrà cercato nei dizionari. Trovato, la coppia (*nome*, *valore*) viene inserita nello stack degli operandi. La ricerca del nome nel dizionario e l'inserimento nello stack dei dati e degli operatori richiede un tempo che sconsiglia l'abuso delle procedure.

I dizionari, quello di sistema con tutti gli operatori predefiniti e quello dell'utente in cui sono memorizzate le variabili e le procedure di un programma, sono memorizzati in un apposito stack, con il dizionario dell'utente nella parte più alta. Il dizionario più alto è il dizionario corrente. Se

l'utente definisce la variabile `/S 5 def`, gli oggetti `S` e `5` formano la coppia più alta del dizionario corrente. La sua collocazione nel dizionario non è diretta, come vedremo ritornando al momento in cui lo scanner incontra la definizione della variabile: i tre elementi della definizione sono inviati uno dopo l'altro, a partire dal primo a sinistra, nello stack degli operandi. Il primo è il nome `/S` inteso in senso letterale (non interpretato), poi il valore e infine l'operatore `def`. Dallo stack degli operandi `def` interpreta `/S` come nome immettendolo, privato dello slash, nel dizionario corrente in coppia col suo valore e liberando lo stack degli operandi. In maniera analoga agisce il `def` della procedura.

Può essere interessante seguire il comportamento di una procedura un po' speciale (Glenn C. Reid, 1990) che, oltre a prevedere operazioni da eseguire, ha il compito di definire alcune variabili:

```
/Stampa{
  /Stringa exch def
  /Y exch def
  /X exch def
   X Y moveto Stringa show
       }def
300 -20 (Sono qui) Stampa
```

Nello stesso programma è presente una invocazione alla procedura che le passa tre parametri.

Nella fase di lettura lo scanner inizia con lo scorrere il codice della procedura immettendolo sotto forma di costrutti sintattici non interpretati nello stack degli operandi. Qui risulta operativo soltanto il `def` della procedura. L'operatore `def` immette la procedura nel dizionario corrente (svuotando lo stack degli operandi): il nome, `Stampa`, non è più inteso in senso letterale, ed è associato con l'array delle operazioni ancora non interpretate. Lo scanner prosegue e termina il proprio compito con la lettura dell'invocazione alla procedura, che porta a due conseguenze:

1. i tre parametri che passa a se stessa e il nome della procedura sono immessi nello stack degli operandi, a partire dal primo a sinistra. Seguendo la regola che l'ultimo elemento immesso è il primo ad operare, l'interprete inizia dal nome `Stampa`.

2. trovato nel dizionario il nome `Stampa`, prosegue l'immissione nello stack degli operandi degli elementi dell'array estratti dal dizionario (il valore associato al nome `Stampa`). Il primo frammento di procedura immesso nello stack è la definizione più in alto (o più a sinistra), di seguito le altre due definizioni e infine il codice che caratterizza il compito della procedura.

Lo stack, pensato con una disposizione orizzontale, ha nel primo elemento a sinistra il primo elemento immesso e si presenta così:

```
300 -20 (Sono qui)/Stringa exch def
/Y exch def/X exch def X Y moveto
Stringa show
```

Leggendo lo stack, come vuole la notazione postfissa, da destra a sinistra, si incontra il primo operatore esecutivo nello `exch` che precede il nome `/Stringa`. Può infatti intervenire su due operandi invertendo il loro ordine. Considerando soltanto questo frammento di codice, `exch` lo fa diventare `/Stringa (Sono qui) def`. È la modalità in cui `def` diventa esecutivo immettendo nel dizionario corrente la coppia formata dal nome (non più inteso in senso letterale) `Stringa` e dal valore `(Sono qui)`. La prima varibile è stata definita e lo stack diventa:

```
300 -20 /Y exch def /X exch def
X Y moveto
Stringa show
```

Ora è operativo il codice `-20 /Y exch def` che `exch` trasforma in `/Y -20 def` che definisce, immettendola nel dizionario, la seconda variabile con il nome `Y` e il valore `-20`. Allo stesso modo sarà definita l'ultima variabile dalla coppia <`X`, `300`>. Nello stack è rimasto il codice `X Y moveto Stringa show`. Ora le tre variabili sono definite, lo stack è esecutivo: il codice stamperà nella posizione richiesta il testo previsto (vedi Sez. 6).

L'esempio è stato considerato per aver il modo di seguire come si sviluppa l'azione di un codice PostScript. Pensando solo alla esecuzione più diretta il codice sarebbe stato:

```
300 -20 moveto (Sono qui) show
```

3 La pagina LaTeX e l'ospite PostScript

L'utilizzo del PostScript in un documento LaTeX (Goossens *et al.*, 2007) implica, tra le altre cose, il coordinamento degli output, che hanno necessità di apparire in continuità.

Se si tratta di un grafico, come nei casi della punteggiata e della curva archimedea, il modo più diretto è quello di utilizzare la macro `\code` all'interno di un comando `\pscustom` del pacchetto PSTricks.

Il comando `\pscustom` è predisposto ad interfacciare PSTricks, e quindi LaTeX, col PostScript. Ad esempio `\pscustom` trasforma l'unità di misura del PostScript, che è $\frac{1}{72}$ inch, direttamente in 1 pt di TeX, che è $\frac{1}{72.27}$ inch. Per TeX 1 pt del PostScript è un bigpoint (1 bp). Entro `\pscustom` la macro `\code` apre una finestra sul PostScript, e il grafico prodotto sarà del tutto inserito nel documento LaTeX. Ad esempio, nella sez. 4.1.3 il PostScript costruisce una circonferenza con il raggio di 42.5197 pt (bp), equivalenti a 1.5 cm. Ma l'output, che avviene in ambiente LaTeX, propone un raggio che pur rimanendo di 42.5197 pt

misura 1.4944 cm. Solo se è necessario che l'immagine rispetti le dimensioni in centimetri bisognerà moltiplicare i 42.5197 pt del PostScript per il fattore di conversione 1.00375, immettendo nel PS i 42.6791 pt (1.5056 cm) che in TEX saranno ridotti a 1.5 cm.

Con la macro `\pstverb` di PSTricks si inserisce una pagina PostScript nella pagina LATEX. L'origine della pagina PostScript è il punto corrente della pagina LATEX[1].

4 Curve mediante cicli

Partendo dal tracciamento di punti distribuiti su una circonferenza e da quello dei caratteri 'A' disposti lungo una spirale archimedea, si vuole mostrare come i cicli che disegnano la circonferenza e la spirale possono essere ottenuti in maniera diretta con la macro `\multido` del PSTricks o, in modo più consapevole e flessibile, con il ciclo `loop ... repeat` di TEX e con quello `n ... repeat` di PostScript.

Per disegnare la punteggiata, il ciclo `\multido` e quello standard di TEX ripetono il comando `\psdot`, quello PostScript effettua ripetute rotazioni del sistema di riferimento. `\psdot` usa coordinate polari, come conferma il ';' che separa i suoi parametri.

Un po' diverso è `\nput`, che costruisce le prime due versioni della spirale archimedea, e che può essere pensato come un'asta rigida di lunghezza variabile: una estremità è legata al nodo di riferimento, l'estremità libera incontra idealmente il carattere `A` nella sua metà più alta; la base di `A` è parallela alla retta dell'asta. `\cnode` crea il nodo attorno a cui `\nput` ruota e distanzia l'etichetta A.

4.1 Circonferenza punteggiata

Ad ogni variazione della coordinata angolo, crescente con incremento costante, il comando `\psdot` stampa un punto all'estremo del raggio, rotante e di lunghezza costante. Agendo sulle opzioni di pspicture — con `dotstyle=square,triangle,...` — il punto può essere sostituito da altri oggetti. Nella seconda versione della punteggiata, il ciclo `\loop` e i comandi `\ifnum` e `\advance` appartengono al codice TEX.

4.1.1 La punteggiata con `\multido`

Nella prima versione della punteggiata (esempio 5.14.1 Goossens *et al.* (2007)) la variazione dell'angolo e l'esecuzione del comando `\psdot` sono gestite dalla macro `\multido`. La `i` della variabile `\iAmp` prepara la macro a ricevere numeri interi.

