Numero 18
Ottobre
2014

ArsTEXnica

Rivista italiana di TEX e LATEX

Atti del GUIT *meeting 2014*

GUIT

http://www.guitex.org/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@guitex.org`.

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.guitex.org`
`guit@sssup.it`

Redazione ArsTEXnica:
`http://www.guitex.org/arstexnica/`
`arstexnica@guitex.org`

Codice ISSN 1828-2369

Stampata in Italia
Pisa: 15 Ottobre 2014

GUIT meeting 2014

UNDICESIMO CONVEGNO NAZIONALE SU TEX, LATEX E TIPOGRAFIA DIGITALE

TEX come strumento per la scrittura accademica

18 ottobre 2014

Verona, Polo Zanotto, *viale dell'Università, 4*

Aula T1

Università di Verona

Programma del convegno

9:00 Benvenuto. Inizio dei lavori.

9:10 *Scrivere la tesi di laurea in LATEX*. Agostino De Marco.
9:40 *Scrivere report statistici con R e Sweave*. Michele Scandola, Nadia Baltieri.
10:10 *Typesetting and highlighting Unicode source code with LATEX: a package comparison*. Roberto Giacomelli, Gianluca Pignalberi.
10:40 *LATEX per la Stesura dei Rapporti di Prova*. Renato Battistin.

11:10 — Pausa caffè (20').

11:30 *Introduzione a Bibfilex*. Massimo Nardello.
12:00 *Funzionalità avanzate del sistema biblatex/Biber*. Ivan Valbusa.
12:30 *Dealing with Ancient Works in Bibliographies*. Jean-Michel Hufflen.

13:00 — Pausa pranzo (90').

14:30 *Greek and Latin hyphenation*. Claudio Beccari.
15:00 *LilyPond: un music engraver integrabile con LATEX*. Tommaso Gordini, Davide Liessi.
15:30 *XML-TEI e Tagged PDF*. Luigi Scarso.
16:00 *Lo stile tipografico: teoria e pratica*. Claudio Vincoletto.

16:30 — Pausa caffè (20').

16:50 *Espressioni regolari in LATEX*. Enrico Gregorio.
17:20 *MenùTEX: una piattaforma per realizzare menù*. Claudio Fiandrino.
17:50 *La produzione di una rivista: ArsTEXnica*. Massimiliano Dominici.
18:20 Discussione sulle prospettive del gruppo.

19:00 Chiusura dei lavori. Arrivederci.

La partecipazione all'evento è libera e gratuita.
Registrazione online: www.guitex.org/home/meeting
Informazioni: Ivan Valbusa, ivan.valbusa@univr.it

Gruppo Utilizzatori Italiani di TEX ★ GUIT

www.guitex.org

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 18, Ottobre 2014

Gruppo Utilizzatori Italiani di TEX

Editoriale

Claudio Beccari

Credo che in passato non abbiamo mai avuto un Meeting così pieno di contributi interessantissimi come quello di quest'anno a Verona.

L'annuncio del Meeting è stato in sordina; voleva essere dedicato alla *Scrittura nel mondo accademico*, come segno di rispetto per Ivan Valbusa (che si era preso l'incarico di organizzarlo e che ringrazio moltissimo per questo pesante impegno) e per il Dipartimento di Filosofia, Pedagogia e Psicologia a cui egli fa capo. Si sarebbe desiderato anche che l'attenzione fosse dedicata prevalentemente alle discipline umanistiche.

Ovviamente il mondo di LaTeX è vasto, come è vasto l'ambito delle discipline umanistiche e accademiche in generale. Quindi i contributi che abbiamo ricevuto sono molto vari. Ne sono molto contento perché l'accento sulle materie e le discipline non collegate alla vocazione iniziale di LaTeX, la matematica e le scienze sperimentali, mostra chiaramente quanto sia alto l'interesse per le discipline umanistiche e di quanto gli studiosi di queste discipline possono beneficiare conoscendo l'enorme versatilità di questo grande sistema di tipocomposizione che ci riunisce a Verona.

Descriverò i contributi ricevuti in ordine alfabetico secondo il cognome del primo o unico autore.

Renato Battistin ci mostra come comporre i "rapporti di prova"; questi sono paragonabili agli articoli che i ricercatori universitari scrivono per presentare le metodologie e i dati sperimentali su cui hanno lavorato durante le loro ricerche, siano esse nel campo tecnologico o nel campo socio-psicologico. Riguardano anche i rapporti aziendali e industriali che hanno scopi simili.

Agostino De Marco presenta una nuova versione del suo articolo sulla composizione delle tesi di laurea; egli l'ha aggiornato agli ultimi sviluppi e alle nuove disponibilità del sistema TeX. Resta una pietra miliare anche tenendo conto che sempre più studenti di ogni facoltà e disciplina compongono le loro tesi con LaTeX.

Massimiliano Dominici ci illustra le novità che riguardano proprio questa rivista. La sua produzione in tre formati, quello a stampa, quello costituito da un unico file PDF e quello sotto forma di diversi file PDF, ognuno dei quali contiene un solo articolo come se fosse un estratto dall'intera rivista. Non è per niente facile, tenendo conto che bisogna coordinare contributi di varie persone, uniformare lo stile delle bibliografie, estrarre i sommari in inglese da inviare al TeX Users Group, per gli annunci sulla rivista internazionale *TUGboat*, eccetera. L'autore ha trovato come far fare buona parte del lavoro a LuaTeX e ce ne spiega i dettagli.

Claudio Fiandrino ci propone un metodo per costruire le schede degli elenchi delle cose da fare, le *to do lists*, elenchi di oggetti, e simili cose molto utili per i lavoro di ricerca universitaria ma con applicazioni anche in altri ambiti completamente diversi; per esempio nella costruzione e pubblicazione dei menù dei ristoranti; anzi usa proprio questo esempio per mostrare la versatilità di questo metodo.

Roberto Giacomelli e Gianluca Pignalberi ci mostrano invece un confronto fra i vari strumenti offerti da LaTeX per stampare i codici di programmazione nel modo più adatto per la loro comprensione, ma nello stesso tempo per eseguire quelle operazioni di composizione a cui bisogna dare la debita attenzione in fase di tipocomposizione, ma che coinvolgono anche problemi delicati come la gestione dei codici UNICODE dei caratteri usati.

Tommaso Gordini e Davide Liessi ci presentano un argomento nuovo nella panoramica delle applicazioni di LaTeX finora pubblicate su ArsTeXnica, vale a dire la composizione della musica; in particolare ci mostrano come sia possibile integrare il programma di composizione musicale Lilypond con LaTeX, in modo da raggiungere la stessa qualità tipografica che usavano gli incisori di spartiti nei secoli passati; la loro analisi storica è altrettanto interessante della descrizione dei mezzi tecnici per arrivare al risultato voluto.

Enrico Gregorio ci dedica un articolo di programmazione mediante il linguaggio L3 di LaTeX e ci mostra un bell'esempio per gestire le "espressioni regolari". Come di consueto, egli ci apre nuove finestre sul mondo della composizione tipografica assistita dal calcolatore. I suoi articoli ci aiutano anche a prendere familiarità con il nuovo linguaggio interno di LaTeX.

Jean-Michel Hufflen ci ha dedicato diversi articoli nei meeting passati sul suo programma per la composizione delle bibliografie, sia sotto l'aspetto dell'elenco bibliografico, sia sotto quello delle citazioni. In questo articolo egli ci mostra come gestire sia le voci dell'elenco, sia le citazioni quando si debba lavorare con riferimenti a testi antichi o antichissimi, di cui talvolta non si conosce con precisione l'autore, e meno che mai l'anno di pubblicazione ma, se va bene, se ne conosce il secolo, talvolta secoli "avanti Cristo".

Massimo Nardello ci illustra un altro program-

ma di composizione bibliografica che consente anche un facile trasferimento dal formato necessario all'elaborazione con BibTEX o con *biber*, adatti alla successiva elaborazione con LATEX, ai formati gestibili dai word processor più usati, visto che nel mondo delle scienze umane molte case editrici non sono ancora organizzate per pubblicare i testi forniti dagli autori in formato PDF.

Michele Scandiola e Nadia Baltieri ci presentano una procedura per integrare LATEX con Sweave e R; R è il tipico programma usato dai ricercatori che lavorano nel campo della statistica, in particolare nella statistica collegata alle indagini psico-sociologiche. La tipocomposizione dei risultati in questo campo presenta problemi specifici che, risolti mediante la procedura indicata, permettono ai ricercatori di pubblicare i loro risultati nel modo più efficace.

Luigi Scarso ci presenta un suo studio sui file di tipo "Tagged PDF" che contengano anche parti scritte in linguaggio XML, come richiesto dalla normativa TEI (Text Encoding Initiative). Va da sé che l'argomento è piuttosto avanzato, ma l'autore riesce a spiegarci queste problematiche e a mostrarci un'ulteriore funzionalità dei programmi di tipocomposizione del sistema TEX.

Ivan Valbusa ci mostra una serie di tecniche per configurare il processo di elaborazione dei database bibliografici con biblatex al fine di ottenere una composizione conforme ai propri desideri o alle specifiche delle case editrici. Queste talvolta, specialmente in campo umanistico, sono molto esigenti per quel che riguarda tanto la composizione dei riferimenti, quanto il modo e la forma delle citazioni.

Claudio Vincoletto ci tiene un'altra bella lezione di storia della tipografia e dei metodi oggi a disposizione per riprodurre testi antichi conservandone il layout e l'aspetto dato dal disegno della pagina e dalla scelta dei font. Egli ci mostra due esempi relativi a testi del XVI e del XVII secolo.

Infine devo presentare il mio lavoro. Avevo promesso che non avrei più afflitto i lettori di ArsTEXnica con i miei articoli sulla sillabazione e l'adattamento di LATEX a varie lingue; ma recentemente è sorto un problema sollevato da un gruppo di latinisti. È risultato che il latino e il greco avevano le impostazioni LATEX piuttosto carenti; i file contenenti i pattern di sillabazione e i file di descrizione della lingua latina mancavano delle funzionalità necessarie alla composizione del latino classico. Il greco era invece abbastanza ben servito quando si componeva con XeLATEX e polyglossia, ma era molto carente nella composizione con pdfLATEX e babel. Credo di aver risolto entrambi i problemi e spiego come.

Bene: quattordici interventi sono molti; non mi pare che in passato abbiamo avuto meeting con questo numero di interventi. Personalmente sono molto contento, ma qui è meglio che finisca la mia presentazione del meeting e di questo numero di ArsTEXnica.

Colgo l'occasione di ringraziare tutti coloro che hanno sostenuto lo svolgimento di questo meeting 2014 presso l'Università di Verona, in particolare il prof. Guglielmo Bottari, direttore del Dipartimento di Filologia, Letteratura e Linguistica, che si è prodigato in modo sostanziale per la buona riuscita del nostro incontro annuale.

▷ Claudio Beccari
Professore emerito
Politecnico di Torino
claudio dot beccari
at gmail dot com

Scrivere la tesi di laurea in LaTeX

Agostino De Marco

Sommario

Lo scopo del presente articolo è fornire gli strumenti per scrivere una tesi di laurea utilizzando LaTeX. Tale obiettivo è conseguito analizzando i problemi tipici incontrati durante la stesura della tesi e le possibili soluzioni, con una particolare attenzione ai pacchetti da usare nelle varie circostanze. I singoli argomenti non vengono approfonditi nei dettagli ma si rimanda, ove necessario, alla letteratura specifica o ai manuali dei pacchetti suggeriti.

Abstract

The goal of this article is to provide the tools to write a thesis with LaTeX. The article analyzes the problems that are usually encountered while writing a thesis and their solution, with a particular emphasis on the packages to use in each case. The topics are not examined in deep and, when necessary, the reader is referred to specific literature or to the manual of the suggested packages.

1 Introduzione

Questo articolo trae spunto dagli articoli di De Marco (2013) e Mori (2007) dedicati alla stesura della tesi di laurea in LaTeX. Dal 2007 al 2014 il sistema TeX ha subito aggiornamenti continui — alcuni dei quali hanno segnato importanti passi in avanti dal punto di vista tecnologico — e si è arricchito di nuove interessanti funzionalità. Si metteranno qui in risalto gli aspetti più significativi per chi ha intenzione di scrivere la tesi di laurea di primo o secondo livello, la monografia di laurea o la tesi di dottorato[1] in LaTeX. Come per gli articoli di Mori e De Marco (2013), il lettore non deve aspettarsi una guida alla redazione della tesi. Per chi voglia saperne di più si rimanda ai testi di Eco (1977), Lesina (2013) e, in particolare per le tesi scientifiche, ai testi di Matricciani (2000), Matricciani (2003), Matricciani (2007) e Beccari *et al.* (2011). Piuttosto, lo scopo dell'articolo è quello di dare delle indicazioni utili e generali per lavorare in modo efficace con LaTeX.

Il testo presume che il lettore conosca già i rudimenti di LaTeX, ovvero che abbia letto — o almeno sia seriamente intenzionato a leggere — una delle numerose guide di base disponibili gratuitamente in rete.

Per i lettori italiani è senz'altro consigliabile consultare *L'arte di scrivere con LaTeX* di Pantieri e Gordini[2] (chiamata in gergo l'*Arte*). Questa fonte è fondamentale soprattutto per i neofiti, che vi troveranno spiegazioni su come procurarsi tutto l'occorrente per usare LaTeX, come installarlo nel proprio calcolatore e come aggiornarne la distribuzione. Inoltre, la guida di Pantieri e Gordini presenta in maniera chiara e organica i concetti fondamentali della composizione tipografica con LaTeX, offrendo un vasto campionario di esempi e di problemi risolti.

Un'alternativa alla guida su menzionata è la *Introduzione all'arte della composizione tipografica con LaTeX* a cura del Gruppo Utilizzatori Italiani di TeX (chiamata in gergo *Guida* GUIt), adatta soprattutto agli utenti desiderosi di approfondire i dettagli del linguaggio LaTeX e i meccanismi della composizione tipografica.

In linea generale, in questo articolo si seguirà la prassi di non scandagliare troppo i vari argomenti: dei pacchetti citati, infatti, si analizzano soltanto le impostazioni più importanti e se ne suggerisce l'uso, indirizzando alla relativa documentazione chi voglia approfondirne la conoscenza.

Si ricorda che la maggioranza dei pacchetti per LaTeX è accompagnata da un manuale che ne descrive l'utilizzo e spesso presenta degli esempi. La posizione del manuale dipende dalla distribuzione TeX che si usa; le distribuzioni più diffuse offrono il comando

`texdoc` ⟨*nome pacchetto*⟩

che cerca e apre il file PDF (Portable Document Format) con il manuale del pacchetto indicato. In alternativa, disponendo di un collegamento a internet, è possibile reperire il manuale di un pacchetto all'indirizzo

`http://texdoc.net/pkg/`⟨*nome pacchetto*⟩

oppure si può cercare per parole chiave il documento che interessa attraverso l'interfaccia del sito `http://texdoc.net`.

2 Prescrizioni di formattazione di una tesi di laurea

Tutti i laureandi si trovano di fronte ad un elenco, più o meno dettagliato, di 'prescrizioni di formattazione' o 'direttive redazionali' per la tesi di laurea. Un tipico esempio potrebbe essere il seguente:

1. Da qui in avanti si userà per brevità la locuzione 'tesi di laurea' o il termine 'tesi' per indicare genericamente questo tipo di documenti.

2. `http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf`

> La tesi deve essere composta scrivendo entrambi i lati delle pagine, su fogli di formato UNI A4.
>
> I margini devono essere: superiore 20 mm, inferiore 15 mm, sinistro e destro 15 mm, rilegatura 15 mm.
>
> La distanza dal bordo per intestazione e piè di pagina deve essere di 12,50 mm.
>
> Il carattere da usare è Times New Roman, 11 pt, interlinea doppia.

Oltre a queste specifiche di tipo generale, devono essere precisate le regole di 'stile', cioè il formato delle testatine, dei piedini, dei titolini, eccetera. Molto spesso lo stile del manoscritto viene sottoposto agli studenti attraverso un file preconfezionato (creato con MS Word o con OpenOffice) che fa da modello. Il modello è esso stesso un documento che chiarisce — ma non sempre — i dettagli del layout da adottare.

Si deve osservare, purtroppo, che le prescrizioni di formato nelle università italiane in alcuni casi sono troppo generiche, in altri differiscono a seconda della scuola o dipartimento di appartenenza del relatore della tesi; in altri casi ancora, niente affatto rari, le direttive redazionali contengono delle vere e proprie castronerie dal punto di vista della tipografia professionale. Verrebbe quasi da pensare che chi ha redatto quelle prescrizioni o non è al corrente del calcolatore e usa ancora la macchina da scrivere, o impiega con scarso successo un word processor (che invece potrebbe anche produrre risultati soddisfacenti se utilizzato in modo corretto), oppure ignora completamente i rudimenti della tipografia.

Chi intende comporre la tesi in LaTeX ha il compito di interpretare le direttive e scegliere la classe di documento e/o i pacchetti di estensione necessari a raggiungere il risultato voluto. La qualità del risultato dipenderà, ovviamente, dalla predisposizione ad apprendere gli aspetti tecnici e dagli strumenti di cui si è in possesso.

È noto che molti studenti che si avvicinano a LaTeX lo fanno proprio in occasione della stesura della tesi. Per essi è fondamentale un lavoro preparatorio che richiede di:

- installare una distribuzione aggiornata del sistema TeX — ad esempio, la distribuzione TeX Live[3] o la distribuzione MikTeX;[4]
- acquisire dimestichezza con il flusso di lavoro necessario a generare un documento minimale — creazione di un file sorgente, compilazione e creazione di un output in formato PDF;
- comprendere almeno i concetti basilari della tipografia — rudimenti sui font, struttura di un manoscritto, layout, stile, eccetera.

A questo punto sarà possibile cimentarsi con il lavoro di design del manoscritto di laurea.

3. http://www.tug.org/texlive
4. http://miktex.org

3 Classi per le tesi di laurea

Dagli archivi CTAN (Comprehensive TeX Archive Network)[5] si possono scaricare diversi pacchetti che contengono l'occorrente per comporre la tesi di laurea o di dottorato con LaTeX. Fra i tanti file di estensione, che servono per estendere le classi di documento predefinite alla composizione delle tesi, esistono alcune classi e pacchetti che vale la pena citare: ClassicThesis (Miede, 2013), sapthesis (Biccari, 2012), suftesi (Valbusa, 2013), TOPtesi (Beccari, 2013), frontespizio (Gregorio, 2013), per lo più scritti da italiani per gli studenti universitari italiani.

La classe ClassicThesis, scritta da un docente tedesco ma adatta a tutte le lingue, offre un design professionale che sarebbe quanto mai indesiderabile da personalizzare, perché così si perderebbe tutto il bello del pacchetto. Siccome il layout della pagina è piuttosto originale, può non adattarsi alle specifiche di questa o quella università.

La classe sapthesis offre una soluzione completa per la composizione di tesi per studenti della Sapienza – Università di Roma.[6]

La classe suftesi fornisce uno stile di documento molto semplice e sobrio, vicino alle abitudini estetiche degli utenti umanisti.[7]

Il pacchetto TOPtesi contiene: il file di classe; un pacchetto omonimo per configurare un certo numero di classi standard; un pacchetto con comandi utili e indipendente dalla classe; un pacchetto per il frontespizio, con il quale probabilmente si possono personalizzare diverse classi, anche se non è stato creato come modulo a parte espressamente per questo scopo. La classe consente di comporre tesi in italiano e in inglese: per comporre il frontespizio in lingua diversa dall'italiano, si possono usare comandi specifici che possono essere inseriti in un file di configurazione; la tesi è personalizzabile per ogni lingua e per molti stili universitari. La classe è stata pensata anche per scrivere le tesi completamente in lingua diversa dall'italiano, in vista del fatto che gli studenti in doppia laurea con i programmi Erasmus devono scrivere la tesi anche (o solo) nella lingua dell'università ospitante.

Merita un'attenzione particolare il pacchetto frontespizio di Enrico Gregorio, esclusivamente dedicato al frontespizio della tesi. Esso è configurabile in qualsiasi dettaglio, per cui è possibile predisporre il frontespizio della tesi virtualmente per ogni prescrizione di segreteria di ateneo.[8] Esempi d'uso di

5. http://ctan.org
6. http://biccari.altervista.org/c/informatica/latex/sapthesis.php
7. http://profs.lettere.univr.it/valbusa/2010/09/17/la-classe-suftesi
8. L'unica cosa da segnalare è che il pacchetto frontespizio non consente di riprodurre le prescrizioni di formato di alcune università italiane, ma tratta solo casi in cui queste direttive sono inaccettabili dal punto di vista tipografico. Gli studenti universitari che si accingono a scrivere la tesi non si scoraggino: se una cosa non si può fare con frontespizio,

questo pacchetto sono riportati nel paragrafo 6.1.

Per coloro che decidono di comporre il manoscritto con una delle classi su menzionate, sarà consigliabile attenersi in primo luogo al manuale della classe scelta. Se si è studenti della Sapienza – Università di Roma e si sceglie sapthesis, oppure si è studenti del Politecnico di Torino e si sceglie TOPtesi, allora gran parte del lavoro è già predisposto e l'attenzione può essere concentrata con una certa disinvoltura sui contenuti anziché sul layout o sullo stile del documento. Se si ha la necessità di personalizzare queste classi perché si appartiene ad un'altra università e si devono rispettare certe direttive di formato, sarà bene valutare se si è veramente in grado di realizzare le personalizzazioni richieste in un tempo dato. I forum di utilizzatori di LaTeX sono pieni di richieste disperate di aiuto da parte di studenti che hanno poco tempo per la consegna del manoscritto e che non riescono a risolvere questo o quel problema di formattazione.

La classe TOPtesi di Beccari può effettivamente rivelarsi una buona scelta, perché ha un manuale d'uso chiaro ed è abbastanza elastica da poter essere personalizzata anche dai meno esperti.

In alternativa alle soluzioni precedenti si può scegliere di utilizzare la classe di documento predefinita book, selezionando via via i pacchetti di estensione che permettono di realizzare le soluzioni tipografiche desiderate.[9]

La parte rimanente di questo articolo propone appunto questa strada. Naturalmente, a parte la scelta della classe di documento e qualche altro aspetto legato al layout e allo stile, il resto dell'articolo contiene argomenti che interessano anche chi sceglie ClassicThesis, sapthesis, suftesi o TOPtesi o altre classi ancora,[10]

4 La tesi con la classe book

Per una tesi di laurea è possibile utilizzare la classe predefinita book. Nelle opzioni della classe, oltre alla dimensione del font di base (`10pt`, `11pt` o `12pt`)[11] e a quella del foglio (tipicamente `a4paper`), è possibile scegliere:

- se avere un documento fronte-retro (`twoside`) o solo fronte (`oneside`);
- se collocare la prima pagina dei capitoli su facciate destre (`openright`) o indifferentemente (`openany`).

Si suggerisce di utilizzare la classe book invece di quella report, in quanto la prima prevede tre comandi (`\frontmatter`, `\mainmatter` e `\backmatter`)[12] che controllano il formato del numero di pagina e la numerazione dei capitoli. Nel *frontmatter* le pagine sono numerate con i numeri romani minuscoli (i, ii, iii, ecc.) e i capitoli non sono numerati come se si utilizzasse il comando asteriscato `\chapter*{}`, ma vanno a finire nell'indice (mentre di solito i capitoli iniziati con `\chapter*{}` non compaiono nell'indice). Nel *mainmatter* le pagine sono numerate con numeri arabi (la numerazione riparte da 1) e i capitoli sono anch'essi numerati con cifre arabe. Nel *backmatter* le pagine sono numerate come nel mainmatter (la numerazione prosegue da questa), ma i capitoli non sono numerati.

Si consiglia inoltre di utilizzare l'opzione fronte-retro (`twoside`), in quanto:

- si dimezza l'uso di fogli di carta;[13]
- è possibile usare testatine differenziate per pagine sinistre e destre;
- i libri sono scritti in questo modo (dunque ci si aspetta che chi legge la tesi sia abituato a questo layout).

Se ad esempio si vuole avere la tesi con dimensione del corpo 11 pt, stampata fronte-retro su fogli A4, con collocazione della prima pagina dei capitoli su facciate destre, va usato il comando:

```
\documentclass[11pt,a4paper,twoside,%
    % ... eventuali altre opzioni
    openright]{book}
```

Alternativamente può essere utilizzata la classe memoir (Wilson, 2010), che risulta particolarmente flessibile e permette di personalizzare molti aspetti del documento (testatine, titoli capitoli, note, indici, ecc.) senza dover caricare altri pacchetti. Si rimanda alla documentazione della classe per i dettagli. L'uso di memoir non è consigliabile per gli utenti neofiti; il manuale d'uso è abbastanza voluminoso, pertanto leggerlo e capirlo senza possedere la dovuta predisposizione per la materia potrebbe risultare un carico di lavoro non tollerabile da alcuni. D'altra parte, il manuale di memoir è un'ottima fonte di informazioni per chi vuole approfondire le sue conoscenze sulla tipografia.

5 Organizzazione dei file

5.1 La codifica dei sorgenti

Il problema della codifica dei file di testo è delicato e spesso difficile da capire per chi non conosce il funzionamento interno del proprio calcolatore. Per

allora vuol dire che è meglio non farla.

9. In tal caso sarà bene assicurarsi di aver installato una versione completa del sistema TEX, così da avere già nel proprio computer i file necessari e pronti all'uso.

10. Ad esempio la classe scrbook del pacchetto KOMA-Script (Kohm e Morawski, 2012).

11. Per avere una buona leggibilità su fogli A4 è consigliabile usare un font di base di dimensione 11 pt.

12. Per l'uso di tali comandi si rimanda al paragrafo 6.

13. Un comportamento comune a molti laureandi consiste nell'usare qualunque strumento tipografico possibile per aumentare il numero di pagine della tesi (allargando i margini, aumentando la dimensione del font, aumentando l'interlinea, inserendo molte figure, stampando solo fronte, ecc.). Tralasciando il fatto che la qualità dei contenuti è più importante della quantità, spesso questi espedienti producono dei risultati tipografici pessimi. Si consiglia dunque di concentrarsi sui contenuti e lasciar perdere l'impostazione tipografica (a questo pensa LaTeX) e soprattutto il numero di pagine prodotto.

approfondire l'argomento si consiglia la guida di Beccari e Gordini (2012).

Dal punto di vista pratico, gli utenti di LaTeX devono preoccuparsi di come il proprio editor[14] gestisce la codifica dei caratteri. Se ne cita qui uno per tutti: TeXworks,[15] l'editor multipiattaforma che si installa quando viene installata la distribuzione TeX Live o la distribuzione MikTeX. Si dice *codifica di input* il modo in cui sono codificati i caratteri che si immettono nei file `.tex` (e nei file di testo in generale), vuoi attraverso tastiera ed editor, vuoi leggendo con quest'ultimo un file preesistente per modificarlo. Normalmente un editor salva i file con la stessa codifica di quella con cui è configurato. L'editor TeXworks può anche salvarli con una codifica diversa da quella di default. Di solito questa funzionalità non è un problema, anzi può essere molto utile per cambiare codifica a un file.

Si consiglia di impostare la codifica dei file sorgenti della tesi come UTF-8. Per informare il programma di composizione sulla codifica con cui il file sorgente è salvato, basta inserire nel preambolo la chiamata al pacchetto inputenc, specificando nel suo argomento la sigla della codifica in questione. In pratica, nel preambolo basta dare il comando:

```
\usepackage[utf8]{inputenc}
```

Prima di caricare inputenc si consiglia di caricare non solo il pacchetto fontenc con le opzioni che si desiderano, ma anche il pacchetto textcomp e ogni altro pacchetto che carichi collezioni di simboli speciali nell'ordine seguente:

```
\usepackage[T1]{fontenc}
% ... eventuali pacch. per font particolari
\usepackage{textcomp}
% ... eventuali pacch. per simboli speciali
\usepackage[utf8]{inputenc}
```

L'editor TeXworks ha il vantaggio di comprendere particolari istruzioni di autoconfigurazione sulla base delle quali adattare 'al volo' le proprie impostazioni, qualunque esse siano. Un utente di TeXworks ha la possibilità di 'configurare il sorgente' immettendo all'inizio del documento delle righe di commento speciali, anche dette in gergo *righe magiche*. Le righe magiche danno a TeXworks alcune importanti informazioni di autoconfigurazione:

- se esiste e come si chiama il file principale (un'ovvietà superflua se il documento è in un unico file, ma molto utile se si suddividono i sorgenti di un lungo documento in più file);
- la codifica usata per scriverlo (ad esempio, `UTF-8 Unicode`);
- il programma di composizione che si userà per comporlo (ad esempio, `pdflatex`, `xelatex` o `lualatex`);
- volendo, il dizionario ortografico della lingua principale del documento.

Ecco come vanno scritte queste righe (gli spazi resi qui con il simbolo ␣ sono significativi):

```
%␣!TEX␣root␣=␣./tesi.tex
%␣!TEX␣encoding␣=␣UTF-8␣Unicode
%␣!TEX␣program␣=␣pdflatex
%␣!TEX␣spellcheck␣=␣it-IT
```

Ciò fatto, anche se si lavora su computer e sistemi operativi diversi, usando l'editor impostato con codifiche diverse (ad esempio, su uno è impostata di default la codifica LATIN1 e sull'altro la UTF-8), si può aprire il file in questione e lavorarci sopra senza dover usare alcuna accortezza preliminare, poiché sarà l'editor ad autoconfigurarsi correttamente.

5.2 Suddivisione dei sorgenti

La gestione di documenti articolati come un libro o una tesi di laurea può diventare complessa, dunque è auspicabile suddividere il testo in più file. LaTeX permette di avere un *main file* che viene compilato per produrre il risultato finale e nel quale sono richiamati altri file sorgenti. Molti usano nominare il file principale `main.tex`; per una tesi di laurea si potrebbe scegliere il nome `tesi.tex`. Eventualmente, se la tesi deve essere consegnata come file PDF, l'output della compilazione può essere rinominato da `tesi.pdf` a `Tesi_Magistrale_Matteo_Rossi.pdf`, per esempio.

Gli altri file sorgenti vengono richiamati dal sorgente principale con i comandi `\include` e `\input`. Il comando

```
\input{⟨nome file⟩}
```

permette il *nesting*, ovvero rende possibile richiamare un file che ne richiama a sua volta un altro. Il comando:

```
\include{⟨nome file⟩}
```

non permette il *nesting*, ma inserisce un comando `\clearpage` prima del testo che contiene e permette di utilizzare il comando:

```
\includeonly{⟨nome file 1⟩,
   ⟨nome file 2⟩, ⟨nome file 3⟩ ... }
```

per inserire solo i file specificati tra parentesi. Quando si usa `\includeonly`, invece, vengono compilati solamente i file tra parentesi graffe e si aggiornano i contatori ad essi collegati (numeri di pagina, numeri di note, ecc.). I contatori dei file già compilati e non inclusi da `\includeonly` non vengono aggiornati.

14. Programma di creazione e di gestione dei file di testo, cioè dei sorgenti.

15. `http://www.tug.org/texworks`. Si può scaricare un'agile introduzione al programma da `http://profs.sci.univr.it/~gregorio/introtexworks.pdf`.

6 Sezioni della tesi

L'organizzazione della tesi di laurea è argomento di specifici manuali di scrittura (Eco, 1977; Lesina, 2013; Matricciani, 2000, 2003) e in particolar modo della normativa ISO relativa alla presentazione dei rapporti scientifici e tecnici UNI-ISO 5966 (1989). Molto dettagliata è la guida di Beccari *et al.* (2011), liberamente scaricabile dal sito del Politecnico di Torino. In questo paragrafo si propone una possibile struttura per la tesi e si affrontano le problematiche relative ad ogni sezione.

Una tesi può in generale presentarsi con la seguente struttura:[16]

- Il frontespizio°
- La dedica*°
- Il sommario*°
- I ringraziamenti*°
- Gli indici°
- I simboli e le notazioni*
- La prefazione*

} frontmatter

- I capitoli interni
- Le appendici*

} mainmatter

- La bibliografia
- L'elenco degli acronimi*
- L'indice analitico*

} backmatter

6.1 Il frontespizio

La struttura e il contenuto del frontespizio sono generalmente imposti dalla scuola presso cui la laurea è conseguita, dunque è necessario crearlo *ad hoc*. Uno dei problemi che spesso si presentano è quello di produrre un frontespizio adeguato che sia ben centrato sulla prima pagina.

Per gli utenti italiani esiste una soluzione già pronta e facilmente personalizzabile, costituita dal pacchetto frontespizio. Il vantaggio di usare questo pacchetto è che i comandi necessari per definire i vari elementi del frontespizio (titolo, candidato, relatore e così via) sono contenuti nello stesso documento.

Per definire il frontespizio si deve usare l'ambiente `frontespizio` che può essere posizionato subito dopo il comando `\begin{document}`. All'interno dell'ambiente vanno dati i comandi che definiscono i vari elementi del frontespizio. Se il documento principale si chiama `tesi.tex`, alla prima compilazione verrà generato automaticamente il documento `tesi-frn.tex`, che si troverà nella stessa cartella che contiene quello principale. Il documento `tesi-frn.tex` va anch'esso compilato per generare il file `tesi-frn.pdf`, che verrà posizionato automaticamente come prima pagina di `tesi.pdf`. La sequenza di comandi è, dunque,

```
pdflatex tesi
pdflatex tesi-frn
pdflatex tesi
```

e, alla fine, il frontespizio sarà al suo posto. Non occorrerà dare ogni volta questi comandi: basta farlo solo quando si modifica il contenuto dell'ambiente `frontespizio`.

Se la classe `book` è chiamata con l'opzione `oneside`, il frontespizio occupa correttamente solo la prima pagina; nel caso di `twoside`, viene prodotta una seconda pagina bianca.

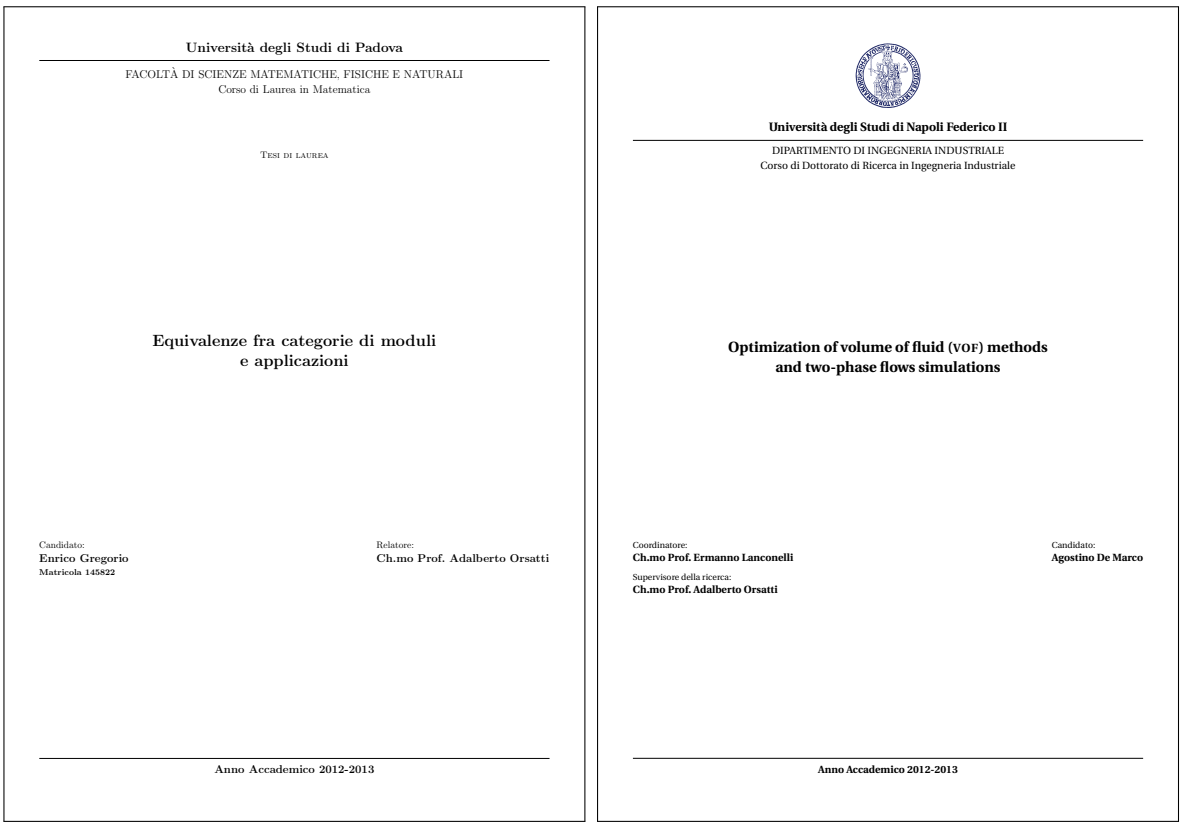
Il documento va impostato dando al comando `\documentclass` l'opzione `titlepage`, per poi caricare nel preambolo il pacchetto frontespizio. Per esempio:

```
\documentclass[a4paper,
  % ... altre opzioni
  titlepage]{book}
% ... altri comandi del preambolo
\usepackage{frontespizio}

\begin{document}
\begin{frontespizio}
\Universita{Padova}
\Facolta{Scienze Matematiche, Fisiche e
    Naturali}
\Corso[Laurea]{Matematica}
\Titoletto{Tesi di laurea}
\Titolo{Equivalenze fra categorie di moduli\\
e applicazioni}
\Candidato[145822]{Enrico Gregorio}
\Relatore{Ch.mo Prof.~Adalberto Orsatti}
\Annoaccademico{2012-2013}
\end{frontespizio}
% ... il resto della tesi
\end{document}
```

produce il frontespizio riportato nella figura 1a. L'esempio seguente:

```
\documentclass[a4paper,titlepage]{book}
\usepackage[swapnames]{frontespizio}
\begin{document}
\begin{frontespizio}
\begin{Preambolo*}
\usepackage{fourier}
\newcommand{\VOF}{\textsc{vof}}
\end{Preambolo*}
\Universita{Napoli Federico II}
\Logo[2.5cm]{Sigillo_UNINA_FedericoII_BLUE}
\Dipartimento{Ingegneria Industriale}
\Corso[Dottorato di Ricerca]{Ingegneria
    Industriale}
```

16. Il simbolo * contraddistingue le sezioni facoltative, mentre ° indica che le sezioni non devono essere presenti nell'indice.



(a) (b)

Figura 1: Esempi di frontespizio.

```
\Titolo{Optimization of volume of fluid
  (\VOF) methods\\
  and two-phase flows simulations
}
\Candidato{Agostino~De~Marco}
\Relatore{Ch.mo Prof.~Ermanno Lanconelli}
\NRelatore{Coordinatore}{}
\Correlatore{Ch.mo Prof.~Adalberto Orsatti}
\NCorrelatore{Supervisore della ricerca}{}
\Annoaccademico{2012-2013}
\end{frontespizio}
...
\end{document}
```

produce il frontespizio della figura 1b e mostra anche la possibilità di inserire un'immagine che rappresenta il logo dell'ateneo.

6.2 La dedica

La dedica, ove presente, può assumere le più svariate forme a seconda dei gusti dell'autore. Di solito (vedi la figura 2) è costituita da una riga allineata a destra, ad esempio, con i comandi:

```
\begin{flushright}
...
\end{flushright}
```

La posizione verticale della riga nella pagina può essere scelta a piacere e per controllarla risulta particolarmente conveniente l'uso di una coppia di comandi `\vspace{\stretch{...}}`. In questo modo è infatti possibile impostare il rapporto tra lo spazio che precede la dedica e quello che segue. Se ad esempio si vuole che lo spazio che segue sia il doppio di quello che precede, è possibile usare i comandi:

```
\null\vspace{\stretch{1}}
\begin{flushright}
  \textit{A Valeria e ai miei genitori}
\end{flushright}
\vspace{\stretch{2}}\null
```

6.3 Il sommario

Le classi article e report — ma non di default la classe book — definiscono un ambiente

```
\begin{abstract}
...
\end{abstract}
```

per il sommario o *abstract* dei contenuti di un documento. Se si utilizza la classe book è necessario inserire nel preambolo la definizione di tale ambiente. Si riporta qui una definizione ispirata a quella della classe report

```
nel preambolo
\usepackage{fancyhdr}

\newenvironment{abstract}%
  {\cleardoublepage%
    \thispagestyle{empty}%
```

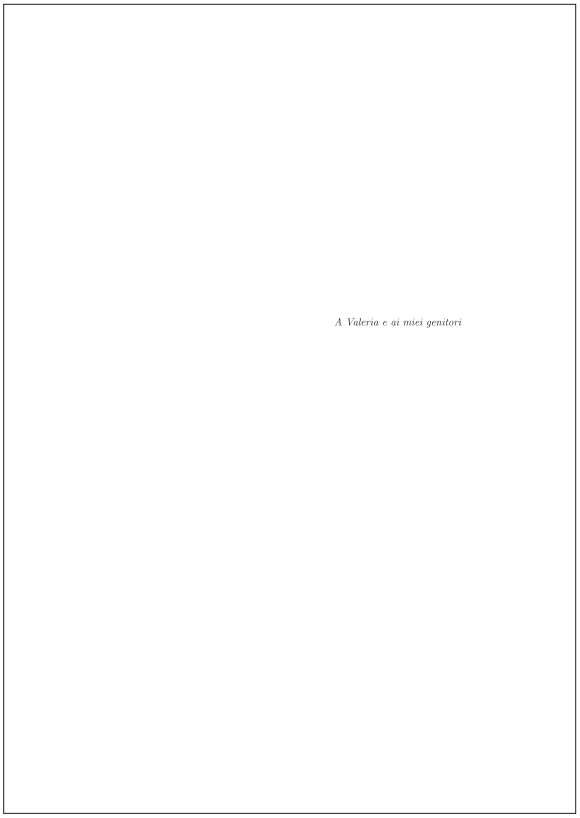


FIGURA 2: Esempio di dedica.

```
    \null \vfill\begin{center}%
      \bfseries \abstractname \end{center}}%
  {\vfill\null}
```

Si noti l'utilizzo del pacchetto fancyhdr al quale si accennerà nel paragrafo 9.2.1.

Per le tesi di laurea in italiano è spesso richiesto che sia presente anche la traduzione inglese dell'abstract. Utilizzando il pacchetto babel è possibile selezionare la lingua per le due versioni dell'abstract, in modo che sia effettuata la corretta sillabazione delle parole e che sia caricato automaticamente il corretto titolo del sommario. Dopo aver richiamato il pacchetto nel preambolo con il comando:

```
\usepackage[english,italian]{babel}
```

è sufficiente inserire i sommari come segue:

```
\begin{abstract}
... versione del sommario in italiano ...
\end{abstract}

\selectlanguage{english}
\begin{abstract}
... English version of the abstract ...
\end{abstract}
\selectlanguage{italian}
```

Il risultato è riportato nella figura 3.

6.4 Gli indici

Gli indici di solito sono posizionati subito dopo il sommario nel seguente ordine:

- indice
- elenco delle figure
- elenco delle tabelle
- altri elenchi

e vengono prodotti automaticamente da LATEX con i comandi:

```
\tableofcontents
\listoffigures
\listoftables
```

Per creare elenchi di oggetti flottanti personalizzati (ad esempio listati di programmi, algoritmi, eccetera), si faccia riferimento al pacchetto float e ai relativi comandi `\newfloat` e `\listof`. Per modificare il layout degli indici è possibile utilizzare il pacchetto tocloft.

6.5 I simboli e le notazioni

Talvolta risulta opportuno far precedere al testo della tesi un elenco dei simboli e delle notazioni utilizzate. A questo scopo può essere utilizzato il pacchetto nomencl. Un'alternativa più potente a nomencl è il pacchetto glossaries, che permette anche di creare un elenco degli acronimi menzionati nel testo e un glossario. Entrambi i pacchetti generano gli elenchi automaticamente per mezzo del programma `makeindex` e, accoppiati all'uso del pacchetto hyperref, generano automaticamente anche i collegamenti ipertestuali tra il simbolo, l'acronimo, il termine menzionato nel testo e la relativa spiegazione nell'elenco. Si rimanda ai rispettivi manuali d'uso per approfondimenti.

Ovviamente, per semplicità, è anche possibile creare manualmente l'elenco, ad esempio utilizzando l'ambiente tabular. Nella figura 4 si riporta un esempio.