Il seguente codice

1. Questo avviene nell'esempio della tabella in cui il punto corrente dato da `\vskip` è l'origine della pagina PostScript (vedi Sez. 6.2). Similmente, nel caso del calcolo del coefficiente angolare dell'asintoto dell'iperbole (vedi Sez. 5.3)

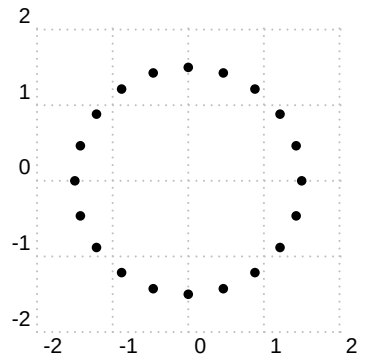


Figura 1: Circonferenza punteggiata ottenuta con `\multido`

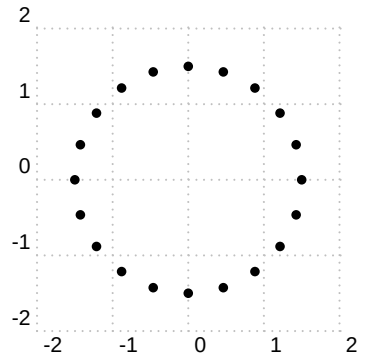


Figura 2: Circonferenza punteggiata ottenuta con un ciclo standard TEX

```
\begin{pspicture}[showgrid=true]
                 (-2,-2)(2,2)
  \multido{\iAmp=18+18}{20}
           {\psdot(1.5;\iAmp)}
\end{pspicture}
```

genera il disegno di figura 1.

4.1.2 La punteggiata e il codice TEX

Il ciclo `\loop ... \repeat` traccia 20 punti cambiando 19 volte la direzione del raggio con un passo di 18°. Il primo punto con il raggio nella direzione 18°, l'ultimo nella direzione 360°. Quest'ultimo è tracciato quando il contatore raggiunge il valore 342. Anche `\ifnum` e `\advance` sono codice TEX.

Di seguito il codice che genera il disegno di figura 2.

```
\begin{pspicture}[showgrid=true]
                    (-2,-2)(2,2)
  \newcount\cont \cont=0
   \loop
     \ifnum\cont<343 \advance
           \cont by 18
       \psdot(1;\the\cont)
   \repeat
\end{pspicture}
```

4.1.3 La punteggiata e il codice PostScript

Meno convenzionale è il caso del codice PostScript. Qui il punto, disegnato come un cerchio pieno di

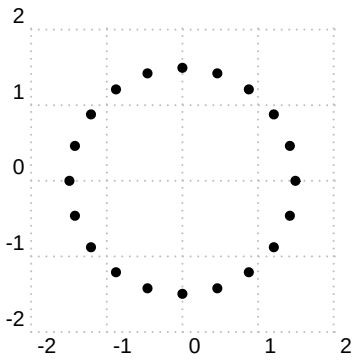


FIGURA 3: Circonferenza punteggiata ottenuta a partire dal codice PostScript

raggio 1.4 pt, è rappresentato 20 volte nella stessa posizione 42.5197, 0). I 42.5197 pt PostScript equivalgono a 1.5 cm, ma nell'ambiente LATEX che ospita la punteggiata misurano, come già detto, 1.4944 cm. Il comando per disegnare la circonferenza è `arc` e i suoi cinque argomenti sono $\mathrm{x}_c, \mathrm{y}_c, \mathrm{r}, \mathrm{ang}_i, \mathrm{ang}_f$. Il comando `rotate` di PostScript agisce sugli assi cartesiani. Sono loro a ruotare di 18° e facilitano le cose, almeno numericamente. Certo, se vivessimo sull'asse x vedremmo disegnare un solo punto. Ad evitare tracciamenti di segmenti il punto corrente coincide col centro del punto.

Il codice seguente

```
\begin{pspicture}[showgrid=true]
                    (-2,-2)(2,2)
 \pscustom{%
  \code{
        42.5197 0 moveto
        20{42.5197 0 1.5 0 360 arc
        gsave fill grestore
        18 rotate
        stroke} repeat
             }}
\end{pspicture}
```

fa ottenere il disegno di figura 3

4.2 Spirale archimedea

Nel caso della spirale archimedea disegnata con il carattere 'A' il punto di riferimento (centro della spirale) è un nodo tracciato come un cerchio pieno da `\cnode` che stabilisce: coordinate del centro, lunghezza del raggio e nome del nodo. Sia `\multido` che `loop ... repeat` spostano e ruotano insieme al raggio il carattere 'A', stampato disponendolo secondo una spirale archimedea con il comando `\nput` (coordinate polari). Detto comando prende come riferimento il nodo e assegna sia la direzione del raggio, variabile di una quantità costante ad ogni passo del ciclo, che la distanza della circonferenza perimetrale (r = 4 pt) dall'etichetta, vista come `labelsep` e crescente con uno step costante. Stabilito ciò, può stampare l'etichetta alla giusta

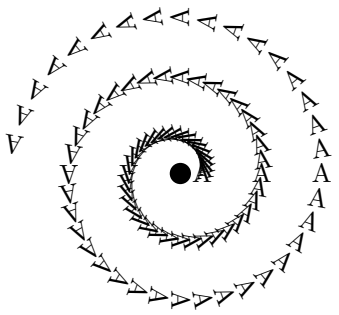


FIGURA 4: Spirale archimedea ottenuta col comando `\multido`

distanza e orientazione. La prima distanza tra il centro del nodo e il carattere 'A' è $(4 + 0.625)$ pt.

4.2.1 Spirale archimedea con `\multido`

`\multido`, che gestisce due variabili indipendenti, traccia facilmente la spirale (esempio 6.2.54 GOOSSENS *et al.* (2007)). In questo esempio la macro `\multido` esegue 90 volte il comando `\nput` e, con una rotazione `rot` che va da 0° a 900°, percorre due rotazioni complete più metà. Le due assegnazioni sono la rotazione `rot=\nA`, e la distanza `labelsep=\rB` pt che separa la circonferenza del nodo dalla etichetta. La prima variabile di `\multido`, `\nA`, accetta solo numeri interi, mentre la seconda variabile, `\rB`, accetta numeri reali; le due variabili sono indipendenti e, avendo entrambe un valore che varia in maniera costante, anche il loro rapporto lo è. Ed è sul valore 16 di tale rapporto che ci si baserà per tracciare con il codice TEX una seconda spirale uguale alla prima. È da notare che `\nput` utilizza due volte il valore assegnato da `\multido` alla variabile angolare `\nA`: la prima volta come l'angolo che assegna la rotazione dell'etichetta, la seconda volta come direzione del raggio. La prima `A` di `\nput` si riferisce al nodo, la seconda all'etichetta.

La figura 4 è ottenuta con il seguente codice:

```
\begin{pspicture}(5,3.5)
  \cnode*(3,1){4pt}{A}
  \multido{\nA=0+10,
           \rB=0+0.625}{90}{
    \nput[rot=\nA,
      labelsep=\rB pt]{\nA}{A}{A}}
\end{pspicture}
```

4.2.2 Spirale archimedea e il codice TEX

Il compito di `loop ... repeat`, che ha una sola vera variabile (intera), presenta una certa artificiosità ma nello stesso tempo è più flessibile di `\multido`. `\cont` è un contatore adeguato a rappresentare i valori interi degli angoli, ma non può rappresentare la lunghezza del raggio, che ha valori razionali. Per questo motivo il valore intero di `\cont` è passato sotto forma di distanza alla variabile razionale `\dimen2`, che può essere divi-

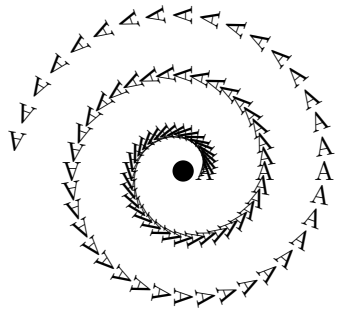


FIGURA 5: Spirale archimedea, identica alla prima, ma con diverso codice

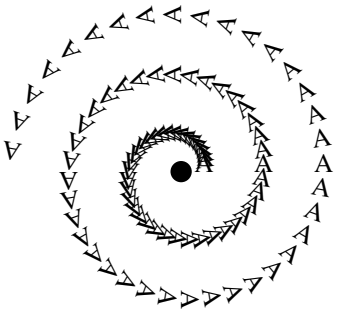


FIGURA 6: Spirale ottenuta con codice PostScript

sa. Ponendo a 10 il passo di `\cont` e dividendo `\dimen2` per 16 si ottiene che le variabili abbiano gli stessi valori del caso precedente e che le due spirali siano sovrapponibili.

Il codice TEX permette anche di mostrare come la rigidità del blocco {*raggio–box del carattere-etichetta*} presente nella spirale è costruita imponendo all'angolo di rotazione dell'etichetta lo stesso valore della rotazione del raggio. Se infatti si definisce un secondo contatore, lo si fa avanzare, ad esempio, con il passo 5 e si assegna il valore di questo contatore alla variabile angolare del raggio, si ottiene che il raggio ruota 1,25 angoli giro mentre l'etichetta effettua 2,5 rotazioni complete.

In figura 5 è mostrata la spirale ottenuta con il programma di seguito riportato:

```
\begin{pspicture}(6,3)
  \newcount\cont \cont=0
  \cnode*(3,0){4pt}{A}
  \loop
   \ifnum\cont<900
   \nput[rot=\the\cont,
      labelsep=\dimen2]
   {\the\cont}{A}{A}
   \advance
   \cont by 10\dimen2=\cont pt
   \divide\dimen2 by 16
  \repeat
\end{pspicture}
```

4.2.3 Spirale archimedea e il codice PostScript

Più semplice il tracciamento col PostScript. Anche in questo caso la rotazione del sistema di riferimento facilita le cose e crea una vera e propria unità del blocco {*raggio–carattere A*}. Qui il raggio è anche base per il carattere A, che risulta più alto rispetto ai due casi precedenti.

Basta pensare di tracciare 90 volte la stringa `(A)`, sull'asse x, a partire dalla posizione 5, di incrementare ad ogni passo la distanza dall'origine di 0.625 bp e di ruotare il sistema di riferimento di 10° in senso antiorario: 22.5 bp in una rotazione di 360°. Il compito è affidato alla procedura `/spiraledia`. Il ciclo `for` passa al corpo della procedura i valori dell'angolo di rotazione, a partire da 0 con incrementi di 10, senza superare il valore 890; il duplicato di 10, diviso per 16 e sommato a 5, dà l'ascissa del punto in cui stampare la stringa `(A)`. Qui non c'è il nodo di riferimento: il cerchio nell'origine è per uniformità grafica con le altre due versioni. La distanza costante tra due spire successive è 22.5 bp (0.8 cm). Il carattere /Times-Roman del PostScript non è uguale al Computer Modern Roman di LATEX.

Il seguente codice disegna la spirale di figura 6:

```
\begin{pspicture}[showgrid=false]
    (-3,-1)(2,2.8)
  \pscustom{\code{
   /Times-Roman findfont 10
     scalefont  setfont
   /spiraledia{0 10 899{dup gsave
     rotate 0 0 moveto 16 div
     5 add 0 rmoveto (A) show
     grestore}for}def
     spiraledia 0 0 moveto
     0 0 4 0 360 arc fill
     showpage }}
\end{pspicture}
```

5 Ellissi e iperboli

Rimanendo nell'ambito delle curve notissime, le ellissi e le iperboli permettono da un lato di toccare due fra le modalità con cui PSTricks traccia le curve e dall'altro di esaminare con una certa attenzione e gradualità il linguaggio PostScript per elaborare le coordinate dei punti della curva e predisporne l'output.

PSTricks traccia una curva in coordinate parametriche, $\big(x(t), y(t)\big)$, con la macro `\parametricplot`, e in coordinate polari, $\big(\alpha, r(\alpha)\big)$, con `\psplot`. In quest'ultimo caso la scelta si esplicita con l'assegnazione `polarplot=true` come argomento di `\psset` o di `\psplot`. La lingua per il calcolo è il PostScript.

5.1 Ellisse in coordinate parametriche

Le coordinate parametriche dell'ellisse ($a > b$) possono essere ottenute pensando di tagliare le circonferenze inscritta (raggio b) e circoscritta (raggio a) all'ellisse con una retta uscente dall'origine e

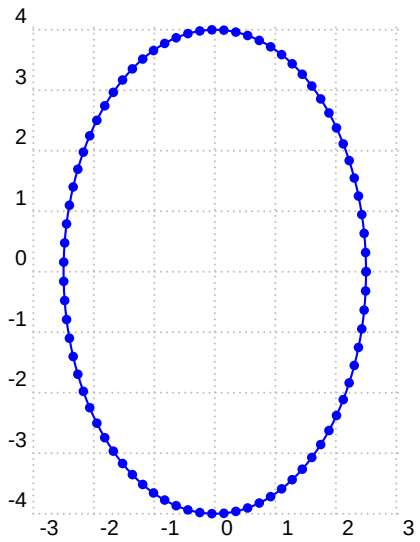


Figura 7: Ellisse in coordinate parametriche

formante un angolo t con l'asse x. L'ascissa dell'intersezione con la circonferenza esterna e l'ordinata di quella con la circonferenza interna si proiettano su uno stesso punto dell'ellisse il cui raggio forma un angolo t con l'asse x; le coordinate del punto, $(a\cos t, b\sin t)$, verificano l'equazione cartesiana dell'ellisse.

Dati $a = 2.5$ e $b = 4$, nell'ellisse in coordinate parametriche l'argomento di `\parametricplot` è `t cos 2.5 mul t sin 4 mul`, in cui è immediato riconoscere $(2.5\cos t, 4\sin t)$.

Con il seguente codice otteniamo l'ellisse in coordinate parametriche visibile in figura 7.

```
\def\Ellissey{t cos 2.5 mul
              t sin 4 mul}
\begin{pspicture}[showgrid=true]
                  (-3,-4)(3,4)
 \psset{linestyle=solid,unit=.8cm}
 \parametricplot[plotstyle=curve,
   plotpoints=80,showpoints=true,
   linecolor=blue]{0}{360}
                 {\Ellissey}
\end{pspicture}
```

5.2 Ellisse in coordinate polari

Per ricavare le coordinate polari $\big(\alpha, r(\alpha)\big)$ dell'ellisse si considera che la conica abbia il centro di simmetria nell'origine del sistema di riferimento. Le coordinate del punto del raggio vettore (modulo r e angolo α con l'asse x) sull'ellisse sono $(r\cos\alpha, r\sin\alpha)$. Sostituendo nell'equazione cartesiana dell'ellisse si ricava $r = \frac{ab}{\sqrt{b^2\cos^2\alpha + a^2\sin^2\alpha}}$. Il radicando è positivo $\forall\alpha$; la curva, se $a > b$, esiste ed è continua in $|\ x\ | \leq a$ (se $a < b \Rightarrow$ sarà continua in $|\ x\ | \leq b$) e non ha complicazioni di tracciamento. In `\psplot` l'equazione del raggio è espressa in codice PostScript; i semiassi `a` e `b` hanno i valori 4 e 2,5: il contrario di quanto fatto nell'ellisse in coordinate parametriche. L'esempio

Codice 1: Codice dell'ellisse in coordinate polari

```
1  \resetOptions
2  \psset{linestyle=solid,
3      unit=0.7cm,polarplot=true}
4  \begin{pspicture}(-4,-1)(4,3)
5    \psaxes[ticks=all]{->}(0,0)
6                    (-4.5,-3)(4.5,3)
7    \psplot[plotstyle=curve,
8      showpoints=false]{0}{360}
9     {/a 4 def /b 2.5 def a b mul
10     x cos dup mul b dup mul mul
11     x sin dup mul a dup mul mul
12     add sqrt  div}
13 \end{pspicture}
```

del tracciamento dell'ellisse in coordinate polari è utile per il confronto con quello più problematico della iperbole.

Nel codice dell'ellisse in coordinate polari l'angolo è espresso dalla variabile x, che sostituisce α, come vediamo nelle righe 9–12 del codice 1.

Interpretando il codice PostScript del raggio dell'ellisse, scritto secondo la Reversed Polish Notation (omologa al modo in cui i dati del PostScript sono salvati in uno stack, disposti in una pila in cui l'ultimo elemento salvato è il primo ad essere estratto), il primo operatore a destra è `div`, che agisce su un dividendo e un divisore, due numeri al momento non disponibili. La posizione che dovrebbe essere occupata dal divisore ospita l'operatore `sqrt`. E alla sua sinistra si incontra l'operatore `add`, che aspetta due addendi. Occorre individuarli scorrendo sempre a sinistra e tenendo presente che in questo contesto sono definiti gli operandi `a`, `b` e l'ampiezza `x`. Si incontra subito un `mul` che agisce su due fattori. Il primo è dato dal frammento di codice `a dup mul`, ossia a^2, il secondo fattore è `x sin dup mul`, ossia $\sin^2 x$. Considerando anche il primo `mul` incontrato, si ottiene il primo addendo che è $a^2\sin^2 x$. Analogamente si individua il secondo addendo che è $b^2\cos^2 x$. La somma dei due addendi, entrambi positivi, è il radicando. Volendo a questo punto fare una istantanea del contenuto dello stack si ottiene `a b mul` $(b^2\cos^2 x + a^2\sin^2 x)$

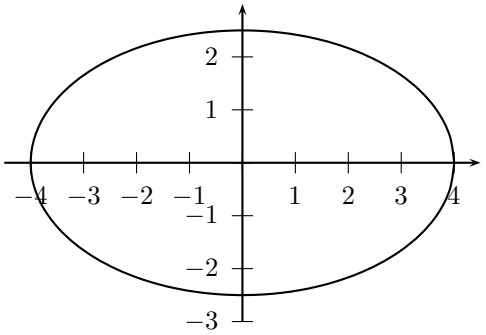


Figura 8: Ellisse in coordinate polari

`sqrt div` dove l'espressione algebrica tra parentesi tonde sta ad indicare il suo valore memorizzato in quella posizione dello stack. L'operatore `sqrt` ne estrae la radice quadrata, che è il divisore che si stava aspettando. Scorrendo il codice ancora a sinistra è facile individuare in `a b mul` il codice del dividendo, espresso nello stack dal valore di ab. Alla fine lo stack occupa una sola locazione con il valore dell'espressione $\frac{ab}{\sqrt{b^2\cos^2 x + a^2\sin^2 x}}$, cioè il risultato del calcolo per il valore assegnato da `\psplot` alla variabile angolo.