6.6 Le appendici

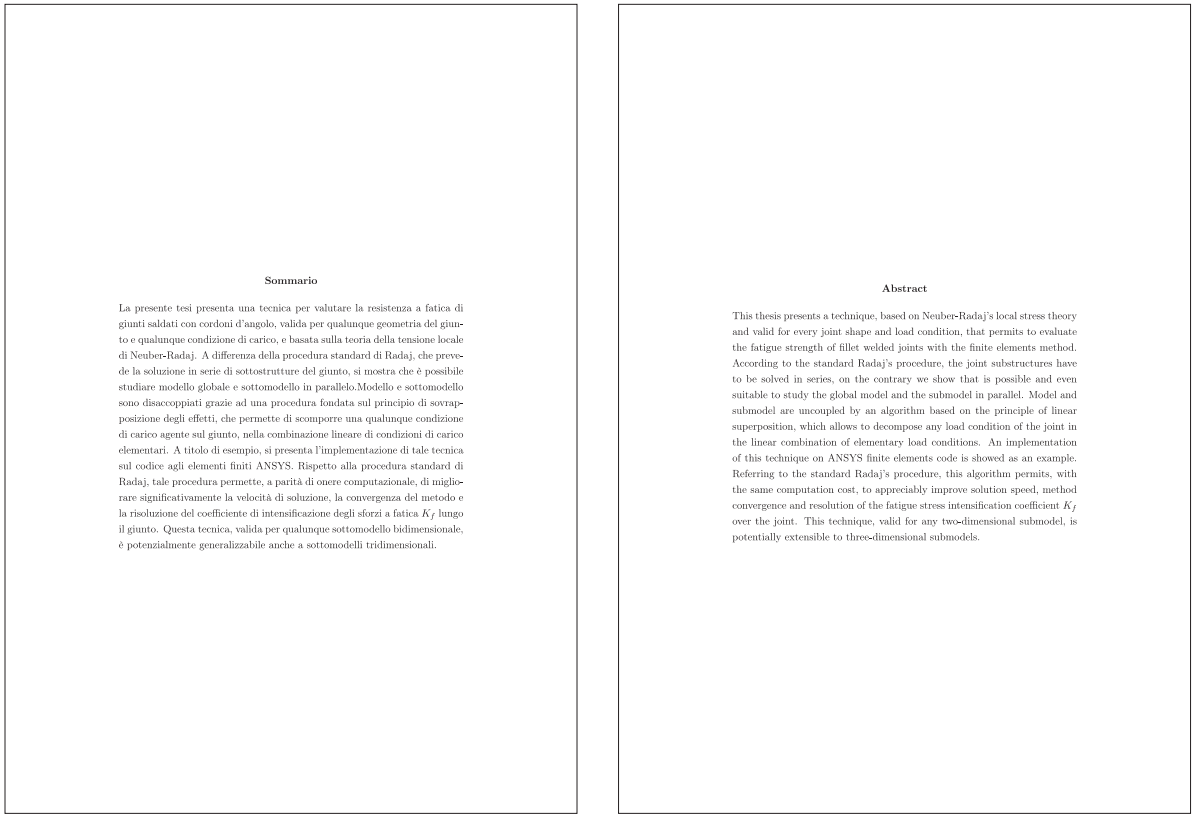
Le appendici sono dei normali capitoli la cui numerazione è però in lettere latine. LATEX permette di crearle semplicemente con il comando `\chapter{...}` preceduto da `\appendix`; se si hanno più appendici, `\appendix` deve essere richiamato solo una volta. Si riporta un esempio:

```
...
\mainmatter
\include{capitolo1}
\include{capitolo2}
\include{capitolo3}

\appendix
\include{appendice1}
\include{appendice2}
...
```

6.7 L'indice analitico

L'indice analitico può essere creato automaticamente per mezzo del pacchetto imakeidx.

L'esempio seguente:



(a) (b)

FIGURA 3: Esempio di abstract in doppia lingua.

```
\usepackage{imakeidx}
...
\makeindex[title=Concept index]
\makeindex[name=persons,title=Index of
    names,columns=3]
...
\begin{document}
...
la relatività.\index{relativity}
...
Einstein.\index[persone]{Einstein, Albert}
...
E fu da quel punto che fu data alla teoria il
nome di \emph{Teoria della relatività}.

\printindex

\indexprologue{\small
  In questo indice troverete un elenco
  di scienziati famosi citati in questa
  tesi.
}
\printindex[persone]

\end{document}
```

produce due indici analitici. Il secondo è preceduto da un breve testo di spiegazione.

Si rimanda al manuale d'uso del pacchetto per le eventuali necessità di personalizzazione del formato.

6.8 La bibliografia

La bibliografia è una parte importante della tesi di laurea. LaTeX offre tutti gli strumenti per realizzarla e gestirla con efficienza e flessibilità. L'argomento richiede la comprensione di alcuni aspetti tecnici e, al solito, si consiglia di approfondirne i dettagli consultando la guida di PANTIERI e GORDINI. Qui si richiamano gli elementi fondamentali per gestire le citazioni bibliografiche e il database delle fonti con il pacchetto biblatex.

La gestione efficiente della bibliografia è basata sulla generazione automatica di un insieme di voci bibliografiche citate durante il testo della tesi. Le voci bibliografiche vengono 'estratte' da una collezione (database) di fonti preparata in precedenza. Il database è un file di testo di estensione `.bib` che va editato a parte inserendovi dei record opportunamente formattati. Esiste un ottimo programma multipiattaforma per la creazione di database bibliografici chiamato Jabref.[17] Esso è dotato di un'interfaccia grafica e di potenti funzioni di gestione.

Un esempio di database bibliografico contenente un certo numero di record è il seguente:

```
@book{eco:tesi,
  author = {Eco, Umberto},
```

17. http://jabref.sourceforge.net

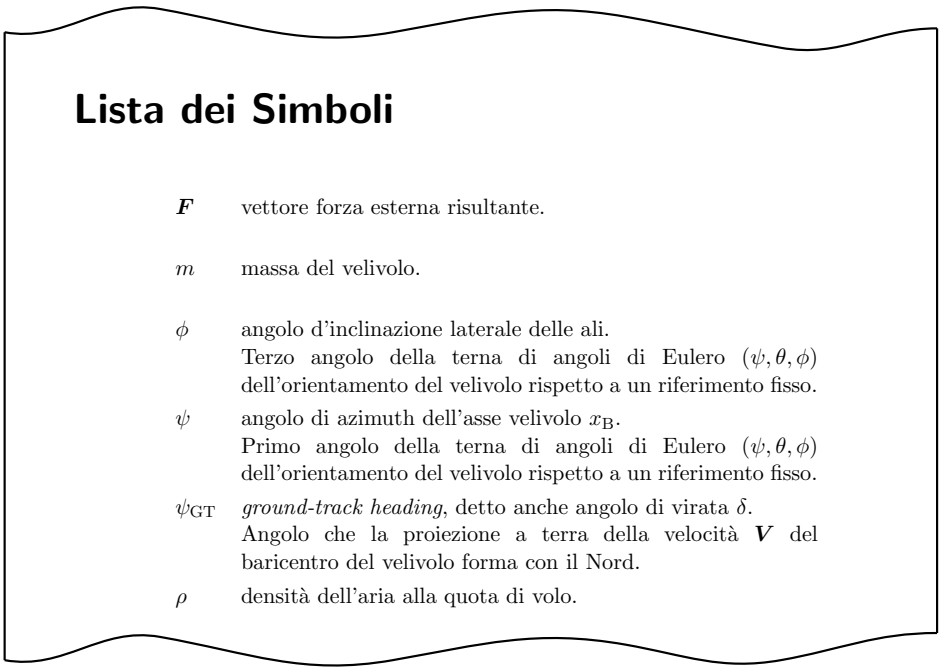


FIGURA 4: Esempio di elenco dei simboli.

```
  title = {Come si fa una tesi di laurea},
  publisher = {Bompiani},
  date = {1977},
  location = {Milano},
}

@article{mori:tesi,
  author = {Mori, Lapo Filippo},
  title = {Scrivere la tesi di laurea con
    \LaTeX},
  journaltitle = {\Ars},
  number = {3},
  date = {2007},
}

@manual{beccari:gordini:codifiche,
  title = {Codifiche in {\TeX} e {\LaTeX}.
    Dal sorgente al PDF, guida pratica per
    lavorare con successo.},
  author = {Beccari, Claudio and Gordini,
    Tommaso},
  publisher = {{\GuIT}},
  year = {2012},
}

@online{wiki:latex,
  title = {\LaTeX{} su Wikipedia},
  date = {2012},
  url = {http://it.wikipedia.org/wiki/LaTeX},
  sortkey = {wiki},
  label = {wiki},
}
```

Il primo record è un esempio di voce bibliografica riferita a un libro (`@book`), il secondo è un esempio di articolo su rivista (`@article`), il terzo record è un riferimento a un manuale (`@manual`), il quarto è un sito internet (`@online`).

Ciascun record ha dei campi che vanno dal titolo all'autore, all'anno di pubblicazione, e così via.[18] I record sono identificati da una *chiave*; per esempio, il record del libro di Eco ha per chiave `eco:tesi`. Le chiavi vengono stabilite dall'utente e devono essere usate nel testo della tesi allorquando si voglia inserire una citazione bibliografica.

Il programma 'estrattore' delle voci bibliografiche dal file `.bib`, che lavora tenendo conto delle effettive citazioni presenti nella tesi, si chiama `biber` e fa parte delle moderne distribuzioni TEX.

Il pacchetto biblatex è un potentissimo strumento — pensato per interfacciarsi con `biber` — con il quale si gestisce automaticamente la bibliografia e si personalizza ogni aspetto degli stili bibliografici e di citazione con poche operazioni. Per funzionare correttamente, il pacchetto richiede di caricare anche il pacchetto babel (o polyglossia, se si compone con XeLATEX) e csquotes con le opzioni indicate di seguito.

Nel preambolo vanno dati i comandi:

```
\usepackage[italian]{babel}% tesi in italiano

\usepackage[
    autostyle,italian=guillemets
    % ... altre opzioni
  ]{csquotes}

\usepackage[
    % ... opzioni
    backend=biber
  ]{biblatex}
```

18. ciascun campo, come si vede, va terminato con la virgola, *anche se è l'ultimo*, pena un errore.

Se, per esempio, la base di dati è stata nominata `bibliografia-tesi.bib`, per indicare a LaTeX di usare questo file per comporre la bibliografia, si deve dare nel preambolo il comando:

```
\addbibresource{bibliografia-tesi.bib}
```

Se si ha più di un database bibliografico, il comando precedente deve essere ripetuto per ogni file specificando sempre l'estensione `.bib`.

A questo punto, nel testo della tesi, la citazione di una fonte bibliografica sarà semplicemente ottenuta con un qualcosa di simile:

```
Si veda~\cite{eco:tesi} per maggiori
    dettagli.
```

cioè con il comando `\cite`. Per ottenere il risultato voluto, che potrebbe essere il seguente:

> Si veda [1] per maggiori dettagli.

la sequenza di comandi di compilazione è:

```
pdflatex tesi
biber tesi
pdflatex tesi
pdflatex tesi
```

Ovviamente questa sequenza è richiesta solo quando si modifica `bibliografia-tesi.bib` (per esempio, dopo aver aggiunto una voce o aver corretto un errore). La doppia compilazione con `pdflatex` dopo l'esecuzione di `biber` è necessaria per la corretta gestione delle informazioni trascritte nei file ausiliari.

Si osservi che il formato finale della citazione dipende dallo stile richiesto tramite le opzioni passate al pacchetto biblatex. L'esempio precedente potrebbe avere l'aspetto seguente:

> Si veda ECO (1977) per maggiori dettagli.

con lo stile della citazione che passa da quello 'numerico' a quello cosiddetto 'autore-anno'.

Il comando `\printbibliography` posizionato al termine del testo della tesi produce la *sezione bibliografica* con relativi titolo e testatina. La sezione bibliografica non è altro che l'elenco dei riferimenti bibliografici, opportunamente ordinati e formattati.

Per mandare nell'indice generale il titolo della bibliografia del documento, va data la sequenza di comandi seguente (valida per la classe di documento `book`):

```
\addcontentsline{toc}{chapter}{\bibname}
\printbibliography
```

Se si usa `babel` per un documento in italiano il comando `\bibname` produce nell'indice generale del documento la voce *Bibliografia*. Se nell'indice si vuole la voce *Riferimenti bibliografici* basta ridefinire `\bibname` tramite il comando

```
\addto\captionsitalian{%
  \renewcommand*{\bibname}%
    {Riferimenti bibliografici}%
}
```

Si osservi che l'aspetto dei riferimenti bibliografici e delle citazioni, che biblatex adatta automaticamente alla lingua principale del documento, si specificano in diversi modi. Il pacchetto fornisce quattro stili bibliografici predefiniti, i quali agiscono nella sezione bibliografica del documento. Essi ordinano le opere (ad esempio, alfabeticamente in base al cognome dell'autore o del curatore); possono contrassegnare o meno l'opera con un'etichetta; sistemano opportunamente i dati nei riferimenti bibliografici.

Si rimanda all'*Arte* di Pantieri e Gordini o al manuale di biblatex (Lehman, 2013) per approfondimenti sugli stili e sugli schemi di citazione.

7 Gli oggetti

7.1 Le figure

Le figure sono uno degli argomenti trattati più estesamente dalle guide. Al solito, l'*Arte* di Pantieri e Gordini è un'ottima fonte di approfondimento.

I problemi incontrati dagli utenti LaTeX durante l'inserimento di figure sono generalmente di due tipi. Una parte dei problemi deriva dalle figure in sé, ovvero dal file che si cerca di inserire in un documento (argomento trattato nel par. 7.1.1), mentre un altro tipo di problemi, totalmente distinto dal precedente, è quello degli oggetti flottanti (argomento trattato nel par. 7.3).

7.1.1 Formati

Esistono due grandi classi di figure, le immagini vettoriali e le immagini bitmap. Le prime sono descritte da forme e possono essere scalate e/o deformate senza perdere definizione; sono soprattutto adatte per i grafici e per gli schemi. Le seconde sono matrici di pixel colorati e sono adatte per le fotografie.

La prima cosa da fare è produrre figure nel formato più adatto per i propri scopi. È inutile salvare grafici o schemi in `.jpeg` per poi convertirli in `.pdf`, in quanto la conversione di un'immagine bitmap in `.pdf` include semplicemente il file bitmap in una "cornice" (tipica del formato Encapsulated PostScript, da cui il PDF deriva) senza migliorare in alcun modo la qualità. È inutile anche fare la conversione opposta, da file vettoriale a bitmap, perché in questo modo si perdono le informazioni sulla geometria contenuta nella figura e quindi si abbassa la qualità del file.

L'elemento più importante di un file PDF è il Bounding Box (BB), che determina la taglia effettiva dell'immagine e che serve a LaTeX per calcolare lo spazio da riservare alla figura. Idealmente i BB

dovrebbero essere al limite massimo del contenuto dell'immagine, ma spesso i programmi di grafica lasciano grandi bordi bianchi attorno alla figura disegnata. Questo porta spesso a grandi confusioni, perché di fatto LaTeX sta lasciando alla figura lo spazio corretto, ma visivamente parte di questo è utilizzato per il bordo bianco, quindi la figura appare troppo piccola, non centrata, con eccessivi margini verticali, eccetera. La prima cosa da verificare è quindi che il programma di grafica generi dei file `.pdf` con BB corretti. Per farlo basta aprire la figura con Ghostview[19] e attivare la visualizzazione dei BB. Se questi non sono corretti bisogna cercare di configurare correttamente il programma di grafica, ma il problema non ha niente a che vedere con LaTeX. Nel caso in cui si abbiano molti file che presentano questo inconveniente bisogna cercare di correggere il problema all'origine.

7.1.2 Pacchetti utili

Per inserire le figure è necessario caricare il pacchetto graphicx, della cui guida si consiglia la lettura. Per ottenere sottofigure (vedi ad esempio la figura. 1) è necessario caricare il pacchetto subcaption.

In casi semplici non è necessario ricorrere a quest'ultimo pacchetto, visto che all'interno degli ambienti figure e table si può mettere più di un grafico o di una tabella. Se si hanno quindi due o più figure che possono essere raggruppate insieme, basta scrivere:

```
\begin{figure}[tb]
  \begin{minipage}{0.48\textwidth}
    \includegraphics[%
      width=\linewidth]{fig_a.pdf}
    \caption{Prima didascalia (destra).}
    \label{fig:a}
  \end{minipage}
  \hspace{4em}
  \begin{minipage}{0.48\textwidth}
    \includegraphics[%
      width=\linewidth]{fig_b.pdf}
    \caption{Seconda didascalia (sinistra).}
    \label{fig:b}
  \end{minipage}
\end{figure}
```

In questo modo si riduce il numero di oggetti flottanti e se ne facilita l'inserimento.

Al fine di mantenere ordine nei file sorgenti, è consigliabile raccogliere tutte le figure in una o più sottocartelle; se ad esempio tali sottocartelle si chiamano `dir_1` e `dir_2`, è sufficiente inserire nel preambolo con il seguente comando del pacchetto graphicx:

```
\graphicspath{{dir_1/},{dir_2/}}
```

L'argomento di `\graphicspath` è relativo alla cartella dove risiede il main file `.tex` che viene compilato.

19. http://pages.cs.wisc.edu/~ghost

La formattazione delle didascalie può essere convenientemente controllata con il pacchetto caption.

Qui vale la pena menzionare il pacchetto di estensione adjustbox che, tra le sue svariate funzionalità, offre il comando `\adjincludegraphics` (simile a `\includegraphics`) che permette di effettuare agevoli operazioni di rifilatura (*cropping*). Ad esempio, il codice:

```
\adjincludegraphics[width=0.7\linewidth,
  trim={{.05\width} {.02\height} 0 0},% lbrt
  clip]{mia-figura.pdf}
```

inserisce l'immagine `mia-figura.pdf` ritagliandone dal lato sinistro (`l`, *left*) una striscia di larghezza pari al 5% della larghezza originale e dal lato in basso (`b`, *bottom*) una striscia di altezza uguale a 2% dell'altezza originale. Il risultato del ritaglio viene poi scalato in modo da avere un'immagine sulla pagina di larghezza uguale al 70% della `\linewidth`.

7.2 Le tabelle

Così come per le figure, anche per le tabelle esistono guide specifiche a cui si rimanda per ogni approfondimento Mori (2006).

Per migliorare la spaziatura dell'ambiente tabular standard è possibile utilizzare il pacchetto ctable, mentre se si vogliono colorare le righe o le colonne è necessario caricare il pacchetto xcolor con l'opzione `table`.

Nel caso di tabelle di grandi dimensioni è possibile ridurre la dimensione della tabella effettuando una scalatura, ad esempio con i seguenti comandi:

```
\begin{center}
  \resizebox{0.95\textwidth}{!}{%
    \begin{tabular}
      ...
    \end{tabular}
  }
\end{center}
```

Si può anche ruotare la tabella di 90° con il pacchetto rotating. Altre tecniche per ruotare immagini e tabelle sono indicate nel manuale del pacchetto hvfloat (Voss, 2013).

Infine, si può spezzare la tabella su più pagine con il versatile pacchetto longtable.

7.3 Controllo degli oggetti flottanti

Spesso gli utenti si lamentano del fatto che LaTeX sposti le figure (e in generale gli oggetti flottanti) lontano dal punto in cui vengono inserite. Nella maggioranza dei casi questo è dovuto ad un utilizzo erroneo delle opzioni di posizionamento degli oggetti flottanti. Qui si vuole sottolineare che alcune scelte devono essere prese nella fase di stesura del testo (paragrafo 7.3.1), mentre altre sono riservate, quando necessarie, alla fase di revisione (paragrafo 7.3.2).

7.3.1 Cosa fare durante la stesura del testo

In primo luogo bisogna accettare il fatto che se LaTeX sposta un oggetto flottante lo fa per motivi estetico-tipografici o perché lo spazio è fisicamente insufficiente. Per esempio LaTeX non metterà mai una figura seguita da un titolo di sezione e da un cambio pagina, ma preferirà stampare la sezione e poi la figura; oppure, se si aggiunge un oggetto flottante in fondo ad una pagina, LaTeX è obbligato a spostarlo almeno nella pagina successiva. Se lo spazio è insufficiente, è inutile cercare di forzare LaTeX a mettere l'oggetto flottante in tale posizione: se lo spazio fisico non c'è, non si può certo inventarlo.

Per fortuna, con un minimo di accortezza, LaTeX fa un ottimo lavoro. Per prima cosa è opportuno utilizzare sempre il posizionamento automatico evitando di aggiungere `\clearpage` o comandi simili: in fase di redazione chi scrive la tesi dovrebbe solo concentrarsi sui contenuti e non sull'impaginazione. In generale i posizionamenti fatti a mano interferiscono con la complessa routine di LaTeX per il posizionamento degli oggetti flottanti e portano a risultati peggiori rispetto a quelli di default. Seguendo le semplici indicazioni che seguono, il posizionamento automatico mantiene gli oggetti flottanti vicini al punto di inserimento e inoltre evita che l'utente si preoccupi continuamente del posizionamento dei float, lasciando più tempo per lavorare sui contenuti.

Una delle origini dei problemi lamentati è l'utilizzo eccessivo dell'opzione `[h]` (che chiede di posizionare la figura nel punto dove compare nel codice): gli oggetti flottanti vengono spesso inseriti con l'opzione `[htbp]` o peggio `[h!t]`. In generale si pensa che questa opzione sia la migliore per mantenere gli oggetti flottanti vicino al punto di inserimento. In realtà può funzionare bene solo quando gli oggetti inseriti sono molto piccoli (dove per piccolo si intende con un'altezza di gran lunga inferiore rispetto a quella del corpo del testo). Il modo migliore per utilizzare le opzioni di posizionamento è quello di domandarsi in primo luogo se l'oggetto flottante sarà abbastanza piccolo per stare in una pagina di testo o se avrà bisogno di una pagina tutta per sé. Nel primo caso lo si introduce quindi con un'opzione di posizionamento `[tb]`; nel secondo con `[p]`. Se non ci sono oggetti flottanti in sospeso, nel primo caso LaTeX potrà spostare l'oggetto subito prima del punto di inserzione (cosa che non può fare se si usa `[h]`) o nella pagina immediatamente successiva. Usando invece `[p]` per i grossi oggetti flottanti, questi verranno immediatamente stampati in una pagina dedicata e non verrano spostati alla fine del capitolo come succede con `[tbp]`. Basta sfogliare un qualunque testo ben impaginato per accorgersi che le figure sono introdotte proprio in questo modo: in generale all'inizio o alla fine della pagina, in una pagina intera se sono grandi, raramente nel corpo del testo se sono davvero piccole. Certi utenti sono infastiditi dal fatto che alcuni oggetti flottanti appaiano prima del testo in cui sono citati (ad esempio una figura in alto nella pagina in cui è citata): per risolvere questo problema è possibile utilizzare il pacchetto flafter, che impedisce agli oggetti flottanti di apparire prima della loro definizione nel testo.

Infine è utile ricordare che LaTeX riesce a posizionare tutte le figure in modo corretto solo se il rapporto

$$\frac{\text{testo}}{\text{figure}}$$

è sufficientemente alto. Ne consegue che è auspicabile, non solo per fini tipografici, scrivere qualche cosa di interessante piuttosto che riempire le lacune con immagini. Se tale rapporto è troppo basso, può accadere che la compilazione si interrompa e venga restituito il seguente errore:

```
! LaTeX Error: Too many unprocessed floats.
```

Questo è dovuto al fatto che LaTeX ha una certa quantità di memoria dedicata al posizionamento degli oggetti flottanti; se troppi oggetti si accumulano durante la compilazione, tale memoria può esaurirsi. Per risolvere questo problema è possibile utilizzare il pacchetto placeins. Esso definisce il comando `\FloatBarrier` che non può essere oltrepassato dagli oggetti flottanti e quindi impone il posizionamento di tutti quelli che sono ancora in memoria. Nel caso che il documento presenti dei posti dove possa essere inserita un'interruzione di pagina, conviene utilizzare `\clearpage`. Tale comando, oltre a creare un'interruzione di pagina, impone il posizionamento di tutti gli oggetti flottanti ancora in memoria in modo analogo a `\FloatBarrier`. Il pacchetto morefloats aumenta il numero di oggetti flottanti che possono essere mantenuti in memoria durante la compilazione da 18 a 36.

Se tutto ciò non fosse sufficiente, nella fase precedente la stampa, e solamente allora, è possibile intervenire manualmente come spiegato nel paragrafo seguente.

7.3.2 Cosa fare durante la revisione del testo

Nella fase che precede la stampa può essere necessario intervenire manualmente per correggere il posizionamento degli oggetti flottanti (quali ad esempio le figure e le tabelle). A questo riguardo esistono numerosi pacchetti, di cui i più utili sono costituiti da float e placeins.

Il pacchetto float permette di forzare il posizionamento dell'oggetto nel punto in cui è situato il relativo ambiente per mezzo dell'opzione `H`. A volte è utile usare questa opzione insieme al comando `\afterpage` del pacchetto afterpage.

Il pacchetto placeins permette di mettere delle barriere invalicabili per gli oggetti flottanti con il comando `\FloatBarrier`.

Il motore di composizione TEX mette a disposizione parametri che controllano gli oggetti flottanti:

`\setcounter{topnumber}{...}` massimo numero di float in posizione `t` per ogni pagina

`\def\topfraction{...}` massima frazione di pagina per i float in posizione `t` per ogni pagina

`\setcounter{bottomnumber}{...}` massimo numero di float in posizione `b` per ogni pagina

`\def\bottomfraction{...}` massima frazione di pagina per i float in posizione `b` per ogni pagina

`\setcounter{totalnumber}{...}` massimo numero di float nella stessa pagina

`\setcounter{dbltopnumber}{...}` massimo numero di float grandi nella stessa pagina

`\def\textfraction{...}` minima frazione di pagina per il testo

`\def\floatpagefraction{...}` minima frazione di pagina per i float in posizione `p`

`\def\dbltopfraction{...}` massima frazione di pagina per i float a piena pagina in composizione a due colonne in posizione `t`

`\def\dblfloatpagefraction{...}` minima frazione di pagina per i float a piena pagina in composizione a due colonne in posizione `p`

Va osservato che l'intervento manuale per forzare il posizionamento degli oggetti flottanti può portare a risultati rovinosi se non si è compreso appieno il meccanismo standard di svuotamento delle code dei float. I comandi su elencati — soprattutto l'opzione `H` per gli ambienti `figure` e `table` e il comando `\floatbarrier` — devono essere usati con estrema cautela.

8 Compilare il codice

8.1 PDF come formato di output

Fino a qualche anno fa il codice LATEX doveva essere compilato per ottenere in output un file in formato DeVice-Independent (`.dvi`); successivamente si otteneva un file in formato PDF per conversione di formato. Questo schema di lavoro non è più usato.

Oggi, con le moderne distribuzioni di TEX, la compilazione attraverso il programma `pdflatex` (`xelatex` o `lualatex`) produce direttamente un file in formato PDF (`.pdf`).

Un altro vantaggio delle distribuzioni moderne è la possibilità di effettuare la cosiddetta *ricerca diretta*[20] e la *ricerca inversa*,[21] molto utili in fase di elaborazione della tesi.

20. facendo CTRL+click sul codice all'interno dell'editor TEXworks, la finestra di visualizzazione del `.pdf` scorre fino a trovare il rispettivo output.

21. facendo CTRL+click all'interno della finestra di visualizzazione del `.pdf`, il cursore viene posizionato sul rispettivo codice all'interno dell'editor

8.2 Il formato PDF archiviabile

La norma ISO 19005-1 del 2005 stabilisce il formato di archiviazione come un formato derivato dal formato PDF mediante alcune aggiunte e modifiche al normale formato PDF, tanto che questo formato di archiviazione si chiama PDF/A.

La tesi, quindi, non potrebbe essere consegnata al momento dell'iscrizione all'esame di laurea in un formato qualsiasi, sia esso DOC, ODT, PS, RTF, o altri formati più o meno esoterici, liberi o proprietari; nemmeno il formato PDF di per sé ha il formato giusto, se manca delle altre piccole modifiche e aggiunte a cui si accennava sopra. Anche il formato PDF scelto per la tesi dovrebbe corrispondere alla versione PDF-1.4 e non dovrebbero essere accettabili né versioni precedenti né versioni successive, perché così prescrive la norma ISO. Le piccole modifiche e aggiunte possono venire inserite su di un file in formato PDF-1.4 mediante opportuni applicativi ancora non molto diffusi e, in particolare, ancora per lo più commerciali.

Il programma `pdflatex`, a partire dagli aggiornamenti del 2008, è in grado di generare direttamente file in formato PDF/A mediante l'ausilio di un file di estensione contenuto nel pacchetto `pdfx` scaricabile dagli archivi CTAN. Su qualunque piattaforma, oltre al pacchetto `pdfx`, bisogna caricare anche i file del modello di colore; questi file si possono installare direttamente nella cartella dove si sono installati tutti i file del pacchetto `pdfx`, in particolare dove si è installato `pdfx.sty`. Il pacchetto può creare sia file conformi allo standard PDF/A sia a quello PDF/X.

Per approfondire questo argomento è vivamente consigliata la consultazione della *Guida* GUIT (Gruppo Utilizzatori Italiani di TEX, 2013).

9 Pacchetti utili

9.1 La lingua italiana

9.1.1 Norme tipografiche

In italiano la maggioranza delle regole tipografiche non sono universali e vincolanti, ma dipendono piuttosto da convenzioni e abitudini o dal gusto dell'autore. Nonostante questo, è importante che l'autore della tesi conosca quali sono le principali "norme" tipografiche italiane. Cevolani (2006) ne offre una sintesi e mostra come applicare ogni regola in LATEX.

9.1.2 La sillabazione

Per attivare la sillabazione e caricare i nomi delle sezioni[22] in lingua italiana, è necessario caricare il pacchetto `babel` con l'opzione dedicata per mezzo del comando:

```
\usepackage[italian]{babel}
```

22. Ad esempio "sommario", "bibliografia", "indice", eccetera.

Ecco il tipico inizio di un sorgente per un documento in italiano con la corretta sequenza dei pacchetti da caricare:

```
\documentclass[11pt,a4paper,twoside,%
    % ... eventuali altre opzioni
    openright]{book}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[italian]{babel}
```

Il pacchetto babel definisce alcuni comandi molto utili per trattare correttamente ciascuna lingua in un documento multilingue. Supponendo di dover scrivere un documento in italiano con alcune parti in inglese, babel andrà caricato così:

```
\usepackage[english,italian]{babel}
```

Per singole parole o brevi frasi in lingua straniera è disponibile il comando `\foreignlanguage`. Eccone un esempio:

```
\foreignlanguage{english}{
   This text in in English!
}
```

Per porzioni di testo in lingua più consistenti è disponibile l'ambiente `otherlanguage` come nell'esempio seguente:

```
\begin{otherlanguage*}{english}
  This is a very long text in English. It
    should be hyphenated correctly.
\end{otherlanguage*}
```

LaTeX 2ε sillaba correttamente quasi tutte le parole italiane, tuttavia esistono casi in cui si utilizzano nomi propri oppure parole rare; in questa eventualità, se la sillabazione tentata da LaTeX 2ε non è soddisfacente, è possibile suggerirla con il comando `\hyphenation` da posizionare nel preambolo: si devono scrivere le parole sillabate tra parentesi graffe, separate da uno spazio, come nel seguente esempio:

```
\hyphenation{sil-la-ba-zio-ne sim-pa-ti-ca}
```

Il precedente comando può anche essere utilizzato quando si vuole che alcune parole *non* vengano sillabate: è sufficiente scriverle senza trattini come nel seguente esempio:

```
\hyphenation{MATLAB Mathematica}
```

Tale comando può anche essere utilizzato per forzare una sillabazione particolare; se ad esempio si vuole che la parola "melograno" sia spezzata tra "melo" e "grano" e non in altri punti, è sufficiente scrivere:

```
\hyphenation{melo-grano}
```

Se la parola in questione compare una sola volta, è possibile suggerirne la sillabazione direttamente nel testo con `\-`; ad esempio:

```
sil\-la\-ba\-zio\-ne
```

Vale la pena segnalare che con l'opzione `italian` di babel il carattere `"` è attivo[23] e serve per svolgere una serie di funzioni, tra le quali:

- introdurre una possibile cesura disabilitando le altre cesure "troppo vicine", ma consentendo la sillabazione di entrambi i monconi della parola; per esempio `dispepsia` viene divisa naturalmente in `di-spep-sia`, mentre `dis"pepsia` viene divisa in `dis-pep-sia`;
- andare a capo negli URL e nelle parole composte in cui i componenti sono separati da una barra; `input"/output` spontaneamente non sarebbe divisibile, ma con il segno `"` lo diventa, e lo diventano i singoli monconi.

A conclusione di questo paragrafo, è doveroso ricordare che gli interventi manuali sulla sillabazione dovrebbero sempre essere fatti nella fase di revisione che precede immediatamente la stampa. Spesso è preferibile riformulare una frase che dà luogo ad un errore di `overfull` piuttosto che imporre particolari sillabazioni.

9.1.3 Il rientro della prima riga

I libri italiani contemporanei generalmente non hanno il primo capoverso dopo il titoletto di sezione rientrato, tuttavia alcuni autori preferiscono avere tale rientro. Per attivare il rientro sulla prima riga di ogni sezione, sottosezione, eccetera, è necessario caricare il pacchetto indentfirst, in quanto la convenzione anglosassone (di default su LaTeX) non lo prevede. Si veda la figura 5.

9.1.4 Caratteri accentati

In LaTeX i caratteri accentati possono essere introdotti con i comandi standard `\'{e}`, `\'{e}`, eccetera, oppure direttamente da tastiera `è`, `é`, e così via, se nel preambolo si carica il pacchetto inputenc con la codifica appropriata. Si consiglia di caricare il pacchetto con l'opzione `[utf8]` (si veda il paragrafo 5.1).

9.2 Il layout

9.2.1 Le testatine e i piè di pagina

Per personalizzare testatine e piè di pagina è possibile usare il pacchetto fancyhdr. Per una tesi è probabile che si abbiano impostazioni differenti a seconda della sezione e dunque è conveniente definire alcuni comandi personalizzati che modifichino testatine e piè di pagina; un esempio per `frontmatter` e `mainmatter` potrebbe essere:

```
\newcommand{\fncyfront}{%
    \fancyhead[RO]{{\footnotesize\rightmark}}
    \fancyfoot[RO]{\thepage}
```

23. In realtà il carattere `"` *può* essere reso attivo mediante il comando `\setactivedoublequote` da dare dopo aver caricato il pacchetto babel e prima di cominciare il corpo del documento; in mancanza di questo comando, il carattere `"` non serve altro che a stampare se stesso. Si rimanda alla lettura del documento `babel-italian.pdf` per maggiori dettagli.

Problema

Il metodo appena presentato nel par. 2.5.1 presenta un serio problema di fondo. Esso è infatti valido nell'ipotesi che l'insieme di vettori $\hat{\Pi}$ costituisca una base di $\mathbb{R}^6$, ovvero che essi siano linearmente indipendenti. Tuttavia è facile dimostrare che $\det \hat{\mathbf{\Pi}} = 0$ indipendentemente dai valori assegnati alle sei coppie di elementi non nulli della matrice. Conseguenza immediata è che la matrice $\hat{\mathbf{\Pi}}$ non è invertibile e che dunque non è possibile risolvere il problema con questa via.

Conclusioni

Interessante è individuare l'origine della lineare dipendenza dei vettori di $\hat{\Pi}$. Come già osservato, l'insieme di vettori Π è un buon candidato a divenire base di $\mathbb{R}^6$, purché nella scelta dei 12 elementi non nulli si rispetti la condizione (2.4).

Tale condizione viene dunque a mancare nel momento in cui si impongono i vincoli (2.5); in altre parole, il metodo viene a fallire quando si impone che le condizioni di carico di base siano una ad una equilibrate. D'altronde il vincolo di condizioni di carico di base equilibrate è imprescindibile dato che altrimenti risulta impossibile risolvere con metodi numerici agli EF il sottomodello. È dunque necessario trovare un'altra strada per risolvere il problema della scomposizione della condizione di carico generica applicata al giunto Φ.

(a) Senza pacchetto indentfirst.

Problema

Il metodo appena presentato nel par. 2.5.1 presenta un serio problema di fondo. Esso è infatti valido nell'ipotesi che l'insieme di vettori $\hat{\Pi}$ costituisca una base di $\mathbb{R}^6$, ovvero che essi siano linearmente indipendenti. Tuttavia è facile dimostrare che $\det \hat{\mathbf{\Pi}} = 0$ indipendentemente dai valori assegnati alle sei coppie di elementi non nulli della matrice. Conseguenza immediata è che la matrice $\hat{\mathbf{\Pi}}$ non è invertibile e che dunque non è possibile risolvere il problema con questa via.

Conclusioni

Interessante è individuare l'origine della lineare dipendenza dei vettori di $\hat{\Pi}$. Come già osservato, l'insieme di vettori Π è un buon candidato a divenire base di $\mathbb{R}^6$, purché nella scelta dei 12 elementi non nulli si rispetti la condizione (2.4).

Tale condizione viene dunque a mancare nel momento in cui si impongono i vincoli (2.5); in altre parole, il metodo viene a fallire quando si impone che le condizioni di carico di base siano una ad una equilibrate. D'altronde il vincolo di condizioni di carico di base equilibrate è imprescindibile dato che altrimenti risulta impossibile risolvere con metodi numerici agli EF il sottomodello. È dunque necessario trovare un'altra strada per risolvere il problema della scomposizione della condizione di carico generica applicata al giunto Φ.

(b) Con pacchetto indentfirst.

Figura 5: Rientro sulla prima riga.

```
    \fancyhead[LE]{\footnotesize{\leftmark}}
    \fancyfoot[LE]{\thepage}
    \fancyhead[RE,LO]{}
    \fancyfoot[C]{}
    \renewcommand{\headrulewidth}{0.3pt}}
\newcommand{\fncymain}{%
    \fancyhead[RO]{{\footnotesize\rightmark}}
    \fancyfoot[RO]{\thepage}
    \fancyhead[LE]{{\footnotesize\leftmark}}
    \fancyfoot[LE]{\thepage}
    \fancyfoot[C]{}
    \renewcommand{\headrulewidth}{0.3pt}}
```

da utilizzare nel seguente modo:

```
\pagestyle{fancy}
\fncyfront
\frontmatter
...
\fncymain
\mainmatter
```

La definizione di tali comandi è differente a seconda che il testo sia fronte-retro (`twoside`) o solo fronte (`oneside`).

Utilizzando l'opzione `openright` può capitare di ottenere una pagina bianca alla fine di un capitolo; per evitare che in questa pagina siano presenti testatine o piè di pagina, è sufficiente includere nel preambolo il pacchetto emptypage.

In alternativa a fancyhdr è possibile usare titlesec. Tale pacchetto ha un'interfaccia d'uso leggermente diversa da fancyhdr e permette di definire stili diversi da applicarsi in diverse parti del documento.

9.2.2 Il layout della pagina

Molto di frequente i regolamenti degli atenei richiedono un layout della pagina differente da quello prodotto di default dalle classi di LaTeX ed è dunque necessario modificarlo. Il primo modo per intervenire è l'utilizzo di comandi interni del LaTeX, quali `\textwidth`, `\oddsidemargin`, eccetera; tuttavia questa strada è sconsigliabile per molte ragioni.

Una migliore soluzione è costituita dal pacchetto geometry (Umeki, 2010) che è completamente configurabile. Nel caso in cui siano necessari degli interventi locali a pagine e/o paragrafi è possibile utilizzare il pacchetto changepage.

Per rilegare la tesi può essere conveniente indicare sulle pagine dove tagliare il foglio; questo può essere agevolmente realizzato utilizzando in coppia i pacchetti geometry e crop. Si veda ad esempio la figura 6.

Di default LaTeX cerca di coprire interamente con il testo o altri elementi l'intera altezza della pagina e, ove necessario, inserisce degli spazi aggiuntivi tra i capoversi oppure dilata gli spazi tra le voci degli elenchi puntati, e così via. Se si vuole disattivare questa impostazione e avere dello spazio bianco a piè di pagina quando non si riesce a coprirla tutta, è sufficiente aggiungere nel preambolo il comando `\raggedbottom`. Il comportamento di default è invece dovuto al comando `\flushbottom`. Per migliorare la copertura delle pagine è possibile permettere che siano spezzate le formule matematiche in *display* aggiungendo al preambolo il comando `\allowdisplaybreaks` (che funziona solo col pacchetto amsmath).

È conveniente non modificare il comportamento di default di LaTeX fino a quando non si arriva alla versione definitiva del testo (che precede immediatamente la stampa). Solo in questa fase è possibile intervenire modificando il posizionamento degli oggetti flottanti (vedi il paragrafo 7.3), oppure intervenendo con i comandi appena citati. Prima di modificare le impostazioni di LaTeX è conveniente provare ad effettuare piccole modifiche al testo che spesso sono sufficienti per risolvere eventuali problemi e permettono di ottenere layout più eleganti.

Per approfondimenti sui layout di pagina si consiglia la guida *Introduzione alla definizione della geometria della pagina* (Beccari, 2012).

9.2.3 L'interlinea

Spesso le prescrizioni redazionali impongono un'interlinea diversa da 1. (valore di default in LaTeX). Per modificare l'interlinea del documento esistono più strade, tuttavia la più indicata consiste nel caricare il pacchetto setspace. Tale pacchetto fornisce tre interlinee predefinite richiamate con i comandi `\singlespacing` (interlinea singola), `\onehalfspacing` (interlinea 1,5) e `\doublespacing` (interlinea doppia). Se è necessaria un'interlinea differente, è sufficiente utilizzare il comando `\linespread{...}` mettendo tra parentesi graffe il numero che rappresenta il fattore di scala per l'avanzamento di riga.

9.3 Lo stile

9.3.1 I fonts

In primo luogo, lavorando con `pdflatex`, è consigliabile utilizzare l'encoding T1 che rappresenta lo standard di codifica dei caratteri di LaTeX. Tale codifica è attivata nel preambolo per mezzo del comando:

```
\usepackage[T1]{fontenc}
```

Se la tesi è di tipo scientifico, è conveniente abilitare i font matematici forniti dall'$\mathcal{AMS}$ (American Mathematical Society) con il comando:

```
\usepackage{amssymb}
```

Il pacchetto amssymb carica il pacchetto amsfonts e definisce i comandi per usare i simboli introdotti da quest'ultimo.

Per la matematica conviene in generale aggiungere il comando:

```
\usepackage{mathtools}
```

che carica anche il pacchetto amsmath, fornendo svariate estensioni per il miglioramento della strut-

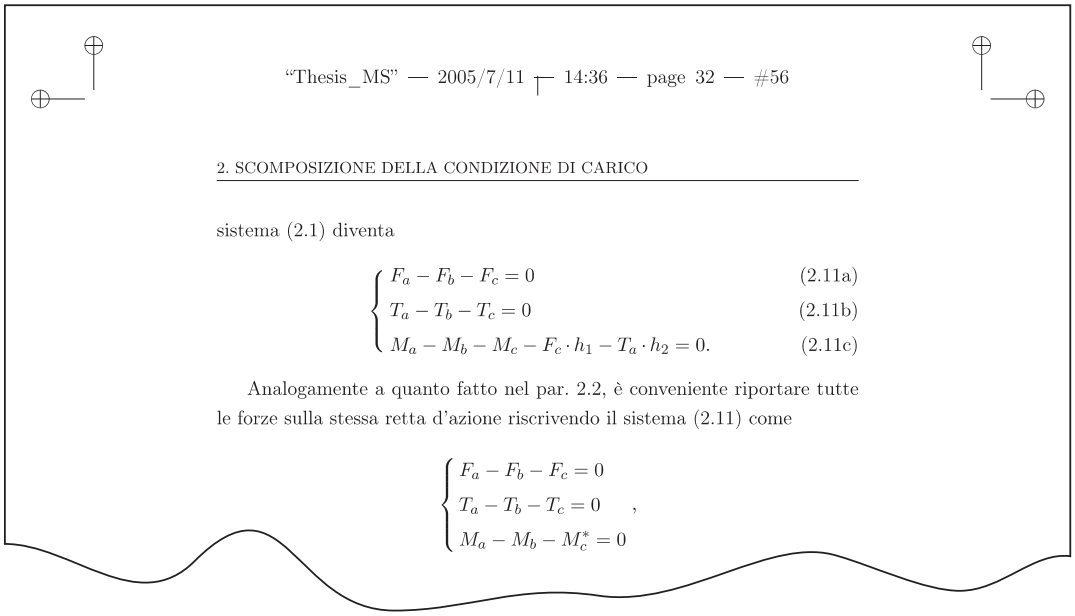


FIGURA 6: Esempio di segni per il taglio del foglio.

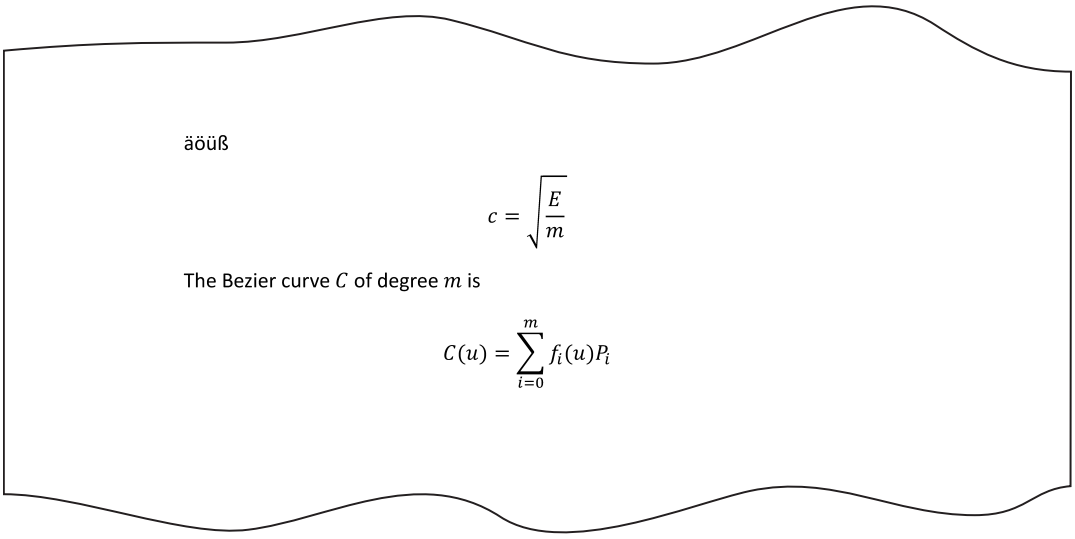


FIGURA 7: Esempio d'uso dei font Calibri per il testo e Cambria Math per la matematica.

tura informativa e della stampa di documenti che contengono formule matematiche.