In figura 8 vediamo l'ellisse disegnata dal codice mostrato in codice 1.

5.3 Iperbole in coordinate polari

L'equazione del raggio polare dell'iperbole è $r = \frac{ab}{\sqrt{b^2\cos^2\alpha - a^2\sin^2\alpha}}$ e, poiché il radicando è una differenza, la curva non è continua, ma esiste per $\mid x \mid \geq a$ e α interno agli angoli tra gli asintoti contenenti l'asse x, e il suo tracciamento richiede attenzione. Infatti la macro `\psplot` che la traccia deve, in un certo senso, essere guidata dall'utente. Se si prova a far calcolare il raggio in $0 \leq \alpha \leq 360$, come per l'ellisse, o in intervalli non congrui, si creano incompatibilità che bloccano la compilazione. Occorre inoltre tracciare i due rami di iperbole separatamente assegnando due intervalli per la variabile α basati sulle direzioni degli asintoti. Per determinare tali direzioni evitando la calcolatrice ho usato un po' di codice PostScript immesso nel documento LaTeX con la macro `\pstverb` di PSTricks. Questa garantisce che l'output sia in una pagina PostScript collegata al punto corrente della pagina LaTeX, creato nell'esempio da `\vskip 2cm`, che diviene l'origine della pagina PostScript. Il codice PostScript calcola gli angoli tra gli asintoti e l'asse x e li stampa in un box accanto all'iperbole permettendo di scegliere gli intervalli per `\psplot`: più gli estremi degli intervalli sono vicini agli angoli fra gli asintoti e l'asse x più i rami di iperbole saranno lunghi. Poiché gli estremi degli archi nel cui intervallo calcolare il raggio polare e tracciare la curva sono alloctoni, è opportuno che `\psplot` effettui un controllo di positività sul radicando che qui, diversamente dall'ellisse, è dato da una differenza. Gli opposti coefficienti angolari degli asintoti obbligano al loro tracciamento separato portando a quattro il numero dei tracciamenti. Nei vari `\psplot` secondo richiesta vengono assegnati i parametri a e c dell'iperbole, definiti il parametro b e il radicando. `\psplot` riconosce solo variabili e procedure definite nel proprio ambiente: pur se identiche vanno ripetute ogni volta.

Il codice riportato di seguito, e commentato successivamente, si divide in due macroblocchi: da riga 2 a riga 35 è il blocco PostScript che effettua i calcoli e traccia il box dei risultati; da riga 38 a riga 81 invece è il codice PSTricks che disegna l'iperbole e gli asintoti. Detto codice genera la figura 9.

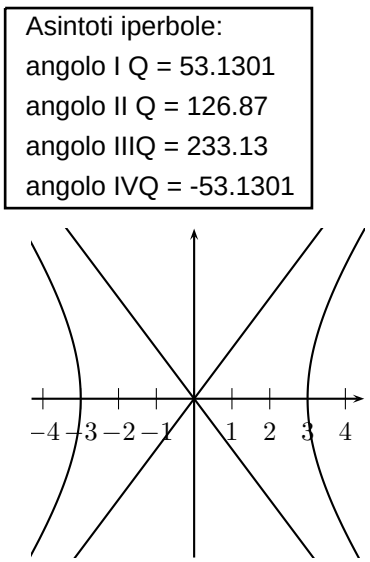


FIGURA 9: Iperbole in coordinate polari e codice PostScript

```
1  \pstverb{
2    /LM 10 def
3    /Helvetica findfont 10
4     scalefont setfont
5    /nstr 8 string def
6    /a 3 def
7    /b 4 def
8    /arctg1{b a atan}def
9    /arctg2{b a neg atan}def
10   /arctg4{b neg a atan 360 sub}def
11   /arctg3{b neg a neg atan}def
12   /prt-n{nstr cvs show}def
13   /prtArctg1{( angolo I
14                 Q = ) show
15      arctg1 prt-n }def
16   /prtArctg2{( angolo II
17                 Q  = ) show
18      arctg2 prt-n}def
19   /prtArctg3{( angolo III
20                 Q  = ) show
21      arctg3 prt-n}def
22   /prtArctg4{( angolo IV
23                 Q = ) show
24      arctg4 prt-n}def
25   LM  15 moveto
26   ( Asintoti iperbole:) show
27   LM    0 moveto prtArctg1
28   LM  -15 moveto prtArctg2
29   LM  -30 moveto prtArctg3
30   LM  -45 moveto prtArctg4
31   newpath 5 25 moveto 5 -50 lineto
32   120 -50 lineto 120 25 lineto
33   5 25 lineto 1 setlinewidth
34   stroke showpage
35 }
36 \vskip 2cm
37 \resetOptions
38 \psset{linestyle=solid,unit=.5cm,
39        polarplot=true}
```

```
\begin{pspicture*}(-4.3,-4.2)
                          (4.5,4.5)
\psaxes[ticks=x]{->}(0,0)
            (-4.3,-4.2)(4.5,4.5)
\psplot[plotstyle=curve]{-50}{50}
{
  /a 3 def /c 5 def
  /b{c dup mul a dup mul
     sub sqrt} def
  /rdcn{b dup mul x cos dup mul
     mul a dup mul x sin dup mul
     mul sub}def
  rdcn  0 gt{a b mul rdcn
             sqrt div} if
}
\psplot[plotstyle=curve]{127}
                           {230}
{
     /a 3 def /c 5 def /b{c dup
       mul a dup mul sub sqrt} def
     /rdcn{b dup mul x cos dup mul
       mul a dup mul x sin dup
 mul
       mul sub}def
   rdcn  0 gt{a b mul rdcn sqrt
                div}{} ifelse
}
\psset{polarplot=false}
\psplot[plotpoints=30]{-4}{4.5}
{
   /a 3 def /c 5 def
   /b{c dup mul a dup mul sub
      sqrt} def
    b a div x mul}
\psplot[plotpoints=30]{-4.1 }
                      {3.5 }
{
   /a 3 def /c 5 def
   /b{c dup mul a dup mul sub
      sqrt} def
    b a div x mul neg
}
\end{pspicture*}
```

Commenti al codice

Il codice del precedente paragrafo è stato ripulito dai commenti, che vengono qui ampliati. Per ogni intervallo di righe di codice si dà una spiegazione sommaria di quanto avviene.
Righe 2–35: Codice PostScript per calcolare gli angoli tra asintoti e asse x e tracciare il box con i risultati.
Righe 2–4: Margine a sinistra 10 pt e definizione del font.
Righe 5–7: Stringa vuota di 8 caratteri.
Righe 8–11: Calcola $\arctan b/a$ per i quadranti 1º, 2º, 4º e 3º.
Riga 12: Converte il numero in stringa e lo stampa.
Righe 13–24: Prepara le stringhe col valore dell'arctan.
Righe 25–30: Crea vari punti correnti per stampare i risultati.
Righe 31–35: Traccia il box dei risultati.
Righe 38–81: Codice PSTricks per tracciare l'iperbole in coordinate polari.
Righe 38–43: Ambiente pspicture e tracciamento assi.
Righe 44–54: Definizioni, positività radicando, tracciamento iperbole 1º e 4º quadrante.
Righe 55–65: Definizioni, positività radicando, tracciamento iperbole 2º e 3º quadrante.
Righe 66–81: Definizioni e tracciamento asintoti.

6 Tabelle calcolanti[2]

LATEX, i pacchetti in circolazione, e l'ambiente newenvironment sono in grado di costruire tabelle per ogni esigenza. Ma non prevedono alcuna reale autonomia circa la elaborazione di quantità numeriche presenti nelle tabelle. Il principale motivo per questa carenza è probabilmente dovuto al codice TEX i cui operatori aritmetici agiscono solo su contatori numerici interi. Effettua calcoli su contatori dimensionali con valori razionali restituendo una dimensione, anche nella divisione tra contatore dimensionale e numerico. Al contrario il PostScript prevede operatori che effettuano calcoli coi numeri reali ma non ha una gestione diretta delle tabelle; occorre definire procedure specifiche. Per mostrare i rispettivi comportamenti si considera il caso di una semplice tabella che rappresenti i punteggi ottenuti da tre gruppi in una ipotetica competizione e che sappia calcolare e rappresentare risultati e percentuali.

Nel primo caso il calcolo delle percentuali, espresse da numeri interi, richiede un elevato numero di contatori e operazioni. Con il PostScript il calcolo delle percentuali con due decimali significativi si ottiene con poche procedure mentre è impegnativa la costruzione delle tabelle che mostrano i dati.

6.1 Tabelle LATEX che calcolano con TEX

Il primo esempio è una tabella che espone i punteggi associati a tre gruppi; questa calcola le rispettive percentuali con i comandi TEX ed in particolare con `\divide` che opera la divisione restituendo il quoziente troncato.

Le percentuali che si ottengono troncando agli interi il quoziente delle divisioni tra i singoli punteggi moltiplicati per 100 e il punteggio complessivo (tolti i casi del dividendo multiplo intero del divisore) sono sempre inferiori al valore effettivo, fino ad una unità (0,99...). Ma il risultato della divisione può essere migliorato con l'utilizzo dei resti. Se il resto supera il duecentesimo del pun-

2. Il mio è un approccio empirico elementare, lontano e indipendente dal lavoro sapientemente fondato di Roberto Giacomelli, *Una tabella che fa i calcoli*, ArsTEXnica, ottobre 2008.

teggio complessivo (pari allo 0,5%) la percentuale assegnata aumenta di un punto, altrimenti rimane invariata. Le percentuali corrette, sempre espresse solo dagli interi, hanno ora una più equilibrata incertezza di ± 0,5%, anche se il centesimo della somma che viene utilizzato per correggere l'effetto del primo troncamento è anche esso troncato: nel caso dei Bianchi porta alla contraddizione di un resto superiore al centesimo della somma dei punti.

Guardando alla quantità di codice sembra un calcolo complicato, ma lo sembra per la numerosità dei contatori necessari. Occorre solo un po' di pazienza. Il tutto è ben compendiato dalla tabella.

```
\newcount\Rossi \newcount\Rcento
\newcount\Bianchi \newcount\Vcento
\newcount\Verdi \newcount\Rcento
\newcount\Rg \newcount\Vg
\newcount\Bg \newcount\sumcento
\newcount\sumduecento
\newcount\Rr \newcount\Vr
\newcount\Rv \newcount\Br
\newcount\Rb \newcount\Rb
\newcount\sum \newcount\sumcentoB
\newcount\sumcentoR
\newcount\sumcentoV
\Rossi=18769
\Verdi=9876
\Bianchi=14728
\Rcento = \Rossi
\Vcento = \Verdi
\Bcento = \Bianchi
\Rr     = \Rossi
\Vr     = \Verdi
\Br     = \Bianchi
\multiply\Rcento by 100
%     (-> \Rcento = 1876900)
\multiply\Vcento by 100
%     (-> \Vcento =  987600)
\multiply\Bcento by 100
%     (-> \Bcento = 1472800)
\sum=0
  \advance\sum by \Rossi
  \advance\sum by \Verdi
  \advance\sum by \Bianchi
%     (-> \sum = 43373)
\sumduecento=\sum
  \divide\sumduecento by 200
%     (-> \sumduecento = 216)
\sumcento=\sum
  \divide\sumcento by 100
%     (-> \sumcento = 433)
  \sumcentoR=\sumcento
  \sumcentoV=\sumcento
  \sumcentoB=\sumcento
  \divide\Rcento by \sum
%     (-> \Rcento = 43)
  \divide\Vcento by \sum
%     (-> \Vcento = 22)
  \divide\Bcento by \sum
```

Tabella 2: Tabella con le percentuali calcolate con TEX

Gruppo	Pnt	% gr	$\frac{\Sigma}{200}$	Rst	% crt
Rossi	18769	43%	216	150	43%
Verdi	9876	22%	216	350	23%
Bianchi	14728	33%	216	439	34%

```
%      (-> \Bcento = 33)
  \Rg=\Rcento \Vg=\Vcento
  \Bg=\Bcento
 \multiply\sumcentoR by -\Rcento
%     (-> \sumcentoR = -18619)
 \advance\Rr by \sumcentoR
%     (-> \Rr = 150)
 \ifnum \Rr>\sumduecento %(->
false )
   \advance\Rcento by 1\fi
%     (-> \Rcento = 43)
 \multiply\sumcentoV by -\Vcento
%     (-> \sumcentoV =  -9526)
 \advance\Vr by \sumcentoV
%     (-> \Vr = 350)
 \ifnum \Vr>\sumduecento  %(->
true )
   \advance\Vcento by 1\fi
%     (-> \Vcento = 23)
 \multiply\sumcentoB by -\Bcento
%     (-> \sumcentoB =  -14289)
 \advance\Br by \sumcentoB
%     (->\Br = 439)
 \ifnum \Br>\sumduecento  %(->
true )
   \advance\Bcento by 1\fi
%   (-> \Bcento = 34)
```

Commenti al codice

Righe 1–6: Contatori per il calcolo della percentuale grezza.
Righe 7–12: Contatori per il calcolo della percentuale corretta.
Righe 13–15: Dati.
Righe 16–38: Calcolo percentuale grezza (le righe commentate — % — indicano un risultato parziale).
Righe 39–49: Percentuali grezze espresse da `\Rg \Vg \Bg`.
Righe 50–70: Calcolo percentuali corrette espresse da `\Rcento \Vcento \Bcento`; i resti sono espressi da `\Rr \Vr \Br`. Nella tabella 2 è visibile il risultato del codice appena analizzato, ottenuta con il sorgente nel codice 2.

6.2 Il PostScript che calcola e costruisce tabelle

In questa sezione si è scelto di immettere tutto il codice nell'ambiente creato dal comando `\pstverb`. Questo crea una pagina PostScript in cui immettere l'output elaborato collegandolo alla pagina

Codice 2: Codice della tabella con valori calcolati con TEX

```
\begin{tabular}{lrrcrr}
 prcnt = \%
\toprule
 Gruppo&Punti& prc gr&$\frac{S}
           {200}$&Rst &prc crrt\\
\midrule
 Rossi&\the\Rossi&\the\Rg prcnt&
     \the\sumduecento &
     \the\Rr&\the\Rcento prcnt\\
 Verdi&\the\Verdi&\the\Vg prcnt&
     \the\sumduecento &
     \the\Vr&\the\Vcento prcnt\\
 Bianchi&\the\Bianchi&\the\
      Bg prcnt&\the\sumduecento &
     \the\Br&\the\Bcento prcnt\\
\bottomrule
\end{tabular}
```

LATEX. L'origine della pagina PostScript è il punto corrente della pagina LATEX. Nel caso specifico quello determinato da `\vglue 170pt`.

Dato che si tratta di un esempio dimostrativo il numero delle righe e colonne è minimo. La loro numerosità non è però un problema.

La tabella con le percentuali a due decimali — i centesimi incerti e arrotondati — mostra i risultati del calcolo che si risolve con poche procedure. Una per sommare i punteggi ed una per ogni gruppo (riga) della tabella per calcolare le percentuali. Nella seconda definizione (righe 16–21 a lato), si nota che il punteggio del singolo gruppo è moltiplicato per 10000 prima di essere diviso per il punteggio complessivo. Se volessimo calcolare una percentuale dovremmo moltiplicare il punteggio parziale per 100 e poi dividere per il punteggio complessivo. Si avrebbe un numero con molti decimali di cui solo due devono rimanere dopo aver approssimato i centesimi. Gli strumenti di calcolo del PostScript hanno l'operatore di arrotondamento, `round`, che agisce approssimando gli interi ed eliminando tutte le cifre decimali. Se nel frattempo anziché per 100 si è moltiplicato il punteggio parziale per 10000, `round` non avrà approssimato le unità ma i centesimi in quanto per riportare le cose al loro posto si dovrà dividere il quoziente ottenuto (un intero) per 100. Il codice per le variabili e le procedure di calcolo è quello riportato nelle righe 12–21 del codice 3.