Per modificare la dimensione del font, in aggiunta ai comandi standard,[24] è utile il pacchetto relsize che consente di assegnare dimensioni relative con i comandi `\smaller` e `\larger`.

Riguardo al tipo di font da utilizzare, l'esperienza conferma che, quasi certamente, la scelta migliore è quella di usare i font che LaTeX carica di default, ovvero la famiglia Computer Modern sviluppata dallo stesso inventore del TeX, Donald Knuth. Questi caratteri possono essere usati nella variante Latin Modern prodotta dal GUST (gruppo utenti di TeX polacco)[25] caricando il pacchetto lmodern:

```
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

Se si vuole a tutti i costi cambiare font, è bene ricordare che è necessario scegliere tre famiglie (Serif, Sans-serif, Typewriter e i font per la matematica) che formino una buona combinazione. A tale proposito, è importante ricordare che i font, tranne alcune eccezioni,[26] non hanno tutti i simboli necessari per la matematica e quindi non possono essere usati se non nel testo.

Per cambiare font è possibile caricare uno dei numerosi pacchetti dedicati. Un elenco dei font dispo-

24. `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge` e `Huge`.

25. `http://www.gust.org.pl/projects/e-foundry/latin-modern`

26. Il gruppo utenti TeX danese ospita una pagina in cui sono riportati tutti i font che supportano la matematica: `http://www.tug.dk/FontCatalogue/mathfonts.html`.

nibili e i pacchetti da caricare per usarli è riportato sul sito del gruppo di utenti TEX danese.[27]

Un'ottima alternativa ai caratteri di default è offerta dai pacchetti newtxtext e newtxmath. Con essi si caricano: il font Times (o un suo clone) per il testo, font appositamente disegnati per la matematica basati su Times Italic e altri font PostScript, un clone del font Helvetica per la famiglia sans serif, vari possibili font per la famiglia typewriter. Considerazioni analoghe si possono fare con i pacchetti newpxtext e newpxmath per usare il font Palatino (o un suo clone) per il testo accoppiato a font matematici appositamente disegnati; un clone del font Helvetica per la famiglia sans serif e vari possibili font per la famiglia typewriter.

Parlando qui di font, non si può fare a meno di menzionare un'importante alternativa al programma pdflatex, cioè xelatex o anche XeLaTeX. La caratteristica principale di XeLaTeX è che può adoperare senza bisogno di installazioni particolari tutti i font noti al sistema operativo in uso, che siano in formato OpenType o TrueType. Questi font sono dotati di tabelle interne con cui XeLaTeX è capace di creare al volo la struttura dati che nel TEX tradizionale risiede in specifici file metrici (.tfm). Altra importante caratteristica di XeLaTeX è che lavora direttamente con file in codifica Unicode, cioè UTF-8 oppure UTF-16. Un esempio di sorgente con caratteri speciali è il seguente:

```
\documentclass{article}

\usepackage{fontspec}
\usepackage{unicode-math}
\setmainfont{Calibri}
\setmathfont{Cambria Math}

\begin{document}
äöüß
\[
c = \sqrt{\frac{E}{m}}
\]
The Bezier curve $C$ of degree $m$ is
\[
C(u) = \sum_{i=0}^m f_i(u)P_i
\]
\end{document}
```

Questo sorgente produce un output come quello della figura 7, in cui si nota l'uso del font Calibri per il testo e del font Cambria Math per la matematica.

Per una introduzione all'uso di XeLaTeX si rimanda alla guida di Gregorio (2011) e ai riferimenti in essa contenuti.

9.3.2 Il titolo dei capitoli

Per personalizzare il formato dei titoli dei capitoli è possibile utilizzare il pacchetto fncychap; si rimanda a Mori (2007) per maggiori dettagli.

Nella figura 8 è mostrato un esempio di personalizzazione attraverso il pacchetto titlesec. Una volta definito lo stile nel preambolo con il comando \titleformat, la formattazione del titolo del capitolo è ottenuta semplicemente nel documento attraverso il comando standard della classe book:

```
\chapter{Definizioni di base e notazioni}
```

9.3.3 Liste

Per personalizzare i tre ambienti standard dedicati alle liste, cioè enumerate, itemize e description, si consiglia il pacchetto enumitem.

9.3.4 I "mini indici"

Quando i capitoli hanno una struttura particolarmente complessa, può essere conveniente riportare nella pagina iniziale l'indice del capitolo (vedi ad esempio la figura 9). Questi "mini indici" possono essere prodotti automaticamente con il pacchetto minitoc.

9.3.5 Le epigrafi

Talvolta si vogliono inserire epigrafi nella pagina iniziale dei capitoli. Per farlo è possibile utilizzare il pacchetto epigraph; un esempio è riportato nella figura 10.

9.3.6 Le note

LaTeX produce di default un layout delle note di alta qualità; tuttavia, qualora lo si ritenga strettamente necessario, esistono alcuni accorgimenti per modificarlo. Il pacchetto footmisc fornisce molti controlli sulle note tra cui la possibilità di forzare le note al fondo della pagina[28] con l'opzione bottom; si veda la figura 11.

Per impedire che le note vengano spezzate su più pagine è sufficiente assegnare al parametro di penalità un valore molto elevato, ad esempio:

```
\interfootnotelinepenalty=10000
```

mentre per controllare la dimensione della zona assegnata alle note a piè di pagina si può usare il comando:

```
\dimen\footins=2cm
```

9.4 La matematica

Si consiglia il lettore di consultare l'*Arte* di Pantieri e Gordini, la guida del Gruppo Utilizzatori Italiani di TEX (2013) e la guida di Voss (2010) per approfondimenti sulla scrittura matematica (semplice e avanzata) in LaTeX. Qui di seguito si accenna ad alcuni aspetti interessanti per chi scrive una tesi di laurea.

9.4.1 I simboli "speciali"

Intendendo con "simboli speciali" tutti quelli che non sono inseribili direttamente dalla tastiera, è necessario distinguere tra quelli matematici e quelli

27. http://www.tug.dk/FontCatalogue

28. Normalmente LaTeX unisce le note con l'ultima riga della pagina e dunque su pagine non piene non si hanno le note a fondo pagina.

```
\usepackage[calcwidth,pagestyles]{titlesec}% loads titleps
\usepackage{adjustbox,xcolor}

% chapter head style via titlesec
\titleformat{\chapter}[display]
    {\bfseries\Large}
    {\color{blue!65!black}\filleft%
        \minsizebox{!}{24pt}{\chaptertitlename}% needs package adjustbox
        \lapbox[0pt]{\width}{%
            \minsizebox{!}{40pt}{%
                \ \colorbox{blue!65!black}{\color{white}\thechapter}% needs xcolor
            }%
        }% needs package adjustbox
    }
    {4ex}
    {{\color{blue!65!black}\titlerule}
        \huge\bfseries\scshape
        \vspace{2ex}%
        \filright}
    [\vspace{2ex}%
        {\color{blue!65!black}\titlerule}]
```

Capitolo 1

DEFINIZIONI DI BASE E NOTAZIONI

Jesce sole, jesce sole, nun ce fa' cchiù suspirà!
– Gatta Cenerentola

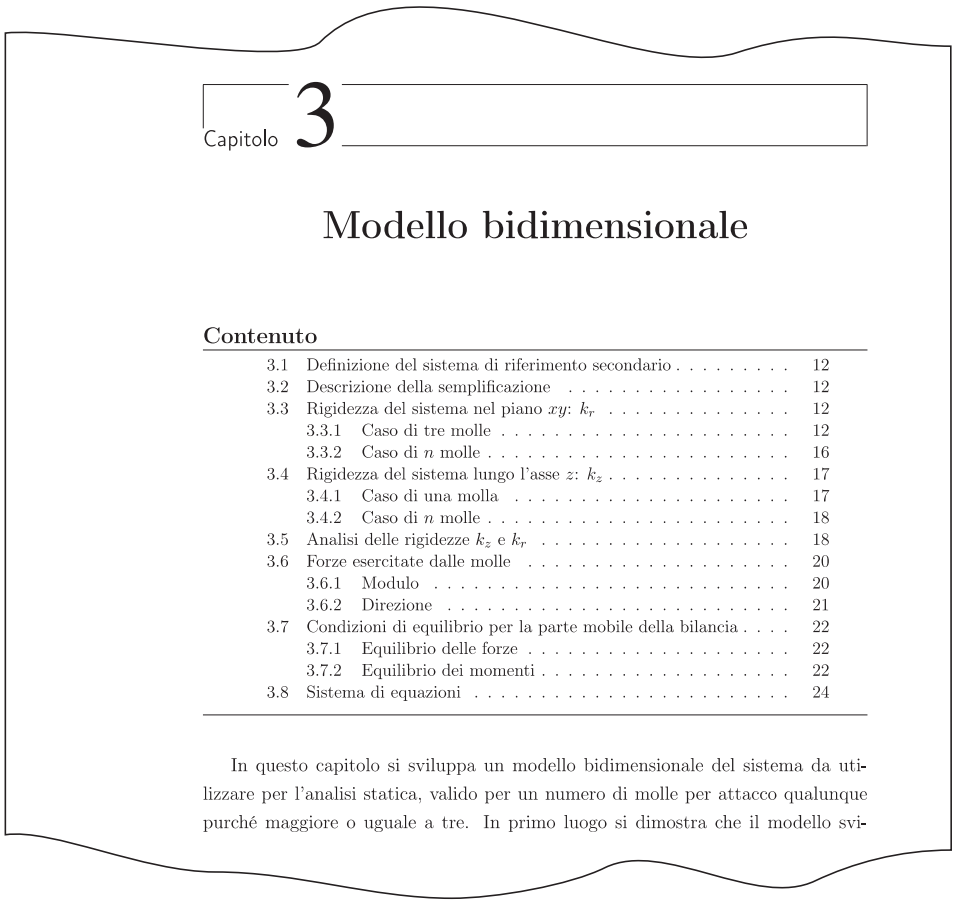
1.1 Introduzione

Lo studio della Meccanica del volo, come altre materie ingegneristiche, poggia le sue basi sui noti concetti della Fisica matematica. Esso richiede di familiarizzare con un certo numero di definizioni, con precise convenzioni sul segno di determinate grandezze e con il sistema di notazione che da esse scaturisce. Più avanti si vedrà che una peculiarità del sistema di notazione della Meccanica del volo, e in particolare dell'Aerodinamica degli aeromobili, è quella di fare largo uso di simboli con pedici multipli.

Scopo di questo capitolo è quello di richiamare i principali elementi di base della materia, a partire dalla definizione dei sistemi di riferimento essenziali e dell'orientamento dei velivoli nello spazio, per passare poi dall'anatomia dei velivoli tradizionali con una panoramica sulle azioni esterne agenti sugli aeromobili in volo. Sarà presentato al tempo stesso il sistema di notazioni adottato nel testo illustrando le motivazioni per cui si scelgono determinati simboli, pedici, eccetera.

Come in tutte le materie ingegneristiche, per le quantità che verranno via via introdotte si utilizzeranno sistemi di unità di misura diversi a seconda del contesto e dell'argomento. Al giorno d'oggi è necessario esprimere le grandezze nel Sistema Internazionale di unità di misura (SI, *International System of Units*). In Italia ne è stato reso obbligatorio l'uso nel 1976 in tutti gli atti pubblici. In Inghilterra e negli USA non vi è alcun obbligo a non utilizzare i sistemi tradizionali di misura basati sulle *Imperial units* e sulle *United States customary units* (o *English Units*). Pertanto, per ragioni storiche, oltre che pratiche, in aeronautica si utilizzano indifferentemente le unità di questi diversi sistemi. Così verrà fatto anche qui.

FIGURA 8: Esempio di definizione del titolo dei capitoli con titlesec. Il comando standard `\chapter` della classe book produce un titolo nel formato personalizzato.



Capitolo 3

Modello bidimensionale

Contenuto

In questo capitolo si sviluppa un modello bidimensionale del sistema da utilizzare per l'analisi statica, valido per un numero di molle per attacco qualunque purché maggiore o uguale a tre. In primo luogo si dimostra che il modello svi-

Figura 9: Esempio di "mini indice".

non matematici: per i primi dovrebbe essere sufficiente caricare i simboli dell'$\mathcal{AMS}$ con il pacchetto amssymb; per tutti gli altri simboli sono necessari pacchetti appositi che possono essere facilmente identificati consultando la preziosa guida *The comprehensive LaTeX symbol list* (Pakin, 2009).

9.4.2 Rappresentazione dei numeri

Un pacchetto molto utile per la rappresentazione di numeri è numprint. Tra le funzioni di tale pacchetto si ricordano l'inserimento di un separatore ogni tre cifre per le migliaia e l'approssimazione automatica. Ad esempio:

```
\numprint{2.742647826672E-01}
```

produce

$$2{,}743 \cdot 10^{-01}$$

9.4.3 Unità di misura

Le unità di misura del Sistema Internazionale possono essere inserite con i comandi del pacchetto siunitx (se ne veda la documentazione), che permette di regolarne molto finemente il formato e di cambiare il risultato nel documento finito operando un'unica modifica nel preambolo anziché agire a mano su ciascuna unità di misura. Dato che le convenzioni tipografiche italiane prevedono la virgola e non il punto (predefinito dal pacchetto) come separatore decimale, il pacchetto va caricato almeno con l'opzione seguente:

```
\usepackage[output-decimal-marker={,}
  % ... altre opzioni
  ]{siunitx}
```

I comandi fondamentali sono `\num` (simile a `numprint`), `\SI` e `\si` che permettono di formattare i numeri, le grandezze fisiche e le unità di misura in maniera configurabile.

Il pacchetto dispone di un modulo di elaborazione dei numeri che consente di avere nei sorgenti dei valori numerici del tipo `30e3` che le macro di scrittura trasformano in 30×10^3. Questo permette di trarre i valori numerici da file scritti dagli stessi strumenti di misura moderni, dove i numeri sono espressi con la notazione informatica dei numeri a virgola mobile.

Si può specificare il numero di cifre da scrivere nei valori numerici delle misure: il modulo di elaborazione dei numeri provvede ad arrotondare quel valore numerico al numero richiesto di decimali. Infatti, se si specifica che si vogliono consistentemente 4 decimali e la virgola decimale, il numero `1.234567`, con il comando:

```
nel preambolo
\sisetup{
  round-mode      = places,
```

Capitolo 3

Implementazione su codici agli elementi finiti

Everything should be made as simple as possible, but not simpler.

Albert Einstein (1879–1955)
German physicist

In questo capitolo si utilizza la strategia di ridurre le condizioni di carico generiche agenti sul giunto ad una combinazione lineare di soluzioni note a priori (risultato del cap. 2), al fine di sviluppare un algoritmo che superi

Figura 10: Esempio di epigrafe.

```
  round-precision = 4
}

nel testo
\num{1.234567}
```

viene stampato nella forma:

1,2346

dove il numero di decimali è quello voluto ma l'ultima cifra tiene conto dell'arrotondamento in alto, visto che la parte scartata è maggiore della metà dell'unità corrispondente all'ultima cifra scritta; inoltre, come richiesto, il punto decimale è consistentemente cambiato nella virgola.

Il seguente frammento di codice:

```
\SI{23.4}{kg.m.s^{-2}} \\
$r=\SI{0,8768(11)e-15}{m}$ \\
\si{\joule\per\mole\per\kelvin}\\
\si{J.mol^{-1}.K^{-1}}\\
\SI{100}{\celsius} \\
\ang{1;2;3}
```

produce la di scrittura di grandezze fisiche nella forma:

$23{,}4\,\mathrm{kg\,m\,s^{-2}}$
$r = 0{,}8768(11) \cdot 10^{-15}\,\mathrm{m}$
$\mathrm{J\,mol^{-1}\,K^{-1}}$
$\mathrm{J\,mol^{-1}\,K^{-1}}$
$100\,^{\circ}\mathrm{C}$
$1^{\circ}2'3''$

9.4.4 Altri pacchetti

Per evidenziare gli ambienti matematici può essere utilizzato il pacchetto empheq. Il seguente risultato:

$$\boxed{\begin{aligned} f(x) &= ax + b && (1)\\ E &= mc^2 + \int_0^T f(t)\,\mathrm{d}t && (2)\end{aligned}}$$

è prodotto con il codice:

```
nel preambolo
\usepackage{empheq}
\newcommand*{\diff}{\mathop{}\!\mathrm{d}}

nel testo
\begin{empheq}[box=\fbox]{align}
f(x) & = a x + b \\
E & = mc^2 + \int_0^T f(t)\, \diff{t}
\end{empheq}
```

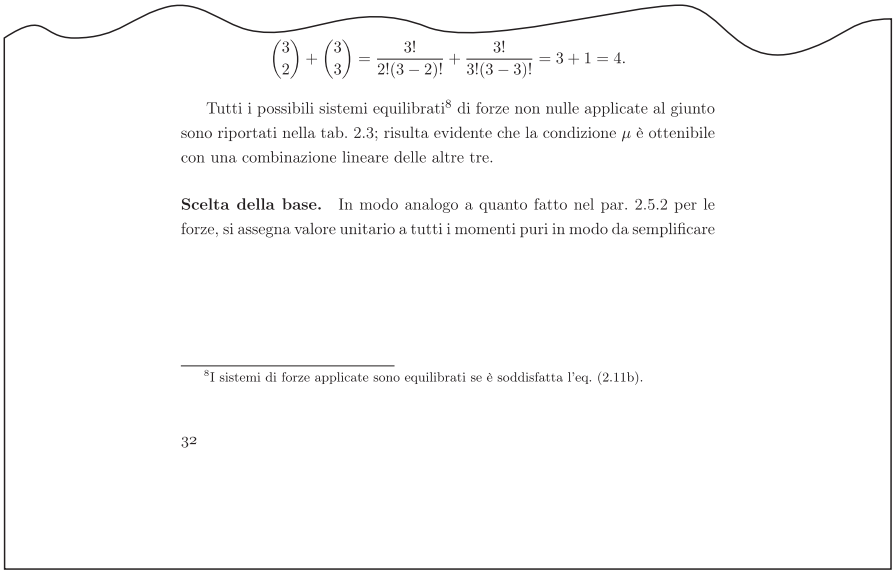
Si noti la definizione del comando `\diff` per il simbolo di differenziale: in ambiente matematico `\diff{t}` permette di ottenere 'dt' come richiesto dalle norme ISO 80000-2:2009 (2009).

Per la personalizzazione degli ambienti "tipo teorema" è necessario il pacchetto ntheorem. Il pacchetto xfrac permette invece di scrivere correttamente le frazioni nel testo e nel testo matematico (ad esempio: $\sfrac{5}{7}$).

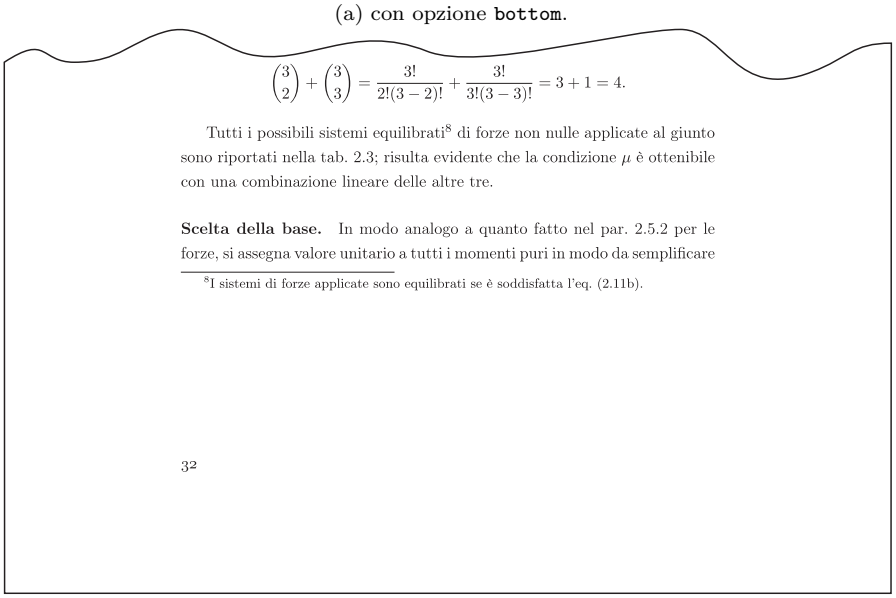
9.5 Codici e algoritmi

Il pacchetto listings è un potente strumento con il quale si gestisce la scrittura di codici in numerosi linguaggi di programmazione, controllandone molto finemente il formato.

Per la formattazione di algoritmi sono invece consigliabili i pacchetti algorithm e algpseudocode:



(a) con opzione `bottom`.



(b) senza opzione `bottom`.

FIGURA 11: Posizione delle note.

il primo genera degli oggetti flottanti, mentre il secondo no.

9.6 Riferimenti incrociati

In molti casi è comodo usare contemporaneamente i comandi `\ref` e `\pageref` per riferirsi a figure e tabelle, specialmente quando ci sono più pagine tra il riferimento e l'oggetto. Per questo alcuni utenti utilizzano comandi come:

```
\newcommand{\fullref}[1]{%
    \ref{#1} a pagina~\pageref{#1}}
```

che semplificano la scrittura del riferimento. Tuttavia, non sapendo a priori dove sia posizionato l'oggetto a cui ci si riferisce, utilizzando un comando del genere può capitare che il `\pageref` punti alla pagina stessa dove si trova il riferimento e produca così un risultato insoddisfacente.

Per rendere automatica la scrittura dei riferimenti completi è possibile utilizzare il pacchetto varioref che introduce il comando `\vref` da usarsi nello stesso modo del comune `\ref`. Tale pacchetto funziona in parallelo a babel e quindi si adatta alla lingua utilizzata nel testo. Ad esempio:

```
si veda la figura~\vref{fig:Mia:Figura}
```

produce, a seconda di dove viene posizionata la figura, qualcosa del tipo:

si veda la figura 3.1 nella pagina successiva

oppure

si veda la figura 3.1 a pagina 24

Per quanto riguarda invece il riferimento ad equazioni, è consigliabile utilizzare il comando `\eqref{...}` di amsmath al posto di `(\ref{...})`. Ad esempio:

```
... grazie all'equazione~\eqref{e2}
```

produce qualcosa del tipo:

... grazie all'equazione (3.6)

Un pacchetto alternativo e per certi aspetti più potente di varioref è il pacchetto cleveref. Si rimanda il lettore al manuale d'uso per approfondimenti.

9.7 Revisione del codice

In fase di revisione del codice è molto utile, oltre ad un'attenta lettura del file di *log* dei messaggi (`.log`), l'utilizzo dei pacchetti refcheck e showkeys che controllano l'utilizzo dei `\label` e dei `\ref`. In aggiunta a questi è anche conveniente abilitare l'opzione `draft` per la `documentclass`: in questo modo i punti in cui il testo sborda dai margini verranno evidenziati con delle barre nere.

10 Siti utili

In aggiunta alle guide e ai manuali citati nella bibliografia, sono disponibili sul Web una serie di risorse utili per risolvere i problemi incontrati durante l'utilizzo di LaTeX.

Il riferimento primario per la comunità italiana di utenti LaTeX è il sito del Gruppo Utilizzatori Italiani di TeX (GuIt),[29] che ospita un forum sull'argomento.[30] e ha anche una sezione documentazione ben organizzata.

Altro sito di interesse è quello del CTAN, che ospita gran parte del materiale su LaTeX disponibile in rete ed è dotato di un motore di ricerca.

Sarovar[31] è un catalogo molto completo di pacchetti e programmi legati a TeX e LaTeX. Permette svariati tipi di ricerca, in particolare è estremamente utile la lista "topical" quando non si conosce il nome di un pacchetto ma solo "quello che deve fare".

Altri due importanti riferimenti sono il sito di domande e risposte su StackExchange[32] e il sito LaTeX Community.[33]

11 Strumenti per la compilazione *in the cloud*

Nel panorama degli strumenti per LaTeX si è recentemente affermata una nuova possibilità di lavoro. Diversamente dal metodo tradizionale in cui si produce un documento tramite un sistema TeX installato sul proprio computer, oggi esistono siti internet che permettono la compilazione online di documenti LaTeX residenti in una *cloud* dedicata.

29. http://www.guitex.org
30. http://www.guitex.org/home/it/forum/index
31. http://texcatalogue.sarovar.org
32. http://tex.stackexchange.com
33. http://www.latex-community.org

Alcuni chiamano queste applicazioni web col nome di strumenti di compilazione *in the cloud*.[34] Essi sono inoltre adatti alla stesura 'collaborativa' di documenti, cioè quelli per i quali le modifiche online sono consentite a più utenti; la gestione incrementale delle modifiche è una delle funzionalità offerte.

Alcuni esempi sono (figura 12):

- ShareLaTeX: https://www.sharelatex.com
- writeLaTeX: https://www.writelatex.com
- verbosus: https://verbosus.com
- Authorea: https://www.authorea.com

Il servizio offerto da questi siti per la gestione dei progetti LaTeX presenta diversi profili possibili. Di solito vi è sempre la possibilità di registrarsi gratuitamente avendo a disposizione un limitato spazio di memorizzazione e un limitato numero di collaboratori.

È utile visitare le sezioni di questi siti che offrono gratuitamente numerosi modelli di tesi di laurea.[35][36]

Riferimenti bibliografici

Beccari, C. (2012). *Introduzione alla definizione della geometria della pagina*. http://www.guitex.org/home/images/doc/GuideGuIT/intropagedesign.pdf.

— (2013). *La classe TOPtesi. Per comporre la tesi al Poli e in molte altre università*. http://texdoc.net/texmf-dist/doc/latex/toptesi/toptesi-doc-xetex.pdf.

Beccari, C. e Gordini, T. (2012). *Codifiche in TeX e LaTeX. Dal sorgente al PDF, guida pratica per lavorare con successo*. http://www.guitex.org/home/images/doc/GuideGuIT/introcodifiche.pdf.

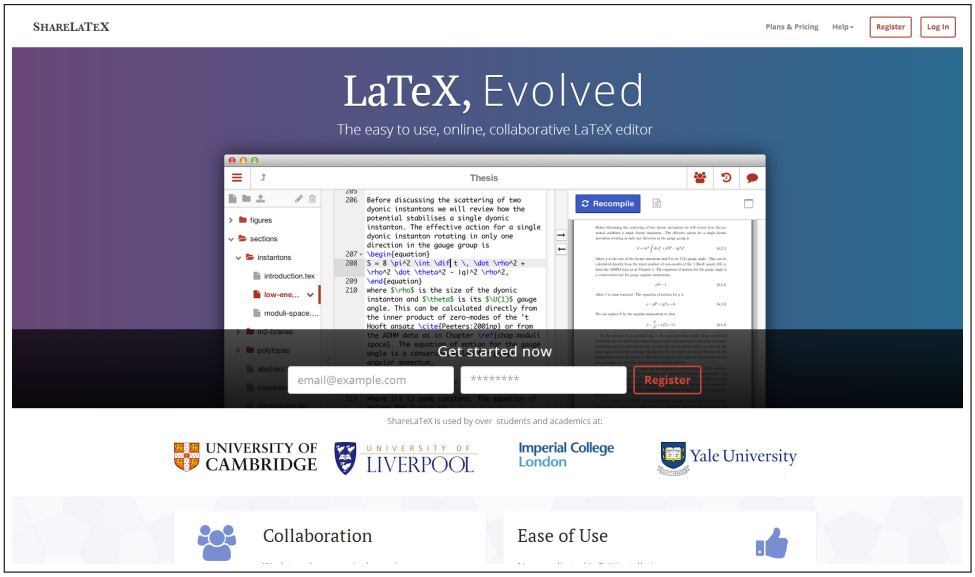
Beccari, C., Canavero, F., Rossetti, U. e Valabrega, P. (2011). *Saper Comunicare. Cenni di scrittura tecnico-scientifica*. Politecnico di Torino, Torino, Italy. URL https://didattica.polito.it/tesi/SaperComunicare.pdf. Ver. 1.11.

Biccari, F. (2012). *Documentation of the LaTeX class sapthesis.cls*. http://texdoc.net/texmf-dist/doc/latex/sapthesis/sapthesis-doc.pdf.

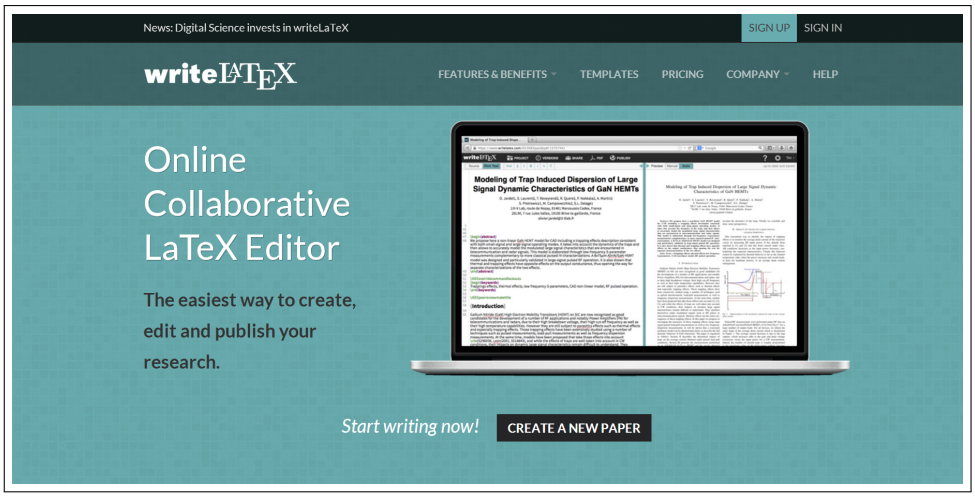
Cevolani, G. (2006). «Norme tipografiche per l'italiano in LaTeX». *ArsTEXnica*, **1** (1).

34. Si veda anche http://en.wikibooks.org/wiki/LaTeX/Collaborative_Writing_of_LaTeX_Documents
35. https://www.sharelatex.com/templates/thesis
36. https://www.writelatex.com/templates

https://www.sharelatex.com



https://www.writelatex.com



https://verbosus.com

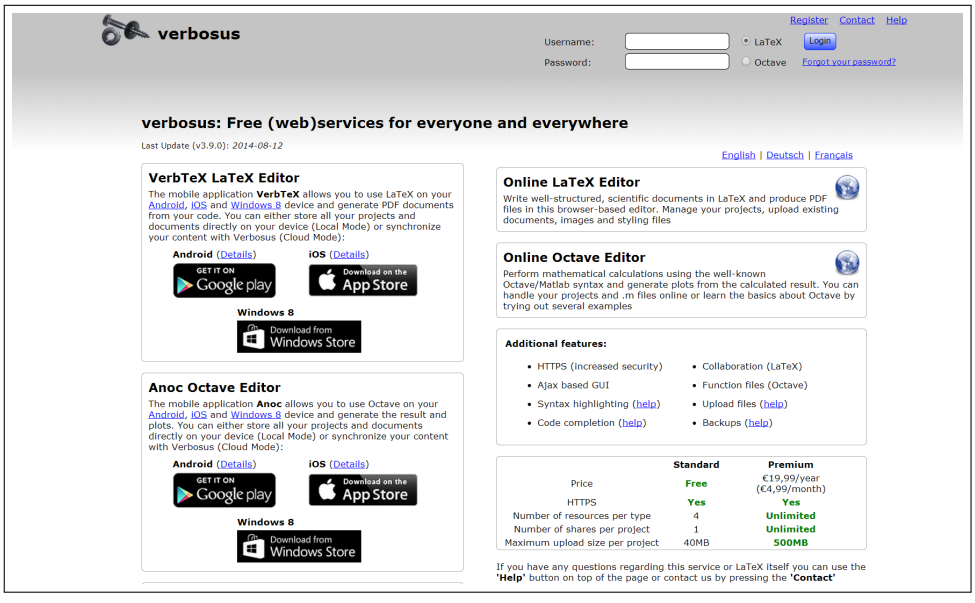


FIGURA 12: Pagine di benvenuto di alcuni siti web per la compilazione di documenti LaTeX *in the cloud*.

De Marco, A. (2013). «Scrivere la tesi di laurea in LaTeX 2ε». *ArsTEXnica*, **1** (16).

Eco, U. (1977). *Come si fa una tesi di laurea*. Bompiani, Milano.

Gregorio, E. (2011). *Introduzione a XeLATEX*. http://profs.sci.univr.it/~gregorio/introxelatex.pdf.

— (2013). *Il pacchetto* `frontespizio`. http://texdoc.net/texmf-dist/doc/latex/frontespizio/frontespizio.pdf.

Gruppo Utilizzatori Italiani di TEX (2013). *Introduzione all'arte della composizione tipografica con LATEX*. http://www.guitex.org/home/images/doc/GuidaGuIT-B5.pdf.

ISO 80000-2:2009 (2009). *Quantities and units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology*. International Organization for Standardization.

Kohm, M. e Morawski, J.-U. (2012). *The* `KOMA-Script` *guide*. http://texdoc.net/texmf-dist/doc/latex/koma-script/scrguien.pdf.

Lehman, P. (2013). *The* `biblatex` *package*. http://texdoc.net/texmf-dist/doc/latex/biblatex/biblatex.pdf.

Lesina, R. (2013). *Il nuovo manuale di stile. Edizione 2.0. Guida alla redazione di documenti, relazioni, articoli, manuali, tesi di laurea*. Zanichelli, Bologna. Ristampa della II edizione 1994.

Matricciani, E. (2000). *La tesi scientifica. Guida alla comunicazione in Ingegneria e nelle Scienze*. Paravia Scriptorium, Torino.

— (2003). *Fondamenti di comunicazione tecnico-scientifica*. Apogeo, Milano.

— (2007). *La comunicazione tecnico-scientifica*. Aracne editrice, Milano.

Miede, A. (2013). *A classic thesis style*. http://texdoc.net/texmf-dist/doc/latex/classicthesis/ClassicThesis.pdf.

Mori, L. (2007). «Scrivere la tesi di laurea con LaTeX 2ε». *ArsTEXnica*, **1** (3).

Mori, L. F. (2006). «Tabelle su LaTeX 2ε: pacchetti e metodi da utilizzare». *ArsTEXnica*, **1** (2), pp. 31–47.

Pakin, S. (2009). *The comprehensive LATEX symbol list*. In http://texdoc.net/texmf-dist/doc/latex/comprehensive/symbols-a4.pdf.

Pantieri, L. e Gordini, T. (2011). *L'arte di scrivere con LATEX*. URL http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.

Umeki, H. (2010). *The* `geometry` *package*. http://texdoc.net/texmf-dist/doc/latex/geometry/geometry.pdf.

UNI-ISO 5966 (1989). *Presentazione dei rapporti scientifici e tecnici*. Ente Italiano di Unificazione, Milano.

Valbusa, I. (2013). *User's guide to* `suftesi`. *A document class for typesetting theses, books and articles*. http://texdoc.net/texmf-dist/doc/latex/suftesi/suftesi.pdf.

Voss, H. (2010). «Math mode». `Mathmode.pdf` in http://texdoc.net/texmf-dist/doc/latex/mathmode/Mathmode.pdf.

— (2013). *Package* `hvfloat`. *Rotating objects and captions*. http://texdoc.net/texmf-dist/doc/latex/hvfloat/hvfloat.pdf.

Wilson, P. (2010). *The* `memoir` *class for configurable typesetting*. http://texdoc.net/texmf-dist/doc/latex/memoir/memman.pdf.

▷ Agostino De Marco
Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Industriale
`agostino dot demarco at unina dot it`

Scrivere report statistici con R e Sweave

Michele Scandola, Nadia Baltieri

Sommario

In questo articolo si cercherà di fornire alcune informazioni introduttive sull'uso di Sweave, uno strumento che combina codice LaTeX con codice R, un linguaggio di programmazione per il calcolo statistico che negli ultimi anni sta guadagnando sempre più consensi, al punto tale da diventare il linguaggio di riferimento della comunità statistica.

Abstract

The purpose of this brief article is to give an introduction to Sweave, a tool that allows to combine LaTeX code and R code. R is a statistical programming language that is gaining popularity in the last years, thus becoming the referential programming language for the statistics community.

1 Introduzione

Ai lettori di ArsTeXnica probabilmente non serve una descrizione di LaTeX, ma forse è opportuno spendere qualche parola introduttiva sull'ambiente di sviluppo R.

R è un ambiente e un linguaggio di programmazione per il calcolo statistico e la rappresentazione grafica dei dati. Apparso per la prima volta nel 1993, R è un dialetto di S, ambiente e linguaggio di programmazione statistico sviluppato nei laboratori Bell da John Chambers e colleghi (Hornik, 2013).

R è disponibile gratuitamente sotto licenza GNU GPL e conta attualmente più di 5200 pacchetti statistici, scaricabili gratuitamente tramite il Comprehensive R Archive Network (CRAN `http://cran.r-project.org/mirrors.html`). Tali pacchetti coprono la maggior parte delle metodologie statistiche e in taluni casi, come nei *mixed effect models* (Bates, 2010; Pinheiro e Bates, 2000) e nella statistica bayesiana (Kruschke, 2011), rappresentano il cosiddetto "gold standard". Nei pacchetti spesso si trovano le più moderne tecniche statistiche: in molti casi essi sono scritti direttamente dagli ideatori delle ultime novità nel campo della statistica.

Fra i numerosi pacchetti che si trovano nel CRAN ve ne sono anche alcuni (più di 40) che permettono delle efficaci ed eleganti rappresentazioni grafiche dei dati usando concetti innovativi e sperimentali, come, ad esempio, la "grammatica dei grafici" (Wilkinson, 2005), pienamente sviluppata in ggplot2 (Wickham, 2009).

Altri pacchetti e funzionalità di R riguardano gli strumenti per la gestione e/o il trattamento dei dati, come ad esempio tuneR (Ligges *et al.*, 2013), utile per il processamento dei dati audio, o il pacchetto RODBC (Ripley e Lapsley, 2010), necessario per le connessioni ODBC[1].

Oggigiorno R non è ancora così diffuso nella comunità scientifica, questo probabilmente anche a causa della grande diffusione di programmi di più lungo corso e dotati di una comoda interfaccia grafica, come SPSS (SPSS Italia, 2014) e SAS (SAS Institute, 2014), che permettono anche a chi ha una conoscenza minima del tema di fare analisi statistiche. Ciò nonostante R risulta essere il più usato fra i "software minori" in ambito statistico, mostrando una tendenza in rapida crescita (Muenchen, 2014). Per gli statistici di professione R ormai è il software di riferimento.

2 I report statistici

Un report statistico viene redatto con lo scopo di informare i lettori sui risultati che possono provenire dalla ricerca sperimentale, dai dati di interviste e questionari o da altre fonti relative a una particolare area o progetto (vedi Figura 1). Qualsiasi ricerca che implichi la raccolta di dati quantitativi (e a volte anche qualitativi) richiede un'analisi statistica, o per lo meno l'utilizzo di statistiche descrittive. Né l'analisi statistica né la ricerca in sé è utile finché non viene redatta in un formato leggibile e comprensibile. La preparazione del report statistico fa parte di questo processo comunicativo.

Se a un profano può sembrare che realizzare report statistici sia un compito banale, ossia un semplice copia-incolla dei risultati, motivazioni che risiedono nelle fondamenta della sperimentazione scientifica mostrano come ciò non basti. In ogni esperimento a carattere scientifico la metodologia utilizzata deve essere chiara e trasparente, per permettere la riproducibilità dell'esperimento e la conseguente verifica dei risultati.

Anche se un report statistico non è un articolo scientifico, elementi quali *introduzione*, *review dello stato dell'arte della letteratura*, *descrizione delle ipotesi* e *metodo* sono caldamente consigliati, se non strettamente necessari. Questo permette di inquadrare la ricerca in un contesto più ampio

1. le connessioni ODBC sono delle interfaccie native che permettono l'accesso al *database management system* utilizzabile da quei linguaggi di programmazione in grado di chiamare funzioni di librerie native (da Wikipedia, `http://it.wikipedia.org/wiki/ODBC`)

anche dai "non addetti ai lavori". L'*introduzione* dovrebbe spiegare la motivazione e la natura della ricerca in atto, nella *review dello stato dell'arte della letteratura* si dovrebbe cercare di indicare i principali lavori scientifici che stanno alla base del proprio lavoro, in modo da giustificare la formulazione delle *ipotesi*. Nel *metodo* va invece descritto come sono stati raccolti i dati, la natura degli esperimenti o dei questionari, la tipologia di partecipanti e qualsiasi aggiustamento che è stato apportato durante lo svolgimento del progetto.

La descrizione delle analisi statistiche utilizzate è una parte molto importante del *metodo*, anche se spesso negli articoli scientifici viene trascurata e si tende, invece, ad includerla nei *risultati*. Nel report statistico questa parte è fondamentale; si indica quale software è stato utilizzato per l'analisi dei dati, la sua versione e se sono stati utilizzati pacchetti aggiuntivi. Inoltre, si dichiara esattamente che tipo di analisi è stata effettuata e, se pertinente, con quali correzioni ai test. Infine, sono importanti le motivazioni sulla scelta dell'uso di alcune analisi e non di altre.

Tutte queste informazioni sono fondamentali per la riproducibilità della ricerca scientifica e per la valutazione della correttezza o dell'opportunità delle tecniche statistiche utilizzate.

Una sezione che dovrebbe essere particolarmente accurata ed estesa, nei report statistici, è la descrizione dei dati. Va descritto accuratamente il campione da cui sono stati presi i dati, utilizzando statistiche descrittive riguardanti l'età (media, deviazione standard, *range*), la distribuzione di genere (numero di maschi e femmine), se pertinente, la manualità (destro o mancino), il grado di istruzione (frequenza per titolo di studio, media e deviazione standard degli anni di studio), eccetera. La stessa cosa va fatta per i dati di interesse della ricerca, magari aiutandosi con grafici, come i *boxplot* o gli istogrammi.

Un altro metodo che rende ancor più trasparente l'analisi, con conseguente giovamento dell'applicazione del metodo scientifico, è rendere disponibili tutti i dati e, se possibile, il codice delle analisi effettuate. I dati possono essere salvati come foglio Excel o in altro formato, mentre il codice delle analisi viene salvato nel formato del software utilizzato.

Nella pratica comune, però, spesso queste indicazioni non vengono seguite e il codice delle analisi statistiche e i dati analizzati vengono tenuti separati, sebbene siano entrambi una parte fondamentale per la verifica dei risultati. Quando si analizzano i dati, è necessario testarli in più modalità per capirne le caratteristiche e comprendere quali siano i modelli statistici a loro più adatti. Riprendere i dati e le analisi più e più volte può portare ad avere numerose versioni di dati, analisi e report e può essere difficile comprendere a quali analisi e a quali dati corrisponda ciascun report (Leisch, 2003).

Una soluzione a tutto questo è l'integrazione di dati, delle analisi e del report statistico, tramite il *literate statistical practice* (Rossini *et al.*, 2003). In questo modo le analisi sono incorporate all'interno dello stesso report. Questa metodologia non è applicabile a tutti i software statistici ma è facilmente effettuabile con R, nei limiti della dimestichezza che l'utente ha con R e più in generale con il mezzo informatico. Nel prossimo paragrafo parleremo di come questo tema sia stato affrontato in R.

3 La *Reproducible Research*

Il CRAN offre l'opportunità di vedere i pacchetti al suo interno organizzati in tematiche (attualmente più di 30) che vanno dalla statistica bayesiana, passando per l'econometria, la genetica, il *machine learning*, finendo in strumenti statistici utili per il Web.

La tematica che ci interessa è quella che si occupa della riproducibilità della ricerca (Kuhn, 2014), ossia di tutti quei pacchetti che permettono di preparare report statistici ben impaginati e comprensibili anche a chi non usa R e non ha familiarità con le sue funzioni, faticando quindi a capirne l'output.

L'idea alla base di tutti questi pacchetti è quella di scrivere un unico script contenente codice R di analisi e codice per la creazione del report. In questo modo, se si vuole fare un aggiornamento dei dati, non è necessario riscrivere tutto il testo ma basta eseguire nuovamente lo script.

In breve, R offre pacchetti che permettono di esportare i report in svariati formati, fra cui HTML, markdown, Rich Text Format e formati Word proprietari.

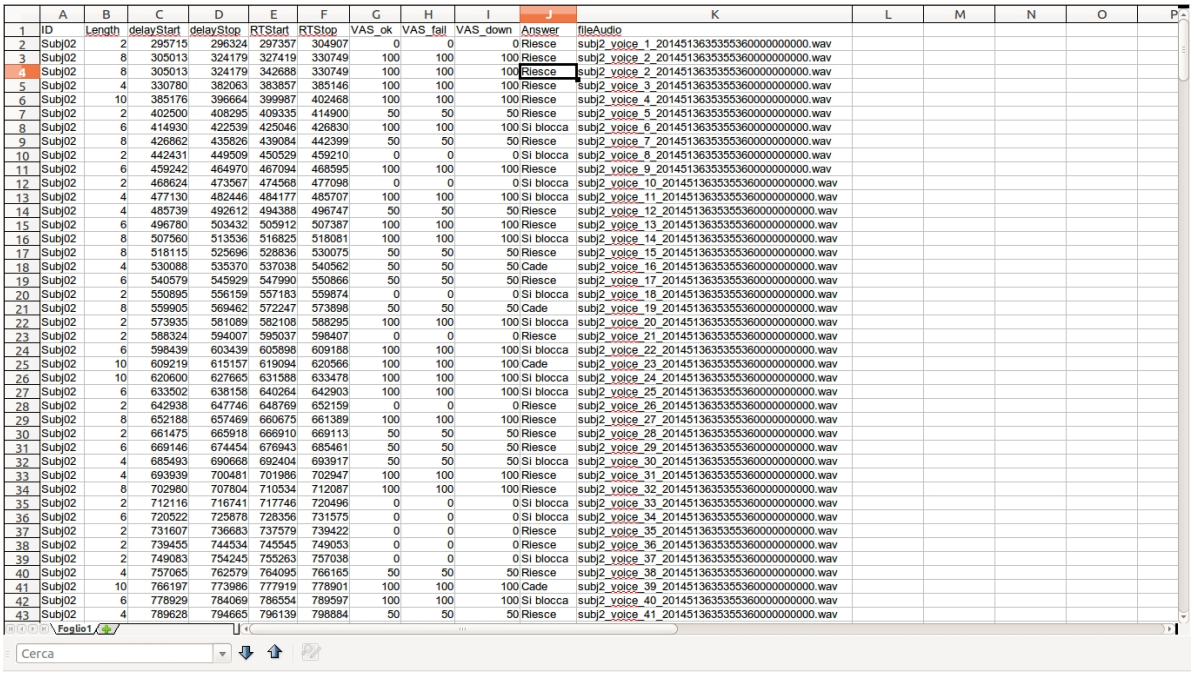
La possibilità più supportata al momento è LaTeX, tramite Sweave (Leisch, 2002) e pacchetti opzionali come xtable (Dahl, 2013), Hmisc (Harrell Jr *et al.*, 2014), animation (Xie, 2013) e tikzDevice (Sharpsteen e Bracken, 2013).

4 Sweave

Sweave fornisce un framework flessibile che permette di mettere insieme, come già detto, la documentazione delle analisi, il codice R e il suo output in un unico file. I file Sweave sono semplici file testuali, con l'estensione `.nw`, `.rnw`, `.Rnw`, `.snw` o `.Snw`.[2]

All'interno di un file Sweave troviamo *code chunks* e *documentation chunks*. All'interno dei *code chunks* troviamo il codice R, mentre all'interno dei *documentation chunks* si scrive il codice LaTeX.

2. Per approfondimenti sull'utilizzo di Sweave, visitate la pagina `http://www.stat.uni-muenchen.de/~leisch/Sweave/`] e consultare l'articolo apparso su ArsTEXnica (Crivellaro, 2011).



	A	B	C	D	E	F	G	H	I	J	K
1	ID	Length	delayStart	delayStop	RTStart	RTStop	VAS_ok	VAS_fail	VAS_down	Answer	fileAudio
2	Subj02	2	295715	296324	297357	304907	0	0	0	Riesce	subj2_volce_1_20145136353553600000000000.wav
3	Subj02	8	305013	324179	327419	330749	100	100	100	Riesce	subj2_volce_2_20145136353553600000000000.wav
4	Subj02	8	305013	324179	342688	330749	100	100	100	Riesce	subj2_volce_2_20145136353553600000000000.wav
5	Subj02	4	330780	382063	383857	385146	100	100	100	Riesce	subj2_volce_3_20145136353553600000000000.wav
6	Subj02	10	385176	396664	399987	402468	100	100	100	Riesce	subj2_volce_4_20145136353553600000000000.wav
7	Subj02	2	402500	408295	409335	414900	50	50	50	Riesce	subj2_volce_5_20145136353553600000000000.wav
8	Subj02	6	414930	422539	425046	426830	100	100	100	Si blocca	subj2_volce_6_20145136353553600000000000.wav
9	Subj02	8	426862	435826	439084	442399	50	50	50	Riesce	subj2_volce_7_20145136353553600000000000.wav
10	Subj02	2	442431	449509	450529	459210	0	0	0	Si blocca	subj2_volce_8_20145136353553600000000000.wav
11	Subj02	6	459242	464970	467094	468595	100	100	100	Riesce	subj2_volce_9_20145136353553600000000000.wav
12	Subj02	2	468624	473567	474568	477098	0	0	0	Si blocca	subj2_volce_10_201451363535536000000000000.wav
13	Subj02	4	477130	482446	484177	485707	100	100	100	Si blocca	subj2_volce_11_201451363535536000000000000.wav
14	Subj02	4	485739	492612	494388	496747	50	50	50	Riesce	subj2_volce_12_201451363535536000000000000.wav
15	Subj02	6	496780	503432	505912	507387	100	100	100	Riesce	subj2_volce_13_201451363535536000000000000.wav
16	Subj02	8	507560	513536	516825	518081	100	100	100	Si blocca	subj2_volce_14_201451363535536000000000000.wav
17	Subj02	8	518115	525696	528836	530075	50	50	50	Riesce	subj2_volce_15_201451363535536000000000000.wav
18	Subj02	4	530088	535370	537038	540562	50	50	50	Cade	subj2_volce_16_201451363535536000000000000.wav
19	Subj02	6	540579	545929	547990	550866	50	50	50	Riesce	subj2_volce_17_201451363535536000000000000.wav
20	Subj02	2	550895	556159	557183	559874	0	0	0	Si blocca	subj2_volce_18_201451363535536000000000000.wav
21	Subj02	8	559905	569462	572247	573898	50	50	50	Cade	subj2_volce_19_201451363535536000000000000.wav
22	Subj02	2	573935	581089	582108	588295	100	100	100	Si blocca	subj2_volce_20_201451363535536000000000000.wav
23	Subj02	2	588324	594007	595037	598407	0	0	0	Riesce	subj2_volce_21_201451363535536000000000000.wav
24	Subj02	6	598439	603439	605898	609188	100	100	100	Si blocca	subj2_volce_22_201451363535536000000000000.wav
25	Subj02	10	609219	615157	619094	620566	100	100	100	Cade	subj2_volce_23_201451363535536000000000000.wav
26	Subj02	10	620600	627665	631588	633478	100	100	100	Si blocca	subj2_volce_24_201451363535536000000000000.wav
27	Subj02	6	633502	638158	640264	642903	100	100	100	Si blocca	subj2_volce_25_201451363535536000000000000.wav
28	Subj02	2	642938	647746	648769	652159	0	0	0	Riesce	subj2_volce_26_201451363535536000000000000.wav
29	Subj02	8	652188	657469	660675	661389	100	100	100	Riesce	subj2_volce_27_201451363535536000000000000.wav
30	Subj02	2	661475	665918	666910	669113	50	50	50	Riesce	subj2_volce_28_201451363535536000000000000.wav
31	Subj02	6	669146	674454	676943	685461	50	50	50	Riesce	subj2_volce_29_201451363535536000000000000.wav
32	Subj02	4	685493	690668	692404	693917	50	50	50	Si blocca	subj2_volce_30_201451363535536000000000000.wav
33	Subj02	4	693939	700481	701986	702947	100	100	100	Riesce	subj2_volce_31_201451363535536000000000000.wav
34	Subj02	8	702980	707804	710534	712087	100	100	100	Riesce	subj2_volce_32_201451363535536000000000000.wav
35	Subj02	2	712116	716741	717746	720496	0	0	0	Si blocca	subj2_volce_33_201451363535536000000000000.wav
36	Subj02	6	720522	725878	728356	731575	0	0	0	Si blocca	subj2_volce_34_201451363535536000000000000.wav
37	Subj02	2	731607	736683	737579	739422	0	0	0	Riesce	subj2_volce_35_201451363535536000000000000.wav
38	Subj02	2	739455	744534	745545	749053	0	0	0	Riesce	subj2_volce_36_201451363535536000000000000.wav
39	Subj02	2	749083	754245	755263	757038	0	0	0	Si blocca	subj2_volce_37_201451363535536000000000000.wav
40	Subj02	4	757065	762579	764095	766165	50	50	50	Riesce	subj2_volce_38_201451363535536000000000000.wav
41	Subj02	10	766197	773986	777919	778901	100	100	100	Cade	subj2_volce_39_201451363535536000000000000.wav
42	Subj02	6	778929	784069	786554	789597	100	100	100	Si blocca	subj2_volce_40_201451363535536000000000000.wav
43	Subj02	4	789628	794665	796139	798884	50	50	50	Riesce	subj2_volce_41_201451363535536000000000000.wav

Figura 1: Esempio di un file contenente dati.

A differenza dei *documentation chunks*, che non necessitano di essere inscritti in nessuna particolare struttura, i *code chunks* iniziano con `<<label, options>>=` e terminano con `@`. Per esempio:

```
<<loading,echo=TRUE,results=hide>>=
 # in questo code chunk creo dei dati
 # casuali d'esempio, il codice viene
 # mostrato nel file .tex risultante,
 # ma i risultati dei comandi R non
 # vengono mostrati. Questo è stato
 # fatto perché in questo caso queste
 # istruzioni non danno alcun risultato

 rm(list=ls())
  # pulisce la memoria da variabili
  # che possono esser state create
  # precedentemente
 graphics.off()
  # libera la memoria da grafici
  # che possono esser stati creati
  # precedentemente

 dati =
  data.frame(
   # creo un data frame, che è un modo
   # per mantenere in memoria
   # dati congiunti

   y=c(rnorm(20,mean=10,sd=3),
       rnorm(20,mean=5,sd=2)),
    # all'interno del data frame
    # creo una variabile
    # y composta da due serie
    # di 20 numeri casuali
    # derivati da una distribuzione
    # gaussiana
    # con media 10 e deviazione
    # standard 3
    # e con media 5 e deviazione
    # standard 2

   group = rep(c("A","B"),each=20)
    # creo una variabile che
    # caratterizza i due
    # gruppi di variabili
  )
@
```

Una volta scritto il file, la successiva compilazione viene fatta tramite una chiamata a Sweave, che compilerà i *code chunks* restituendo come output, in accordo con le opzioni selezionate, i risultati del codice R in un file `.tex`. Questo potrà successivamente essere compilato con un compilatore di file TEX adatto al codice usato. Ecco i passaggi per la compilazione di un file Sweave su Terminale:

```
user@YourPC:~$ R CMD Sweave yourSwv.Rnw
user@YourPC:~$ pdflatex yourSwv.tex
user@YourPC:~$ pdflatex yourSwv.tex
```

Le opzioni di Sweave possono essere dichiarate in due modi:

- fra le parentesi acute all'inizio del *code chunk*, per opzioni che riguardano il singolo chunk;
- da qualsiasi parte in un *documentation chunk* col comando:

  `\SweaveOpts{op1=val1, op2=val2, ...}`

che permette di modificare le opzioni generiche del report. Il comando può essere scritto in qualsiasi punto di un *documentation chunk*.

Le opzioni principali sono:

label: con questa opzione si può indicare il nome del *code chunk*. È sempre utile e consigliato impiegarlo con label univoci per ogni *chunk*, sia per un possibile *debugging* del codice sia per un eventuale riutilizzo del *chunk*. Inoltre, se nello stesso *chunk* vengono create immagini, l'opzione label viene utilizzata per creare immagini automaticamente salvate col nome del *chunk* o, se il documento lo prevede, per creare file multipli di output. Il label può essere dichiarato come prima opzione del *chunk*, senza specificare che è un label (es. `<<postHoc-tests>>=`). Quando è dichiarato in qualsiasi altro ordine è necessario specificare che si tratta di un label (es. `<<opt1=value1,label=postHoc-tests>>=`).

echo: questa opzione, che può essere settata a `TRUE` o `FALSE`, viene utilizzata per indicare se si vuole che i commandi R usati nel *code chunk* siano poi riportati nel report statistico oppure no. Per una migliore trasparenza nella ricerca scientifica è bene che quest'opzione sia sempre settata a `TRUE`.

results: può prendere come valori `hide`, `verbatim` o `tex`. In questo modo è possibile indicare se e come si vuole che i risultati del *code chunk* vengano riportati, vale a dire se si vuole che vengano inseriti come codice LaTeX (opzione `tex`), che vengano inseriti all'interno dell'ambiente *verbatim*, oppure se si preferisce che non siano riportati affatto (opzione `hide`).

fig: con questa opzione si indica che come output dal *code chunk* si vuole produrre un'immagine da inserire nel documento. Se impostata a TRUE, Sweave creerà, infatti, un'immagine in formato *pdf*, che sarà nominata con lo stesso nome del label del *code chunk*. In caso di più immagini, non è possibile usare sempre lo stesso label perché ogni nuova immagine verrebbe memorizzata sostituendo quella precedentemente salvata.

Un semplice esempio di come scrivere una pagina in Sweave è mostrato nel codice seguente (per l'output, vedere la Figura 2).

```
\documentclass{article}

  \title{Un semplice esempio di come
  scrivere una pagina in
  \emph{Sweave}}
 \author{Michele Scandola}

\begin{document}

\SweaveOpts{concordance=TRUE,
              width=12, height=8}

\maketitle

\section{Introduzione}

 In questa pagina \emph{Sweave} si
 mostra un semplice esempio di come
 integrare codice \LaTeX a codice
 \emph{R}.

 <<loading,echo=TRUE,results=hide>>=
  ... # vedere esempio di codice
      # riportato in precedenza
 @

\section{Plot dei dati}

 <<boxplot,echo=TRUE,fig=TRUE>>=
  boxplot(y~group,data=dati)
  # con questo comando si esegue
  # un boxplot dei dati creati in
  # precedenza divisi per gruppo
 @

\section{T-test sui dati}

 <<ttest,echo=TRUE,
   results=verbatim>>=
  t.test(subset(dati$y,
                dati$group=="A"),
         subset(dati$y,
                dati$group=="B"))
  # con questo comando si esegue
  # un test statistico t sulla
  # differenza media fra i dati
  # provenienti dal gruppo "A"
  # ed il gruppo "B"
 @

\end{document}
```

Come si può vedere da questo esempio con Sweave è possibile riportare non solo i risultati ma anche la analisi effettuate, favorendo così la riproducibilità della ricerca. Ciò nonostante, bisogna tenere presente che l'utilizzo di Sweave non è di per sé garanzia di trasparenza e riproducibilità: in ogni caso il documento deve essere redatto in modo trasparente (ossia i comandi utilizzati devono essere resi visibili, tramite l'opzione `echo=TRUE`) e i dati devono essere accessibili.

È possibile rendere accessibili i dati sia inserendoli nel documento, preferibilmente in forma estesa (non riportando, ad esempio, solo le medie e deviazioni standard), sia rappresentandoli in maniera aggregata (tramite *boxplot* o altre tipologie di grafici). L'accessibilità dei dati è stata recentemente valutata anche per la realizzazione di report nell'ambito di articoli scientifici (Simonsohn, 2013). La proposta è stata facilmente resa effettuabile negli ultimi anni, per quanto si tratti di una oppor-

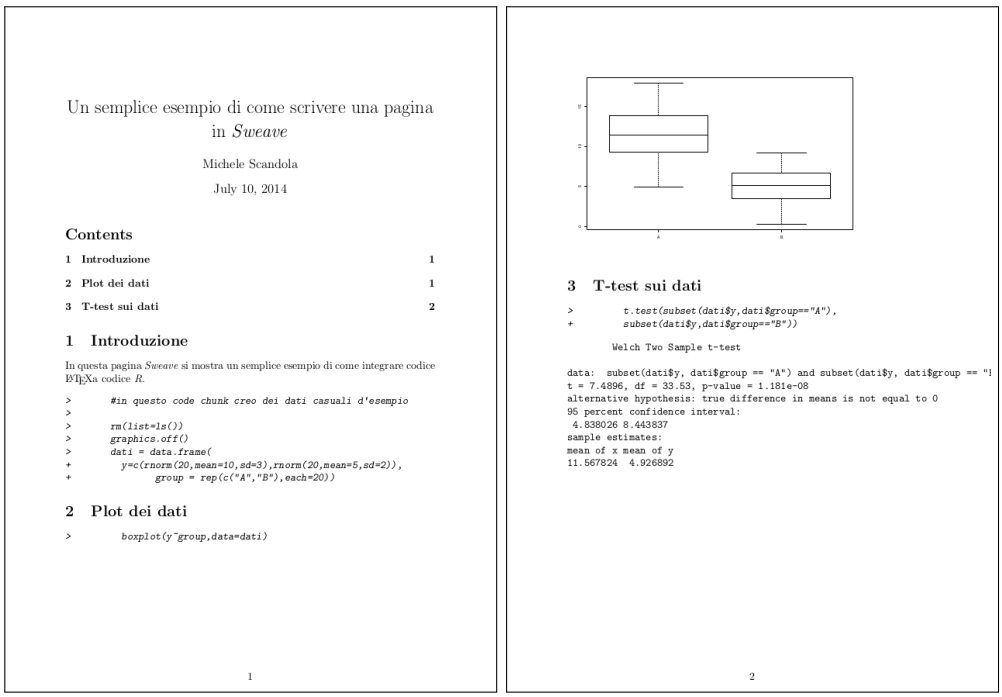


Figura 2: Output della pagina d'esempio in Sweave.

tunità non ancora completamente sfruttata, grazie alla possibilità di caricare materiale supplementare online in importanti riviste scientifiche internazionali, come nelle riviste della serie di *Frontiers*, su *Cortex*, ecc. Il procedimento per fare ciò è molto semplice ed è possibile eseguirlo nella fase di caricamento dell'articolo nei siti di riferimento delle varie riviste citate e non solo.

Al fine di realizzare report statistici completamente trasparenti, appare però opportuno rendere accessibili non solo i dati ma anche il codice Sweave e, conseguentemente, il codice delle analisi statistiche, caricando anch'esso online come materiale supplementare.

4.1 Come usare Sweave su progetti che includono molteplici *files*

Nella scrittura di un documento molto ampio, come può essere una tesi di dottorato o un libro, è normale, pratico e utile non limitarsi alla scrittura di un unico *file* ma suddividerlo in un progetto composto da più *files*.

Un ampio progetto fatto in Sweave può essere costituito da un *file* principale *.Rnw* contenente diverse chiamate a `\input` per collegarsi ad altri *files*. Tali *files* possono essere:

- *files* `.tex`
- *files* `.Rnw` o simili, che necessitano di essere convertiti in *files* `.tex`
- alcune figure statiche
- eccetera

Procedere in tal modo significa che bisogna eseguire il commando Sweave su ogni *file* `.Rnw` in modo da convertirli in *files* `.tex` e, solo in seguito, lanciare il compilatore TEX che si vuole usare sul file principale.

Questo procedimento può risultare piuttosto macchinoso, soprattutto se si vuole controllare che effetto hanno avuto alcuni cambiamenti appena fatti in un *file* secondario sul documento complessivo. Un modo per semplificare questo processo è usare il commando *SweaveAll* nel pacchetto patchDVI (Murdoch, 2013a) per R. Questa funzione necessita che due variabili vengano impostate per funzionare in modo ottimale:

.TexRoot una stringa al cui interno si trova il nome del *file* principale del progetto

.SweaveFiles un *array* di stringhe al cui interno si trovano i nomi di tutti i *files* che si vuole vengano compilati col commando Sweave

In particolare, il *file* `.Rnw` principale deve contenere un *code chunk* simile al seguente codice:

```
<<echo=FALSE>>=
 .SweaveFiles = c("main.Rnw",
                  "Chapter1.Rnw",
                  "Chapter2.Rnw",
                  "Chapter3.Rnw")
@
```

In questo modo, i *files* `main.Rnw`, `Chapter1.Rnw`, `Chapter2.Rnw` e `Chapter3.Rnw` verranno compilati con Sweave. Nell'esempio precedente abbiamo inserito anche il *file* principale *main.Rnw*, in tal modo anche il codice R presente in esso verrà ricompilato con gli altri *files*. Nel caso non vi fosse codice R nel *file* principale, lo si può omettere dalla stringa *.SweaveFiles*.

In tutti i *files* subordinati (che sono inclusi nel *file* principale tramite il commando `\input`) è invece necessario l'assegnamento della variabile *.TexRoot*. Un esempio è riportato nel codice seguente:

```
<<echo=FALSE>>=
 .TexRoot = "main.tex"
@
```

In questo modo il commando *SweaveAll* è informato che, dopo aver eseguito Sweave su tutti i *files* contenuti in *.SweaveFiles*, deve eseguire il compilatore TEX sul *file* contenuto in *.TexRoot*.[3]

5 Conclusione

Questo interesse del mondo scientifico a rendere disponibili a tutti i dati e i metodi di analisi utilizzati dimostra come si senta la necessità di una maggior trasparenza e controllo. In questo modo chiunque può verificare i calcoli eseguiti e le conclusioni da essi dedotte, tutelando così la ricerca scientifica da ogni possibile obiezione che potrebbe essere sollevata in ambito accademico in relazione alla trasparenza della metodologia di analisi impiegata. In questo sia R che Sweave possono giocare un ruolo importante.

Riferimenti bibliografici

Bates, D. M. (2010). *lme4: Mixed-effects modeling with R*. URL http://lme4.r-forge.r-project.org/book.

Crivellaro, M. (2011). «Un dialogo tra GNU-R e LaTeX: xtable e Sweave». *ArsTeXnica*, **11**, pp. 31–36. URL http://www.guitex.org/home/images/ArsTeXnica/AT011/AT11-crivellaro.pdf.

Dahl, D. B. (2013). *xtable: Export tables to LaTeX or HTML*. URL http://cran.r-project.org/package=xtable. R package version 1.7-1.

Harrell Jr, F. E. *et al.* (2014). *Hmisc: Harrell Miscellaneous*. URL http://cran.r-project.org/package=Hmisc. R package version 3.14-0.

Hornik, K. (2013). «The R-project website». URL http://www.r-project.org.

Kruschke, J. K. (2011). *Doing Bayesian data analysis : a tutorial with R and BUGS*. Academic Press, Burlington, MA. URL http://www.worldcat.org/search?qt=worldcat_org_all&q=0123814855.

Kuhn, M. (2014). «CRAN Task View: Reproducible Research». URL http://cran.r-project.org/web/views/ReproducibleResearch.html.

Leisch, F. (2002). «Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis». In *Compstat 2002 — Proceedings in Computational Statistics*, a cura di W. Härdle e B. Rönz. Physica Verlag, Heidelberg, pp. 575–580. URL http://www.stat.uni-muenchen.de/~leisch/Sweave. ISBN 3-7908-1517-9.

— (2003). «Sweave and Beyond: Computations on Text Documents». In *Proceedings of 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, a cura di K. Hornik, F. Leisch e A. Zeileis. Vienna, Austria, Dsc. URL http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/Leisch.pdf.

Ligges, U., Krey, S., Mersmann, O. e Schnackenberg, S. (2013). *tuneR: Analysis of music*. URL http://r-forge.r-project.org/projects/tuner.

Muenchen, R. A. (2014). «The Popularity of Data Analysis Software». URL http://r4stats.com/articles/popularity.

Murdoch, D. (2013a). *patchDVI: Package to patch .dvi files*. URL http://cran.r-project.org/package=patchDVI. R package version 1.9.1601.

— (2013b). «The patchDVI package». URL http://cran.r-project.org/web/packages/patchDVI/vignettes/patchDVI.pdf.

Pinheiro, J. C. e Bates, D. M. (2000). *Mixed-Effects Models in S and S-Plus*. Springer. ISBN 0-387-98957-0.

Ripley, B. e Lapsley, M. (2010). *RODBC: ODBC Database Access*. URL http://cran.r-project.org/package=RODBC. R package version 1.3-2. Michael Lapsley ha collaborato dal 1999 fino all'ottobre 2002.

Rossini, A., Heiberger, R. M., Sparapani, R. A., Mächler, M. e Hornik, K. (2003). «Emacs Speaks Statistics (ESS): A multi-platform, multi-package intelligent environment for statistical analysis».

SAS Institute (2014). «The SAS website». URL http://www.sas.com/offices/europe/italy.

Sharpsteen, C. e Bracken, C. (2013). *tikzDevice: R Graphics Output in LaTeX Format*. URL http://cran.r-project.org/package=tikzDevice. R package version 0.7.0.

Simonsohn, U. (2013). «Just post it: the lesson from two cases of fabricated data detected by statistics alone». *Psychological Science*, **24** (10), pp. 1875–1888.

3. Per ulteriori approfondimenti consultare (Murdoch, 2013b).

SPSS Italia (2014). «The SPSS website». URL http://www.spss.it.

Wickham, H. (2009). *ggplot2: elegant graphics for data analysis*. Springer New York. URL http://had.co.nz/ggplot2/book.

Wilkinson, L. (2005). *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Xie, Y. (2013). «animation: An R Package for Creating Animations and Demonstrating Statistical Methods». *Journal of Statistical Software*, **53** (1), pp. 1–27. URL http://www.jstatsoft.org/v53/i01.

▷ Michele Scandola
IRCCS Fondazione S. Lucia, Rome, Italy
SCNLab, Dipartimento di Psicologia, Università degli Studi di Roma "La Sapienza"
NPSY-Lab.VR, Dipartimento di Filosofia, Pedagogia e Psicologia, Università degli Studi di Verona
michele dot scandola at univr dot it

▷ Nadia Baltieri
Dipartimento di Filologia, Letteratura e Linguistica, Università degli Studi di Verona
nadia dot baltieri at univr dot it

Typesetting and highlighting Unicode source code with LaTeX: a package comparison

Roberto Giacomelli, Gianluca Pignalberi

Abstract

LaTeX offers its users several packages to typeset and highlight source code, ranging from the very basic one (verbatim) to the almost complete one (listings). How do they perform when their input is a Unicode source code? This paper addresses this question with focused tests and tries to lead readers to a choice, be it either definitive or case-driven.

Sommario

LaTeX offre ai suoi utenti diversi pacchetti per comporre ed evidenziare del codice sorgente; questi pacchetti vanno dal più spartano (verbatim) a quello pressoché completo (listings). Come si comportano quando il loro input è un sorgente Unicode? L'articolo risponde a questa domanda con dei test specifici e cerca di condurre i lettori a una scelta, sia essa definitiva o da prendere di volta in volta.

1 Introduction

Global world needs a global approach to languages. Computer programs are no exception: they have local versions that should run smoothly on every computer. Documenting those programs and the corresponding source code for local communities should be as straightforward as it is running them. So it is extremely important that typesetting programs, and especially TeX, manages different languages and different alphabets.

In the TeX world multilingual support is very well developed: fonts representing non-Latin alphabets are included and supported, babel and polyglossia support a lot of languages with specific hyphenation and typographic rules; inputenc supports a nearly-Unicode input; XeLaTeX and LuaLaTeX are Unicode-compliant; the Junicode font (Baker, 2014), distributed with TeX, is an Open Type font directly usable with XeLaTeX/LuaLaTeX and is an invaluable tool for medievalists. Linguists who write documents containing "exotic" alphabets are indeed very well supported. Can computer scientists and engineers expect such a support when typesetting source code? This paper aims at addressing such a question.

After a short introduction on the Unicode standard presented in section 2, in section 3 we list the packages considered for this comparative test and talk about a couple of external tools (`pygments` and `fold`) that can (or have to) support some of the listed packages. Section 4 reports on the two small source codes (Unicode-encoded) used as typesetting benchmark. The final section 5 reports on the results of our test examining in detail every typeset code and summarizing the same results in a synoptic table.

2 The Unicode World

2.1 Why is Unicode so important?

Human kind developed thousands of languages and tens of thousands of symbols to write them—the letters.[1] Up to some years ago, computers could handle 128 or 256 characters at a time thanks to character codes like ASCII or EBCDIC. Those codes were born when memory, storage devices, processors and, therefore, computers and computation time were extremely expensive and precious.

Then Unicode came—a consortium that aimed at unifying all of the alphabets on earth and at creating *the* standard. It is impossible to represent tens of thousands of letters, ideograms, numbers and symbols with a narrow code like ASCII, so Unicode encodes characters with 8 bits or more. Needless to say, this standard needs more memory and computation resources than the past standards, but modern computers are powerful enough to quickly manage Unicode.

Unicode also offers a unique way to help linguists to write their multilingual documents and computer developers to write their international programs without worrying about local characters codes: every character has a code number not shared by any other character. The first 128 characters are the same of ASCII (i.e., Unicode keeps the ASCII order) though the corresponding binary representation may vary according to the chosen format (i.e., UTF-8 encodes the letter A with 41_{h}, like ASCII, while UTF-32 encodes the same letter with 00000041_{h}; see section 2.2).[2] While linguists can benefit from the straight way to write and store multilingual documents, programmers are not just bound to Unicode text strings: they can even use Unicode characters in variable names. In our opinion this is a great improvement in code readability.

Be careful: you need a Unicode editor to encode/decode a Unicode (source) file. Having such

1. Not to mention ideographic written languages.
2. From now on, stretching the term meaning, we will refer to characters beyond the 127^{th} as "Unicode characters".

an editor does not ensure file intelligibility. Indeed the editor can only show those glyphs forming the font in use and not necessarily a Unicode font has all the glyphs recommended by the standard.

2.2 How Unicode was designed

The Unicode encoding is a freely-available set of specifications for multilingual information interchange. As every other text encoding, Unicode provides a unique number—the *code point*— for every character to be represented. The code point identifies a specific character which is stored in memory as a sequence of bytes. The code point is independent of the hardware[3] and of the software platform and comes in a variety of multibyte formats called UCS *Transformation Format*—UTF. UCS stands for Universal Character Set. The Unicode standard ensures that every represented character will need at most 21 bits.

There are three main UTFs: UTF-8, co-created by Rob Pike and Ken Thompson, UTF-16 and UTF-32. The first one is the most widely used format and it is the *de facto* standard encoding for text, XML and JSON files;[4] the detailed UTF-8 encoding/decoding algorithm follows.

A Unicode code point is transformed from/into a sequence of one or more bytes according to the UTF rules. Table 1 shows such lookup table for UTF-8. It is straightforward to understand that a UTF-8 character is 1 to 4 bytes long according to its code point.

For example (as in Korpela (2006)), UTF-8 encodes the code point U+212B—associated to the character Å—with 3 bytes because it belongs to the third interval of table 1 as $800 < 212B < FFFF$. What about those 3 bytes? The first step is to convert the hexadecimal code point into a binary number: $21\,2B_{16} \rightarrow 00100001\,00101011_2$. Now we use this number, or better, its 16 bits, to fill in the blanks (we highlight them in italics; the numbers in roman are those set by the standard) in range 3 of table 1: 1110*0010* 10*000100* 10*101011*. Now convert this 3-bytes binary sequence into the hexadecimal equivalent: $11100010\,10000100\,10101011_2 \rightarrow E2\,84\,AB_{16}$. $E2\,84\,AB_{16}$ is the 3-bytes sequence we are looking for: the UTF-8 representation of Å in memory. If we revert the procedure starting from the E2 84 AB sequence we immediately realize that the first byte starts with 1110 and so the sequence, and hence the character, belongs to the third range. Then we go back to get the code point 212B.

Please notice that the encoder always maps characters supposing that the input data are represented according to the requested format but without any guarantee that this is true.

It should be clear that data are still written in a file as a stream of bytes. A second consequence is that a Unicode string of fixed length (i.e., containing a fixed number of glyphs) cannot be represented by a fixed number of bytes. Generally speaking, UTF-8 does not allow a direct indexing of Unicode strings (we are not considering the trivial case of a text of only ASCII symbols that have a direct one byte representation because we cannot rely on this specific case). Hence programming languages that support Unicode strings are not likely to provide this indexing operator.

Other transformations like UTF-16 and UTF-32 are much simpler than UTF-8: the first one uses 16 or 32 bits to encode every character, while the second one uses 32 bits. A text encoded in UTF-32 is of fixed length in terms of memory request and it is straight to decode such a text without any of the passages described for UTF-8; even indexing is straightforward. Differently, UTF-16 needs to be encoded similarly to UTF-8 and hence does not provide file indexing. Despite being slightly more complex than UTF-32, UTF-16 saves memory when used to encode a text with a lot of characters that UTF-8 would encode with 3 bytes (mostly Asian). Nevertheless, UTF-8 holds a number of big advantages[5] over the simpler UTF-16 and UTF-32 formats:

- it is ASCII-compatible: any UTF-8 code point less than 128 is guaranteed to have the same binary representation of ASCII;
- it saves memory: 4-bytes encoded characters are not particularly common in practice; the most part of web pages is plain ASCII (i.e., HTML tags and CSS properties);
- it has no endianness issues: only UTF-16- or UTF-32-encoded texts can be big- or little endian-ordered;
- it is the standard (for the internet, at least): storing files in UTF-8 allows direct reading and writing, without any re-encoding step.

2.3 Unicode, programmers and typesetters

Many modern programming languages take advantage of the Unicode standard and their compilers or interpreters have to read Unicode source files. Go by Google (Griesemer *et al.*, 2014) and Rust by Mozilla Foundation (Hoare e Rust Project Developers, 2014) are such new mainstream languages which also attempt to be *system programming languages*. They both require UTF-8-encoded source files.

3. The sequence of bytes is often not: it should be noticed that some Unicode formats take into account the endianness.

4. The JSON Javascript Object Notation is a data description format used over the internet in the client-server communication.

5. More information can be retrieved on `http://utf8everywhere.org/website`.

TABLE 1: The multibyte UTF-8 encodes code points to numbers from 1 to 4 bytes long. Ranges, here hexadecimally represented, are grouped by length and have fixed data in specific positions just like a prefix notation. This technique immediately addresses the decoding process and, most important, how many bytes are needed to represent the current character.

Byte 0	Byte 1	Byte 2	Byte 3
Range 1 → 0 ... 7F			
0 a_6 a_5 a_4 a_3 a_2 a_1 a_0			
Range 2 → 80 ... 7FF			
1 1 0 a_{10} a_9 a_8 a_7 a_6	1 0 a_5 a_4 a_3 a_2 a_1 a_0		
Range 3 → 800 ... FFFF			
1 1 1 0 a_{15} a_{14} a_{13} a_{12}	1 0 a_{11} a_{10} a_9 a_8 a_7 a_6	1 0 a_5 a_4 a_3 a_2 a_1 a_0	
Range 4 → 10000 ... 10FFFF			
1 1 1 1 0 a_{20} a_{19} a_{18}	1 0 a_{17} a_{16} a_{15} a_{14} a_{13} a_{12}	1 0 a_{11} a_{10} a_9 a_8 a_7 a_6	1 0 a_5 a_4 a_3 a_2 a_1 a_0

Even LaTeX programmers/typesetters can easily compile UTF-8-encoded documents, provided

1. they use XeTeX or LuaTeX—Unicode compliant— or pdfTeX with the `utf8` input encoding as typesetting engines;

2. they use a Unicode-compliant font that has, at least, all of the needed glyphs.

Typesetting a Unicode source code into a LaTeX document (i.e., for documentation purpose) can be trickier than typesetting Unicode text and additionally requires that

3. the specific package correctly handles Unicode sources.

3 Packages and external tools

LaTeX users who need to typeset source code in a fancy yet typographically perfect way have several packages to pick from. Every one of them has pros and cons: one is light but too limited, one is way too powerful and configurable yet heavy; one lacks some features though excels in some other tasks. Some of them are incapable of correctly typesetting *and/or* highlighting source code containing those Unicode characters longer than one byte.

Our test will take into account the following packages: verbatim, fancyvrb, jvlisting, listings, minted, verbments, and pythontex and will unveil how they manage Unicode source files and how flexible they are. No other feature, i.e., line numbering, will be taken into account though we will remark which packages can mark (in draft mode) or wrap lines too long.

As already stated, some of those packages can or have to use the tools described in sections 3.1 and 3.2.

3.1 Pygments

Pygments (BRANDL *et al.*, 2014) is a Python project dedicated to syntax highlighting. Every programming language obeys to several syntax and semantic rules and needs a process that transforms a source code written in that language into a real executable. The same mechanism that transforms the source code into a program can be used to highlight or beautify (i.e., indent according to specific styles) the source code.

Pygments, which is a very accurate (meta)lexer, easily highlights different languages;[6] it can also expand its capabilities to include new programming languages. It outputs the results in different formats including LaTeX.

Pygments considers a *lexer* as a programming language analyzer to recognize both syntax and semantic items of a grammar while considers a *style* as a typographical set of rules to apply to the language elements such as keywords, operators, literal values, comments and so on.

Three out of the tested packages, i.e., minted, verbments and pythontex, can only highlight source files through Pygments accessing its script `pygmentize`. The script is also capable to show the user the available lexers and styles respectively with the following commands issued in a terminal session:

```
$ pygmentize -L lexer
$ pygmentize -L style
```

6. The list of lexers supported by Pygments is very long, as reported in the official web site `http://pygments.org/docs/lexers/`.

In addition, we can read the web page http://pygments.org/docs/lexers/ to get the same information on lexers.

It can be useful a short "user's point of view" of the three packages.

3.1.1 The Verbments package

The latest available version of the Dejan Živković's LaTeX package is currently the 1.2 of the year 2011. Verbments is the union of the words 'verbatim' and 'pygments', the two modules verbments is based on.

One run of the chosen typesetting engine—`pdflatex`, `xelatex` or `lualatex`—with the option `-shell-escape` on the source TeX document suffices. Indeed, the verbatim material is passed to Pygments and the result is saved in an external file (a buffer, as we will refer it from now on) for normal inclusion via `\input`. The working cycle shall be the following: write the source code to highlight in a pyglist environment in the TeX source file—verbments has no macro to include verbatim code from an external file—and run the typesetting engine whenever the content changes.

Subsequent compilations can miss the `-shell-escape` option because the verbments macro simply loads the already generated buffer. If the verbatim material changes, the buffer is not automatically updated and no warning is raised by verbments to alert the user that the document keeps having the older verbatim material. This apparently handy behavior may be dangerous. If our document contains two or more pyglist environments, the buffer always contains the code in the last pyglist. When compiling without `-shell-escape`, all of the source codes typeset in the document are the same: that one saved in the buffer. It might be convenient working this way until the moment we need to compile the camera-ready document or to check sizes. In this case it is necessary to use `-shell-escape`.

The package verbments seems to be an agile and simple tool especially useful for those users who are writing documents with short code fragments such as software manuals, programming books or other kind of teaching materials, that can be "hard-wired" in the document.

3.1.2 PythonTeX as highlighting engine

As we can read from its manual, PythonTeX's goal is primarily to execute and typeset Python code from within LaTeX; as a secondary achievement it provides the way to syntax highlight source code written in any of the languages supported by Pygments.

This package is delivered with TeX Live and MikTeX, so it is immediately available to nearly all TeX users. The Python code included in a TeX document is executed—so some troubles might happen—and the execution results are included back in the document.

The compilation process involves three step:

1. the first run of any TeX typesetting engine on the TeX document produces an external file containing the Python code;
2. the subsequent execution of the `pythontex.py` script on the external file outputs the results of the original Python code;
3. the final re-run of the chosen TeX engine pulls the pythontex output back in the document.

During the upcoming user's editing activity PythonTeX is able to run the Python code only if it has been changed and, in any case, when something has been changed in the `tex` source file; otherwise there is no need to run it again and PythonTeX will not do it.

The time has come to show a minimal example with PythonTeX to highlight the LaTeX code chosen for our benchmark. We picked the code in figure 1, stored in an external file named `alcestis.tex`.

First of all, we need to know which is the lexer to be used by Pygments to highlight a LaTeX code. (Needless to remember that PythonTeX uses Pygments as a backend to highlight code.) The lexer is, of course, `latex`. The command to perform the task is simply expressed by

`\inputpygments[`⟨*option*⟩`]{`⟨*lexer*⟩`}{`⟨*ext file*⟩`}`

Then we can write a LuaLaTeX source file as the following:

```
% !TEX program = lualatex
\documentclass{article}
\usepackage{fontspec}
\usepackage{pythontex}
\setmonofont[Scale=MatchLowercase]{%
    cmuntt.otf}

\begin{document}
\inputpygments{latex}{alcestis.tex}
\end{document}
```

to finally get the LaTeX document with the pretty-printed (highlighted) source code of `alcestis.tex` in it.[7]

3.1.3 The minted package

The Konrad Rudolph's package seems to be a middle ground between the aforementioned packages:

7. All of the shown source codes written to be compiled with XeLaTeX/LuaLaTeX will use True/OpenType fonts distributed along with TeX. Users do not need to install those fonts on their systems provided that they load them by file name instead of by font name: the typesetting engine will look for the wanted font in standard directories, including those under the distibution main directory. In this way users can try our examples as is, without any other preliminary operation.

it has more features than verbments but is not as complex as pythontex.

minted current version is the 1.7, last changed on 17/9/2011, and it is maintained from 2009 by Geoffrey Poore—the same author of PythonTeX. The author develops both packages using `git` as revision control system. The packages are hosted on `github`, respectively at `https://github.com/gpoore/minted` and `https://github.com/gpoore/pythontex`.

minted is available on CTAN and with TeX Live and MikTeX. The package documentation is also useful to successfully install Pygments, especially on Windows machines.

The workflow is made up of only one step: do *always* compile the source `tex` file with the `-shell-escape` option. Pygments is executed behind the scenes according to the idea to create a complete LaTeX user interface to the Python library.

Remember that the typeset code appearance is a Pygments responsibility, so do not blame it on minted if something goes wrong in syntactic or semantic highlighting of your code. It is not so difficult to write a new lexer in Pygments to manage particular cases.

Equipped with minted, the user is capable of typesetting math within code comments and easily setup a Pygments style. The following example shows such features:

```
% !TEX program = pdflatex

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\usepackage{minted}

% set a different highlighting style
% see the style list running the command
% $ pygmentize -L style
\usemintedstyle{colorful}

\thispagestyle{empty}

\begin{document}
\begin{minted}[mathescape]{rust}
/*
  A Rust (stupid) program to compute
  the sum of the first 100 integer
  as $\sum_{i=1}^{100}i$
*/
fn main() {
    let mut s = 0_u; // infer uint type
    for i in range(1, 101) {
        s += i; // $s \leftarrow s+i$
    }
    println!("{}", s);
}
\end{minted}
\end{document}

% file end
```

The result is the following—please notice that the `colorful` style adds a background light color to the literate string value:

```
/*
  A Rust (stupid) program to compute
  the sum of the first 100 integer
  as ∑_{i=1}^{100} i
*/
fn main() {
    let mut s = 0_u; // infer uint type
    for i in range(1, 101) {
        s += i; // s ← s+i
    }
    println!("{}", s);
}
```

3.1.4 *Pygments stand alone*

Pygments and its script `pygmentize` can be used directly.[8] The procedure is the same implemented by the discussed packages and has two steps:

1. the user executes the script in a terminal to obtain the LaTeX source file containing the syntax-highlighted version of the input code;
2. then `\inputs` or copies and pastes the previous output into the `tex` document.

The `pygmentize` flag `-O` accepts a specific option—`full`— to produce a complete LaTeX source. For example, `pygmentize` processes a Rust source file called `sum.rs` using the `rust` lexer and saves the `latex` output in a file called `output.tex`:

```
$ pygmentize -f latex -l rust \
             -O full -o output.tex \
              sum.rs
```

The file `output.tex` is a complete LaTeX file—though containing some errors—that we can compile as usual. It contains a series of custom macros and colors used in the Verbatim environment to syntax highlight the code.

Without the `full` option, `pygmentize` only produces the Verbatim environment with syntax highlighted code. This is the output file we can directly include in other documents, provided we use fancyvrb.

This can help us typeset and manage code in complex cases, for instance, those cases in which code fragments to be included in the document are in external files and these files must be compiled to verify the correctness of the code itself and even executed. No manual effort is needed when an automatic tool can store the code in one place, extract the interesting portions and syntax highlight them with Pygments. Such a tool could make it very easy creating a complex book containing many code snippets of executable source code.

8. Though the need of doing this operation by hand may seem odd, just think of a document with an occasional source code and no need to burden the compiler including another package or no chance to reliably use the package. This paper has plenty of such latest cases.

3.2 Fold

Fold is a Unix command used for making a text file with long lines more readable on a limited width terminal. Most Unix terminals have a default screen width of 80 characters, and therefore reading files with long lines could get annoying. The fold command puts a line feed every x characters if it does not reach a new line before that point.[9] The user can pass the x value to the command with the `-w` option.

This program—a filter—can be used as a preprocessor when the used package does not break input text lines. The newest versions of fold manage Unicode characters.

4 The benchmark source codes

We are going to test how LaTeX and the aforementioned packages typeset the source files shown in figures 1 and 2. The first one is a normal LaTeX code; the second one is a Rust code snippet representing a function. Those figures show both source codes as typeset by XeLaTeX and minted using the Deja Vu Mono font which is almost as complete as FreeSerif.

We will compile both the benchmark codes with pdfLaTeX, XeLaTeX and LuaLaTeX and all of the packages involved in the test.

Before letting the results speak for themselves, we want to point out that we expect pdfLaTeX to poorly perform in these tests regardless of the package. The reason is simple: pdfLaTeX cannot show characters other than Latin and a few other symbols unless it uses some specific commands provided by babel. We just cannot use these commands in a verbatim environment or alike because they would appear verbatim. Even in case a method to escape those commands existed so that they would be correctly compiled by LaTeX, it would be impossible to use them in an external code to be included in a document because they would hardly be part of the language grammar and thus accepted by the language compiler. We included pdfLaTeX in this work for completeness' sake.

5 Test and results

This section reports the detailed results. At the end of the section we summarize them in a synoptic table (table 2) that can be useful to have them all at a glance.

Please notice that we compiled every test with the draft option to let LaTeX mark lines too long. We will see which cases will bypass this option though this is one of the features we did not take into account.

9. http://en.wikipedia.org/wiki/Fold_%28Unix%29.

5.1 verbatim

The verbatim package is the simplest package of our test, probably the obvious alternative to the basic LaTeX's verbatim environment. This simple package, unable to syntax highlight or to break lines, can manage Unicode characters with pdfLaTeX and with the other Unicode-compliant typesetters. As we can see in figure 3, Greek letters are not represented. This is not verbatim fault: the first error message we got is