I risultati ottenuti sono mostrati in due tabelle (entrambe in tabella 3) che allineano in maniera diversa i dati delle colonne: nella prima tutte le colonne sono imbandierate a sinistra, nella seconda la prima colonna è imbandierata a sinistra, le altre a destra. Le intestazioni della prima tabella sono centrate, quelle della seconda sono allineate con i dati. In codice 3 troviamo l'intero codice usato, che commenteremo nei successivi paragrafi.

Codice 3: Codice della tabella con valori calcolati con PostScript

```
\vglue 170pt
\pstverb
{
/grassetto/Helvetica-bold findfont
  10 scalefont def
/Helv/Helvetica findfont 10
  scalefont def
/helv/Helvetica findfont 30
  scalefont def
/nstr 8 string def
/prt-n{nstr cvs show}def
/lstA 127856 def
/lstB 6473 def
/lstC 48265 def
/somma {lstA lstB add lstC add}def
/percA{lstA 10000 mul somma div
       round 100 div}def
/percB{lstB 10000 mul somma div
       round 100 div}def
/percC{lstC 10000 mul somma div
       round 100 div}def
/intestazionegCentro{
   grassetto setfont (Gruppo)
   dup dup stringwidth  pop
   50 exch sub 2 div 0 rmoveto
   show stringwidth pop 50 exch
   sub 2 div 0 rmoveto
   (Punti) dup dup stringwidth
pop
   50 exch sub 2 div 0 rmoveto
   show stringwidth pop 50 exch
   sub 2 div 0 rmoveto
   (Percnt) dup dup stringwidth
   pop 50 exch sub 2 div 0
   rmoveto show stringwidth
   pop 50 exch sub 2 div 0
   rmoveto}def
%/intestazionegCCC
%{grassetto setfont
% /diff1{(Gruppo) stringwidth pop
%        50 exch sub 2 div}def
% /diff2{(Punti)  stringwidth pop
%        50 exch sub 2 div}def
% /diff3{(Percnt) stringwidth pop
%        50 exch sub 2 div}def
% diff1 0 rmoveto  (Gruppo) show
  diff1 0 rmoveto
% diff2 0 rmoveto  (Punti)  show
  diff2 0 rmoveto
% diff3 0 rmoveto  (Percnt) show
  diff3 0 rmoveto
%}def
/intestazionegSSS{
 grassetto setfont
 (Gruppo) dup show stringwidth pop
 50 exch sub 0 rmoveto (Punti)
```

```
 dup show stringwidth pop 50 exch
 sub 0 rmoveto (Percnt) show}def
/rigaAgs{Helv setfont (Listone A)
 dup show stringwidth pop 50 exch
 sub 0 rmoveto lstA nstr cvs dup
 show stringwidth pop 50 exch sub
 0 rmoveto percA nstr
 cvs show}def
/rigaBgs{Helv setfont (Lista B)
 dup show stringwidth pop 50 exch
 sub 0 rmoveto lstB nstr cvs dup
 show stringwidth pop 50 exch sub
 0 rmoveto percB nstr cvs show}def
/rigaCgs{Helv setfont (Lista C)
 dup show stringwidth pop 50 exch
 sub 0 rmoveto lstC nstr cvs dup
 show stringwidth pop 50 exch sub
 0 rmoveto percC nstr cvs show}def
0 146 moveto intestazionegCCC
0 130 moveto rigaAgs
0 115 moveto rigaBgs
0 100 moveto rigaCgs
newpath -6 156 moveto 156 156
lineto  0.8 setlinewidth  stroke
newpath -4 141 moveto 154 141
lineto  0.6 setlinewidth  stroke
newpath -6  95 moveto 156  95
lineto  0.8 setlinewidth  stroke
showpage

/intestazionegSDD{
   grassetto setfont
   (Gruppo) dup show stringwidth
   pop 50 exch sub 0 rmoveto
   (Punti) dup stringwidth pop
   50 exch sub  0 rmoveto show
   (Percnt)dup stringwidth pop
   50 exch sub  0 rmoveto show
   }def
/rigaAgSDD{
   Helv setfont (Listone A)
   dup show stringwidth pop 50
   exch sub 0 rmoveto lstA nstr
   cvs dup stringwidth pop 50
   exch sub  0 rmoveto show percA
   nstr cvs dup stringwidth pop
   50 exch sub  0 rmoveto show
   }def
/rigaBgSDD{
   Helv setfont (Lista B)
   dup show stringwidth pop 50
   exch sub 0 rmoveto lstB nstr
   cvs dup stringwidth pop 50
   exch sub  0 rmoveto show percB
   nstr cvs dup stringwidth pop
   50 exch sub  0 rmoveto show
   }def
/rigaCgSDD{
   Helv setfont (Lista C)
   dup show stringwidth pop 50
   exch sub 0 rmoveto lstC nstr
   cvs dup stringwidth pop 50
   exch sub  0 rmoveto show percC
   nstr cvs dup stringwidth pop 50
   exch sub  0 rmoveto show
   }def
0 60 moveto intestazionegSDD
0 45 moveto rigaAgSDD
0 30 moveto rigaBgSDD
0 15 moveto rigaCgSDD
%newpath -6 70 moveto 156 70
%lineto 156 9 lineto -6 9 lineto
%closepath .8 setlinewidth
%stroke
%newpath -5 55 moveto  155 55
%lineto  0.6 setlinewidth  stroke
%newpath 50 69 moveto 50 8
%lineto stroke
%newpath 105 69 moveto 105 8
%lineto stroke
newpath -6 70 moveto 156 70
lineto  0.8 setlinewidth
stroke
newpath -4 55 moveto 154 55
lineto 0.6 setlinewidth stroke
newpath -6 9 moveto 156 9
lineto  0.8 setlinewidth stroke
showpage
}
```

Prima tabella

Le righe 4–21 contengono le definizioni preliminari, i dati e i relativi calcoli.

Considerato che le definizioni della riga delle intestazioni e dei dati non assegnano il punto corrente iniziale e che utilizzano, con `rmoveto`, gli spostamenti relativi, il codice che le invoca per stamparle può assegnare il punto corrente a una qualsiasi posizione della pagina.

L'operatore `stringwidth` calcola e mette nello stack gli spostamenti relativi a cui sarebbe sottoposto il punto corrente dalla stampa della stringa considerata, tenendo conto anche delle caratteristiche e delle dimensioni del font utilizzato: nella posizione più alta dello stack lo spostamento verticale, che è zero e che viene eliminato dal successivo `pop`, e subito sotto lo spostamento orizzontale. Nel fare queste operazioni viene cancellata la stringa a cui l'operatore `stringwidth` si è riferito.

Nelle righe 22–36 si trova il codice per formattare i titoli delle colonne, centrati. Per centrare una stringa nella dimensione di una colonna occorre calcolare la differenza tra le ampiezze della colonna (50) e della stringa (`stringwidth pop`) e assegnarne una metà (`stringwidth pop 50 exch sub 2 div`) all'ascissa di `rmoveto` per determinare il punto da cui iniziare la stampa della stringa e l'altra metà per spostare, con `rmoveto`, il punto

corrente all'inizio della seconda colonna. In questo fare si utilizza tre volte la stringa: una stampa e due `stringwidth` e questa tripla disponibilità si ottiene una volta dalla stringa stessa, le altre dalla doppia duplicazione della stringa `(Gruppo)`.

Nelle righe 37–48 è presente un codice simile, che usa le procedure. Il loro uso chiarisce il flusso dei dati, ma rallenta l'esecuzione. Da riga 49 a riga 70 possiamo vedere l'imbandieramento a sinistra sia dei titoli (49–60) che dei dati(61–70).

Per seguire la dinamica operativa del PostScript (sempre in relazione alle righe 22 - 36 dell'intestazione centrata) bisogna guardare al codice della prima colonna come ad un codice inserito nello stack degli operandi, dall'alto verso il basso, o più comodamente da destra verso sinistra. Nella prima lettura a partire da destra si nota che per trovare un operatore esecutivo è necessario scorrere fino al `dup` situato alla destra della stringa `(Gruppo)`, di cui fa una copia. Le ultime tre posizioni ora sono `(Gruppo) (Gruppo) dup`: è operativo `dup` che crea una copia della seconda stringa. Le ultime cinque posizioni dello stack diventano: `(Gruppo)(Gruppo)(Gruppo) stringwidth pop`. L'operatore `stringwidth` memorizza l'incremento che avrebbero le coordinate del punto corrente dalla stampa della terza stringa, che toglie dallo stack (l'incremento dell'ascissa è ovviamente l'ampiezza della stringa); `pop` estrae l'ordinata del punto corrente. A questo punto le ultime posizioni dello stack sono `(Gruppo)(Gruppo) stringwidth 50 exch sub 2 div`; `exch` può scambiare i suoi operandi, `sub` sottrarli e `2 div` calcolare la metà della differenza, che è l'ascissa di `rmoveto` che, per comodità, è riportato nel codice tra virgolette. Andando ancora a guardare alla parte bassa dello stack il codice, che è `(Gruppo)(Gruppo) "ascissa" rmoveto 0 show`, crea il nuovo punto corrente da cui iniziare la stampa della stringa che gli è contigua, eliminandola dallo stack. Che nella parte più bassa ora è `(Gruppo) stringwidth pop 50 exch sub 2 div 0 rmoveto` e ha il compito di portare il punto corrente all'inizio della seconda colonna. Lo fa aggiungendo all'ascissa del punto corrente, cioè in coda alla stringa, la seconda metà della differenza di cui si è detto. Nella parte bassa dello stack è ora presente il codice della seconda colonna, uguale in tutto a quello della prima tranne che per la stringa che è diversa. Lo stesso per la terza colonna.

Da riga 55 a riga 70 troviamo il codice che predispone l'imbandieramento a sinistra dei dati della prima tabella e da riga 71 a riga 81 il codice per stampare i titoli centrati, i dati imbandierati a sinistra e le tre filettature orizzontali.

Di seguito si commenta la dinamica operativa delle righe 55–60 riguardanti l'imbandieramento a sinistra della prima riga di dati. Si sceglie il font da utilizzare, si immette la prima stringa `(Listone A)`, si duplica per utilizzarla due volte (`dup show stringwidth`): per stamparla (`dup show`) e misurarne la lunghezza con `stringwidth pop` che estrae la lunghezza della stringa. Si immette la larghezza della prima colonna (50) e poiché si deve spostare il punto corrente verso destra, nell'ascissa in cui inizia la seconda colonna, lo spazio rimasto alla destra della stringa dovrà essere espresso da un numero positivo: per questo si scambiano i ruoli del minuendo e del sottraendo con `exch` e si assegna a questa differenza il ruolo di spostamento relativo (`'dx' 0 rmoveto`). Lo 0 per l'ordinata ci dice che questa non cambia valore. Ora il punto corrente è l'ascissa del punto zero della seconda colonna. La seconda immissione della riga, `lstA`, è un numero e dovrà essere trasformato in stringa che, per i motivi detti, sarà duplicata e stampata iniziando dalla posizione attuale. La trasformazione in stringa avviene con la procedura `nstr` e l'operatore `cvs`. La prima, `nstr`, usa l'operatore `string` per creare una stringa vuota di una determinata lunghezza, nello specifico di otto caratteri. Il secondo operatore, `cvs`, trasforma il numero in una stringa al massimo di otto caratteri. Ora, come già fatto nella prima colonna, si calcola la lunghezza della nuova stringa (`stringwidth`), si elimina l'ordinata, si immette la larghezza della seconda colonna (50) e si calcola lo spazio rimasto alla destra della stringa per poter portare il nuovo punto corrente allo zero della terza colonna (`stringwidth pop 50 exch sub 0 rmoveto`). Lo stesso per la terza immissione (`percA`).

Tabella 3: Le due tabelle generate interamente col PostScript

Gruppo	Punti	Percnt
Listone A	127856	70.02
Lista B	6473	3.55
Lista C	48265	26.43

Gruppo	Punti	Percnt
Listone A	127856	70.02
Lista B	6473	3.55
Lista C	48265	26.43

Seconda tabella

Il codice per questa tabella, da riga 83 a a 141, funziona allo stesso modo, con l'evidente differenza di imbandieramento. La parte commentata traccerebbe una filettatura per il contorno e la linea orizzontale interna.

Per la prima colonna (giustificata a sinistra) vale quanto detto per tabella precedente e porta il punto corrente allo zero della seconda colonna. La seconda immissione (giustificata a destra) è un numero

che dovrà essere trasformato in stringa e duplicato, se ne misurano le dimensioni, si elimina l'ordinata dalla sommità dello stack, si immette la larghezza della seconda colonna, si sottrae dalla larghezza della colonna la larghezza della stringa e si fa di questo valore il punto corrente per iniziare la stampa della stringa, che andrà a terminare nel punto zero della terza colonna facendone il nuovo punto corrente (`lstA nstr cvs dup stringwidth pop 50 exch sub 0 rmoveto show`). La stessa cosa per la terza colonna.

7 Procedura che si autodefinisce e l'operatore bind

Il filo del mio discorso, che potrebbe ritenersi completato con l'*excursus* sulle tabelle, ha invece lasciato in sospeso l'argomento della procedura con due livelli di definizione, che non è stato verificato. Inoltre non volevo chiudere senza neanche accennare a quell'operatore `bind` presente almeno nella metà delle pagine di Adobe System Inc. (1988). Sarebbe stato come non lasciare neanche un esile *legame* con il PostScrpt.

Diciamo che riprendere quell'esempio e completarlo con l'operatore `bind` è quel legame.

Non ritornerò sulla procedura di cui ho già detto molte cose. Accennerò rapidamente soltanto al comportamento del nuovo operatore la cui esecutività consiste nell'andare a cercare nel dizionario i nomi presenti nella procedura. Se il nome cercato è nel dizionario, `bind` interviene sulla procedura legando il nome al corrispondente oggetto operatore. Questo ricorsivamente per ogni nome presente nella procedura, anche all'interno di un array, o in presenza di incapsulamenti. Se il nome non è nel dizionario o non è di un operatore, `bind` non agisce. Quando la procedura sarà eseguita, l'interprete troverà operatori di cui conosce le modalità di intervento e le azioni che sono in grado di compiere (definite nel dizionario e che, in coppia con il nome, costituiscono il valore dell'operatore), e l'esecuzione sarà immediata.

Ad esempio, il seguente codice è esemplificativo di quanto detto. Il risultato è visibile immediatamente dopo.

```
/helv/Helvetica findfont 30
  scalefont def
/F{
   /Stringa exch def
   /Y exch def
   /X exch def
  X Y moveto  Stringa show
   }bind def
 helv setfont
 0 -50 (?<- Sono qui) F
```

?<- Sono qui

8 Ringraziamenti

Per me si è trattato di una prima volta e non avrei potuto organizzare alcunché senza i riferimenti e la disponibilità del *The TEXBook* di Donald Knuth, il *LATEX* di Claudio Beccari, il *TEX* di Gianni Gilardi, *The LATEX Companion* di Frank Mittelbach ed altri, il *Guide to LATEX* di Helmuth Kopka, senza il *Tutorial and Cookbook* di Adobe e senza gli *Appunti di Programmazione* in LaTeX e TeX di Enrico Gregorio.

Riconosco anche che senza la disponibilità — non dichiarata ma espressa dalla qualità e apertura dei rapporti — di Massimiliano Dominici e di Gianluca Pignalberi non mi sarei così impegnato nel cercare di portare avanti il mio lavoro. Innanzi tutto ringrazio il mio anonimo Editor per aver saputo vedere elementi di validità nel mio scritto. Un ulteriore ringraziamento voglio fare a Gianluca Pignalberi e all'anonimo revisore per essere riusciti, con un gran lavoro e occhio di lince, con suggerimenti e anche il recupero di un paio di errori, a dare al mio scritto, che era nato per essere letto dagli amici, la forma (spero sia insieme a un po' di sostanza) di articolo.

Riferimenti bibliografici

Adobe System Inc. (a cura di) (1984). *PostScript Language, Tutorial and Cookbook*. Addison-Wesley, Reading, MA, USA. The BlueBook.

— (1988). *Postscript language program design*. Addison-Wesley, Reading, MA, USA. The GreenBook.

— (1991). *PostScript Language, Reference Manual*. Addison-Wesley, Reading, MA, USA. The RedBook.

Glenn C. Reid (1990). *Thinking in PostScrpt*. Addison-Wesley, Reading, MA, USA.

Goossens, M., Mittelbach, F., Rahtz, S., Roegel, D. e Voss, H. (2007). *The LATEX Graphics Companion Second Edition*. Addison-Wesley, Reading, MA, USA.

Utting, I. (1992). «Postscript tutorial and reference». Technical report. URL `http://www.cs.kent.ac.uk/pubs/1992/109`.

▷ Riccardo Nisi
`r underscore nisi at tin dot it`

Eventi e novità

BachoTeX 2009

Il XVIII convegno BachoTeX, organizzato dallo User Group polacco (GUST), si terrà dal 29 Aprile al 3 Maggio 2009 a Bachotek, nei pressi di Brodnica, in Polonia, e sarà dedicato al tema: "TEX: at a turning point, or at the crossroads?".