```
! LaTeX Error: Command \textSigma
      unavailable in encoding T1.
```

which means that our forecast was correct (see section 4). We tried to add LGR to the font encoding, but the result is awkward with all of the Latin letters transliterated into Greek. The same figure shows no problems with XeLaTeX and LuaLaTeX.

5.2 fancyvrb

The same problem is common to fancyvrb. It shows exactly the same pros and cons as verbatim and the results are almost identical. The only difference is that overfull hboxes are marked when compiling in draft mode with verbatim and not marked with fancyvrb (figure 4).

5.3 jvlisting

Another tested package is jvlisting. Of course it has the same problems of verbatim and fancyvrb with Greek letters and pdfLaTeX (because of the latter) and cannot highlight keywords but breaks lines too long. This package can mark lines too long that cannot be broken (figure 5).

5.4 listings

The listings package seems to be far better than those listed so far: it highlights a source code according to the rule for a specific language (though the supported languages are quite outdated) and it breaks lines to avoid overfull hboxes. It seems to correctly recognize Unicode characters when used with XeLaTeX and LuaLaTeX but we can see that it writes those characters in quite arbitrary positions. Moreover, when used with pdfLaTeX it has problems with Unicode sequences that it judges malformed, though it blames it on the inputenc package that issues the following message:

```
! Package utf8x Error:
  MalformedUTF-8sequence.
```

You can see the poor results in figure 6.

5.5 minted

Another tested package is minted. It is one of the packages requiring the compile option `-shell-escape` because it invokes the external program `pygmentize`. Though originally written for XeLaTeX, it can be used with pdfLaTeX too. It is

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBΣD Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ', ὁρᾷς γὰρ τἀμὰ πράγμαθ' ὡς
ἔχει,\ledsidenote{ ΑΛ.}\\
\pend
\endnumbering
\end{document}
```

FIGURE 1: The LaTeX code of our benchmark.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π _nums : Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *π _nums.get(i) {
            let π = 2*i + 3;          // prime value
            let mut k = (n/ π -3)/2; // vec index
            while k+1 > i {
                *π _nums.get_mut(( π *(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }
    return π _nums;
}
```

FIGURE 2: The Rust code of our benchmark.

unable to break lines too long or to mark them, but the cooperation with Pygments gives wonderful results in terms of highlighting, as readers can see in figure 7. The typesetting of the Rust code shows that Pygments fails at recognizing the π as a valid symbol, so Pygments' lexer issues a syntax error and surrounds the symbol with a red square.

5.6 verbments

The following tested package has been verbments which also requires the `-shell-escape` flag in compilation. Differently than minted, we can avoid this flag if the source code to include has not been changed. Indeed verbments, during the first compilation, saves a processed version of the file to include in another file. Unless we change the original source file, we have no need to make verbments update it.

The results obtained with XeLaTeX/LuaLaTeX are as good as usual (but no line break is performed and too long lines are not marked) while results with pdfLaTeX are odd: TeX commands attempting to render missing Unicode characters substitute the real glyphs. We obviously read those commands as they are typeset verbatim (figure 8).

5.7 pythontex

The last tested package is pythontex. We already mentioned its features in section 3.1.2 and will not repeat them here. Its results are exactly the same allowed by minted and the only difference is that pythontex currently runs faster.

5.8 Packages' performances at a glance

Now that we examined in detail all of the tested packages, we want to give the readers the relevant results at a glance, that is why we made the synoptic table 2.

Conclusion

Typesetting source codes in LaTeX can be as straight as loading a package (i.e., verbatim). Typesetting and highlighting source codes is trickier, especially when users need automatic line break. When Unicode characters are involved, the game gets even harder.

This paper tested seven packages written for source code typesetting and summarized their performances with three typesetters: pdfLaTeX, XeLaTeX and LuaLaTeX. Those readers who need to typeset and highlight Unicode source files in their documents have now an overview to address the best choice for themselves.

Having a last look at the results, we only can hope that developers improve the lexers performances and that the TeX community will widen the Unicode support.

Acknowledgment

We wish to acknowledge Claudio Beccari and Tommaso Gordini for writing their guide to code maps and for answering some questions of ours. They told us how they encompassed the problems we found with pdfLaTeX using a trick that avoids using any verbatim-like environment.

We also wish to acknowledge the reviewers and the proof-readers for making the paper far better than it originally was. Despite we thoroughly checked any single piece of information reported, every surviving error is just our fault.

References

Baker, P. S. (2014). *Junicode. The font for medievalists*. URL `http://junicode.sourceforge.net`. Lo leggete dando `texdoc junicode` al prompt del terminale o di DOS.

Beccari, C. e Gordini, T. (2012). «Codifiche in TeX e LaTeX». `http://www.guitex.org/home/images/doc/GuideGuIT/introcodifiche.pdf`.

Brandl, G., Ronacher, A., Pocoo team e Hatch, T. (2014). «Pygments python syntax highlighter». Sito web ufficiale `http://pygments.org/`.

Griesemer, R., Pike, R. e Thompson, K. (2014). «Go programming language». Sito web ufficiale `http://golang.org/`.

Hoare, G. e Rust Project Developers (2014). «Rust programming language». Sito web ufficiale `http://rust-lang.org/`.

Korpela, J. K. (2006). *Unicode Explained*. O'Reilly, Sebastopol, CA.

Unicode (2014). «The unicode consortium». Sito web ufficiale `http://www.unicode.org`.

▷ Roberto Giacomelli
`giaconet dot mailbox at gmail dot com`

▷ Gianluca Pignalberi
`g dot pignalberi at alice dot it`

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBD Con.:
\textit{Tournier, Pierson}.}}
\noindent ‘, ` ` ’
,\ledsidenote{ A.}\\
\pend
\endnumbering
\end{document}
```

(a) pdfLaTeX + verbatim do not show Greek letters but mark lines too long.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut _nums: Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *_nums.get(i) {
            let  = 2*i + 3;       // prime value
            let mut k = (n/-3)/2; // vec index
            while k+1 > i {
                *_nums.get_mut((*(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }

    return _nums;
}
```

(b) pdfLaTeX + verbatim do not show Greek letters but mark lines too long.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBED Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ', ὀρᾷς γὰρ τἀμὰ πράγμαθ' ὡς
ἔχει,\ledsidenote{ ΑΛ.}\\
\pend
\endnumbering
\end{document}
```

(c) XeLaTeX/LuaLaTeX + verbatim show Greek letters *and* lines too long.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π_nums: Vec<bool> = Vec::from_elem(
       if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                             // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *π_nums.get(i) {
            let π = 2*i + 3;       // prime value
            let mut k = (n/π-3)/2; // vec index
            while k+1 > i {
               *π_nums.get_mut((π*(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }

    return π_nums;
}
```

(d) XeLaTeX/LuaLaTeX + verbatim show Greek letters *and* mark lines too long.

Figure 3: Two Unicode source codes typeset with verbatim. This package lets LaTeX mark lines too long in draft mode.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBD Con.:
\textit{Tournier, Pierson}.}}
\noindent ‘,  ` ` ’
,\ledsidenote{ A.}\\
\pend
\endnumbering
\end{document}
```

(a) PDFLaTeX + fancyvrb do not show Greek letters *nor* mark lines too long.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut _nums: Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *_nums.get(i) {
            let  = 2*i + 3;       // prime value
            let mut k = (n/-3)/2; // vec index
            while k+1 > i {
                *_nums.get_mut((*(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }

    return _nums;
}
```

(b) PDFLaTeX + fancyvrb do not show Greek letters *nor* mark lines too long.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBΣD Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ`, ὁρᾷς γὰρ τάμὰ πράγμαθ' ὡς
ἔχει,\ledsidenote{ ΑΛ.}\\
\pend
\endnumbering
\end{document}
```

(c) XeLaTeX/LuaLaTeX + fancyvrb show Greek letters but do not mark lines too long.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π_nums: Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *π_nums.get(i) {
            let π = 2*i + 3;       // prime value
            let mut k = (n/π-3)/2; // vec index
            while k+1 > i {
                *π_nums.get_mut((π*(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }

    return π_nums;
}
```

(d) XeLaTeX/LuaLaTeX + fancyvrb show Greek letters but do not mark lines too long.

FIGURE 4: Two Unicode source codes typeset with fancyvrb. The results are almost identical to those of verbatim but the lines too long are not marked.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it
VLPBD Con.:
\textit{Tournier, Pierson}.}}
\noindent ‘, ` ` ’
,\ledsidenote{ A.}\\
\pend
\endnumbering
\end{document}
```

(a) pdfLaTeX + jvlisting show no Greek and break lines too long, or mark those unbreakable.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut _nums: Vec<bool> =
Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2},
// length
        true                                //
value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *_nums.get(i) {
            let  = 2*i + 3;       // prime
value
            let mut k = (n/-3)/2; // vec
index
            while k+1 > i {
                *_nums.get_mut((*(2*k+3) -
3)/2) = false;
                k -= 1;
            }
        }
    }

    return _nums;
}
```

(b) pdfLaTeX + jvlisting show no Greek and break lines too long, or mark those unbreakable.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBΣD
Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ᾽, ὁρᾷς γὰρ τἀμὰ πράγμαθ' ὡς
ἔχει,\ledsidenote{ Αλ.}\\
\pend
\endnumbering
\end{document}
```

(c) XeLaTeX/LuaLaTeX + jvlisting show Greek.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π_nums: Vec<bool> = Vec::from_elem(
       if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *π_nums.get(i) {
            let π = 2*i + 3;       // prime value
            let mut k = (n/π-3)/2; // vec index
            while k+1 > i {
               *π_nums.get_mut((π*(2*k+3) - 3)/2)
= false;
                k -= 1;
            }
        }
    }

    return π_nums;
}
```

(d) XeLaTeX/LuaLaTeX + jvlisting show Greek.

Figure 5: Two Unicode source codes typeset with jvlisting. The package performs like verbatim and fancyvrb but can break lines too long and can mark those non-breakable.

```
\documentclass[a4paper,12pt]{article
   }
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{
   FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.:
   \it ΥΒ[U+FFFD] Con.:
\textit{Tournier, Pierson}.}}
[U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
[U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
J+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD][U+FFFD]
   [U+FFFD][U+FFFD][U+FFFD][U+FFFD]note{[U+FFFD]\\
\pend
\endnumbering
\end{document}
```

(a) PDFLaTeX + listings show broken Greek letters but highlight *and* break lines too long.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd
   numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool>
   {
    let lim = (n as f64).sqrt() as
       uint;
    let mut[U+FFFD]_numsVec<bool> = Vec
       ::from_elem(
        if n%2 == 0 {n/2 - 1} else {
           n/2}, // length
        true

            // value
    );

    for i in range(0, (lim-3)/2 + 1)
        {
        if[U+FFFD]_numget(i) {
            let[U+FFFD] 2*i + 3;
                // prime value
            let mut k = ([U+FFFD])/2;
                // vec index
            while k+1 > i {
                [U+FFFD]_numget_mut[U+FFFD]
                    *(2*k+3) - 3)/2)
                    = false;
                k -= 1;
            }
        }
    }

    return[U+FFFD]_nums
}
```

(b) PDFLaTeX + listings show broken Greek letters but highlight *and* break lines too long.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it ΣVLPBD
    Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ᾽, ὁρᾷςγὰρτάμὰπράγμαθ    ' ὡςἔχει
,\ledsidenote{ ΛA.}\\
\pend
\endnumbering
\end{document}
```

(c) XeLaTeX/LuaLaTeX + listings put Greek letters in arbitrary and wrong positions.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π_nums: Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, //
            length
        true                              //
            value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if π*_nums.get(i) {
            let π = 2*i + 3;       // prime
                value
            let mut k = (nπ/-3)/2; // vec index
            while k+1 > i {π
                *_nums.get_mutπ((*(2*k+3) - 3)
                    /2) = false;
                k -= 1;
            }
        }
    }

    return π_nums;
}
```

(d) XeLaTeX/LuaLaTeX + listings put Greek letters in arbitrary and wrong positions.

FIGURE 6: Two Unicode source codes typeset with listings. While the package works bad with XeLaTeX and LuaLaTeX (arbitrarily positioned Unicode characters), it poorly performs with PDFLaTeX: it does not recognize any Unicode character and scrambles the result.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBD Con.:
\textit{Tournier, Pierson}.}}
\noindent ‘, ` ` ’
,\ledsidenote{ A.}\\
\pend
\endnumbering
\end{document}
```

(a) PDFLaTeX + minted do not show Greek letters and cannot break lines but highlighting is very good.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut | _nums : Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *| _nums.get(i) {
            let | = 2*i + 3;          // prime value
            let mut k = (n/ | -3)/2; // vec index
            while k+1 > i {
                *| _nums.get_mut(( | *(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }
    return | _nums;
}
```

(b) PDFLaTeX + minted do not show Greek letters and cannot break lines but highlighting is very good. Pygments' lexer fails at recognizing π as a valid symbol according to the syntax and so surrounds it with a red square.

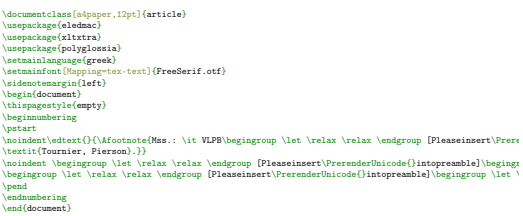
```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBΣD Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ᾽, ὁρᾷς γὰρ τἀμὰ πράγματθ' ὡς
ἔχει,\ledsidenote{ ΑΛ.}\\
\pend
\endnumbering
\end{document}
```

(c) XeLaTeX/LuaLaTeX + minted perform almost perfectly. It only lacks line break.

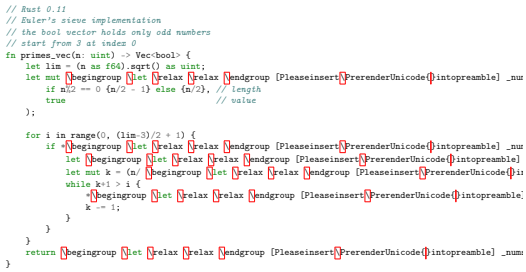
```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π _nums : Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *π _nums.get(i) {
            let π = 2*i + 3;          // prime value
            let mut k = (n/ π -3)/2; // vec index
            while k+1 > i {
                *π _nums.get_mut(( π *(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }
    return π _nums;
}
```

(d) XeLaTeX/LuaLaTeX + minted perform almost perfectly. It only lacks line break. Pygments' lexer fails at recognizing π as a valid symbol according to the syntax and so surrounds it with a red square.

FIGURE 7: Two Unicode source codes typeset with minted. The only drawback is that it does not perform line break (users should preprocess source code with `fold`). Pygments' Rust lexer fails at recognizing π as a valid symbol according to the syntax (and so do Go and C lexers).



(a) pdfLaTeX + verbments substitute Greek letters with TeX commands and cannot break lines but highlighting is very good.



(b) pdfLaTeX + verbments substitute Greek letters with TeX commands and cannot break lines but highlighting is very good. Pygments' lexer fails at recognizing π as a valid symbol according to the syntax and so surrounds it with a red square.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBΣD Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ', ὁρᾷς γὰρ τάμὰ πράγμαθ' ὡς
ἔχει,\ledsidenote{ ΑΛ.}\\
\pend
\endnumbering
\end{document}
```

(c) XeLaTeX/LuaLaTeX + verbments perform almost perfectly. It only lacks line break.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π _nums : Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *π _nums.get(i) {
            let π = 2*i + 3;          // prime value
            let mut k = (n/ π -3)/2; // vec index
            while k+1 > i {
                *π _nums.get_mut(( π *(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }
    return π _nums;
}
```

(d) XeLaTeX/LuaLaTeX + verbments perform almost perfectly. It only lacks line break. Pygments' lexer fails at recognizing π as a valid symbol according to the syntax and so surrounds it with a red square.

FIGURE 8: Our benchmark source codes typeset with verbments. It performs like minted with XeLaTeX/LuaLaTeX but results with pdfLaTeX are odd.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBD Con.:
\textit{Tournier, Pierson}.}}
\noindent μ`,  ` μ` μ'
,\ledsidenote{ A.}\\
\pend
\endnumbering
\end{document}
```

(a) pdfLaTeX + pythontex do not show Greek letters and cannot break lines but highlighting is very good.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut _nums : Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *_nums.get(i) {
            let p = 2*i + 3;          // prime value
            let mut k = (n/p -3)/2; // vec index
            while k+1 > i {
                *_nums.get_mut((p*(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }
    return p _nums;
}
```

(b) pdfLaTeX + pythontex do not show Greek letters and cannot break lines but highlighting is very good. Pygments' lexer fails at recognizing π as a valid symbol according to the syntax and so surrounds it with a red square.

```
\documentclass[a4paper,12pt]{article}
\usepackage{eledmac}
\usepackage{xltxtra}
\usepackage{polyglossia}
\setmainlanguage{greek}
\setmainfont[Mapping=tex-text]{FreeSerif.otf}
\sidenotemargin{left}
\begin{document}
\thispagestyle{empty}
\beginnumbering
\pstart
\noindent\edtext{}{\Afootnote{Mss.: \it VLPBΣD Con.:
\textit{Tournier, Pierson}.}}
\noindent Ἄδμηθ`, ὁρᾷς γὰρ τἀμὰ πράγμαθ' ὡς
ἔχει,\ledsidenote{ ΑΛ.}\\
\pend
\endnumbering
\end{document}
```

(c) XeLaTeX/LuaLaTeX + pythontex perform almost perfectly. It only lacks line break.

```
// Rust 0.11
// Euler's sieve implementation
// the bool vector holds only odd numbers
// start from 3 at index 0
fn primes_vec(n: uint) -> Vec<bool> {
    let lim = (n as f64).sqrt() as uint;
    let mut π_nums : Vec<bool> = Vec::from_elem(
        if n%2 == 0 {n/2 - 1} else {n/2}, // length
        true                              // value
    );

    for i in range(0, (lim-3)/2 + 1) {
        if *π_nums.get(i) {
            let π = 2*i + 3;          // prime value
            let mut k = (n/π -3)/2; // vec index
            while k+1 > i {
                *π_nums.get_mut((π*(2*k+3) - 3)/2) = false;
                k -= 1;
            }
        }
    }
    return π _nums;
}
```

(d) XeLaTeX/LuaLaTeX + pythontex perform almost perfectly. It only lacks line break. Pygments' lexer fails at recognizing π as a valid symbol according to the syntax and so surrounds it with a red square.

FIGURE 9: The same Unicode source codes typeset with pythontex which results are exactly like those obtained with minted.

Table 2: Summary of the typesetting and highlighting Unicode source code packages parallel test.

Package	`-shell-escape`	Highlights	Breaks lines	Program	Program + package typeset Unicode
verbatim	✗	✗	✗	P X L	✗[1] ✓ ✓
fancyvrb	✗	✗	✗	P X L	✗[1] ✓ ✓
jvlisting	✗	✗	✓[2]	P X L	✗[1] ✓ ✓
listings	✗	✓	✓	P X L	✗[3] ✗[4] ✗[4]
minted	✓	✓[5]	✗	P X L	✗[1] ✓ ✓
verbments	✓[6]	✓[5]	✗	P X L	✗[7] ✓ ✓
pythontex	✓	✓[5]	✗	P X L	✗[1] ✓ ✓
Legenda	✗ No	✓ Yes	P: pdfLaTeX	X: XeLaTeX	L: LuaLaTeX

[1] The package does manage Unicode characters but the compiler does not and, of course, does not show them along with Latin letters.

[2] The package will mark those long but non-breakable lines.

[3] The package conflicts with `utf8` of inputenc and every Unicode character is judged malformed. The typeset source code shows wrong code points instead of nothing or wrong characters.

[4] The compilation successfully ends but the Unicode characters are arbitrarily, and incorrectly, positioned.

[5] Through Pygments. Its lexer fails at recognizing the π symbol as valid and so marks it with a red square.

[6] The option can be avoided after the first compilation of the source code if the latter does not change.

[7] Unicode characters are replaced by a sequence of TeX commands that are shown as is in the verbatim-like environment.

LaTeX per la Stesura dei Rapporti di Prova

Renato Battistin

Sommario

Motivazioni, vantaggi e svantaggi dell'impiego di LaTeX nella stesura di rapporti tecnici relativi a prove di conformità a norme armonizzate vengono esaminati in un contesto di gestione strutturata dell'informazione. L'implementazione di contenuti ed aspetti stilistici del rapporto di prova è analizzata in un contesto volto al conseguimento di obbiettivi di efficienza e di riduzione dei costi di stesura.

Abstract

Motivations, advantages and disadvantages of the use of LaTeX in the preparation of technical reports relating to tests of conformity to harmonized standards are examined in the context of a structured management of the information. The implementation of content and stylistic aspects of the test report is analyzed in a context aimed at achieving the objectives of efficiency and reducing the costs of writing.

1 Introduzione

L'ambito industriale, artigianale ed istituzionale richiede spesso l'applicazione di norme armonizzate a livello nazione o internazionale. Tali norme specificano ad esempio i requisiti che deve soddisfare un particolare prodotto industriale. I requisiti possono essere della natura più varia, ad esempio possono riguardare l'aspetto qualitativo del prodotto quale, solo per citarne alcuni, il colore, la forma, la finitura superficiale, la robustezza; oppure possono riguardare aspetti quantitativi quali grandezze fisiche o chimiche del prodotto come le dimensioni, la capacità di riflettere la luce oppure quella di disciogliersi in un particolare solvente, la tenuta nel tempo agli agenti atmosferici, anche qui solo per citarne alcuni.

L'accertamento dei requisiti richiesti da una norma per un particolare prodotto richiede l'esecuzione di prove cosiddette di *conformità*. Tali prove a seconda della loro natura possono produrre dei risultati di tipo qualitativo (osservazioni) o quantitativo (dati sperimentali). In entrambi i casi il risultato va confrontato con gli eventuali requisiti richiesti dalla norma per il superamento della prova di conformità oppure con dei requisiti presupposti quando la norma specifica solo il metodo di prova.

In generale un prodotto può essere sottoposto a più prove e su più campioni e l'insieme dei risultati spesso è raccolto assieme ad altre informazioni in un documento tecnico detto *rapporto di prova*. La stesura di un rapporto di prova può essere una parte economicamente rilevante dell'applicazione di una norma, sia in termini di tempo, che di mezzi e di personale, quindi è necessario in un contesto lavorativo implementare tecniche e mezzi adeguati affinché la gestione della stesura del rapporto di prova diventi un processo efficiente e razionale.

Esamineremo, limitatamente all'esperienza dell'autore, le motivazioni che hanno condotto all'implementazione di LaTeX per la stesura dei rapporti di prova; gli obbiettivi specifici perseguiti nel cercare di ottenere l'efficienza e la razionalizzazione sopra citati.

2 Alternative, Dubbi, Svantaggi

In ambito lavorativo, così come in ambito formativo, a volte ci si può trovare nella situazione di dovere, o potere, decidere di cambiare gli strumenti editoriali impiegati quotidianamente per la gestione e la stesura di documenti. La decisione può rivelarsi piuttosto ardua, sia perché la sua implementazione indubbiamente costituirà un impegno oneroso, sia perché non è facile prevedere l'aumento di efficienza specie quando vengono introdotti strumenti editoriali oggettivamente difficili e poco conosciuti.

La decisione deve quindi essere supportata da un'analisi dei costi e dei benefici, non solo relativamente alla singola operazione di implementazione ma soprattutto all'operazione di mantenimento degli strumenti che si vogliono introdurre. In ambito lavorativo diventa quindi scontato il confronto con i tradizionali metodi di stesura di tipo WYSIWYG (*What You See Is What You Get*) caratteristico della gran parte degli elaboratori di testo ad interfaccia grafica, GUI[1], quale quello incluso nel pacchetto *LibreOffice* ed in altri elaboratori di testo commerciali.

I metodi WYSIWYG hanno l'indubbio grosso vantaggio dell'immediatezza del risultato e la (apparente) semplicità d'uso ma, a parte questo, presentano alcuni svantaggi che nell'ambito industriale si fanno pesamentemente sentire: difficoltà di integrazione automatica dei dati sperimentali, difficoltà di gestione organizzata e razionale del contenuto informativo dei documenti prodotti ma soprattutto mancanza quasi assoluta di scalabilità ossia l'effetto di riduzione dei costi all'aumentare della dimensione.

1. *Graphical User Interface.*

Le alternative soprattutto in ambito industriale esistono e sono costituite dall'implementazione di pacchetti software integrati ossia comprendenti sia la gestione dei dati che la stesura dei documenti. Se le esigenze di modifica del formato dei documenti sono poco frequenti, questi pacchetti possono essere molto utili una volta superati gli scogli iniziali dell'implementazione e della curva di apprendimento[2]. Va anche detto che il loro principale vantaggio nel lato documentale è legato strettamente alla produzione efficiente di documenti stilisticamente "statici" ossia documenti in cui l'inserimento dei dati è comunque automatizzato ma il loro aspetto stilistico è modificato raramente. Un esempio è la stesura dei prospetti paga mensili ove i dati vengono recuperati automaticamente da un database e riportati in regioni (campi) ben precise di un modello di documento predefinito. Rimanendo nell'ambito delle prove di conformità è ovvio che questi software sono vantaggiosi se la gran parte delle norme applicate per l'esecuzione delle prove richiedono l'esecuzione di un'unica prova senza varianti, caso piuttosto frequente ad esempio nell'ambito chimico e farmaceutico dove una prova può essere finalizzata alla sola determinazione di un preciso composto chimico. La possibilità di scalabilità è essenzialmente limitata alla possibilità di automazione del processo di stesura del rapporto di prova ma difficilmente può essere ottenuta nella gestione efficiente del contenuto informativo dei documenti.

Esistono norme che richiedono l'esecuzione di più prove eventualmente non tutte obbligatorie, magari con la presenza di varianti o da eseguire su un numero non prestabilito di campioni oppure prove che presentano varianti di esecuzione a seconda dei risultati parziali ottenuti. Norme che includono più prove spesso non sono autocontenute e richiamano anche altre norme per il completamento dell'esecuzione di un specifica prova oppure per la sua applicazione su particolari tipi di campioni. Queste norme comportano la stesura di rapporti di prova dove è richiesta un'elevata versatilità sia in termini stilistici che di contenuti. Sono questi rapporti di prova che, per garantire anche i requisiti di efficienza e scalabilità, hanno bisogno di strumenti editoriali versatili come LaTeX. Pur tuttavia la sola dotazione stilistica di LaTeX non basta per prendere una decisione sugli strumenti da adottare per la stesura dei rapporti. Il grosso e ben noto problema con LaTeX è il suo difficile impiego, la sua ostica curva di apprendimento, la sua mancanza di intuitività e va aggiunta anche la scarsità di mezzi per la gestione del contenuto informativo, tutte cose piuttosto evidenti vista la sua "vocazione" tipografica.

2. Senza dimenticare i costi di acquisizione, mantenimento e quelli di aggiornamento sui quali si basa il vero profitto dei fornitori di software commerciale.

Tuttavia il grosso ed innegabile vantaggio di LaTeX è la sua natura puramente testuale. Qualsiasi altro sistema di elaborazione (nel senso più ampio del termine) di testo o tipografico, specie se proprietario, deve ricorrere ad interfacce grafiche WYSIWYG o compendiare con software ausigliari. Tutto ciò complica la possibilità di operare direttamente sui file sorgente ad esempio per intercalare altro testo, magari originato da un database o più in generale da un altro software per gli scopi più vari. Ma non solo: è l'intero file sorgente LaTeX che può essere generato direttamente nella forma e con i contenuti desiderati senza l'obbligo di affrontare la decodifica di un file binario o doversi confrontare con formati proprietari. Questa caratteristica rende LaTeX ideale per sovrapporre ad esso una struttura separata dall'aspetto tipografico e che gestisca il contenuto informativo; una struttura che permetta operazioni quali la validazione, la gestione efficiente e razionale del contenuto, la scalabilità. In definitiva tutte le caratteristiche necessarie per stendere rapporti di prova.

Nelle sessioni successive analizzeremo i contenuti di un rapporto di prova in termini stilistici quindi vedremo come sovrapporne una struttura che gestisca il contenuto informativo ed infine faremo alcune considerazioni sui costi e i benefici associati a questa scelta editoriale.

3 Analisi Stilistica

Un rapporto di prova deve avere un aspetto per certi versi austero e deve essere asettico nei contenuti ossia privo di commenti od osservazioni sui risultati. Per sua natura il rapporto richiede una cerca sistematicità nella presentazione dei risultati ed uno schema tipografico con elevata ricorrenza nei suoi elementi distintivi e tutto questo per concentrare l'attenzione del lettore sull'informazione veramente importante in esso contenuta: i risultati delle prove effettuate. Il rapporto quindi sarà composto da elementi stilistici comuni a tutti i rapporti indipendentemente dalle norme e prove applicate.

3.1 Elementi stilistici comuni

Gli elementi stilistici comuni ad ogni rapporto possono essere suddivisi tra quelli richiesti obbligatoriamente dalle norme sulla stesura dei rapporti di prova e quelli non obbligatori ma che verosimilmente vengono inclusi.

Tra gli elementi informativi che obbligatoriamente devono essere inclusi almeno in una, se non altrimenti indicato, delle parti stilisticamente fisse del rapporto di prova quali la prima pagina, le intestazioni e piè di pagina, vi sono:

- un numero oppure una stringa alfanumerica quale identificativo unico del rapporto; tale identificativo deve comparire su tutte le pagine del rapporto

- le date di ricezione dei campioni, di inizio e fine delle prove, di emissione del rapporto; la data di emissione deve comparire su tutte le pagine
- l'identificazione dell'oggetto delle prove e il codice con cui il cliente identifica quel particolare oggetto
- il numero totale di pagine di cui è costituito il rapporto e tale numero va riportato vicino al numero di ogni pagina
- identificazione dell'ente emittente il rapporto, completo della sua denominazione, del suo indirizzo e di altre informazioni necessarie per il suo corretto rintracciamento
- il riferimento alla norma applicata completo di data di pubblicazione
- le firme
- una dichiarazione circa le condizioni di divulgabilità parziale del rapporto di prova

Tra gli altri elementi non obbligatori ma che possono essere inclusi vi sono:

- i loghi dell'ente emettitore e dell'eventuale ente accreditante, sulla prima ed eventualmente sulle pagine successive
- le sigle relative alla versione del modello del documento
- le eventuali dichiarazioni circa la natura, l'uso e la validità del rapporto di prova
- l'identificativo di commessa di lavoro

Tutti gli elementi stilistici associati alle informazioni sopra elencate sono comuni a tutti i rapporti di prova indipendentemente dalla norma applicata, dal numero di prove effettuate e in definitiva dai contenuti del rapporto medesimo quindi tutti questi elementi stilistici potranno essere inclusi in un *documentclass*.

A questo punto è chiaro che la prima pagina del rapporto può avere delle differenze stilistiche rilevanti rispetto alle pagine successive e quindi di conseguenza può essere necessario stabilire questa differenza nel *documentclass* ad esempio introducendo uno stile di default per l'intestazione e il piè di pagina di tutte le pagine ed una intestazione e piè di pagina appositi per la prima pagina.

Il *documentclass* dovrà comunque servire sia per specificare i vari elementi stilistici fondamentali per l'aspetto del rapporto oppure da utilizzare solamente per scopi particolari durante la stesura del rapporto, sia per definire dei comandi o ambienti comuni a tutti i rapporti:

- il modello LaTeX originale (es. `article` oppure `report`)
- vari *package* standard tra cui ad esempio `fancyhdr` per modificare le intestazioni e i piè di pagina, `lastpage` per il totale delle pagine, `ifthen` utile per costruire dei comandi condizionali
- comandi relativi a certi aspetti stilistici ricorrenti nel rapporto; ad esempio la clausola (o punto) applicata la quale deve obbligatoriamente essere specificata in modo da individuare esattamente la parte della norma che è stata applicata relativamente ad un certo risultato riportato
- comandi relativi a frasi sintetiche che specificano ad esempio la conformità o meno dell'esito di una prova ai requisiti della norma applicata, oppure che specificano l'applicazione di una particolare condizione di prova comune a più norme e prove
- comandi comuni per gestire in maniera univoca grafici o figure come ad esempio l'immagine di un campione o l'esito visivo di una prova

Il *documentclass* potrà contenere ovviamente oltre agli elementi stilistici comuni anche codice per la gestione delle opzioni, per la definizione di comandi interni ed un elenco dei *package* necessari. Al fine di garantire efficienza e razionalità nella gestione del codice LaTeX, quello che si dovrà evitare di includere nel codice del `documentclass` è il codice specifico per ciascuna norma applicata e per le prove in essa previste. Questo codice auspicabilmente dovrà essere collocato in un *package* apposito e gestirà i contenuti del rapporto.

3.2 I contenuti del rapporto

La regola d'oro è "scrivere lo stretto necessario" tuttavia questa regola può essere derogata in funzione della capacità comprensiva della persona destinataria del rapporto di prova. A seconda del tipo di norma applicata, e più spesso di quanto si pensi, invece di essere letto da una persona di cultura tecnica, il rapporto finisce in mano a persone di cultura non tecnica preposte a prendere comunque decisioni sulla base dei risultati del rapporto. Giova in tal caso, al fine di limitare un volume altrimenti eccessivo di scambi epistolari tra ente emettitore del rapporto e destinatario del medesimo, la presenza di spiegazioni relative alla modalità di esecuzione delle prove, senza però sconfinare nel commento.

Ad ogni norma applicata generalmente corrisponde un *package* il quale include i comandi di ridefinizione di quelli generici presenti nel *documentclass*, i comandi specifici per la gestione delle opzioni, ma soprattutto i comandi e gli ambienti che servono alla stesura dei risultati delle singole prove. Ad esempio la dichiarazione della norma applicata che deve comparire per esteso nella prima pagina ed eventualmente in forma abbreviata

nell'intestazione delle pagine successive, è resa mediante la ridefinizione di una coppia di comandi già presenti in forma generica nel *documentclass*.

Esistono norme di prova che hanno un carattere generale e che richiamano per alcune delle prove altre norme "satellite" alle quali viene demandato il metodo di prova ed eventualmente anche i requisiti. La gestione a livello di *package* di queste norme generiche non presenta difficoltà particolari e può essere gestita con le medesime tecniche di separazione del codice adottate tra il *documentclass* del rapporto e i *package* relativi alle norme.

Chiaramente a parte gli elementi stilistici comuni a tutti i rapporti di prova che sono definiti nel *documentclass*, è evidente che la gestione stilistica complessiva dei rapporti di prova per la tipologia di norme che abbiamo qui considerato si gioca nei *package* dove la descrizione della prova e la presentazione dei risultati deve essere gestita in una forma il più possibile comune. I risultati saranno preferibilmente presentati in forma tabulare in special modo se vanno distinti per campione provato. Dovrà essere stilisticamente sempre ben separata la parte descrittiva della prova da quella dei risultati ad esempio utilizzando semplicemente due titoletti introduttivi distinti per le due parti e sempre i medesimi per tutti i rapporti e quindi inclusi come comandi nel *documentclass*:

```
\newcommand{\specifiche}{%
\subsubsection*{\textit{Specifiche}}}%
\newcommand{\rilevazioni}{%
\subsubsection*{\textit{Rilevazioni}}}%
```

Trucchi come questi garantiscono un aspetto consistente dei rapporti di prova a prescindere dalla norma applicata e dal tipo e quantità di prove associate. Tuttavia durante la stesura di un rapporto è facile utilizzare il comando o l'ambiente errato specie se ci sono prove simili e quindi definizioni corrispondenti simili entro la stessa norma o tra norme simili. Inoltre è ben noto che l'uso efficace di un *package* richiede una sua conoscenza piuttosto approfondita se non altro per conoscerne i comandi. Tutto ciò in ambito lavorativo industriale significa scrivere procedure e documentazioni alle quali l'operatore preposto alla stesura del rapporto si deve attenere. Tuttavia questo è solo meccanismo di controllo passivo degli errori di stesura i quali sono sempre in agguato. Ad esempio un rapporto di prova deve contenere la firma del responsabile del laboratorio; se tale firma è dimenticata in quanto il comando LaTeX corrispondente non è stato inserito, allora ciò potrà non andare a detrimento dell'aspetto stilistico del rapporto ma lo andrà ad esserlo per la sua validità legale. Altri esempi sono l'ordine di stesura dei risultati delle prove eseguite oppure l'assenza di un grafico associato ad una prova.

Una soluzione alternativa è l'introduzione di meccanismi di controllo attivo della struttura informativa del documento in modo tale che sia impossibile produrre rapporti di prova incoerenti od inconsistenti, seppure stilisticamente corretti.

4 Struttura Informativa

Il termine "struttura informativa" di un documento va qui inteso come un complesso di regole che stabiliscono dove e quando le informazioni che vogliamo includere nel documento da stendere vanno collocate. Quando stendiamo un documento come ad esempio una lettera, un articolo, sappiamo che alcune informazioni vanno collocate relativamente ad altre in un certo ordine. Ad esempio sappiamo che il titolo di una sessione di testo va posto all'inizio della stessa e non alla fine anche se dal punto di vista quantitativo l'informazione trasmessa sarebbe la stessa. Questa è una tacita regola nella stesura di un articolo oppure di una lettera. Tuttavia esistono documenti come ad esempio i manuali tecnici dove le regole possono essere più complicate o comunque non così scontate. Ad esempio potrebbe essere imposto che non ci deve essere testo tra il titolo di una sessione e quello di una sottosessione se almeno una sottosessione è presente[3] oppure ci potrebbe essere un limite sul numero stesso di sottosessioni o sul loro livello di annidamento. Nel caso di rapporti di prova le regole possono riguardare la presenza delle prove; ad esempio se è presente una tal prova allora necessariamente devono essere presenti i risultati anche di un'altra. Oppure le prove di un certo tipo devono essere raggruppate in una particolare sessione a sé stante del rapporto. Queste regole non sono regole stilistiche in senso stretto e quindi difficilmente possono essere gestite da LaTeX.

L'alternativa è utilizzare un linguaggio di marcatura del testo[4], uno che fu concepito appositamente per questo scopo e che fu il primo linguaggio di marcatura ad essere standardizzato: l'SGML (*Standard Generalized Markup Language*[5]). L'uso dell'SGML presume la progettazione di un DTD (*Document Type Descriptor*) un file che descrive la struttura informativa del documento. Qui va subito chiarita una cosa: la progettazione del DTD non può essere avulsa da quella del complesso del *documentclass* e dei *package* correlati, quindi deve essere ben chiaro dove fermarsi con la strutturazione dell'informazione entro LaTeX e dove fermarsi con la dettagliatura dell'informazione entro il DTD. Un sapiente dosaggio dell'uno e dell'altro è la chiave per la semplicità dello strumento editoriale che si vuole costruire; DTD troppo dettagliati dove

3. La mancata osservanza di questa regola si può trovare ad esempio nel testo di alcune norme e può essere fonte di ambiguità nell'applicabilità delle clausole.

4. LaTeX è un linguaggio di marcatura. Tuttavia nel presente articolo diremo che un linguaggio è di marcatura se accompagnato da uno strumento di validazione della struttura informativa del testo entro il quale viene utilizzato.

5. ISO 8879:1986, *cfr.* anche Goldfarb (1990).

si arrivano a distinguere gli aspetti stilistici di un testo (es. un corsivo oppure un grassetto) risultano spesso troppo complicati per essere applicati da un umano e possono indicare limitatezza o inadeguatezza del motore stilistico impiegato. Per contro DTD troppo generici non possono essere utilizzati per verificare la correttezza formale di alcuni elementi informativi per loro natura dettagliati (es. i marcatori <indirizzo> ed </indirizzo> per descrivere esaustivamente un indirizzo postale.).

Il DTD del rapporto di prova è quindi impiegato per definire quali parti vanno stese prima delle altre, ad esempio: la prima pagina con tutte le sue informazioni che contraddistinguono il rapporto come il suo identificatore numerico, le sue date, ecc; la collocazione delle fotografie dei campioni, ad esempio nella parte terminale del rapporto prima delle firme; le norme in forma alternativa una alle altre, ciascuna con il suo insieme strutturato di prove e ciascuna prova con la sua struttura caratteristica in funzione del numero e del tipo di dati che vengono forniti come risultati della prova medesima; gli elementi strutturali semplici ma comuni a tutte le prove indipendentemente dalla norma applicata, come ad esempio note di commento a margine di certi tipi di risultati come un avviso standard relativo a certi esiti di conformità quando i valori di misura sono vicini al limite normativo.

Tale strutturazione dell'informazione è visibile nel frammento di DTD seguente relativo ad una prova che implica la misurazione di una grandezza fisica

```
<!ELEMENT lucediffusa - - (LDrecord*) >
<!ELEMENT LDrecord o o EMPTY >
<!ATTLIST LDrecord
        campione cdata #required
        misura cdata #required
        esito cdata "&pass;" >
```

e il cui valore di misura rilevato viene trascritto nel rapporto, assieme all'esito di conformità rispetto al requisito della norma, come codice SGML seguente:

```
<lucediffusa>
<LDrecord campione='999999 1'
  misura='0.20' esito='&pass;'>
<LDrecord campione='999999 2'
  misura='0.12' esito='&pass;'>
</lucediffusa>
```

Un rapporto di prova steso in SGML è del tutto analogo ad una pagina web scritta in HTML ed ha in comune con questa una caratteristica del tutto evidente: la scarsa leggibilità. L'informazione è ora strutturata ma dispersa in mezzo a testo di marcatura. Le pagine web in HTML vengono rese "visibili" dal *browser* mentre la "G" dell'acronimo SGML che sta per *Generalized* ci rammenta che dobbiamo stabilire noi cosa farne e che significato dare alla marcatura. Le daremo un significato tipografico.

5 Resa Tipografica

Il testo marcato SGML è composto in generale di elementi, attributi ed entità e il significato che daremo agli elementi e agli attributi dell'esempio dato nella precedente sessione sarà chiarito tra breve. Prima però vediamo come è definita la prova dell'esempio nel *package*:

```
\newenvironment{lucediffusa}{%

  \newcommand{\row}[3]{%
    ##1 & ##2 & ##3 \\ \hline
  }%

  \subsection*{Scattered Light}
        \clausole{Clause 2.6}
        \specifiche
The Reduced Luminance Factor,
 \lstar, is an index of
 the scattered light by the filter.
 The measurement of \lstar, performed
 by the primary method described in
 Appendix G, of the Standard,
 shall be not superior to 0.65 \cdmqlx.
\rilevazioni
The measure  value of \lstar, expressed
 in \cdmqlx, and the results
 of test are:\\[0.3cm]
\indent
 \begin{tabular}{|l|c|c|} \hline
 Sample & \lstar (\cdmqlx) & Test \\
 \hline
}{%
 \end{tabular}
}% lucediffusa
```

La definizione consiste in un ambiente LaTeX che descrive una sessione di testo con tanto di titolo, informazioni relative alla clausola applicata della norma, una descrizione succinta della prova separata da l'elencazione dei risultati in forma tabellare grazie all'impiego di un comando interno, `row`, che ne definisce le righe.

Un esempio applicativo di tale ambiente è:

```
\begin{lucediffusa}
\row{999999 1}{0.20}{\pass}
\row{999999 2}{0.12}{\pass}
\end{lucediffusa}
```

Il collegamento tra il codice SGML e il codice LaTeX è fornito da un file detto di *mappatura* (oppure anche di rimpiazzo o sostituzione) scritto nello storico formato ASP, *Amsterdam SGML Parser*, Warmer e Van Egmond (1989):

```
<lucediffusa> + "\\begin{lucediffusa}" +
</lucediffusa> + "\\end{lucediffusa}" +
<LDrecord> +  "\\row{[CAMPIONE]}
{[MISURA]}{[ESITO]}" +
</ldrecord> +
```

Si osservi come gli attributi dell'elemento `<LDrecord>` vengono passati al comando `row` definito entro l'ambiente `lucediffusa`.