Così gli organizzatori presentano l'evento:

"The community is putting a lot of effort and thought into possible strategies for promoting and developing TEX and related products for the foreseeable future, including:

- TEX, and TEX-based engines
- enhanced graphics engines
- new or improved macro packages
- user interfaces
- new fonts

Work progresses in many different directions, and thus there is clearly hope for an ever better future, even though the more pessimistic may wonder whether TEX represents an evolutionary dead-end.

In this context the feedback between users and developers of new tools and engines is immensely important — it might decide whether within a few years we will be looking back at today as a successful turning point or as a bad decision at the crossroads of TEX's history.

Some say that they have never believed that conference themes/mottos have any impact on submissions anyway. We, the Program Committee, think otherwise; please help us to prove our point! Although we do not restrict what your presentations should be about, we would be more than happy if your contribution will be user-centric and oriented towards the future of TEX with special emphasis on the needs, hopes and dangers."

TUG Conference 2009

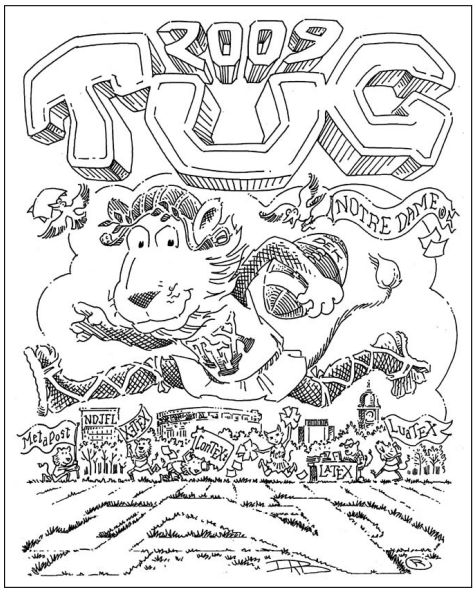
Il convegno annuale 2009 del TEX Users Group si terrà dal 28 al 31 luglio 2009 presso l'Università di Notre Dame, a Notre Dame, Indiana, USA.

L'organizzazione locale è curata principalmente da Martha Kummerer, del *Notre Dame Journal of Formal Logic* (`http://www.nd.edu/~ndjfl/`).

Il programma, ancora da definire nei dettagli, è mirato ad illustrare tecniche pratiche per la produzione di documenti con TEX, LATEX, ConTEXt, METAPOST e strumenti affini.

Il convegno vero e proprio, della durata di tre giorni, sarà preceduto, il 28, da una giornata di *workshop* aperti a tutti.

Ulteriori informazioni saranno via via pubblicate sulla pagina internet dedicata al convegno: `http://www.tug.org/tug2009/`.

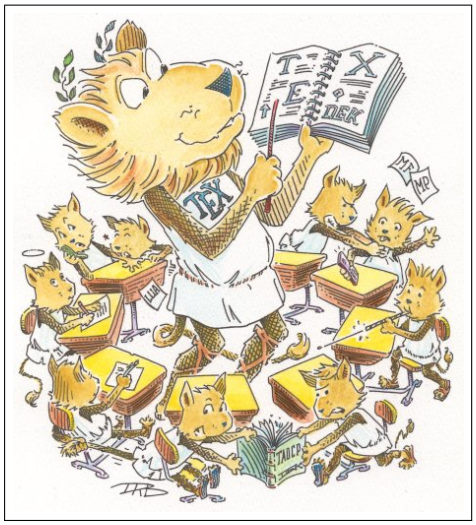


EuroTeX Conference 2009 e 3rd ConTeXt Meeting

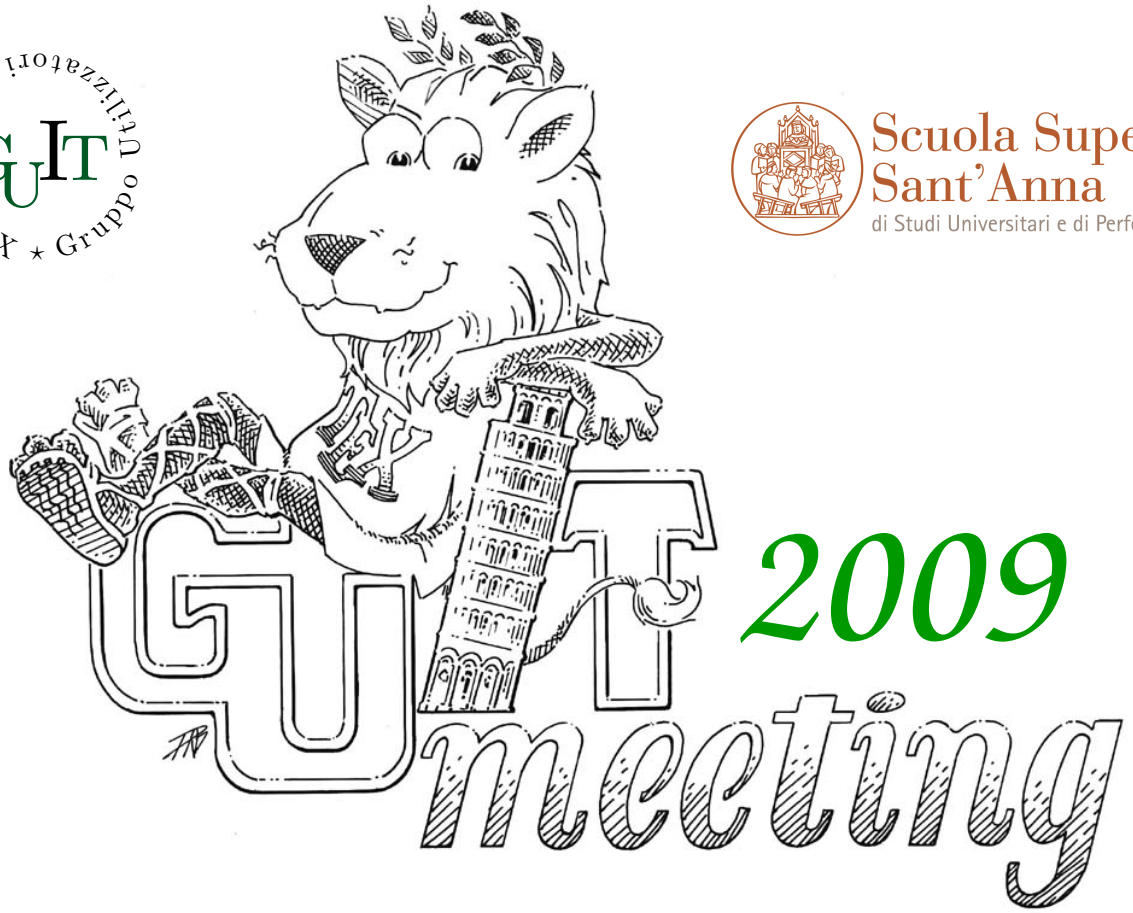
Nel 2009, i convegni EuroTeX e ConTeXt si terranno entrambi congiuntamente all'Aia, in Olanda, dal 31 Agosto al 4 Settembre, organizzati dal locale User Group (NTG) con la collaborazione della *ConTEXt Task Force*.

Il convegno, pur accettando interventi su ogni argomento d'interesse per chi usa TEX, sarà particolarmente incentrato sull'uso di TEX in ambito educativo.

Il programma e altri particolari sull'organizzazione del convegno verranno comunicati in seguito sulla pagina internet `http://www.ntg.nl/EuroTeX2009/index.html`.



Questa rivista è stata stampata
presso Logo S.r.l. Servizi di Stampa Digitale, Borgoricco (PD)
su carta ecosostenibile Vision Trend White
prodotta da Steinbeis Temming Papier GmbH & Co.



Sesto convegno nazionale su TEX, LATEX e tipografia digitale

Pisa, 17 ottobre 2009

Aula Magna, Scuola Superiore Sant'Anna

Call for Paper

Sabato 17 ottobre 2009 presso l'Aula Magna della Scuola Superiore Sant'Anna di Pisa si terrà il quinto Convegno annuale su TEX, LATEX e tipografia digitale organizzato dal Gruppo Utilizzatori Italiani di TEX. Il Convegno sarà un momento di ritrovo e di confronto per la comunità LATEX italiana, tramite una serie di interventi atti sia a contribuire all'arricchimento sia a supportarne lo sviluppo.
Maggiori informazioni sul Convegno e sulle modalità di presentazione degli interventi sono disponibili all'indirizzo:

`http://www.guit.sssup.it/guitmeeting/2009/`

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 7, Aprile 2009

90007
9 771828 236001