La mappatura da SGML a LaTeX può essere eseguita mediante `sgmlsaps` sul file generato da un parser validante, come l'`nsgmls` di James Clark, applicato al rapporto di prova steso in SGML:

```
cat RAPPORTO.sgml | nsgmls -c CATALOG |
 sgmlsasp MAP > RAPPORTO.tex
```

dove il file di catalogo solitamente contiene dichiarazioni che servono per reperire DTD, entità pubbliche e locali (ad esempio l'entità `&pass;`) che si vogliono rendere disponibili al software di elaborazione del file SGML e all'operatore all'atto della stesura SGML del rapporto di prova, ad esempio:

```
PUBLIC "ISO 8879:1986//ENTITIES
 Added Latin 1//EN" "ISOlat1.tex"
PUBLIC "ISO 8879:1986//ENTITIES
 Diacritical Marks//EN" "ISOdia.tex"
PUBLIC "ISO 8879:1986//ENTITIES Numeric
 and Special Graphic//EN" "ISOnum.tex"
PUBLIC "-//IstitutoXXX//ENTITIES Names,
 Standards, Abbreviations, etc.//IT"
 "ISTXXXentity.tex"
PUBLIC "-//IstitutoXXX//DTD Reports and
 Other Documents//IT" "istitutoxxx.dtd"
DOCTYPE "istitutoxxx" "istitutoxxx.dtd"
```

dove il file `ISTXXXentity.tex` contiene le entità specifiche dell'ente che stende il rapporto, ad esempio:

```
<!-- entita' LaTeX -->
<!ENTITY barrainversa CDATA
 "\small{\tt\char`\\}\normalsize">
...
<!-- entita' relative agli esiti  -->
<!ENTITY pass CDATA "\pass">
<!ENTITY fail CDATA "\fail">
...
```

Un altro sistema di mappatura, che qui però ci limitiamo solo a citare per la sua rilevanza, è basato su `sgmlspl`:

```
nsgmls RAPPORTO.sgml |
sgmlspl SPEC.pl > RAPPORTO.tex
```

dove al posto del file di mappatura viene utilizzato un file contenente delle specifiche di conversione basate su eventi e scritto ad esempio in Perl. Questo meccanismo di conversione è più versatile rispetto a quello basato su `sgmlsasp` ed è ritenuto anche più semplice da alcuni autori (MacGrath (1998)), tuttavia richiede specifiche competenze di programmazione che contrastano con la sintassi intuitività del file di mappatura.

A questo punto abbiamo un sistema di marcatura che separa la struttura dell'informazione che vogliamo trasmettere dalla forma stilistica che vogliamo invece preservare, rendendola inaccessibile per quanto possibile all'operatore che stende il rapporto. Inoltre la struttura informativa imposta con il DTD obbliga l'operatore a rispettare un prefissato pseudo-ordine nella stesura e quindi ad inserire e completare le varie parti del rapporto solo dove sono previste pena l'invalidazione del documento SGML. La validazione con gli strumenti SGML (`nsgmls` nella fattispecie) ovviamente non evita tutti gli errori che possono essere prodotti durante una stesura, tuttavia li limita fortemente e questo in ambito lavorativo significa maggiore efficienza e riduzione dei costi.

La stesura direttamente in LaTeX sarebbe senza dubbio più semplice. Tuttavia saremmo privi di strumenti di validazione automatica del documento e dovremmo crearceli appositamente; inoltre all'atto della stesura manuale del documento ci priveremo dei meccanismi di riconoscimento automatico della struttura informativa del documento che sono presenti in alcuni software di videoscrittura (in primis tra tutti Emacs). Quello che è più rilevante in ambito lavorativo è che caricheremmo tutto il compito della validazione sulle spalle dell'operatore che stende il rapporto con conseguente aumento dei tempi di stesura e del volume di errori. Si tenga presente che per la stesura in SGML esistono software che riescono a leggere il DTD ed a crearsi una rappresentazione della struttura del documento mettendo a disposizione dell'operatore in qualsiasi punto del documento solo gli elementi (con i loro attributi) che possono validamente essere collocati in quel punto. La stesura diretta in LaTeX non ha simili benefici in quanto non si può costruire un parser sufficientemente generale da poter raffigurare la struttura di un complesso di *documentclass* e *package*. Chiunque abbia provato a stendere un documento in LaTeX si rende conto di questa situazione: i comandi e gli ambienti possono essere resi disponibili dal programa di videoscrittura quando questo è stato configurato per farlo, ma senza un minimo di conoscenza di come va steso un documento LaTeX alla prima compilazione si otterranno con ogni probabilità degli errori.

6 Stesura ed Inserimento dei Dati

I metodi per l'inserimento dei dati sperimentali possono essere dei più vari e molteplici (ossia più tecniche per il medesimo rapporto). I fattori che influenzano la decisione sono anch'essi vari ma in definitiva la scelta dipende da una valutazione pesata delle risorse materiali ed umane in gioco. Una norma con poche prove (se non una sola) ed applicata raramente non giustifica la creazione di software *ad hoc* per la stesura e meno che meno per l'inserimento automatico dei dati nel rapporto; sarà sufficiente un buon programma di videoscrittura che operi con l'SGML. Per contro una norma applicata con continuità e composta da una decina di prove necessità senz'altro di automatismi per l'inserimento dei dati e per la stesura. Quest'ultimo caso può essere gestito contemporaneamente per i due aspetti ossia conviene creare un software il più possibile efficiente (quindi privo di GUI) che recu-

peri i dati da un database e che generi il rapporto direttamente in SGML[6], ad esempio una cosa del tipo[7]:

```
void
sgml_fetch_lucediffusa (MYSQL *conn,
 MYSQL_RES *res_set) {
MYSQL_ROW       row;
unsigned int    i;
if ((unsigned long)
 mysql_num_rows (res_set) > 0){
printf("<lucediffusa>\n");
while ((row = mysql_fetch_row (res_set))
 != NULL) {
printf("<LDrecord campione='%s'
 ld='%1.2f' esito='%s'>\n", row[1],
 atof(row[2]), row[3]);
} // while
if (mysql_errno (conn) != 0)
print_error (conn, "<!-- lucediffusa
 mysql_fetch_row() failed -->");
printf("</lucediffusa>\n");
} // if
} // sgml_fetch_lucediffusa
```

Oppure può essere conveniente specie per prove che si ripresentano su più norme[8] creare un software comune per l'inserimento dei dati nel rapporto ad esempio una volta che questo è stato in parte già steso come [9]:

```
my $sth=$dbh->prepare("SELECT
 campione,ld,esito FROM lucediffusa
 WHERE rapporto='$rapporto'
 ORDER BY campione;");
$sth->execute();
my $lucediffusa = "\n";
while (my $ref=$sth->fetchrow_hashref()){
my $temp = $ref->{'campione'};
if ($infilestring !~ /LDrecord.*$temp/g){
 $lucediffusa .= "<LDrecord campione='".
 $ref->{'campione'}."'ld='".
 $ref->{'ld'}."' esito='".
 $ref->{'esito'}."'>\n";
}; # if
} # while
$sth->finish();
...
$infilestring =~ s/\<lucediffusa\>/
 \<lucediffusa\>$lucediffusa/g;
```

dove `$infilestring` è una stringa che include inizialmente l'intero rapporto parziale e che viene completata nelle ultime due righe del codice mostrato con l'inserimento di altro codice SGML completo di tutti i marcatori e dei dati registrati nel database relativamente alla prova considerata nell'esempio.

Gli esempi sopra sono solo due delle possibili tecniche di stesura ed inclusione dei dati. L'uso di `sgmlspl` combinato con codice C, Perl o Python può condurre a soluzioni anche più raffinate ed integrate (*cfr.* MacGrath (1998)).

Ovviamente la scelta dell'automazione richiede nel computo delle spese, oltre alla creazione del software di stesura, anche la creazione e gestione del database, nonostante quest'ultimo possa essere disponibile a priori grazie a politiche preesistenti di archiviazione digitale dei dati di prova.

7 Costi e Benefici

L'esame dei costi e dei benefici in qualsiasi attività umana ed in special modo quella lavorativa è importante per poter programmare le scelte future, per potersi migliorare. L'impiego di LaTeX per la stesura dei rapporti assieme ad una gestione strutturata dell'informazione non è esente da queste valutazioni.

L'impiego di un file di stile comune come il *documentclass* comporta un grosso impegno progettuale iniziale ma lo sforzo viene ripagato non appena viene richiesta una modifica allo stile di tutti i modelli[10] di rapporto di prova indipendentemente dalla norma applicata. Basta per fare un confronto pensare cosa comporterebbe la medesima operazione se i modelli fossero basati sul WYSIWYG: la stessa modifica su ognuno dei modelli. Se il numero di modelli impiegati è verosimilmente anche solo di svariate decine, se non dell'ordine delle centinaia, quell'unica modifica apportata al *documentclass* deve essere moltiplicata per svariate decine, se non aumentata di due ordini di grandezza. La cosa può non sembrare rilevante se la modifica riguarda una frase ma diventa proibitiva se comincia a riguardare ad esempio l'introduzione di un logo o la correzione dell'altezza di una intestazione di pagina.

L'implementazione di una nuova norma in termini redazionali ha dei costi. Il DTD deve essere modificato e le modifiche sono tanto più pesanti quanto più la struttura della norma è differente da quelle già implementate. Norme simili hanno prove simili e quindi le modifiche al DTD possono essere veramente minime; se la norma è profondamente diversa, e quindi con prove del tutto nuove o collegate tra loro in modo nuovo, allora come caso limite potrebbe essere necessario stendere un DTD apposito, satellite di quello principale. In un caso o nell'altro per il DTD l'effetto scala è senz'altro rilevante e quindi la sua adozione ha una conseguenza positiva per i costi.

6. Non direttamente in LaTeX, per i motivi citati in precedenza.
7. Il codice è C.
8. Caso piuttosto frequente quando le norme applicate di distinguono tra loro più nell'uso dell'oggetto sottoposto a prova che nella natura dell'oggetto stesso (es. indumenti protettivi). Le norme in questo caso possono risultare piuttosto simili tra loro presentando nelle prove comuni solo differenze di metodo o sul limite normativo ammesso per la medesima grandezza fisica misurata.
9. Il codice è Perl.
10. Il termine *modello* in questo contesto è da intendersi come variante associata ad esempio ad una norma.

Una nuova norma richiede generalmente uno o più nuovi *package* ed un nuovo file di mappatura. Se la norma viene applicata molto spesso allora la stesura dovrà essere automatizzata con la creazione di un software apposito di tipo compilato od interpretato a seconda dei casi. L'effetto scala può essere solo parziale se non marginale però i costi di implementazione verranno ripagati rapidamente grazie all'automazione.

8 Considerazioni Finali

Terminiamo l'articolo con una serie di considerazioni che possono contribuire al quadro complessivo circa l'impiego di LaTeX per la stesura dei rapporti di prova.

L'impiego di SGML sembra una scelta obsoleta in quanto l'XML (*Extended Markup Language*), nato come un sottoinsieme di regole dell'SGML, lo ha ampiamente sostituito nelle applicazioni editoriali mentre è principalmente adatto alle applicazioni internet assieme all'HTML. L'HTML nato indipendentemente dall'SGML ha poi subito una iniziale naturale razionalizzazione proprio come applicazione SGML[11] (si confronti anche Goossens e Rahtz (1999)). XML fu concepito per ridurre la complessità associata all'estrema versatilità delle regole di SGML[12], però questo ebbe un risvolto notevole nell'uso editoriale diretto dei linguaggi di marcatura: l'XML non è adatto alla videoscrittura diretta in quanto irrimediabilmente troppo prolisso e per rendersene conto basta solo tentare di scrivere una formula di media complessità usando direttamente MathML. Tale prolissità si estende naturalmente a tutti quegli strumenti di traslazione dell'XML per eventuali applicazioni tipografiche, quali XSL ed XSLT, visto che sono a loro volta delle applicazioni XML come lo è una proposta alternativa al DTD ossia XML-Schema. Tutto ciò è ovvio se si pensa che l'XML fu concepito per essere gestito ed elaborato da un software e non per essere usato direttamente da un umano. SGML fu progettato proprio allo scopo di essere utilizzato come strumento editoriale diretto e quindi ha una serie di caratteristiche che possono rendere la marcatura piuttosto sintetica mantenendo un basso rapporto tra quest'ultima e testo informativo. L'esempio di codice SGML adottato lungo tutto l'articolo non rende giustizia alle peculiari caratteristiche editoriali di questo linguaggio.

L'importanza di LaTeX nella stesura di rapporti di prova può sembrare ridimensionata dall'analisi condotta in questo articolo. Basti pensare che nulla vieta di optare per una mappatura dei rapporti da SGML addirittura al classico RTF. Ovviamente chi conosce un minimo le potenzialità degli strumenti tipografici di LaTeX non avrà dubbi sulla scelta.

I rapporti secondo specifiche tecniche di prova concordate con il cliente oppure relativi a prove definite su disciplinari tecnici di settore possono essere impossibili da strutturare fino al dettaglio delle prove[13] come invece è possibile per una norma armonizzata. In tal caso si potrebbe pensare che la stesura del rapporto possa essere effettuata direttamente in LaTeX tuttavia è importante anche in questi casi mantenere un minimo di marcatura e quindi di possibilità di validazione del documento per lo meno per la parte delle informazioni standard come il numero identificativo del rapporto, le varie date, ecc. che possono essere comunque importanti in ambito lavorativo per ragioni di archiviazione, recupero ed identificazione del rapporto. In questo caso la parte non codificata esplicitamente nel DTD dovrà essere necessariamente stesa direttamente in LaTeX. Questo ovviamente porrebbe un problema al parser che viene invece evitato in SGML con il concetto di *notazione* nel DTD:

```
<!ELEMENT latex - - CDATA>
<!ATTLIST latex format NOTATION
 (LaTeX) "LaTeX">
```

In questo modo tutto quello che verrà compreso entro i tag <latex> e </latex> non verrà esaminato dal parser ma verrà passato inalterato durante la mappatura al programma che auspicabilmente sarà in grado di interpretarlo ossia il questo caso LaTeX. Si tenga presente però che l'impiego delle NOTATION può essere un problema sia per l'impossibilità di estendere la validazione al testo costituente la notazione, sia per il buon esito a priori della compilazione del codice LaTeX immesso direttamente. L'uso delle notazioni comporta quindi un costo sia in termini di tempo di verifica del documento finale, sia in termini di addestramento degli operatori che dovranno in questo caso conoscere almeno i rudimenti di LaTeX.

I grafici meritano una menzione specifica in quanto nulla vieta di includerli come immagini analogamente alle fotografie dei campioni. Tuttavia il vantaggio di operare con LaTeX è che esso spalanca le porte a molti strumenti per la creazione di grafici direttamente mediante codice LaTeX. Ad esempio è possibile generare un grafico usando i comandi resi disponibili nell'ambiente `pspicture` dai *package* `pstricks` e `pst-plot`, il codice potrebbe

11. Il nuovo standard HTML5 ha scisso il legame con SGML stabilendo che un DTD, seppure utilizzabile, non è più un sistema validante sufficiente per verificarne la sintassi di HTML5. In pratica nonostante la somiglianza nell'espressione dei marcatori, l'HTML5 è un linguaggio con regole di *parsing* differenti da quelle di SGML e quindi da quelle delle precedenti versioni di HTML.

12. Basti pensare che SGML permette anche la ridefinizione dei simboli di apertura e chiusura, "<" e ">", dei *tag*.

13. I disciplinari possono manifestare un periodo di aggiornamento piuttosto breve oppure essere soggetti a frequenti emendamenti.

essere generato da un programma che esegue il *parsing* di un file di dati in formato CSV. Ovviamente LaTeX è ideale anche per generare durante la compilazione disegni o immagini stilizzate che possono servire per meglio spiegare i risultati delle prove, ad esempio la posizione di impatto di un proiettile su un visore protettivo. Nel DTD risulta comodo descrive il grafico o il disegno stilizzato ad esempio come

```
<!ELEMENT grafico - - EMPTY >
<!ATTLIST grafico file cdata #required >
```

che verrà utilizzato ad esempio come:

```
<grafico file='graficomio.tex'>
```

che verrà mappato

```
<grafico> + "\\grafico\[tex\]{[FILE]}" +
</grafico> +
```

dove il comando `grafico` nel *documentclass* può essere per altro concepito per un uso più generale:

```
\newcommand{\grafico}[2][]{%
\ifthenelse{\equal{#1}{tex}}{
\input{#2}
}{%
\begin{figure}
\begin{minipage}{5.0cm}
\includegraphics{#1}
\end{minipage}
\end{figure}
}% ifthenelse
}% grafico
```

dove se il primo argomento opzionale è 'tex' allora viene importato un file in formato LaTeX altrimenti in un altro formato processabile (ad es. JPG se la compilazione è condotta con pdfLaTeX). Si tenga presente che il file di mappatura non è unico ma generalmente associato ad una norma applicata e che il motore di compilazione può quindi essere variato (es. LaTeX nel caso i grafici e le fotografie siano in formato EPS). Infine si noti che non è stato necessario ricorrere alle NOTATION nel DTD in quanto l'inclusione del codice LaTeX avviene in questo caso addirittura dopo la mappatura e cioè durante la compilazione.

Il controllo delle revisioni è un meccanismo importante per gestire lo storico delle modifiche apportate ai file chiave utilizzati per la stesura dei rapporti come sicuramente il DTD, i file di mappatura oltre ovviamente al *documentclass* ed ai *package*, ma anche i file delle entità locali. Pensare di estendere il controllo di revisione anche ai singoli rapporti di prova può essere un vantaggio specie nel caso in cui il rapporto sia compilato da più persone in tempi distinti, come ad esempio nel caso in cui il rapporto includa prove che coinvolgono più laboratori.

Talvolta si rende necessaria una modifica al volo al testo descrittivo della prova ad esempio per evidenziare una certa caratteristica di un campione provato che può influire sull'esito di una prova. In tal caso la modifica può essere eseguita inserendo un blocco <latex>...</latex> nel rapporto in formato SGML dove verrà incluso il codice di ridefinizione dell'ambiente LaTeX che definisce la prova evitando in questo modo di modificare il *package*. Oppure l'alternativa brutale, ma più veloce e parimenti efficace, è quella di modificare (temporaneamente!) la copia locale del *package*. In quest'ultimo caso se il *package* è sottoposto a controllo di revisione, ad esempio tramite CVS (*Concurrent Versions System*), è possibile "agganciare" la modifica apportata al *package* alla versione SGML o LaTeX del rapporto in formato di commento. La cosa è fattibile anche per il formato PostScript del rapporto sia che questo sia compilato con LaTeX , sia quando ottenuto per conversione dal formato PDF in seguito ad una compilazione con pdfLaTeX[14]:

```
cvs diff -y --suppress-common-lines \
  norma.sty | sed -s "s/^./%\ &/g" \
  >> rapporto.ps
```

9 Conclusioni

L'articolo ha offerto un esempio di quale possa essere il ruolo di LaTeX in ambito lavorativo nella stesura di rapporti di prova per una tipologia particolare di norme armonizzate quali sono quelle ad elevato contenuto di prove. In questo particolare tipo di applicazione il LaTeX dovrebbe essere utilizzato solamente per il compito per il quale è più adatto ossia quello tipografico mentre per la gestione organica dell'informazione, e quindi per la sua struttura, è opportuno ricorrere ad altri strumenti informatici quali l'SGML. In definitiva LaTeX garantisce il conseguimento di obbiettivi di scalabilità all'aumentare del numero di norme e quindi di prove implementate, razionalizzazione e riduzione dei costi nella gestione dei formati stilistici dei rapporti di prova e dei loro contenuti.

Riferimenti bibliografici

Goldfarb, C. (1990). *The SGML Handbook*. Oxford University Press.

Goossens, M. e Rahtz, S. (1999). *The LaTeX Web Companion: integrating TeX, HTML, and XML*. Addison-Wesley Longman.

MacGrath, S. (1998). *PARSEME.1ST: SGML for Software Developers*. Prentice Hall.

Warmer, J. e Van Egmond, S. (1989). «The implementation of the Amsterdam SGML Parser». *Electronic Publishing*, **2** (2), pp. 65–90.

▷ Renato Battistin
Fermo Posta - 31041 CORNUDA TV
rbattistin at apf dot it

14. Comandi dalla *shell* Bash.

Introduzione a Bibfilex. Un software libero per gestire dati bibliografici in formato Biblatex

Massimo Nardello

Sommario

L'articolo si propone di presentare Bibfilex, un software libero e multipiattaforma utile a gestire archivi bibliografici in formato Biblatex pensato particolarmente per chi usa LaTeX nell'ambito umanistico. Dopo averne indicato la caratteristiche fondamentali e le principali funzionalità, ci si sofferma sul sistema di composizione delle citazioni. Questo consente di sostituire le chiavi presenti in un file in LaTeX con le relative citazioni complete, rendendo non più necessario l'impiego di BibTex o di Biber, e quindi agevolando enormemente la sua conversione in formato Word o Open Document.

Abstract

This paper describes Bibfilex, a free multiplatform software suitable to manage bibliographic databases to be processed with Biblatex; this software is intended mainly for use in the humanities. We first describe the main characteristics and the software functionalities, then we discuss in detail on the particular way of creating citations. This approach allows to replace the keys present in a LaTeX file with the complete citations, so that it is not necessary any more to use the bibliographic programs BibTex and Biber. By so doing it becomes very easy to convert source LaTeX documents to the Open Document format.

1 Un altro *bibliographic manager*?

Già da tempo sono a disposizione degli utenti di LaTeX diversi software, sia liberi che proprietari[1], che consentono di gestire degli archivi bibliografici in formato BibTex e/o Biblatex.[2] Ciascuno di essi ha caratteristiche un po' differenti, ma fondamentalmente tutti mettono a disposizione degli utenti un ampio numero di campi nei quali inserire le informazioni bibliografiche, ricerche e filtri per reperire il materiale desiderato, funzionalità di importazione di informazioni da banche dati pubbliche, come PubMed[3], o dal web, e infine la possibilità di esportare i dati in diversi formati. Alcuni di questi programmi, poi, consentono di sincronizzare la propria base dati su diversi dispositivi attraverso un servizio cloud. In questo modo si possono consultare e modificare con un browser[4], ma pure con uno smartphone o un tablet.[5]

Taluni di questi software consentono anche di inserire automaticamente le citazioni del materiale catalogato in un *word processor*, come Word o Writer.[6] Altri sono pensati per aiutare chi elabora i propri documenti in LaTeX e si serve di BibTeX o Biber per le citazioni, e lavorano direttamente su file di testo in cui le informazioni bibliografiche sono già strutturate nel formato corretto.[7] In ogni caso, anche i programmi pensati per essere utilizzati con un *word processor* sono in grado di esportare i dati in formato BibTeX o Biblatex, sebbene talora con qualche criticità, per cui possono essere utilizzati fruttuosamente anche da utenti di LaTeX.

In questo quadro caratterizzato da un'offerta così ampia ed eccellente ci si può domandare se e perché un utente possa necessitare di un software per la gestione bibliografica che sia diverso da quelli attualmente disponibili. Come autore di Bibfilex, ovviamente, mi sono posto a lungo questa domanda prima di cimentarmi nel faticoso lavoro della sua scrittura. Sono arrivato alla conclusione che lo strumento di cui avevo bisogno non era ancora disponibile, e che quindi sarebbe stata una buona idea cercare di realizzarlo.[8] La motivazione che mi ha orientato a procedere in questa direzione è la necessità di poter disporre di un software che avesse *tutte* le seguenti caratteristiche, quelle che poi ho cercato di implementare in Bibfilex, e che non ho reperito nei prodotti già disponibili.

1.1 Un software libero

In primo luogo, il software che corrisponde alle mie attese deve essere libero, cioè deve essere distribuito con una licenza d'uso che consenta a chiunque di utilizzarlo, di condividerlo, eventualmente di migliorarlo e di ridistribuirlo con le proprie modifiche, sempre alle condizioni previste dalla stessa licen-

1. Uso i due termini nel senso inteso da R. Stallman e dalla *Free Software Foundation*: cf. https://www.gnu.org/philosophy/free-sw.en.html.
2. La pagina di Wikipedia sui software di *reference management* ne menziona più di 30, e diversi di essi supportano il formato BibTex e/o Biblatex: http://en.wikipedia.org/wiki/Comparison_of_reference_management_software.
3. Cf. http://www.ncbi.nlm.nih.gov/pubmed.
4. Cf. ad es. Zotero: https://www.zotero.org.
5. Cf. ad es. Mendeley: http://www.mendeley.com.
6. Come, ad esempio, i citati Zotero e Mendeley.
7. Cf. ad es. JabRef e KBibTex: http://jabref.sourceforge.net, http://home.gna.org/kbibtex.
8. Bibfilex è liberamente disponibile nel sito https://sites.google.com/site/bibfilex, dove è possibile scaricare anche il manuale in inglese.

za.[9] La ragione di questa preferenza è anzitutto di tipo etico: ritengo infatti che il software non sia un prodotto materiale, ma piuttosto conoscenza formalizzata, e poiché esso rappresenta un bene fondamentale per la vita delle società odierne, è auspicabile che esso sia liberamente e completamente accessibile a tutti, anche a livello del codice sorgente, al pari delle altre forme di conoscenza (la matematica, la fisica, la filosofia, la teologia, la musica, ecc.). Inoltre la possibilità di poter visionare il codice sorgente di un software libero consente di verificare che non vi siano al suo interno funzionalità svantaggiose per l'utente, come ad esempio quella di inviare a terzi i suoi dati personali. Infine, se un software è libero può essere migliorato e condiviso, e questo lo rende strumento di collaborazione e di aiuto reciproco. È evidente come sia importante anche oggi promuovere nella società dinamiche virtuose di questo genere. Bibfilex è rilasciato con una licenza d'uso GNU Public License (GPL) versione 3[10], che lo rende software libero a tutti gli effetti. Anche il compilatore e l'ambiente di sviluppo utilizzati per crearlo sono software libero.[11]

1.2 Un software che non obbliga a condividere i propri dati

Il software deve poi rispettare la proprietà dei dati dell'utente, non obbligandolo a condividere in un archivio pubblico le proprie considerazioni personali su un'opera. Alcuni software, al contrario, richiedono che chi li utilizza sia disponibile a veder pubblicati on line i propri abstract e le proprie recensioni delle opere che ha catalogato, seppure in modo anonimo, con il risultato che valutazioni soggettive – talora temporanee e soggette a cambiamento – diventano improvvisamente di dominio pubblico. Bibfilex archivia i dati su un file che risiede nel computer dell'utente, e anche se questo può ovviamente essere caricato in un servizio cloud, come Dropbox, di per sé questo non è richiesto per il funzionamento del programma o per poter disporre di tutte le sue funzionalità. Insomma, l'utente resta l'unico proprietario del materiale che produce, e lo condivide solo se lo vuole.

1.3 Un software rigorosamente aderente alla struttura dei dati di Biblatex

Il software in esame deve ricalcare esattamente la struttura dei dati prevista dal manuale di Biblatex[12], per poter essere utilizzato con LaTeX e BibTex o Biber senza la minima incompatibilità. Alcuni programmi esportano i loro dati in formati molto simili, ma non perfettamente aderenti a quanto previsto dal suddetto manuale, cosa che comporta talora delle difficoltà durante la compilazione delle citazioni. Del resto, in molti software il formato Biblatex non è quello nativo, per cui l'esportazione dei dati dal formato originale a quello di destinazione può comportare dei compromessi, che diventano criticità quando si utilizzano i file esportati con BibTex o con Biber. Bibfilex, al contrario, utilizza una struttura dei dati aderente a quanto indicato nel manuale di Biblatex, fatta eccezione per alcuni campi opzionali.

1.4 Un software reattivo

Il software deve essere molto veloce, sia in fase di partenza che durante l'esecuzione delle varie funzionalità, e richiedere poca memoria, per poter essere utilizzato agevolmente anche in computer non recenti. Deve poi essere reattivo anche in presenza di una base dati molto consistente. Questo porta all'esclusione di software realizzati con linguaggi di scripting o interpretati[13], come Python o JavaScript, ma anche con Java, visto che esso comporta la necessità di utilizzare una macchina virtuale, e questa a sua volta comporta un notevole uso della memoria e una maggiore lentezza operativa. Bibfilex è scritto in Free Pascal[14], ed è quindi un software il cui codice è compilato e viene eseguito direttamente dal sistema operativo in uso. Questo gli consente di avere le alte prestazioni che sono richieste.

1.5 Un software multipiattaforma

Il software deve essere disponibile per GNU/Linux, sia con le librerie grafiche Gtk che Qt, per Windows e per Apple OS X. Esso però deve funzionare in modo nativo su ciascuna piattaforma, cioè sfruttare le librerie grafiche esistenti e non richiederne l'installazione di altre. In questo modo l'utente si trova di fronte ad un programma che ha un *look* consistente con la piattaforma che utilizza e le cui componenti funzionano secondo gli standard a cui è abituato. Bibfilex è scritto con Lazarus[15], un ambiente di sviluppo per Free Pascal che consente di generare codice sorgente multipiattaforma e di compilarlo in modo nativo per ciascun sistema operativo senza bisogno di modifiche.

1.6 Un software semplice

Il software deve essere il più semplice possibile, anche a costo di disporre di meno funzionalità. Esistono già programmi molto completi che hanno tantissime opzioni, ma il loro utilizzo può diventare troppo complesso per chi non ha buone competenze informatiche. Nella scrittura di Bibfilex, ho cercato di creare un'interfaccia utente molto intuitiva, in

9. Cf. http://www.gnu.org/philosophy/free-sw.en.html.
10. Cf. http://www.gnu.org/copyleft/gpl.html.
11. Si tratta di Free Pascal e di Lazarus, rilasciati con una licenza GPL e LGPL modificata: cf. http://wiki.lazarus.freepascal.org/licensing.
12. Cf. http://mirrors.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf.
13. Si tratta di linguaggi di programmazione che generano un codice che deve essere tradotto da un interprete immediatamente prima di essere eseguito dal computer.
14. Cf. http://www.freepascal.org
15. Cf. http://lazarus.freepascal.org.

modo da consentirne un uso agevole anche da parte di utenti non esperti.

1.7 Un software che dispone di un sistema di compilazione delle citazioni alternativo a BibTex e a Biber

La maggior parte delle editrici in ambito umanistico richiede che i contributi che sono loro inviati per la pubblicazione siano in formato Word o Open Document, quello utilizzato da LibreOffice e da OpenOffice. Purtroppo, a mio giudizio, non esistono a tutt'oggi dei software in grado di convertire correttamente in tali formati i documenti in LaTeX che contengono delle chiavi BibTex. Questa limitazione rischia purtroppo di disincentivare coloro che studiano e insegnano discipline umanistiche all'uso di LaTeX. Il software deve allora consentire la sostituzione delle chiavi con le relative citazioni complete, in modo da poter ottenere un file in LaTeX che, non necessitando più di BibTex o di Biber, può essere tradotto molto più facilmente in Word o in Open Document con programmi come TeX4ht o Pandoc.

Bibfilex consente proprio di operare questa sostituzione. Le citazioni sono costruite a partire da un modello definito dall'utente all'interno del programma, che può essere caricato o salvato come file XML, e quindi scambiato con altri. Il software poi mette e disposizione alcune opzioni di citazione che non sarebbero realizzabili con la modellazione possibile all'utente (citazioni abbreviate, collocazione del curatore al posto dell'autore se questo manca, ecc.). Anche se questo sistema di citazione risulta di gran lunga meno duttile di un foglio stile in Biblatex, sembra l'opzione più vantaggiosa per poter risolvere il problema di cui sopra.

Esso, poi, conferisce al programma un ulteriore vantaggio. Non di rado tra le università e le editrici vi sono piccole ma fastidiose differenze metodologiche che impongono agli studenti o agli autori di modificare la struttura delle citazioni prevista da un foglio stile di BibLatex. Nonostante molti di essi possano essere ampiamente personalizzati, tale duttilità potrebbe non essere sufficiente a fronte di richieste particolari. A titolo di esempio, possiamo citare la modifica o l'aggiunta di elementi testuali all'interno della citazione (virgole, virgolette, ecc.) o una disposizione originale delle opere in bibliografia (particolari distinzioni tra fonti e studi, ecc.).

L'approccio utilizzato da Bibfilex consente di lasciare all'utente la possibilità di modificare molti aspetti delle citazioni direttamente all'interno del programma, per cui a fronte di particolari richieste anche chi non è esperto può facilmente apportare i cambiamenti necessari per ottenere delle citazioni congruenti con le indicazioni ricevute. Inoltre, ottenendo un file in LaTeX con le citazioni complete, sia nel testo che nella bibliografia, l'utente può facilmente apportare piccole correzioni o riorganizzare il materiale bibliografico in modo da rispondere esattamente alle richieste tipografiche e metodologiche. Quindi può compilare il testo in LaTeX, ottenendo il consueto pdf, oppure, se necessario, convertirlo facilmente in Word o in Open Document.

Evidentemente questo approccio, il cui funzionamento verrà dettagliato in seguito, non ha la duttilità di un foglio stile di Biblatex, e resta una soluzione auspicabilmente temporanea per chi si trova in difficoltà con il sistema tradizionale di compilazione delle citazioni.

2 Le principali funzionalità di Bibfilex

Dopo aver indicato le principali caratteristiche di Bibfilex, che sintetizzano anche le motivazioni che hanno portato alla sua scrittura, prendiamo ora in considerazione le sue principali funzionalità.

2.1 Il formato dei dati

Bibfilex non gestisce direttamente i file di testo contenenti i dati in formato Biblatex (quelli, cioè, che solitamente hanno l'estensione ".bib"), perché in caso di archivi molto consistenti questo approccio comporterebbe una diminuzione delle prestazioni, oppure un ampia occupazione della memoria del computer. Al contrario, il software archivia i dati in un file di Sqlite,[16] un database di pubblico dominio ampiamente utilizzato anche da parte di grandi aziende. L'uso di questo database consente di archiviare anche quantità molto consistenti di dati, nell'ordine delle centinaia di migliaia di elementi, mantenendo comunque una notevole reattività nelle operazioni di modifica e di ricerca.

In questo modo, però, un file di Bibfilex non può essere referenziato all'interno di un documento in LaTeX. È dunque possibile esportare il suo contenuto in un file di testo correttamente formattato secondo le indicazioni di Biblatex. Questa esportazione può anche avvenire in automatico all'uscita del programma. Viene così creato o aggiornato un file con lo stesso nome di quello in uso che può essere effettivamente referenziato all'interno di un documento in LaTeX. Questo file può anche essere importato in altri software, come JabRef o, fatta eccezione per gli eventuali allegati, Zotero e Mendeley.

Ovviamente Bibfilex può importare file di Biblatex, fatta eccezione per gli eventuali allegati, come quelli creati con JabRef o KBibTex, o esportati in formato Biblatex con Mendeley o Zotero, come pure le citazioni in formato BibTex disponibili in Google Book.

Bibfilex archivia i dati secondo il formato di Biblatex, ma la disposizione dei campi è stata lievemente modificata. Sono messi in evidenza non

16. Cf. http://www.sqlite.org.

solo quelli richiesti, ma anche altri campi che normalmente devono essere compilati nell'ambito umanistico, come il luogo di edizione o la casa editrice. Sono pure a disposizione un campo per l'abstract ("abstract") e uno per le recensioni ("review"), nei quali è possibile introdurre un testo libero a piacere che però non può essere formattato.

2.2 La gestione della chiavi BibTex e della copia in ritaglio

Il software consente di creare le chiavi BibTex secondo un modello definito dall'utente nelle opzioni del programma, come pure di introdurle manualmente o di modificare quelle esistenti. In ogni caso, il software verifica che la chiave immessa sia unica nel file in uso, e nel caso non lo sia, la rende tale aggiungendo una lineetta e una lettera progressiva al termine di essa. È anche disponibile una *stop list*, con cui si possono escludere alcune parole che si possono trovare all'inizio del titolo nella creazione della chiave.

È poi possibile copiare in ritaglio la chiave della scheda bibliografica corrente o di quelle selezionate, in modo da poterle incollare all'interno di un testo in LaTeX, con o senza il comando `\cite` e l'eventuale *postnote*. È anche possibile copiare la citazione completa della scheda corrente o di quelle selezionate: tale citazione viene composta a partire dal modello definito nelle opzioni del programma, di cui si parlerà in seguito.

La copia in ritaglio della citazione completa può avvenire in due formati differenti. Il primo è quello testuale, in cui vengono mantenuti i comandi LaTeX di formattazione del testo (corsivo, maiuscoletto, ecc.) presenti nel modello di citazione definito nelle opzioni; questo formato consente di incollare la citazione all'interno di un documento in LaTeX. Il formato HTML, invece, traduce i comandi di formattazione in codici HTML, per cui la citazione può essere incollata in un *word processor* come Word o Writer mantenendo tutte le sue caratteristiche di formattazione. Questa funzionalità rende Bibfilex utilizzabile anche da chi non compone i propri testi con LaTeX, ma con un elaboratore di testi.

2.3 La selezione dei dati bibliografici

È possibile filtrare i dati bibliografici in modi diversi. Nell'interfaccia principale del software vi è la possibilità di eseguire rapidamente un filtro su un singolo campo, scelto tra quelli più comunemente utilizzati, o sulle parole chiave. È poi a disposizione una maschera apposita nella quale si possono eseguire dei filtri incrociati, al massimo su tre campi. Le condizioni di selezione che vengono create dal filtro, in formato SQL, sono visibili all'utente, e se questi ha le competenze necessarie può modificarle manualmente, e creare così filtri molto più complessi su un qualsiasi numero di campi.

Un'altra funzionalità molto utile è quella che consente di filtrare nel file in uso le schede la cui chiave BibTex è contenuta in un documento in LaTeX. In questo modo è possibile selezionare solo le opere effettivamente citate nel testo, per poi esportarle in un file di Biblatex e costituirne un archivio a parte, da tenere insieme al documento.

2.4 L'immissione dei dati

Bibfilex agevola anche l'immissione dei dati da parte dell'utente. È disponibile una maschera che consente di immettere con facilità i caratteri non presenti nella propria tastiera, o selezionare le parole chiave da inserire nella scheda corrente all'interno di una lista di quelle già immesse. In ogni campo è poi possibile attivare l'autocomposizione: basta scrivere almeno un carattere e poi premere "Ctrl + barra spaziatrice" in un qualsiasi campo perché esso venga completato con la prima voce tra quelle già immesse che inizia con i caratteri inseriti. Nel caso il completamento non fosse quello desiderato, basta continuare la scrittura e rieseguire l'autocompletamento dopo che altri caratteri sono stati inseriti, fino a che il campo non viene completato con la voce desiderata. Nel campo relativo alle parole chiave l'autocompletamento viene effettuato non sul suo intero contenuto, ma su ogni singola parola, mentre quello "crossref" viene completato con le chiavi BibTex presenti nell'archivio.

In quest'ultimo campo è possibile immettere la chiave di un'altra scheda bibliografica da cui dipende quella corrente (ad esempio, una miscellanea e uno dei suoi contributi). Una volta immessa la chiave, il software consente di copiare nella scheda corrente (nell'esempio fatto, quella relativa al singolo contributo) il contenuto di tutti quei campi della scheda referenziata (quella relativa alla miscellanea) che sono previsti dal manuale di Biblatex[17], con alcune integrazioni. Questa opzione, ovviamente, non serve se si compilano le citazioni con Biber, ma è necessaria se si utilizza il sistema di citazioni di Bibfilex.

2.5 Gli allegati

Bibfilex consente anche di allegare a ciascuna scheda bibliografica un numero a piacere di allegati, cioè file di qualsiasi genere, aventi però nomi diversi all'interno della medesima scheda. Quando si allega un file, questo viene compresso in formato zip e copiato in una cartella collocata nella stessa posizione in cui si trova il file in uso e avente il suo stesso nome. Se non è presente, la cartella viene automaticamente creata dal software, e viene rimossa qualora l'ultimo allegato al suo interno venisse cancellato. Poiché gli allegati non risiedono nei file di Bibfilex ma sono esterni ad essi, il loro

17. Cf. P. Lehman, *The Biblatex Package. Programmable Bibliographies and Citations*. Version 2.9a, 251–253, in http://ctan.mirror.garr.it/mirrors/CTAN/macros/latex/contrib/biblatex/doc/biblatex.pdf.

numero e le loro dimensioni sono limitati solo dallo spazio disponibile nel disco fisso del computer in uso.

2.6 I limiti di Bibfilex

Bibfilex ha anche molte limitazioni, se paragonato ad altri diffusi *bibliographic manager* disponibili oggi. Per menzionarne solo alcune, è solo in lingua inglese, non consente di eseguire ricerche e di importare automaticamente i dati bibliografici da portali come PubMed, né di annotare gli eventuali file in pdf che fossero allegati alle schede. Non è neppure possibile la modifica o l'integrazione dei nomi dei campi per la catalogazione, come l'importazione e l'esportazione in formati diversi da Biblatex. Anche se in futuro alcune di queste funzionalità potranno essere implementate, se se ne ha bisogno in modo non saltuario, Bibfilex non rappresenta il software più adeguato per le proprie esigenze.

3 La funzionalità di composizione delle citazioni

L'aspetto più originale di Bibfilex è la possibilità di elaborare citazioni complete a partire dalle chiavi BibTex secondo un modello definito dall'utente. A questo aspetto vogliamo ora volgere la nostra attenzione.

Aprendo le opzioni del programma e scegliendo la linguetta "Citations pattern", si vede una griglia simile a quella utilizzata nei fogli di calcolo. Sulla sinistra vi sono i nomi dei vari tipi di voce (*entry types*) previsti da Biblatex, e per ciascuno di essi vi sono tre righe corrispondenti ad altrettanti modelli di citazione.

- Il modello *normal* è quello che viene seguito da Bibfilex quando viene composta la citazione della scheda corrente o di quelle selezionate per copiarle in ritaglio, oppure nella procedura di sostituzione delle chiavi di BibTex con le citazioni complete all'interno di un file in LaTeX, solo però per la prima citazione che compare nel testo.

- Il modello *short* è quello che viene seguito da Bibfilex nella procedura di sostituzione delle chiavi BibTex per le citazioni che seguono la prima, quelle cioè che indicano le opere che sono già state citate almeno una volta nel testo in LaTex. In alcune norme metodologiche, infatti, queste citazioni devono essere in un formato ridotto, che può essere definito in questo modello. Ulteriori aggiustamenti possono essere fatti attivando alcune opzioni del sofware, e di questo si parlerà in seguito.

- Il modello *biblio* è quello che viene seguito da Bibfilex nella procedura di sostituzione delle chiavi di BibTex con le citazioni complete all'interno della bibliografia. Questa viene collocata al posto del comando `\printbibliography`.

Nella bibliografia, poi, le opere sono elencate in ordine alfabetico, e a tale scopo il cognome dell'autore viene posto prima del nome. Al contrario, nelle citazioni all'interno del testo il nome precede sempre il cognome. Il nome, poi, può essere puntato, a seconda dell'impostazione di un'opzione del software.

Nelle righe corrispondenti alle differenti voci (*entry types*) di Biblatex e ai tre modelli indicati è possibile inserire la struttura desiderata della citazione. È sufficiente compilare le varie celle inserendo in ciascuna di esse il nome del campo che si desidera in quella posizione tra due sbarrette verticali (si tratta del carattere |), eventualmente con altri elementi (virgole, punti, spazi, preposizioni, ecc.) che devono risultare nella citazione accanto a quel campo. È pure possibile inserire in una cella solamente elementi testuali, senza alcun campo. I nomi dei campi e gli altri elementi possono essere formattati in corsivo, in grassetto o in maiuscoletto utilizzando i comand LaTeX.

Ad esempio, per un articolo il modello *normal* e quello *biblio* potrebbero essere composti nel modo seguente:[18]

`{\textsc{|author|},␣`
`«|title|»`
`,␣in␣\textit{|journaltitle|}`
`␣|volume|`
`␣(|year|)`
`␣|number|`
`,␣|pages|`

Il modello *short* potrebbe invece essere composto nel modo seguente:

`{\textsc{|author|},␣`
`«|title|», art. cit.`
`,␣|pages|`

Il software compone le citazioni secondo queste regole:

- identifica nella griglia delle citazioni la riga corretta sia per il tipo di voce (*entry type*) che per il modello (*normal*, *short* o *biblio*), e quindi compone la citazione mettendo insieme il contenuto di tutte le celle della riga, da sinistra verso destra;

- se in una cella è presente il nome di un campo e se in quel campo della scheda corrente vi è effettivamente un testo, tutto il contenuto della cella verrà introdotto nella citazione.

18. Si noti che qui le celle sono disposte in verticale anziché in orizzontale, e che lo spazio è indicato dal carattere ␣.

- se in una cella è presente il nome di un campo ma se quel campo della scheda corrente è vuoto, tutto il contenuto della cella *non* verrà introdotto nella citazione;
- al contrario, se in una cella vi sono degli elementi testuali ma non è riportato nessun campo, il contenuto della cella verrà introdotto nella citazione in ogni caso.

In questo modo si possono formattare le citazioni in modo abbastanza aderente alle più svariate richieste metodologiche. Evidentemente, però, alcune peculiarità specifiche non possono essere ottenute con questo strumento. Per questo Bibfilex mette a disposizione dell'utente alcune ulteriori possibilità di formattazione delle citazioni che non sarebbero ottenibili inserendo campi ed elementi testuali nella griglia.

In primo luogo, è possibile indicare il carattere di separazione tra gli autori contenuti nei campi *author*, *bookauthor* e *editor*, nei quali devono essere separati dalla parola *and*. Solitamente, si usa la lineetta lunga (–).

È poi possibile indicare come Bibfilex debba gestire le citazioni in cui manca un autore e vi è invece un curatore. In questo caso occorre procedere nel modo seguente. Nei modelli, occorre inserire il curatore, preceduto o seguito da indicazioni del tipo "a cura di", nella collocazione in cui dovrebbe trovarsi nel caso in cui l'autore sia effettivamente indicato. È poi possibile attivare un'opzione del software per la quale, se l'autore manca, il curatore viene collocato al suo posto, ereditando la sua stessa formattazione (ad es. il maiuscoletto). Vi è anche la possibilità di specificare il testo che in questo caso specifico deve seguire il curatore, come ad esempio "(a cura di)".

È anche possibile specificare se il nome dell'autore debba essere puntato oppure se debba essere indicato per esteso.

Nel caso del modello *short* di citazione, si può poi indicare se abbreviare il titolo e/o l'autore. Nel primo caso, il titolo viene terminato all'eventuale punto che si trova al suo interno, oppure, in alternativa, al punto esclamativo, o a quello interrogativo. Nel secondo caso, viene citato soltanto il cognome dell'autore, senza introdurre alcun riferimento al suo nome.

In futuro sarà possibile introdurre nuove funzionalità nella procedura di composizione delle citazioni di Bibfilex che consentano di ottenere ulteriori formati di citazione, sempre che ciò non sia già possibile operando direttamente sulla composizione dei vari modelli.

Con il tempo, poi, sarà possibile costituire una banca dati di modelli di citazione, in modo che gli utenti meno competenti possano caricare quello più affine alla metodologia a cui devono attenersi senza preoccuparsi di costruirselo da soli. Nel sito di Bibfilex è già presente un file di questo genere, che contiene dei modelli di citazione per le voci *article*, *book* e in *inbook* con caratteristiche tipiche dell'ambito umanistico.

4 Conclusione

Bibfilex è stato scritto per venire incontro alle necessità di chi, come il sottoscritto, ricerca e pubblica nel campo umanistico, sebbene possa essere utilizzato fruttuosamente anche da utenti impegnati in altri ambiti del sapere. Questo software potrebbe contribuire a divulgare l'uso di LaTeX tra coloro che fino ad oggi sono stati costretti a non adottarlo perché non del tutto compatibile con le norme metodologiche a cui devono attenersi, o al formato dei file che devono consegnare. Il mio auspicio è che sempre più utenti che si trovano in queste condizioni siano spinti anche dallo strumento che ho prodotto a conoscere lo straordinario mondo di LaTeX e ad utilizzarlo al meglio per la propria attività.

▷ Massimo Nardello
Studio Teologico Interdiocesano
Reggio Emilia
massimo at msnardello.it

Funzionalità avanzate del sistema Biblatex/Biber

Ivan Valbusa

Sommario

In questo articolo si descrivono alcune funzioni poco note messe a disposizione dal pacchetto biblatex usato assieme al motore bibliografico Biber. È richiesta una conoscenza di base del funzionamento di questi programmi.

Abstract

This article describes some features available with the biblatex package when using Biber as the bibliographic engine. A basic knowledge of these programs is required.

1 Biber versus BibTEX

Nel 2010 Philipp Lehman ha rilasciato la prima versione stabile del pacchetto biblatex (Lehman, 2014) e da quel momento la composizione delle bibliografie con LaTeX ha intrapreso una nuova rotta.[1] Prima di allora non mancavano certo gli strumenti ma erano senza dubbio (troppo) poco elastici, in particolare per le esigenze degli utenti umanisti.

Il pacchetto di Lehman ha rivisto profondamente il modo di gestire la bibliografia e, in particolare, di modificare e creare stili bibliografici, attraverso un'interfaccia più intuitiva (usa direttamente le macro di LaTeX) rispetto alla complessa sintassi dei file di stile `.bst` (quelli, per intenderci, che si caricano con `\bibliographystyle`). Nell'arco di questi quattro anni sono nati parecchi stili bibliografici per biblatex e piano piano questo pacchetto è diventato il nuovo standard per la composizione delle bibliografie con LaTeX.

Ma biblatex da solo non è sufficiente per generare una bibliografia con le relative citazioni ed ha bisogno, esattamente come altri pacchetti (vedi il notissimo natbib), di un motore esterno che elabori gli archivi bibliografici e sovrintenda alla generazione delle etichette e all'ordinamento delle voci. Fino a poco tempo fa il motore bibliografico di riferimento era BibTeX, e con la sintassi prevista da questo motore si compongono ancora oggi molti archivi bibliografici. Con la diffusione di biblatex, soprattutto in ambito umanistico, sono però aumentate anche le esigenze e presto si è sentita la necessità di un nuovo motore bibliografico specificamente pensato per il potente pacchetto di Lehman. È nato così Biber (Kime e Charette, 2014), che è diventato ben presto il motore bibliografico predefinito di biblatex.[2] Un approccio alternativo è stato proposto già nel 2003 da Jean-Michel Hufflen con MlBibTeX, una reimplementazione di BibTeX con funzionalità multilingue, che è stato discusso in diversi contributi su ArsTEXnica e TUGboat. Si veda per esempio il recente Hufflen (2012).

A differenza del suo antesignano, Biber consente un controllo profondo dei dati contenuti all'interno degli archivi bibliografici: permette il *parsing* dei dati e la gestione avanzata dei riferimenti incrociati, di insiemi di voci (i cosiddetti *entry sets*) e di voci collegate (*related entries*). Inoltre permette di gestire in maniera elastica le liste di nomi (in particolare per i problemi di disambiguazione). Ha il supporto completo per la codifica Unicode. Prevede l'uso di opzioni *per-type* e molte altre funzioni. Nel seguito ne descriveremo alcune che riteniamo più significative e ancora poco note.

2 Il type `@xdata`

Alcune voci bibliografiche possono avere in comune uno o più dati. Per esempio, i volumi editi da una stessa casa editrice hanno i medesimi campi `publisher` e `location`. Con Biber si ha a disposizione lo speciale "tipo" di voce bibliografica `@xdata`, che funziona come un contenitore di dati che possono essere ereditati dalle altre voci attraverso l'apposito campo `xdata`.

Il sistema è molto utile, e la sua utilità aumenta con l'aumentare delle dimensioni dell'archivio bibliografico. In particolare permette di limitare al minimo i refusi, che sono sempre in agguato. Vediamo come funziona.

Innanzitutto creiamo una voce `@xdata` contenente solo due campi, nome dell'editore e luogo di edizione, e assegnamovi una chiave, esattamente come si fa con le altre voci standard:

```
@xdata{lat,
  Publisher = {Laterza},
  Location = {Roma-Bari}}
```

Dopodiché, nelle voci dello stesso editore, richiamiamo i dati attraverso il campo `xdata` che conterrà la chiave (in questo caso `lat`) assegnata alla voce `@xdata`:

1. Il pacchetto è ora mantenuto da Philip Kime, Audrey Boruvka e Joseph Wright.

2. Naturalmente, è ancora possibile usare BibTeX, passando al pacchetto l'opzione backend=bibtex.

```
@book{Cellucci:2008,
  Author = {Carlo Cellucci},
  Title = {PerchÃ© ancora la filosofia},
    xdata = {lat},
  Year = {2008}}

@book{casati:2012,
  Author = {Roberto Casati},
  Title = {Prima lezione di filosofia},
    xdata = {lat},
  Year = {2012}}
```

Il campo `xdata` può addirittura contenere una lista di chiavi che rimandano a più voci `@xdata`, come mostrato nella documentazione di biblatex:

```
@xdata{macmillan:name,
  publisher = {Macmillan},
}

@xdata{macmillan:place,
  location = {New York and London},
}

@xdata{macmillan,
  xdata = {macmillan:name,
                macmillan:place},
}

@book{...,
  author = {...},
  title = {...},
  date = {...},
xdata = {macmillan},
}
```

Il funzionamento di questo campo è simile a quello dei campi `crossref` e `xref`, con la differenza che non stabilisce alcuna relazione del tipo *parent/child*, limitandosi semplicemente ad ereditare i dati. In questo senso assomiglia forse più alla funzione delle voci `@string`:

```
@string{jams = {J.~Amer. Math. Soc.}}

@article{bertram,
  author = {Bertram, Aaron and
    Wentworth, Richard},
  title = {Gromov invariants for
    holomorphic maps on Riemann surfaces},
  journaltitle = {jams},
  date = 1996,
  volume = 9,
  number = 2,
  pages = {529-571},
}
```

Un approccio consigliabile a tutti potrebbe essere di creare delle voci `@xdata` per ogni editore che si incontra, assegnandovi come chiave semplicemente il nome dell'editore, eventualmente abbreviato. Per esempio: `mulino` (il Mulino), `olms` (Georg Olms Verlag), `bollati` (Bollati Boringhieri), ecc. Come vedremo nella sezione 9, adottando questo approccio si potranno ottenere risultati ancor più performanti.

3 Riferimenti incrociati

Il campo `crossref` consente di gestire in maniera ottimale situazioni tipiche nel campo umanistico. Per esempio record relativi a parti di opere in uno o più volumi, come le voci `@inbook`/`@incollection`, `@bookinbook` e `@suppbook`/`@suppcollection`.

In situazioni del genere abbiamo una voce *parent*, poniamo un libro che raccoglie le opere di un autore (voce `@book`), che avrà un proprio campo `title`, e una voce *child*, poniamo un articolo (voce `@inbook`), che a sua volta avrà un proprio campo `title`. Nel momento in cui la voce *child* eredita i dati dalla voce *parent*, il titolo di quest'ultima dovrebbe diventare il *booktitle* della prima. BibTEX non è in grado di fare questa mappatura e di conseguenza nella voce *parent* è necessario duplicare il campo `title` nel campo `booktitle`.[3] Questa è solo una delle limitazioni di BibTEX e la conseguenza principale di tutto ciò è che in questo modo i record bibliografici diventano inutilmente "pesanti", contenendo di fatto numerosi e irrazionali doppioni, necessari solo per aggirare le limitazioni di BibTEX.

Con Biber queste limitazioni non esistono più, perché questo motore permette di definire delle regole molto flessibili che sovrintendono al sistema di *data inheritance*. Consideriamo il semplice caso seguente:

```
@Book{book,
  author = {Author},
  title = {Booktitle},
  publisher = {Publisher},
  location = {Location},
  date = {1995}}

@InBook{inbook,
  crossref = {book},
  title = {Title},
  pages = {5-25}}
```

Nella costruzione della voce `@inbook` Biber mappa automaticamente il campo `title` della voce `@book` nel campo `booktitle` della voce `@inbook` che vi si collega tramite `crossref`. Nel caso più semplice di una relazione `@book`/`@inbook` la regola (in forma semplificata) definita dal pacchetto è la seguente:

```
\DeclareDataInheritance{book}{inbook}{%
    \inherit{title}{booktitle}
    \inherit{subtitle}{booksubtitle}
  }
```

Ancora più importante è la possibilità di impedire che la voce *child* erediti un determinato campo. Consideriamo i seguenti record:

```
@collection{Facchinetti:2009,
  Booktitle = {Studies on English Modality},
  Date = {2009},
```

3. La cosa è da tempo nota, tanto che molti gestori di archivi bibliografici, tra cui BibDesk e JabRef, hanno opzioni che permettono di duplicare automaticamente i campi necessari.

```
  Editor = {Anastasios Tsangalidis
    and Roberta Facchinetti},
  Location = {Bern},
  Publisher = {Peter Lang},
  Title = {Studies on English Modality},
  Note = {The note of the parent entry}}

@incollection{Degani:2009,
  Author = {Marta Degani and Elisabetta
    Adami and Anna Belladelli},
  Crossref = {Facchinetti:2009},
  Pages = {13-54},
  Title = {The Use of Modal Verbs in
    Interpersonal Contexts}}
```

Se non forniamo al programma alcuna istruzione particolare, il campo `note` verrà ereditato dalla voce *child* (in questo caso `Degani:2009`) e il risultato sarà qualcosa di simile (quanto simile dipende dallo stile bibliografico scelto):

> Degani, Marta, Elisabetta Adami, and Anna Belladelli. «The Use of Modal Verbs in Interpersonal Contexts». In: *Studies on English Modality*. Ed. by Anastasios Tsangalidis and Roberta Facchinetti. **The note of the parent entry**. Bern: Peter Lang, 2009, pp. 13–54

Se al contrario introduciamo la seguente regola

```
\DeclareDataInheritance
  {collection}{incollection}{\noinherit{note}}
```

la voce `@incollection` non erediterà il campo `note` dalla voce `@collection`, generando questo risultato:

> Degani, Marta, Elisabetta Adami, and Anna Belladelli. «The Use of Modal Verbs in Interpersonal Contexts». In: *Studies on English Modality*. Ed. by Anastasios Tsangalidis and Roberta Facchinetti. Bern: Peter Lang, 2009, pp. 13–54

4 Il campo `ids`

Sarà capitato a molti di cambiare idea sull'etichetta assegnata a una voce bibliografica. Poniamo di aver usato la chiave `kant:1968` per indicare i *Kants Werke* e di accorgerci che, probabilmente, è più opportuno identificarli con la chiave `kant:werke`, anche per una maggiore leggibilità del sorgente. Certo, potremmo modificare la chiave direttamente nell'archivio, ma ciò comporterebbe che i file composti precedentemente, e che usano la prima chiave, non sarebbero più compilabili (a meno che non si modifichino tutte le citazioni manualmente).

Se si usa Biber esiste una soluzione razionale a questo inconveniente che, specialmente in ambito umanistico, è più frequente di quanto si possa immaginare. È infatti sufficiente inserire nel campo `ids` una lista di chiavi alternative, che funzionano come degli alias per la vera e propria *entrykey*, che deve in ogni caso essere presente:

```
@mvbook{kant:1968,
  ids = {kant:werke,kant:KW,kantswerke},
  Author = {Kant, Immanuel},
  Title = {Kants Werke. Akademie Textausgabe},
  Publisher = {Walter de Gruyter},
  Location = {Berlin},
  Year = {1968},
  Volumes = {9}}
```

Le chiavi alternative contenute nel campo `ids` permetteranno di richiamare la stessa voce indifferentemente con uno dei comandi

```
\cite{kant:1968}
\cite{kant:werke}
\cite{kant:KW}
\cite{kantswerke}
```

Questa funzionalità risulta particolarmente utile, per esempio, nel caso di lavori di gruppo, dove ciascun autore può verosimilmente aver usato chiavi diverse per le stesse fonti.

5 L'opzione `safeinputenc`

Come anticipato, Biber fornisce sopporto completo per la codifica Unicode e questa caratteristica lo rende lo strumento elettivo per l'uso con XeLaTeX o LuaLaTeX. Quando usiamo (pdf)LaTeX su file codificati Unicode, però, potremmo imbatterci in situazioni poco piacevoli. Come noto, il pacchetto `inputenc` (Jeffrey e Mittelbach, 2014), che in questo caso carichiamo con l'opzione `utf8`, non copre tutti i caratteri della tabella Unicode. Per rendersene conto è sufficiente compilare con (pdf)LaTeX questo semplice esempio minimo:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\begin{document}
  ș
\end{document}
```

Mentre il successivo esempio, se compilato con XeLaTeX, non restituisce alcun errore e mostra correttamente il carattere ș:

```
\documentclass{article}
\usepackage{fontspec}
  \setmainfont{Times New Roman}

\begin{document}
  ș
\end{document}
```

Quando compiliamo con (pdf)LaTeX ci si può quindi trovare nella situazione in cui alcuni caratteri inseriti nel file `.bib` non vengano riconosciuti, ottenendo tipicamente l'errore:

```
Package inputenc Error: Unicode char \u8:
  not set up for use with LaTeX
```

La soluzione drastica sarebbe di sostituire ogni occorrenza di un carattere non riconosciuto con

l'apposito comando LaTeX (e di fatto è quello che spesso si fa). Per esempio modificando tutti i caratteri ș in `\d{s}`. Così facendo, però, in un certo senso stiamo "depotenziando" il file `.bib` per venire incontro ai limiti di (pdf)LaTeX e se in futuro volessimo usare lo stesso file `.bib` con XeLaTeX, avremmo un archivio contenente complessi comandi al posto dei più leggibili caratteri Unicode.

In questo caso ci viene in aiuto l'opzione safeinputenc=true (o semplicemente safeinputenc), che forza l'opzione texencoding=ascii, ma solo nel caso sia caricato il pacchetto inputenc con l'opzione utf8.

```
\usepackage[utf8]{inputenc}
\usepackage[safeinputenc]{biblatex}
```

Grazie a quest'opzione, tutti i dati non-Ascii presenti nel file `.bib` verrano convertiti in formato Ascii. Per esempio, il carattere ș verrà convertito in `\d{s}`, permettendo quindi una corretta compilazione. Va però precisato che questo meccanismo funziona solo relativamente alle lettere latine con diacritici. Se infatti nel file `.bib` sono presenti lettere greche (ma anche cirilliche, ebraiche, arabe, giapponesi, ecc.) safeinputenc risulta inefficace.[4]

Si capisce quindi che questa soluzione è accettabile solo se nel file `.bib` ci sono pochi caratteri che causano il problema. Se invece si vuole avere un pieno supporto Unicode bisogna appunto rivolgersi a motori più evoluti come XeTeX o LuaTeX.

6 Formati non-BibTeX

L'utente LaTeX in genere compone i propri archivi bibliografici secondo la sintassi prevista da BibTeX, ma questa naturalmente non è l'unica sintassi attualmente disponibile. Vi sono infatti numerosi software di gestione della bibliografia che hanno un loro specifico formato e alcuni di questi sono anche molto diffusi. Naturalmente Biber riconosce la sintassi di BibTeX, ma con questo potente motore è possibile lavorare anche con archivi scritti in formati diversi.

Il comando `\addbibresource`, con il quale si carica un archivio bibliografico, accetta diverse opzioni, tra le quali l'opzione datatype, che permette di indicare il formato dell'archivio caricato. Per il momento i formati supportati, e ancora in fase sperimentale, sono i seguenti:

datatype=ris
: Formato RIS realizzato dalla Research Information Systems.[5]

datatype=zoterordfxml
: Formato Zotero RDF/XML. È la sintassi usata dall'estensione di Mozilla Firefox (notissimo *browser* multipiattaforma e *open source*).

datatype=endnotexml
: Formato Endnote XML. Si tratta di un software a pagamento molto diffuso in ambiente Windows.

Questa funzionalità si rivela utile nel caso in cui si debbano recuperare dati bibliografici da archivi *online* che non permettono di esportare in formato BibTeX ma, per esempio, solo in formato RIS, oppure nel caso di archivi condivisi tra più persone di un gruppo di ricerca. Un archivio in formato RIS, per esempio, si carica con

`\addbibresource[datatype=ris]{`⟨*filename*⟩`.ris}`

Per testare questa funzionalità si possono seguire questi due semplici passaggi:

1. Creiamo un semplice file in formato RIS,[6] che chiameremo `test.ris`, contenente un unico record bibliografico (lasciando una riga bianca sotto `ER -`!) identificato dalla chiave `test` (*tag* `ID`):

   ```
   TY  - BOOK
   ID  - test
   AU  - Lemmon, E.J.
   TI  - Beginning Logic
   PY  - 1965
   PB  - Thomas Nelson and Sons Limited
   ER  -
   (questa riga va lasciata vuota)
   ```

2. Compiliamo l'esempio minimale seguente

   ```
   % !TEX TS-program = pdflatex
   % !BIB TS-program = biber
   \documentclass{article}
   \usepackage[backend=biber]{biblatex}
   \addbibresource[datatype=ris]{test.ris}

   \begin{document}
   \cite{test}
     \printbibliography
   \end{document}
   ```

L'esempio è stato testato su una TeXLive 2014 aggiornata al 3 agosto 2014 e dovrebbe funzionare anche per le prossime versioni. A volte, tuttavia, possono sorgere situazioni apparentemente inspiegabili, spesso dovute a errori di sintassi nel file dell'archivio bibliografico. Per esempio, se non si lascia la riga vuota di cui abbiamo detto sopra, la voce bibliografica non verrà stampata.

7 Liste bibliografiche

Tra le funzionalità più utili di biblatex c'è senza dubbio la possibilità di creare facilmente una lista

4. Sono molto grato a Claudio Beccari, che mi ha fatto comprendere l'essenza di questo problema. Per un approfondimento sulla questione delle codifiche in LaTeX si rimanda a Beccari e Gordini (2014).

5. La gestione di questo formato in ConTeXt, attraverso il modulo publications, è stata recentemente trattata in Scarso (2014).

6. Per un'elenco completo dei *tag* RIS si rimanda alla pagina http://en.wikipedia.org/wiki/RIS_(file_format).

delle sigle, grazie al comando `\printshorthands`. Questa lista viene generata recuperando il valore del campo `shorthand`, se presente, e componendovi a fianco l'intera voce bibliografica secondo il layout definito dall'ambiente shorthand impostato dal comando stesso (e modificabile con `\defbibenvironment`). Oltre alla possibilità di stampare la lista delle sigle, biblatex prevede un meccanismo generale che permette di creare una lista di qualsiasi campo presente nei record bibliografici (di fatto `\printshorthands` non è altro che un'istanza particolare di questo meccanismo più generale).

Immaginiamo di voler creare una lista delle abbreviazioni dei nomi delle riviste, un elemento paratestuale molto diffuso in testi di ambito giuridico, ma non solo. La *Rivista italiana di diritto e procedura penale*, per esempio, è generalmente siglata con RIDPP e un record di un articolo pubblicato su questa rivista dovrà avere almeno le seguenti righe:

```
@article{entrykey,
...
journaltitle = {Rivista italiana
  di diritto e procedura penale},
shortjournal = {ridpp},
...
}
```

Si noti che il campo `shortjournal` non ha alcuna formattazione, la quale viene gestita all'interno del file `.tex`, lasciando così l'archivio bibliografico "pulito" e disponibile anche per esigenze stilistiche differenti.

Per poter stampare la lista delle abbreviazioni delle riviste si deve innanzitutto creare un nuovo "tipo" di voce (tecnicamente un nuovo *driver*) che come unico compito abbia quello di stampare il campo `journaltitle`, ovvero il nome della rivista. Chiameremo questo driver `@shortjournal`:

```
\DeclareBibliographyDriver{shortjournal}{%
  \printfield{journaltitle}}
```

Fatto questo, per stampare la lista è sufficiente il comando

```
\printbiblist[title={Elenco abbreviazioni
    delle riviste citate}]{shortjournal}
```

Questo comando ha ha due argomenti. Il primo è opzionale e può contenere le stesse opzioni di `\printshorthands`; in particolare, in questo caso, il titolo da usare per la lista. Il secondo contiene il nome del driver appena definito, nome che viene usato anche per filtrare le voci che hanno il campo `shortjournal` ed, eventualmente, per formattare la lista delle abbreviazioni.

Il filtro è definito da `\DeclareBiblistFilter` e può essere modificato dall'utente in base alle proprie esigenze. La definizione di default che si ricava dal file `biblatex2.sty` corrisponde a

```
\DeclareBiblistFilter{shortjournal}{%
  \filter[type=field,filter=shortjournal]
  }
```

Sono possibili anche altri filtri più complessi, ma la documentazione di biblatex è ad oggi incoerente, visto che definisce il comando `\filter` in un modo diverso da come viene usato nel file di stile. In ogni caso, a titolo di esempio, possiamo creare un filtro per avere un elenco dei nomi delle sole riviste che appaiono in record che hanno il campo `shortjournal` ma non hanno il campo `series`:

```
\DeclareBiblistFilter{shortjournal}{%
\filter[type=field,filter=shortjournal]
\filter[type=notfield,filter=series]
}
```

Nell'esempio riportato nella figura 1 si noterà che l'unica voce contenente il campo `series` non compare nella lista delle abbreviazioni.

Il comando `\printbiblist`, analogamente a `\printshorthands`, racchiude la lista in un ambiente, che ha lo stesso nome assegnato al driver e al filtro. Il comando messo a disposizione dal pacchetto è il seguente:

```
\defbibenvironment{⟨name⟩}
  {⟨begin code⟩}
  {⟨end code⟩}
  {⟨item code⟩}
```

⟨*name*⟩ è il nome che assegnamo all'ambiente e allo stesso tempo la chiave da passare all'opzione `env` di `\printbiblist`; ⟨*begin code*⟩ e ⟨*end code*⟩ sono i comandi da far eseguire all'inizio e alla fine dell'ambiente (analogamente a quello che succede con il classico `\newenvironment`); ⟨*item code*⟩ contiene il codice da eseguire prima di ogni *entry* della lista. Per modificare l'ambiente di questa lista, ovvero per crearne uno nuovo, sono però necessarie anche altre macro. Di seguito vediamo un esempio molto semplice.

Immaginiamo di voler comporre le sigle in neretto e il titolo delle riviste in tondo. Siccome le sigle sono contenute nel campo `shortjournal`, sarà sufficiente usare

```
\DeclareFieldFormat{shortjournal}{%
  \bfseries#1}
```

Per avere i titoli in semplice tondo, ma solo nella lista delle abbreviazioni, servirà invece

```
\AtBeginBiblist{shortjournal}{%
\DeclareFieldFormat{journaltitle}{#1}
}
```

A questo punto definiamo un nuovo ambiente, che chiameremo shortjournal, con

```
\defbibenvironment{shortjournal}
  {\list{}{%
       \labelsep\biblabelsep
       \labelwidth=2cm
       \leftmargin\labelwidth
       \advance\leftmargin\labelsep
```

```
% !TEX TS-program = pdflatex
% !BIB TS-program = biber

\begin{filecontents}{printbiblist.bib}
@article{angenendt,
  author = {Angenendt, Arnold},
  title = {In Honore Salvatoris~-- Vom Sinn und Unsinn der Patrozinienkunde},
  journaltitle = {Revue d'Histoire EcclÃ©siastique},
  shortjournal = {rhe},
  date = 2002,
  volume = 97,
  pages = {431-456, 791-823}}
@article{gillies,
  Author = {Gillies, Alexander},
  date = 1933,
  Journaltitle = {Publications of the English Goethe Society},
  Shortjournal = {pegs},
  Series = {newseries},
  Title = {Herder and the Preparation of Goethe's Idea of World Literature},
  Volume = 9}
@article{sigfridsson,
  Author = {Sigfridsson, Emma and Ryde, Ulf},
  date = 1998,
  Journaltitle = {Journal of Computational Chemistry},
  Shortjournal = {jcc},
  Title = {Comparison of methods for deriving atomic charges
    from the electrostatic potential and moments},
  Volume = 19}
\end{filecontents}

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[english,italian]{babel}
\usepackage[style=philosophy-verbose]{biblatex}
  \addbibresource{printbiblist.bib}

\DeclareBibliographyDriver{shortjournal}{\printfield{journaltitle}}
\DeclareFieldFormat{shortjournal}{\scshape\bfseries#1}
\AtBeginBiblist{shortjournal}{\DeclareFieldFormat{journaltitle}{#1}}

\defbibenvironment{shortjournal}
  {\list{}{%
      \labelsep\biblabelsep
      \labelwidth=2cm
      \leftmargin\labelwidth
      \advance\leftmargin\labelsep
      \itemsep\bibitemsep
      \parsep\bibparsep
    \let\makelabel\printshortjournal}}
  {\endlist}
  {\item}
\newcommand*{\printshortjournal}{\printfield{shortjournal}}

\DeclareBiblistFilter{shortjournal}{%
  \filter[type=field,filter=shortjournal]
   \filter[type=notfield,filter=series]}

\begin{document}
\nocite{*}
\printbiblist[env=shortjournal,title={Elenco abbreviazioni delle riviste citate}]{shortjournal}
  \printbibliography
\end{document}
```

Codice 1: Esempio per la creazione di una lista delle abbreviazioni delle riviste citate.

Elenco abbreviazioni delle riviste citate

RHE Revue d'Histoire Ecclésiastique
JCC Journal of Computational Chemistry

Riferimenti bibliografici

Angenendt, Arnold, "In Honore Salvatoris – Vom Sinn und Unsinn der Patrozinienkunde", *Revue d'Histoire Ecclésiastique*, vol. 97 (2002), p. 431-456, 791-823.

Gillies, Alexander, "Herder and the Preparation of Goethe's Idea of World Literature", *Publications of the English Goethe Society*, nuova ser., vol. 9 (1933).

Sigfridsson, Emma e Ulf Ryde, "Comparison of methods for deriving atomic charges from the electrostatic potential and moments", *Journal of Computational Chemistry*, vol. 19 (1998).

Figura 1: Output del codice 1.

```
      \itemsep\bibitemsep
      \parsep\bibparsep
    \let\makelabel\printshortjournal}}
  {\endlist}
  {\item}
\newcommand*{\printshortjournal}{%
  \printfield{shortjournal}}
```

Abbiamo dovuto definire, come di consueto, lo stile dell'etichetta di ogni *item* (`\printshortjournal`), che in questo caso consiste semplicemente nel campo `shortjournal`. Come si può notare, nella definizione dell'etichetta non abbiamo usato nessuna macro di formattazione, che abbiamo invece passato precedentemente a `\DeclareFieldFormat`.

Questo approccio è consigliabile soprattutto se stiamo definendo un file di stile, in quanto consente all'utente finale di apportare facilmente, nel caso servisse, modiche stilistiche. Se per esempio si volessero le sigle in maiuscolo basterebbe

```
\DeclareFieldFormat{shortjournal}
  {\MakeUppercase{#1}}
```

A questo punto per stampare la lista nel formato appena definito sarà sufficiente aggiungere l'opzione `env` a `\printbiblist`:

```
\printbiblist[env=shortjournal,
  title={Elenco abbreviazioni delle riviste
    citate}]
```

Oltre allo stile delle sigle, questo nuovo ambiente aumenta a 2 cm la larghezza del box che contiene la sigla stessa. È possibile ovviamente personalizzare anche l'intestazione attraverso il comando `\definebibheading`, analogamente a quello che si fa con le classiche bibliografie.

Il listato 1 mostra un codice mimino compilabile contenente i comandi descritti in questa sezione, mentre nella figura 1 si può vedere il risultato finale.

8 Il campo `related`

Una voce bibliografica, nella sua veste più semplice, corrisponde a una fonte, sia essa un volume, un articolo o altro. Consideriamo il caso di un volume in lingua francese del quale è disponibile anche una traduzione italiana. A rigor di termini si tratta di due fonti distinte, ma è consuetudine comporle in un'unica voce. Una delle soluzioni più usate, almeno in Italia, è mostrata nell'esempio seguente ed è implementata negli stili del pacchetto biblatex-philosophy (Valbusa, 2014):

> Jules-Henri Poincaré (1968), *La science et l'hypothèse*, Flammarion, Paris; trad. it *La scienza e l'ipotesi*, a cura di Corrado Sinigaglia, Bompiani, Milano 2003.

Con biblatex-philosophy abbiamo due meccanismi diversi per creare voci simili a quella mostrata in precedenza. Il primo prevede di creare un'unica voce che contenga tutti i dati necessari, usando campi particolari per i dati della traduzione:

```
@book{Poincare:1968-ORIG,
   Author = {Jules-Henri PoincarÃ©},
   Title = {La science et l'hypothÃ¨se},
   Date = {1968},
   Location = {Paris},
```

```
  Publisher = {Flammarion},
  Transtitle = {La scienza e l'ipotesi},
  Transnote = {a cura di Corrado Sinigaglia},
  Transpublisher = {Bompiani},
  Translocation = {Milano},
  Transdate = {2003}}
```

Questa soluzione, sebbene possa andar bene per la maggior parte dei casi, ha dei limiti importanti. Innanzitutto i campi per la traduzione sono limitati, quindi se avessimo bisogno di inserire altre informazioni, le dovremmo mettere, con qualche stratagemma, in uno dei campi disponibili. Nell'esempio sopra si è usato il campo `transnote` perché non sono disponibili campi specifici per il curatore, il traduttore, il prefatore, ecc. della traduzione, e tutte queste figure vanno eventualmente inserite in questo campo. (Per inciso, i nuovi campi `trans*` sono stati definiti grazie a un comando che richiede Biber e che vedremo fra poco.)

A partire dalla versione 2.5, tuttavia, biblatex fornisce un meccanismo molto più potente e razionale se si usa Biber come motore bibliografico. Testo originale e traduzione vengono trattati come voci bibliografiche autonome (in effetti si tratta generalmente di due "oggetti" distinti) e il primo record richiama la voce collegata attraverso il campo `related`:

```
@book{Poincare:1968-ORIG,
  author = {Jules-Henri PoincarÃ©},
  title = {La science et l'hypothÃ¨se},
  publisher = {Flammarion},
  location = {Paris},
  date = {1968},
  related = {Poincare:1968-ITA}}

@book{Poincare:1968-ITA,
  author = {Jules-Henri PoincarÃ©},
  editor = {Corrado Sinigaglia},
  title = {La scienza e l'ipotesi},
  publisher = {Bompiani},
  location = {Milano}}
```

Poiché le voci sono indipendenti, con questo sistema abbiamo a disposizione tutti i campi previsti da biblatex, anche per la voce collegata. Ma con questo sistema si può fare molto di più.

Grazie al campo `relatedstring` possiamo cambiare la stringa di default ("trad. it", per i documenti in lingua italiana) che precede la voce collegata, in una stringa a piacere. Per esempio "prima ed. it.":

```
@book{Poincare:1968-ORIG,
   author = {Jules-Henri PoincarÃ©},
   title = {La science et l'hypothÃ¨se},
   publisher = {Flammarion},
   location = {Paris},
   date = {1968},
   related = {Poincare:1968-ITA},
   relatedstring = {prima ed. it.}}
```

Inoltre, a partire dalla versione 1.6 di Biber le potenzialità sono ancora maggiori. Si possono per esempio creare voci "a cascata", anche molto complesse, in cui inserire i dati dell'edizione originale e di due diverse traduzioni:

> Popper, Karl R. 1934 *Logik der Forschung*, Springer, Wien; trad. ingl. *The Logic of Scientific Discovery*, 3ª ed., Hutchinson, London 1959; trad. it. *Logica della scoperta scientifica*, 3ª ed., Einaudi, Torino 1998.

I record concatenati che generano la voce appena citata sono i seguenti:

```
@book{popper-logik,
  Author = {Karl R. Popper},
  Date = {1934},
  Location = {Wien},
  Publisher = {Springer},
  Related = {popper-logik:ing},
  Relatedstring = {trad. ingl.},
  Title = {Logik der Forschung}}

@book{popper-logik:ing,
  Author = {Karl R. Popper},
  Date = {1959},
  Edition = {3},
  Location = {London},
  Publisher = {Hutchinson},
  Related = {popper-logik:ita},
  Title = {The Logic of Scientific Discovery}}

@book{popper-logik:ita,
  Author = {Karl R. Popper},
  Date = {1998},
  Edition = {3},
  Hyphenation = {italian},
  Location = {Torino},
  Publisher = {Einaudi},
  Title = {Logica della scoperta scientifica}}
```

Gli stili di biblatex-phylosophy permettono di comporre bibliografie commentate, attraverso il campo `annotation`. Nel caso queste voci avessero anche questo campo, verrebbe in ogni caso stampato solo quello della voce madre (ovvero la prima che contiene il campo `related`). Inoltre, di default le voci collegate non vengono incluse nella bibliografia a meno che non siano a loro volta citate.

Sono a disposizione anche altri campi per la personalizzazione delle voci "related". Per esempio, il campo `relatedtype` prevede di inserire una stringa di localizzazione definita nel file `.lbx` o attraverso `\DefineBibliographyStrings`. Il campo `relatedtype` negli stili biblatex-philosophy è impostato di default sulla stringa `translationas` (sic) definita come "trad. it.". Il campo `relatedstring` permette quindi di sovrascrivere la stringa specificata con `relatedtype`. Da questo punto di vista le due sintassi seguenti sono equivalenti:

```
Relatedstring = {trad. ingl.},
Relatedtype = {translationas},
```

Il vantaggio della seconda forma è che se si sta componendo un testo in inglese, la stringa usata

sarà automaticamente "trans.", sempre che la stringa sia stata definita nel file di localizzazione per la lingua inglese.

9 Modificare in modo dinamico i record bibliografici

Veniamo infine a quello che può essere considerato la vera bacchetta magica del sistema biblatex/Biber. Poniamo di avere un archivio molto corposo, nel quale in ciascuna voce di tipo `@book` abbiamo inserito nome dell'editore e luogo di edizione. Se ci viene richiesto di eliminare, per esempio, il nome dell'editore, dovremmo intervenire massicciamente nel file `.bib`, compromettendo in maniera seria l'archivio. Ma ci potrebbe venir richiesto di eliminare il nome dell'editore solo per un sottoinsieme specifico di voci in archivio.

Grazie a Biber è possibile agire sulle singole voci direttamente dal sorgente `.tex`, senza nemmeno aprire il file `.bib`. Si può decidere di eliminare il campo `publisher` per tutte le voci in archivio, solo per alcuni tipi, solo per alcuni archivi, ecc. Le possibilità sono pressoché infinite.

Visto che i comandi previsti per Biber accettano anche le espressioni regolari, si possono applicare filtri ancora più potenti. Questo codice, per esempio, permette di eliminare il campo `publisher` solo per le voci il cui campo `date` è un numero maggiore di 1900:

```
\DeclareSourcemap{
\maps[datatype=bibtex]{
\map{
\step[fieldsource=date,
match=\regexp{(1[^9]\d{2})},final]
\step[fieldset=publisher, null]
    }
  }
}
```

Il primo comando `\step` cerca in tutti gli archivi bibliografici caricati nel preambolo con `\addbibresource` le voci in cui il campo `date` corrisponde al valore di `match`, che è appunto un'espressione regolare. L'espressione regolare individuerà i numeri di 4 cifre che iniziano per 1 e che non hanno 9 come seconda cifra, ovvero tutti gli anni dal 1000 al 1899.[7] Il secondo `\step` elimina il campo `publisher` (attraverso la chiave `null`) delle voci individuate dal primo `\step`.

Con questo comando si possono anche creare nuovi campi, oppure aggiungere una *keyword* solo per quelle voci che rispondono a una determinata condizione. Poniamo di voler inserire le voci pubblicate fino al 1900 in una bibliografia indipendente. Le soluzioni consuete consistono o nell'usare delle categorie o nell'inserire una parola chiave all'interno dei record interessati. Entrambe queste strade comportano di individuare manualmente tutte le voci da includere e in alcuni casi ciò potrebbe essere frustrante, oltre che rischioso, potendo sempre dimenticare qualche voce.

Grazie a `\DeclareSourcemap` possiamo risolvere il tutto con questo "semplice" codice:

```
\DeclareSourcemap{
\maps[datatype=bibtex]{
\map{
\step[fieldsource=date,
match=\regexp{(1[^9]\d{2})},final]
\step[fieldset=keywords,
      fieldvalue={ottocento}]
    }
  }
}
```

e stampare quindi la bibliografia con

```
\printbibliography[
  title={Bibliografia (--1899)},
  keyword=ottocento]
```

Vediamo ora un caso di applicazione di `\DeclareSourcemap` ancora più interessante. Può succedere che in una voce bibliografica ci siano troppe informazioni, ovvero troppi campi, relativamente alla bibliografia che stiamo componendo. È da escludere, per i motivi detti sopra, di "emendare" la voce in archivio: se abbiamo inserito dei dati è un peccato elminarli perché in futuro ci potrebbero servire.

Fortunatamente con Biber si possono non solo eliminare (o aggiungere) campi specifici per insiemi di voci che rispondono a determinati criteri, ma anche per singole voci. Se ci si pensa bene, la chiave univoca assegnata a ogni voce, ovvero la *entrykey*, è di fatto uno speciale campo e non a caso la documentazione di biblatex, lo inserisce, per primo, nella sezione "Generic Fields". Ciò significa che se si può ricercare il contenuto di un determinato campo, come visto sopra, si può anche ricercare il contenuto del campo `entrykey`, ovvero il nome della chiave assegnata a una specifica voce. Vediamo innanzitutto il comando:

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=entrykey,
            match=\regexp{key1},
            fieldset=note,null]
    }
  }
}
```

La chiave `fieldsource=entrykey` predispone alla ricerca nella chiavi delle singole voci; `match=\regexp{key1}` individua la voce etichettata con `key1`. Infine, `fieldset=note,null` elimina da quella voce il campo `note`. Questo è forse lo strumento più potente di Biber, perché permette di agire su singole voci in maniera precisa, visto che le *entrykey* devono essere univoche.

7. Per una prima introduzione all'uso delle espressioni regolari si può fare riferimento alla pagina http://it.wikipedia.org/wiki/Espressione_regolare.

Vediamo un altro esempio che mostra l'utilità di questo sistema. Immaginiamo che ci venga richiesto di modificare il titolo breve (campo `shorttitle`) solo di alcune specifiche voci in archivio. Come sempre, vogliamo evitare di mettere mano all'archivio stesso. Servirà un codice simile al precedente, ma con un'aggiunta fondamentale:

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map[overwrite]{
      \step[fieldsource=entrykey,
            match=\regexp{key2}]
     \step[fieldset=shorttitle,
            fieldvalue={The new short title}]
    }
  }
}
```

Come si può notare, è stata passata al comando `\map` l'opzione overwrite (equivalente a overwrite=true). Questa opzione, infatti, è impostata di default su false per impedire che, se un campo contiene già dei dati, possa essere sovrascritto ovvero modificato (anche se può essere eliminato, come abbiamo visto precedentemente). Attivandola, quindi, viene permessa la sovrascrittura.

Un'altra funzionalità molto utile messa a disposizione da `\DeclareSourcemap` consiste nella possibilità di usare per i campi nomi nuovi definiti dall'utente stesso. Immaginiamo di dover stampare una determinata giurisprudenza, ovvero un elenco di sentenze. Le sentenze sono delle fonti analoghe alle altre e pertanto le si possono trattare come normali record bibliografici. Ecco un esempio di una voce di una giurisprudenza nazionale:

Cons. Stato, Sez. V, 28 set. 1980, n. 713

Qui l'autore è una precisa corte (in questo caso il Consiglio di Stato), inoltre v'è un'informazione specifica, ovvero l'indicazione della sezione. L'obiettivo finale è quello di poter scrivere un record con la seguente sintassi:

```
@jurisdiction{cstat:713/1980,
  Court = {Cons. Stato},
  Section = {5},
  Eventdate = {1980-09-28},
  Number = {713},
  Date = {1980},
  Volume = {1},
  Pages = {1499-}}
```

Innanzitutto dobbiamo creare il nuovo driver `@jurisdiction`, con eventuali macro di supporto, attraverso

```
\DeclareBibliographyDriver{jurisdiction}
{...}
```

Chi fosse interessato può leggere il codice nel file `philosophy-standard.bbx`, reperibile nella distribuzione LaTeX, all'interno della cartella `texmf-dist/tex/latex/biblatex-philosophy`.

I campi `court` e `section` non sono definiti nativamente da biblatex. Per poterli usare dobbiamo quindi mapparli su dei campi riconosciuti. Scegliamo quindi di mappare il primo nel campo `author` e il secondo nel campo `nameaddon`:

```
\DeclareStyleSourcemap{
\maps[datatype=bibtex]{
    \map{
     \step[fieldsource=court,
       fieldtarget=author]
     \step[fieldsource=section,
       fieldtarget=nameaddon]
      }
    }
  }
```

In questo modo il record sopra è già perfettamente utilizzabile. Si devono soltanto formattare i campi vecchi e nuovi, per ottenere il risultato finale atteso. Per il campo `nameaddon`, per esempio, servirà

```
\DeclareFieldFormat[jurisdiction]{nameaddon}
  {\ifinteger{#1}{%
    \bibcpsstring{section}~\RN{#1}}{#1}%
}
```

In questo modo, inserendo nel campo `nameaddon` un numero intero n, si avrà nell'output la stringa "Sez. n", con n in caratteri romani maiuscoli (`\RN`). Se invece si vuole indicare, per esempio, una "Adunanza plenale", si scriverà semplicemente

```
nameaddon = {Ad\adddotspace plen\adddot}}
```

Nella sezione 2 si è mostrata una modalità di creazione di record bibliografici attraverso il campo `xdata` molto promettente. Ora che abbiamo compreso, o almeno intuito, il funzionamento di `\DeclareSourcemap` possiamo mostrare una soluzione ancora più potente.

Immaginiamo di voler creare un archivio di editori, composto essenzialmente da voci `@xdata`. Sarebbe molto comodo assegnare a questo tipo di voci un nome più consono, per esempio `@mypublisher` e, analogamente, avere al posto del campo `xdata` un campo `mypublisher`. Riprendendo l'esempio che abbiamo mostrato precedentemente in 2, vorremmo pertanto poter definire uno scenario del genere:

```
@mypublisher{laterza,
  Publisher = {Laterza},
  Location = {Roma-Bari}}

@book{Cellucci:2008,
  Author = {Carlo Cellucci},
  Title = {PerchÃ© ancora la filosofia},
  mypublisher = {laterza},
  Year = {2008}}

@book{casati:2012,
  Author = {Roberto Casati},
  Title = {Prima lezione di filosofia},
  mypublisher = {laterza},
  Year = {2012}}
```

Perché la soluzione funzioni bisogna far conoscere a biblatex il nuovo driver e il nuovo campo. Per il primo è sufficiente:

```
\DeclareBibliographyAlias{mypublisher}{xdata}
```

Ma non è possibile far riconoscere il nuovo campo `mypublisher`, come ci si aspetterebbe, con

```
\DeclareFieldAlias{mypublisher}{xdata}
```

È necessario pertanto mappare il nuovo campo su `xdata` attraverso

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=mypublisher,
            fieldtarget=xdata]
    }
  }
}
```

10 Conclusione

In questo contributo abbiamo mostrato una piccolissima porzione delle potenzialità del sistema biblatex/Biber, ma si spera che almeno gli esempi riportati possano essere di spunto per nuove e più ingegnose soluzioni. In sede conclusiva mi permetto una riflessione sul senso "filosofico" di tutto questo.

Il sistema che abbiamo trattato si è sviluppato in modo così imponente principalmente per venire incontro alle esigenze degli umanisti. Il fatto che abbia un'infinità di funzioni e comandi è rassicurante, perché possiamo dire, quasi a priori, che probabilmente c'è una soluzione per ogni problema (bibliografico). Il vero limite, tuttavia, è che per scoprire *ogni* soluzione serve una conoscenza veramente profonda del programma stesso. E spesso, lo si deve ammettere, il gioco non vale la candela.

Come tutti i sistemi di gestione bibliografica, anche con biblatex si è adottato un approccio del tipo *bottom-up*. Invece di sforzarsi a rendere il più possibile razionali le pretese di certi utenti/autori/editori, eventualmente cassandole, le si sono assecondate, con la conseguenza che molti programmi (non solo biblatex) sono diventati troppo ricchi e troppo poco funzionali, rischio che sempre si corre quando si vuole accontentare tutti. Ma l'irrazionale, alla fine, sfugge sempre anche alla logica più potente e non è da escludere che vi siano esigenze umanistiche che non troveranno mai una soluzione con i meccanismi seppur vari previsti da biblatex.

È da augurasi che in futuro chi detiene il potere di stabilire le usanze, le consuetudini e le norme abbia la forza di fare un passo indietro, favorendo in questo modo il perfezionamento di sistemi razionali di gestione della bibliografia, che già esistono ma che rimangono confinati in una nicchia perché ognuno cerca di perseguire il meglio per sé piuttosto che il bene di tutti.

Riferimenti bibliografici

Beccari, C. e Gordini, T. (2014). *Codifiche in TEX e LATEX*. URL http://www.guitex.org/home/images/doc/GuideGuIT/introcodifiche.pdf.

Hufflen, J.-M. (2012). «Beyond BibTEX». *ArsTEXnica*, (14), pp. 7–14.

Jeffrey, A. e Mittelbach, F. (2014). «Il pacchetto inputenc». Versione 1.2b.

Kime, P. e Charette, F. (2014). «biber. a backend bibliography processor for biblatex». URL http://biblatex-biber.sourceforge.net. Versione 1.9.

Lehman, P. (2014). «The biblatex package». URL http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/. Versione 2.9a.

Scarso, L. (2014). «Un primo sguardo al modulo publications». *ArsTEXnica*, (17), pp. 29–36.

Valbusa, I. (2014). «Il pacchetto biblatex–philosophy». Disponibile su CTAN all'indirizzo: http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex-contrib/biblatex-philosophy/.

▷ Ivan Valbusa
Dipartimento di
Filosofia, Pedagogia e Psicologia
Università degli Studi di Verona
ivan dot valbusa at univr dot it

Dealing with Ancient Works in Bibliographies

Jean-Michel Hufflen

[. . .] The race of men that the immortals who dwell on Olympus made first of all was of gold. They were in the time of Kronos, when he was king in heaven; and they lived like gods, with carefree heart, remote from toil and misery. Wretched old age did not affect them either, but with hands and feet ever unchanged they enjoyed themselves in feasting, beyond all ills, and they died as if overcome by sleep. All good things were theirs, and the grain-giving soil bore its fruits of its own accord in unstinted plenty, while they at their leisure harvested their fields in contentment amid abundance. [. . .]

Works and Days (8th century BC), (Hesiod, 1988, Lines 109–121).

Abstract

Bibliography processors used in conjonction with word processors have been designed in order to deal with recent works. That is, authors and publishing dates are known. Often, these informations are used to sort bibliographies, so they are supposed to be precise. This is not the case if ancient works are cited. We show how an extension of our MlBibTeX bibliography processor can allow us to deal with such cases by means of *patterns*.

Keywords Citing ancient works, inexact year specification, unknown authors, patterns, translated works, cross-referencing, BibTeX, extension of MlBibTeX, `mlbiblatex` program.

Sommario

Le elaborazioni bibliografiche collegate con i word processor sono state pensate per gestire lavori recenti. In altre parole per questi lavori sono noti sia gli autori, sia le date di pubblicazion. Spesso queste informazioni sono usate per ordinare gli elenchi bibliografici, e perciò ci si aspetta che esse siano precise. Non è così quando ci si riferisce a lavori antichi. Qui mostriamo come una estensione al nostro programma di elaborazione bibliografica MlBibTeX ci permette di gestire questi casi mediante *pattern*.

Parole chiave Citazione di lavori antichi, incertezza sull'anno di pubblicazione, autori ignoti, pattern, lavori tradotti, riferimenti incrociati, BibTeX, estensioni per MlBibTeX, programma `mlbiblatex`.

0 Introduction

Modern typesetting systems have been designed in order for new written documents to be developed, processed, and updated easily. Even if an ancient document's body reached a final state and does not have to be developed or updated, using such a modern processor to typeset ancient documents may be interesting, too, in order to digitalise works of the past or build new critical editions of such works. Good examples related to this approach are given by Delprat and Orevkov (2012) and also by Pignalberi et al. (2011, 2012). Now let us consider *bibliography processors*, extracting *references* from bibliography database files. They have also been designed in order to build 'References' sections for new written documents. But often bibliographies are *sorted*, that is, bibliographical references are sorted with respect to authors' names, and different works written by the same authors are sorted chronologically within the final list[1]. Such a *modus operandi* is suitable if authors and publication dates are known precisely. That is obviously true for recent works, not for ancient ones:

> Ἡσίοδος. *Ἔργα καὶ ἡμέραι*. 8th century BC.

It can be noticed that such a reference has only historical interest, it does not point to a resource—printed or on-line—that readers can get. Most often, modern translations are used and cited, and such a translation has a publisher and a precise publication date:

> Hesiod. *Theogony — Works and Days* [Θεογονία — Ἔργα καὶ ἡμέραι], translated from ancient Greek by Martin Lichtfield West (Oxford University Press, 1988).

However, several translations of the same ancient work may be cited throughout the same article, if we aim to avoid the repetition of the original title and metadata associated with all these translations, a possible solution may look like:

> [1] Hesiod. *Theogony — Works and Days*. English translation of [3] by Martin

1. There exist *unsorted* bibliographies, in which case the order of the references is the order of first citations of these items throughout the text. Unsorted bibliographies are traditional in some specific areas such as Medicine or History, but it seems to us that most bibliographies are sorted.

Lichtfield West. Oxford University Press, 1988.

[2] Hésiode : *Théogonie — Les travaux et les jours*. Traduction française de [3] par Paul Mason. Société d'Édition « Les Belles lettres », Paris, 1979.

[3] Ἡσίοδος. *Θεογονία — Ἔργα καὶ ἡμέραι*. 8th century BC.

that is, by using a cross-referencing system for translations, as sketched in Hufflen (2008b). In order for such a system to be put into action, the third item has to be specified as a separate entry, being type `@MISC` or `@UNPUBLISHED` if the BibTEX bibliography processor (Patashnik, 1988a) is used[2].

In this article, we do not go thoroughly into this notion of cross-references, but focus instead on the features provided by a new experimental extension of MlBibTEX[3] (Hufflen, 2003), so-called `inexact`, for *imprecise* metadata, such as inexact dates of publication or surmised author's identity. As introduced before, we consider that an **inexact date** is a date whose year is not precisely known, e.g., *Oedipus the King*, a tragedy by Sophocles (495 BC–406 BC), was first performed around 429 BC. Another example is given by Hesiod's *Works and Days*, as shown before. Concerning doubtful identity of authors, the following example is related to Italian music:

> [Mersenne Marin? (1588–1648)] *Campanae Parisienses — Aria*. In *Ancient Airs and Dances, 2nd Suite*, by Ottorino Respighi (1879–1936).

Another suite arranged by Ottorino Respighi gives an example of an unknown author:

> [Ignoto] *Italiana*. In *Ancient Airs and Dances, 3rd Suite*, by Ottorino Respighi.

In Section 1, we explain why such metadata are difficult to handle if 'old' BibTEX is used. Section 2 briefly recalls the perspectives open by BibTEX's successors. Section 3 introduces and discusses MlBibTEX's new syntactic features for ancient works' metadata. Throughout the article, we mention that MlBibTEX provides advanced features for sorting bibliographical items, described in Hufflen (2012b). In an annex that may be viewed as a complement to this document, we show how they have been updated. Reading this article only requires basic knowledge about BibTEX and MlBibTEX.

2. In this article, we use MlBibTEX's terminology: a bibliography database file contains bibliographical *entries*, an article's 'References' section contains bibliographical *references*.

3. **M**ulti**L**ingual BibTEX.

1 Doing it with BibTEX

BibTEX was designed in the '80s and, as mentioned above, is suitable for 'modern' documentation: bibliographical entries refer to works that came out for a short period of time. Years are supposed to be written using the same format, in practice, either two digits[4] or four ones. As a counter-example, let us consider two bibliographical items specified this way:

```
@...{i0, ..., YEAR =  800, ...}
@...{i1, ..., YEAR = 1492, ...}
```

the i_0 item will be ranked after the i_1 item because '`800`' and '`1492`' are viewed as *strings* rather than numbers; a string begining with '`1`' takes precedence over a string beginning with '`8`'. In fact, the sort key used by BibTEX's standard bibliography styles for an entry is a concatenation of authors' names, year, and title[5]. As a consequence, any string may be associated with an entry's `YEAR` field. For example, we could write:

```
YEAR = {8th century BC}
```

That would not cause any error, BibTEX would not crash, but if there are other bibliographical entries with the same author name—e.g., some modern translations—the sort's result might be strange. For the same reasons, negative years are syntactically allowed within `YEAR` fields[6]:

```
YEAR = {-753}
```

but BibTEX does not handle them correctly.

The situation is better about uncertain identity for authors, because we can use a command for a dummy accent:

```
AUTHOR = {Mersenne Marin{\unsure}}
```

provided that this `\unsure` command has been defined within LaTeX documents using this entry. If this command is surrounded by additional braces within values associated with `AUTHOR` fields, BibTEX views it as an accent command and ignores it during the sorting step. However, let us notice that the two strings '`Mersenne Marin`' and '`Mersenne Marin{\unsure}`' are viewed equivalent during the sort step. That is, the 'actual' works of this author are merged with the works that could be attributed improperly[7].

4. When BibTEX was launched, this choice was very common: that moment was a long time before Y2K (the *Year 2000 problem* or *Millenium bug*).

5. This operation is fully explained in Hufflen (2008a).

6. Let us recall that in BibTEX, delimiters—braces or double quotes—can be omitted around natural numbers. If a '`-`' sign is used for a negative number, the complete value must be surrounded with delimiters. Concerning years, we also recall that the '`+`' sign is never used for a year AD (***A**nno **D**omini*), so only years BC (***B**efore **C**hrist*) are written using a sign.

7. In fact, BibTEX's sort procedure is not very discriminating (see Hufflen, 2008a for complete details). As another

2 BibTEX's successors

For a long time, BibTEX was unrivalled as *the* bibliography processor associated with the LaTeX word processor (Mittelbach and Goossens, 2004). However this program is ageing and difficultly meets some new requirements about bibliographies, as we explained in Hufflen (2011). So modern programs aim to replace it. In particular, they aim to get rid of BibTEX's language for bibliography styles (Patashnik, 1988b), old-fashioned and based on handling a stack. This replacement process is long, but more and more LaTeX users build their documents' bibliographies with the biblatex package (Lehman, 2014). The MlBibTEX program (Hufflen, 2003) is less widespread but some big-sized projects using it have succeeded.

2.1 The biblatex package

Let us recall that when the biblatex package (Lehman, 2014) is used, formatting 'References' sections is deferred to LaTeX[8]. So the files generated by BibTEX in this case contain texts marked up with LaTeX commands customised when this package is loaded[9]. A new bibliography processor, biber (Kime and Charette, 2014), is able to build files suitable for this package, and aims to replace BibTEX within this framework. When it works, biber is able to deal with more fields. For example, a `SORTYEAR` field can take precedence over the `YEAR` field when entries are sorted, even if only `YEAR` fields' contents appear within references[10]. For our purpose, these two fields `YEAR` and `SORTYEAR` might be used as follows:

```
YEAR = {154?}, SORTYEAR = 1549
```

that is, the `SORTYEAR` field allows such an entry to be ranked at the end of the entries originating from the same decade (1540's), whereas the `YEAR` field expresses that the unit digit is not really known. However, biber—like BibTEX—uses only string sorts, that is, negative years are not handled correctly. Besides, no syntax for authors' uncertain identity is provided.

2.2 MlBibTEX

Initially, MlBibTEX (Hufflen, 2003) was designed in order to provide a 'better' BibTEX, with particular focus on multilingual features. Later, it integrated other services, as detailed in Hufflen (2012b). When BibTEX's users experiment MlBibTEX, they immediately notice that the latter is less permissive than the former. A particular case of this feature: only non-zero integers—including negative ones[11]—are allowed as values associated with `YEAR` fields. This kind of *type-checking* has some advantages: for example, whenever field names are unknown, warning messages are emitted. On the contrary, BibTEX would have silently ignored such fields if they are optional. As another advantage, MlBibTEX can perform numerical sorts on numerical data, such as years[12]. At first glance, this feature runs counter to the introduction of more expressive power. But within MlBibTEX's source files, associating a field name with a new definition of the corresponding parsing function is easy. In fact, we developed MlBibTEX's parser in order for syntactical extensions to be added easily.

3 MlBibTEX's `inexact` extension

The features we present hereafter belong to the `inexact` extension. This extension can be used as an option of the commands `mlbibtex` and `mlbibtex2xml`[13]:

```
mlbibtex [-inexact] job-name
```

where '`[...]`' stands for an optional part and '*job-name*' refers to an auxiliary (`.aux`) file, as in 'old' BibTEX. The `inexact` extension increases expressive power about inexact years and uncertain identity, while performing strict type-checking for some specialised fields such as `YEAR`, `AUTHOR`, or `EDITOR`.

3.1 Inexact dates

Syntactically, a value associated with the `YEAR` field must be either an exact year, that is, an integer beginning with a non-zero digit and possibly preceded by the '`-`' sign, or an inexact one. If the prefix '`ca`' (for '*circa*') is used before an exact year, this prefix means 'around this date':

```
YEAR = {ca1492}          YEAR = {ca-429}
```

Other inexact years may be specified in means of *patterns*, in which case '`?`' is for an unknown digit. Examples:

'`154?`' means a year between 1540 and 1549
'`15??`' ..... 1500 ... 1599

An imprecise digit cannot be followed by a precise one. In other words, the question mark '`?`' cannot

example, BibTEX's standard bibliography styles do not use `MONTH` fields during the sorting step. As a consequence, works of the same authors and same years of publication cannot be sorted according to this month information.

8. The biblatex package has been introduced in Italian in Pantieri (2009).

9. A comparable approach exists within ConTEXt's `bib` module (CONTEXTGARDEN, 2012).

10. Let us mention that the biblatex package allows a *complete* date—that is, year, month, and day—to be specified by means of a `DATE` field, taking precedence over BibTEX's standard fields `YEAR` and `MONTH`. In practice, this `DATE` field is more and more used, but does not really apply for inexact dates.

11. But not zero, because there was no Year zero, Year 1 immediately succeeded Year −1.

12. Let us recall that on the contrary, BibTEX and biber only run lexicographic sorts.

13. See Hufflen (2012b) about the commands provided by MlBibTEX.

be followed a digit from 0 to 9, so '15?5' is an incorrect value for a YEAR field. The zero digit is allowed at the leftmost position if it is immediately followed by a question mark, e.g., '0?' is for a year between 1 AD and 9 AD, '-0?' is for a year between 9 BC and 1 BC.

The rules for MlBibTeX's increasing sort operation are organised as follows:

- an inexact year is ranked *after* an exact year being the same value; e.g.:

  1492 < ca1492 -429 < ca-429

- a pattern for an inexact year is ranked *after* the greatest year it can replace;

- inexact years are sorted according to *imprecision levels*, that is, the number of occurrences of the '?' sign. For example:

  1599 < ca1599 < 159? < 15??

because the imprecision levels of the last three—inexact—years are respectively 0, 1, and 2. If years before Christ are considered, applying the same rules yields:

-1500 < -ca1500 < -150? < -15??

Let us go back to syntactical rules about years and denote them by means of *regular expressions*, as allowed in programming languages like Perl[14] or taxonomy description languages like XML Schema (W3C, 2008). Exact years can be specified by the following regular expression[15]:

```
-?[1-9][0-9]*
```

Both exact and inexact years are described by the deterministic regular expression:

```
ca-?[1-9][0-9]* | -?[1-9][0-9]*\?* |
-?0\?+
```

3.2 Uncertain identity

An unknown author is specified by the '??' sequence. Using this sequence before an author name means that this identity is uncertain.

```
AUTHOR = {?? and Mersenne Marin}
```

is for two authors, the first is unknown, the second is Mersenne Marin. The notation:

```
AUTHOR = {?? Mersenne Marin}
```

means that the author's identity is surmised. You can also write:

```
AUTHOR = {?? Marin, Mersenne}
```

or put this '??' notation before MlBibTeX's keywords introducing a person name's parts (Hufflen, 2003):

```
AUTHOR = {?? first => Mersenne,
          last => Marin}
```

It is well-known that an organisation name can be used an author's name, provided that additional braces allow this organisation name to be viewed as the *Last* part of a name handled by BibTeX (Patashnik, 1988b):

```
AUTHOR = {{Gruppo Utilizzatori Italiani
           di \TeX}}
```

If the identity of such a name is uncertain, do not put the '??' sequence between braces—it would loose its special meaning—but proceed as follows:

```
AUTHOR = {?? {Gruppo Utilizzatori
              Italiani di \TeX}}
```

or use MlBibTeX's org keyword:

```
AUTHOR = {?? org => Gruppo Utilizzatori
                    Italiani di \TeX}
```

The rules for increasing sort operation are:

- an unknown author is ranked *before* any known one; e.g.:

  ?? < Claudio Beccari

- an author with uncertain identity is ranked *after* the same author when its identity is sure; e.g.:

  Mersenne Marin < ?? Mersenne Marin

4 References' look

Let us recall that MlBibTeX is written using the Scheme programming language (Kelsey et al., 1998) and running our parser on bibliography database files results in a structure that may be viewed as an XML[16] tree. Technical details related to our XML format fall out of the scope of this paper, but we mention that this format has been enlarged in order to include markers related to inexact years and uncertain identity. Likewise, MlBibTeX's initial library now includes Scheme functions able to deal with such data. For example, given an inexact year, such a function allows us to know its imprecision level. As a consequence, it is easy to write a function that displays the corresponding year or century with a question mark if the imprecision level is respectively 1 or 2. Let us recall some previous examples, applying this function when a 'References' section is built results in:

14. **P**ractical **E**xtraction and **R**eport **L**anguage (see Wall et al., 2000 for an introductory book about this language).

15. Readers unfamiliar with regular expressions can consult Stubblebine (2007) for a good introduction.

16. e**X**tensible **M**arkup **L**anguage.

`YEAR = {154?}`	$\Longrightarrow$	154?
`YEAR = {15??}`	$\Longrightarrow$	16th century
`YEAR = {-7??}`	$\Longrightarrow$	8th century BC

Likewise, the full power of a programming language easily allows us to display a negative year—`YEAR = {-753}`—as such or as a year BC:

-753 753 BC

At the present time, MlBibTeX's `inexact` extension is experimental. Since BibTeX's bibliography styles cannot deal with these new features, it is impossible to use the compatibility mode allowing such 'old' styles to be applied. So only 'new' styles using the nbst[17] language (Hufflen, 2003) could be updated. Another solution consists of bibliography styles wholly written using Scheme, the only style available now being an extension of plain, that is, references are sorted—as we showed above—and labelled by numbers[18]. The only way to customise this style is to redefine some Scheme functions[19]. Of course, we plan to go on with developing new styles and interfaces if users are interested. Criticisms and suggestions to do that are welcome.

5 Conclusion

On the one hand, we think that the features introduced by the `inexact` extension of MlBibTeX should remain optional. As mentioned above, most bibliographies—in particular, for recent works—are composed of items with precise metadata about dates and authors, and MlBibTeX's default type-checking procedures allow us to be ensured about that. On the other hand, the results we got within our experiments of this extension are encouraging, but we only tried basic examples. We could certainly improve our 'inexact' features, but we need further experiment in collaboration with people involved in ancient studies in order to go on with this direction.

A Sorting with MlBibTeX

In this annex, we do not detail all the features of the sort procedures used within MlBibTeX, we just aim to show how the framework described in Hufflen (2012b) has been extended.

Let us recall that when MlBibTeX processes an .aux files, it builds a list of citation keys, according to the order of first citations using these keys throughout the text. If the bibliography to be built is unsorted, we use this list as it is, as soon as corresponding entries are associated with each member. Otherwise, this list of citation keys is *sorted*. All the sorts performed by MlBibTeX are *stable*[20]; all the lexicographic sorts are language-dependent, the default language being English. The generation of such sorts is explained in Hufflen (2007) and additional details concerning person names are given in Hufflen (2008a).

As shown in Hufflen (2012b), all the order relations used by our sort procedures have a *thunk*[21] as an additional argument representing what is to be run when the two data are equal. This feature allows us to chain the use of several sort keys easily: if two data are viewed as equal by a sort key, the comparison based on the next sort key is immediately called. Our numerical order relations are written within this framework, so do our new comparison procedures including inexact dates. Lexicographic order relations have additional optional arguments controlling the sort's direction and the case comparisons (Hufflen, 2012b). The `<authors<?` function, allowing us to compare bibliographical items regarding authors or editors has been extended:

$$\texttt{((<authors<?}\ \mathit{rel?}\texttt{)}\ i_0\ i_1\ k_0\ f_0\ f_1\texttt{)}$$

where i_0 and i_1 are the two items to be compared, *rel?* the language-dependent order relation used to compare strings, k_0 the thunk that it called if i_0 and i_1 are equivalent. The new optional arguments f_0 and f_1 are respectively called when comparisons involve unknown authors and uncertain identity. Default values for f_0 and f_1 behave as we shown in § 3.2, but other functions can be used. In other words, processing such cases can be customised.

Often the order relation used by the sort procedure is closely related to the bibliography style used and does not appear explicitely when the bibliography processor is invoked. Practically, bibliographies are usually increasingly sorted with respect to authors' or editors' names, then dates, then titles. As shown in Hufflen (2012a,b), the `mlbiblatex` program, a variant of MlBibTeX that builds bibliographies for the biblatex package, uses another approach: the order relation used for sorting bibliographical items is given within the command line, by means of mnemonics. In a future version, we could do the same for the order relations used to sort inexact metadata.

Acknowledgements

I thank Claudio Beccari for his patience when he was waiting for this article, and for his Italian translations of the abstract and keywords. Many thanks also to Gianluca Pignalberi who proof-read a first version.

17. **N**ew **B**ibliography **ST**yles.

18. At the present time, if you use both the `inexact` extension and a `\bibliographystyle` command within your LaTeX document, a warning message is emitted, and this command is useless.

19. That can be done by means of Scheme expressions stored into a .smlbibtex file belonging to the current user's home directory.

20. That is, the relative order of items with equal sorting keys is maintained.

21. A zero-argument function, with respect to Scheme's terminology.

References

CONTEXTGARDEN. *Bibliographies in MkIV*. http://wiki.contextgarden.net/Bibliography_mkiv, July 2012.

Bruno Delprat and Stepan Orevkov. MayaPS: Maya hyeroglyphics with LaTeX. TUGboat, 33(3):289–294, 2012.

Hesiod. *Theogony — Works and Days [Θεογονία — Ἔργα καὶ ἡμέραι]*. Oxford University Press, 1988. Translation from ancient Greek by Martin Lichtfield West.

Jean-Michel Hufflen. MlBibTeX's version 1.3. TUGboat, 24(2):249–262, July 2003.

Jean-Michel Hufflen. Managing order relations in MlBibTeX. TUGboat, 29(1):101–108, 2007. EuroBachoTeX 2007 proceedings.

Jean-Michel Hufflen. Revisiting lexicographic order relations on person names. In *Proc. BachoTeX 2008 Conference*, pages 82–90, April 2008a.

Jean-Michel Hufflen. Specifying translated works in bibliographies. *ArsTEXnica*, 6:93–97, October 2008b. In Proc. GUIT 2008 meeting.

Jean-Michel Hufflen. A comparative study of methods for bibliographies. TUGboat, 32(3):289–301, October 2011. Proc. TUG 2011 conference, Trivandrum, India.

Jean-Michel Hufflen. MlBibTeX and its new extensions. In *Proc. 6th ConTEXt Meeting & EuroTEX 2012*, pages 82–91, Breskens, The Netherlands, October 2012a.

Jean-Michel Hufflen. Beyond BibTeX. *ArsTEXnica*, 14: 7–14, October 2012b. In Proc. GUIT meeting 2012.

Richard Kelsey, William D. Clinger, and Jonathan A. Rees, with Harold Abelson, Norman I. Adams iv, David H. Bartley, Gary Brooks, R. Kent Dybvig, Daniel P. Friedman, Robert Halstead, Chris Hanson, Christopher T. Haynes, Eugene Edmund Kohlbecker, Jr, Donald Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy Lewis Steele, Jr, Gerald Jay Sussman, and Mitchell Wand. Revised^5 report on the algorithmic language Scheme. HOSC, 11(1):7–105, August 1998.

Philip Kime and François Charette. *biber. A Backend Bibliography Processor for biblatex. Version biber 1.9 (biblatex 2.9)*. http://ftp.oleane.net/pub/CTAN/biblio/biber/documentation/biber.pdf, May 2014.

Philipp Lehman, with Philip Kime, Audrey Boruvka, and Joseph Wright. *The biblatex Package. Programmable Bibliographies and Citations. Version 2.9a*. http://ctan.mirrorcatalogs.com/macros/latex/contrib/biblatex/doc/biblatex.pdf, June 2014.

Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig, and Joachim Schrod. *The LaTeX Companion*. Addison-Wesley Publishing Company, Reading, Massachusetts, 2 edition, August 2004.

Lorenzo Pantieri. L'arte di gestire la bibliographia con biblatex. *ArsTEXnica*, 8:48–60, October 2009.

Oren Patashnik. *BibTEXing*. Part of the BibTeX distribution, February 1988a.

Oren Patashnik. *Designing BibTEX Styles*. Part of the BibTeX distribution, February 1988b.

Gianluca Pignalberi, Salvatore Schirone, and Jerónimo Leal. Introduzione agli strumenti per grecisti classici. *ArsTEXnica*, 11:15–30, April 2011.

Gianluca Pignalberi, Salvatore Schirone, and Jerónimo Leal. Ancora sugli strumenti per grecisti classici. *ArsTEXnica*, 13:11–16, April 2012.

Tony Stubblebine. *Regular Expression Pocket Reference*. O'Reilly, 2 edition, July 2007.

W3C. XML *Schema*. http://www.w3.org/XML/Schema, December 2008.

Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl*. O'Reilly & Associates, Inc., 3 edition, July 2000.

▷ Jean-Michel Hufflen
FEMTO-ST (UMR CNRS 6174) & University of Franche-Comté,
16, route de Gray,
25030 Besançon CEDEX
France
jmhuffle@femto-st.fr

Greek and Latin hyphenation – Recent advances

Claudio Beccari

Abstract

A recent package of mine revealed some unacceptable inconveniences in the hyphenation of the Greek language: *pdflatex*, that can manage only 8-bit encoded fonts, could not differentiate between the actual three varieties of Greek (the modern monotonic, the modern polytonic, and the ancient polytonic ones) and could hyphenate more or less correctly provided that the source text was input with the special Latin transliteration connected with LGR encoded fonts.

The hyphenation patterns for the three varieties had been part of any complete format file for years, but the Greek language description file used only the modern polytonic Greek patterns.

The problem was detected; the solution was found; work is in progress in order to perform a fine tuning of the new facilities.

At the same time the hyphenation patterns for standard Latin did not work well for classical Latin; the professional latinists did not work well with LaTeX and the existing hyphenation patterns. A new set of patterns was developed an a suitable set-up was created in order to switch pattern set according to the type of Latin that is being typeset.

The solutions that have been found for Greek and Latin are similar, and for this reason we deal with them in a single article.

Sommario

Mediante un mio recente pacchetto è venuto alla luce un problema relativo alla sillabazione del greco, nelle sue tre varietà, quella del greco moderno monotonico, di quello moderno politonico e di quello antico politonico, che si manifesta quando si compone il testo con *pdflatex*, il quale riconosce solo i font con codifiche di otto bit. Di fatto da diversi anni i file di formato contengono i pattern di sillabazione delle tre varietà di greco, ma il file di descrizione di questa lingua usa solo quelli per il greco moderno politonico; inoltre questi pattern funzionano correttamente solo se il testo sorgente è scritto mediate la traslitterazione latina e i font con la codifica speciale LGR.

Il problema è stato individuato; è stata trovata una soluzione; il lavoro è ancora in corso d'opera per mettere a punto i nuovi file.

Nello stesso tempo i pattern di sillabazione per il latino corrente non erano adatti per comporre testi in latino classico; i latinisti non potevano lavorare bene con LaTeX e i pattern esistenti. È stato creato un nuovo set di pattern e si è creato un tipo nuovo di impostazione per cambiare i pattern da usare a seconda del tipo di testo latino che si sta componendo.

Le soluzioni trovate per le due lingue sono simili e vengono raccolte in quest'unico articolo.

1 Introduction

1.1 Greek

The TeX system is continuously enriched with new facilities for typesetting in other languages, also those that are being typeset with glyph sets that are not part of the so called *Latin alphabet*.

Greek has been supported for some 15 years now; at the beginning there existed only one font created by Silvio Levy (Levy and Murphy, 2010) but no hyphenation patterns nor any language description file did exist.

I created an almost full collection of Computer Modern and Latin Modern compatible Greek bitmapped fonts suited for typesetting with *latex* and, later on, vector fonts for use with *pdflatex*; since both programs are only 8-bit-encoding aware typesetting ones, they could manage only fonts containing just 256 glyphs. Actually for typesetting Greek about 300 glyphs should be necessary. I was aware that a complete solution was described in Mylonas and Witney (1992), but my fonts became the *de facto* default Greek fonts since the beginning of the nineties.

Apostolos Syropoulos wrote the first version of the Greek language description file, in order to access the babel package facilities (Braams and Bezos, 2014; Syropoulos and Milde, 2014). I wrote an initial pattern file for typesetting in polytonic (ancient) Greek; this file was supposed to be adequate for the western European scholars who need to write ancient Greek, without having available a Greek polytonic keyboard. The Greek language description file was updated several times and a new maintainer, Günter Milde, is taking care of updating/upgrading it. Version 1.9 of this language description file (Syropoulos and Milde, 2014) has just been uploaded to CTAN; it contains not only the possibility of selecting the specific variant of the Greek language, but it allows also the direct input of literal Greek.

My Greek hyphenation pattern file was taken care by a succession of Greek maintainers and eventually Dimitrios Filippou became the actual maintainer (Filippou, 2004). My file was eventu-

ally substituted with three other different pattern files, each one suited for the corresponding Greek language variant. But all these files still referred the patterns to the Latin transliteration; they would not work with direct input of literal Greek text.

Meanwhile *xetex* and *xelatex* were introduced in the TEX system; after a short time also *luatex* and *lualatex* were added. These typesetting engines can work with OpenType UNICODE encoded fonts; at the same time they have a more performant system for managing languages, realized through polyglossia (Charette and Reutenauer, 2011) and its commands for selecting languages and their variants. Apostolos Syropoulos decided to support the OpenType fonts and polyglossia, and of course there are no problems as those exhibited by 8-bit encoded fonts and by the babel package with its collection of language description files.

Actually babel can be used also with *xelatex* and *lualatex*; and the `babel-greek` file bundle has a partial support also for being used with the latter typesetting engines. But I suppose that when these two typesetting engines are used, most if not all users, prefer using polyglossia and its special selecting commands that accept the variant specifications by means of *key* = *value* options.

This was the situation by the end of June 2014. Then the problem was spotted; a dense e-mail exchange between the official actors (Günter Milde, Dimitrios Filippou, Mojca Miklavec) and myself was started; we decided a plan to correct this situation, and things are under way to be fine tuned and released to CTAN.

1.2 Latin

The modern and medieval varieties of Latin have received babel and polyglossia support in the past twenty years or so. Classic Latin was not supported. As the author of the existing files, I actually did not know that classic Latin had to receive such a different treatment from the medieval variety. My only excuse is that I am neither a latinist nor a linguist.

Recently I gave a seminar on LATEX to a group of international PhD students in humanities who complained about the fact that they could not use the Latin support for typesetting classic Latin, because spelling and hyphenations were not suited for that Latin variety.

I sought support from a real latinist: my old-time friend Raffaella Tabacco, professor at the university of Vercelli, Italy, and director of the local Department of Humanities that, besides other important activities, is the seat of digilibLT (Biblioteca digitale di testi latini tardo-antichi, *Digital library of late-ancient Latin texts*). My friend, whom I warmly thank for her kind help, gave me access to a book, that is now difficult to find, that contains the hyphenation rules for classic Latin (Farina and Marinone, 1979).

This book, actually, is not very deep in the question of hyphenation, but it gives the essential grammar rules and good advice for the composition of Latin texts and theses on subjects that deal with classic Latin.

In the past I became sort of expert in transforming grammar hyphenation rules into TEX code. I am not a guru nor a wizard in this field, but I have already done this work for more than a dozen languages; some hyphenation pattern files have not been uploaded to CTAN; some of them have been uploaded to CTAN but have been superseded by newer and more correct ones created by other experts; some have been prepared for LATEX 2.09, and of course they became obsolete as soon as LATEX 2_ε became available; the advent of program *xetex*, with its special packages, among which polyglossia, required some upgrading, but in general the existing hyphenation patterns remained available, although with minor but essential adaptations.

I thought I should not encounter any problem in preparing the classic Latin hyphenation patterns, but I did; actually classic Latin requires etymological hyphenation; this kind of hyphenation style requires huge pattern files that have nothing or little to do with grammar rules, in the sense that there is no grammar rule that says what is a prefix or a suffix, or a compound word, or where a certain word comes from. A complete dictionary should be examined and the set of words should be completed with all the flexed forms of verbs, nouns, adjectives, pronouns, and the like.

In this paper I discuss first the problems concerning Greek and the solutions I found; I show some results that I temporarily obtained from the provisional files I used for my tests. Then I discuss what concerns classical Latin and the solutions I found to switch hyphenation patterns when classic Latin is being typeset.

By the time when this paper is available Greek with babel should be almost fixed or shall be completely fixed in a short time with the new patterns files tested and verified by Dimitrios Filippou and his team. With polyglossia there should not be any problem.

At the same time the Latin set up with the three varieties of Latin, modern, medieval, and classical, are fixed when using *pdflatex* and babel, while with polyglossia the upgrading of the `gloss-latin.ldf` language definition file and the new patterns will take e little more time.

2 The LGR encoded Greek fonts

Before going into the details it is necessary to recall the properties of the 8-bit LGR encoded Greek fonts[1]. A table is shown in figure 1.

1. The encoding names starting with L are considered "Local", but for the GReek fonts the LGR encoding has become the *de facto* standard for Greek fonts; any TEX

	’00	’01	’02	’03	’04	’05	’06	’07	’10	’11	’12	’13	’14	’15	’16	’17	
’000	– 0	◌͡ 1	𝛢 2	𝛨 3	𝛸 4	𝙼 5	ϛ 6	ϗ 7	ι 8	Αι 9	Ηι 10	Ωι 11	Α 12	Ϋ 13	α 14	ϋ 15	"00
’020	◌́ 16	◌̀ 17	ι 18	ϙ 19	◌̆ 20	ϙ 21	Ϝ 22	ϡ 23	ϵ 24	‰ 25	ə 26	⅁ 27	‘ 28	’ 29	◌̑ 30	– 31	"10
’040	◌̔ 32	! 33	◌̈ 34	◌́ 35	◌̀ 36	% 37	· 38	ʼ 39	(40	) 41	* 42	+ 43	، 44	- 45	· 46	/ 47	"20
’060	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	· 59	◌̊ 60	= 61	◌̓ 62	; 63	"30
’100	◌̔ 64	A 65	B 66	◌̓ 67	Δ 68	E 69	Φ 70	Γ 71	H 72	I 73	Θ 74	K 75	Λ 76	M 77	N 78	O 79	"40
’120	Π 80	X 81	P 82	Σ 83	T 84	Υ 85	◌̔ 86	Ω 87	Ξ 88	Ψ 89	Z 90	[91	◌̔ 92	] 93	◌̆ 94	◌̓ 95	"50
’140	` 96	α 97	β 98	ς 99	δ 100	ε 101	φ 102	γ 103	η 104	ι 105	ϑ 106	κ 107	λ 108	μ 109	ν 110	ο 111	"60
’160	π 112	χ 113	ρ 114	σ 115	τ 116	υ 117	118	ω 119	ξ 120	ψ 121	ζ 122	« 123	، 124	» 125	◌̃ 126	— 127	"70
’200	◌̀ 128	◌́ 129	◌̇ 130	◌̃ 131	◌̔ 132	◌̔́ 133	◌̔̀ 134	◌̔̃ 135	◌́ 136	◌̆́ 137	◌̆̀ 138	◌̆̃ 139	◌̔́ 140	◌̔̆́ 141	◌̔̆̀ 142	◌̔̆̃ 143	"80
’220	◌̃ 144	◌̆́ 145	◌̆̀ 146	◌̆ 147	◌̔̃ 148	◌̔̆ 149	◌̔̆́ 150	◌̑ 151	◌̓̀ 152	◌̓́ 153	◌̓̃ 154	155	◌̓̀ 156	◌̓́ 157	◌̓̃ 158	159	"90
’240	◌́ 160	◌̈́ 161	◌̈̀ 162	◌̈̃ 163	◌́ 164	◌̈́ 165	◌̈̀ 166	◌̈̃ 167	◌̃ 168	◌̈́ 169	◌̈̀ 170	◌̈̃ 171	◌̃́ 172	◌̃̈́ 173	◌̃̈̀ 174	◌̃̈̃ 175	"A0
’260	◌̀ 176	◌́ 177	◌̇ 178	◌̃ 179	◌̔ 180	◌̔́ 181	◌̔̀ 182	◌̔̃ 183	◌́ 184	◌̆́ 185	◌̆̀ 186	◌̆̃ 187	◌̔́ 188	◌̔̆́ 189	◌̔̆̀ 190	◌̔̆̃ 191	"B0
’300	◌̃ 192	◌̆́ 193	◌̆̀ 194	F 195	◌̔̃ 196	◌̔̆ 197	◌̔̆́ 198	199	◌̓̀ 200	◌̓́ 201	◌̓̃ 202	◌̑ 203	◌̓̀ 204	◌̓́ 205	◌̓̆ 206	◌̓̆ 207	"C0
’320	◌́ 208	◌̈́ 209	◌̈̀ 210	◌̈̃ 211	◌́ 212	◌̈́ 213	◌̈̀ 214	◌̈̃ 215	◌̃ 216	◌̈́ 217	◌̈̀ 218	Ϊ 219	◌̆ 220	◌̆́ 221	◌̆̀ 222	Ϋ 223	"D0
’340	◌̀ 224	◌́ 225	◌̇ 226	◌̃ 227	◌̀ 228	◌́ 229	◌̇ 230	◌̃ 231	◌́ 232	◌̆́ 233	◌̆̀ 234	◌̆̃ 235	◌́ 236	◌̆́ 237	◌̆̀ 238	◌̆̃ 239	"E0
’360	◌̈ 240	◌̀ 241	◌́ 242	◌̃ 243	◌̈ 244	◌̀ 245	◌́ 246	◌̃ 247	◌̔ 248	◌̓ 249	◌̔ 250	◌̔ 251	◌̔ 252	253	◌̓ 254	◌̓ 255	"F0
	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F	

Font: grmn1000.

ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ

αβγδεζηθικλμνξοπρςστυφχψω

Font parameters	
slant per pt	0.0pt
interword space	3.33252pt
interword stretch	1.66626pt
interword shrink	1.11084pt
x-height	4.3045pt
quad width	9.99756pt
extra space	1.11084pt

Figure 1: The layout of the LGR encoded Greek fonts.

Almost all the 256 positions in the table contain a glyph; some of these glyphs are very unusual in any font, even in the OpenType ones, for example the four glyphs at code points 2–5, that represent the Attic numerals; for example the capital Alpha, Eta and Omega with adscript capital iota at code points 9–11.[2] Some cells appear to be empty, but they are not; for example code point 118 appears to be empty, but it contains an invisible character with vanishing width but as high as a lower case letter without ascenders. It has the usual specification of normal lower case characters, it takes place in the hyphenation process and forbids the lower case closed sigma to see the end of a word; in facts the code point 115 (corresponding to the Latin letter ‘s’) is such that if this letter is used at the end of a word, instead of the open sigma at code point 99 (corresponding to the Latin letter ‘c’), it detects the end of the word and changes itself to an open sigma by means of a ligature process; in fact, in order to typeset an isolated closed sigma it is necessary to input `sv` so as to get σ, because if you enter just `s` you get ς.

The ligature process is used very often with these LGR encoded fonts; the actual ligature commands are contained in the `.tfm` files as well as the kerning commands and the individual glyph dimensions. By this ligature process it is possible to use only the Latin keyboard keys corresponding to the ASCII characters in order to input suitable character sequences that produce “complicated” Greek glyphs; for example by setting `>’a|` in the source file, the output will result in ᾷ.

This process is very handy, but it has a drawback: since the ligature mechanism can process just two characters at a time, the ligated glyph is not kerned

system distribution contains also other packages that allow using the so called BETA code.

2. The decimal code points are written within each table cell; the octal code points are obtained by adding the left row index with the top column index of each cell; the hexadecimal code points are obtained by adding the right row index and the bottom column index.

any more; in other words kerning commands and ligature commands interfere with each other so that each command of one kind forbids the action of the other kind.

For this reason it would be better to have a direct access to pre-composed characters at typesetting level, not at the font level. This is what I tried to do in my teubner package (Beccari, 2013a,b), but Günter Milde introduced a much more versatile method that heavily relies on advanced LaTeX core font commands such as `\DeclareTextCommand`, `\DeclareTextComposite` and the like.

In version 1.8 of the Greek language description file, Milde introduced a new idea: the direct access to actual pre-composed glyphs could be extended by making it independent from the input encoding of the source file. This implied the definition of intermediate macros, the so called LICR macros (LICR stands for "LaTeX Internal Character Representation") so that input characters could be mapped to the LICR macros, which in turn select the pre-composed characters. In this way the literal input of Greek text becomes possible. For example with the previous method, in order to get

> Τῃ πάντα διδούσῃ καὶ ἀπολαμβανούσῃ φύσει ὁ πεπαιδευμένος καὶ αἰδήμων λέγει· «δὸς, ὃ θέλεις, ἀπόλαβε, ὃ θέλεις». Λέγει δὲ τοῦτο οὐ καταθρασυνόμενος, ἀλλὰ πειθαρχῶν μόνον καὶ εὐνοῶν αὐτῃ.

you had to typeset[3]:

```
Th| p\'anta dido\'usih| ka\`i
\>apolambano\'ush| f\'usei
\<o pepaodeum\'enos ka\`i
a\>id\'hmwn l\'egei; ((\<`o
j\'eleis, \>ap\'olabe, \<`o
j\'eleis)). L\'egei d\`e
to\~uto o\>ukatajrasumn\'omenos,
\>all\`a peijarq\~wn m\'onon ka\`i
e\>uno\~wn a\>uth|.
```

With the new LICR method you simply enter the Greek text in the source file and you need not use any diacritic macro to select the pre-composed characters. Of course you need to specify the `utf8` input encoding, but the rest is done directly by the LICR macros.

Obviously in order to do any literal Greek input you need a suitable real or virtual keyboard, unless you copy and paste some text from other electronic documents. Virtual keyboards today are not uncommon, so the first method is actually usable. The copy and paste method is dangerous, because even if the source file to copy some text from appears to contain pre-composed characters, they might not be pre-composed but they might be formed with *self combining diacritics*, which superimpose themselves on the base characters; a sort of mechanism similar to the process used also for Latin diacritics when using *pdflatex* with the obsolete OT1 encoding.

Here I would not enter into the details of the Greek-Latin transliteration, but the example above gives a general insight into this method. Notice that the diacritic macros, as well as the LICR macros at the moment do not access directly the glyphs with iota subscript but resort to the ligature process. Among the fine tuning actions, this is one to be dealt with. The postfixed position of the ASCII symbol for iota subscript does not harm the ligature-kerning process and works pretty well; nevertheless it would be a more general approach if the LICR macros could access also the glyphs with iota subscript.

3 The Greek language description file

As it was previously remarked Milde has already published version 1.9 of `greek.ldf` file. He already added the definitions for the attribute `ancient`, but he also declared the monotonic Greek variant as the default one; with version 3.9 of babel the attributes can be used also as modifiers, therefore the user selects one of the three Greek variants with one of these three self commenting specifications:

```
\usepackage[greek,english]{babel}
\usepackage[greek.polutoniko,english]{babel}
\usepackage[greek.ancient,english]{babel}
```

in the preamble of a document where the main language is English, and the secondary language is one of the three Greek variants. Needless to say that each variant is exclusive and it would be an error to specify both `polutoniko` and `ancient`.

It may be worth noting that the obsolete "pseudo" language `polutonikogreek` might not be supported any more, not even for backwards compatibility with legacy documents. In any case changing the option to babel from `polutonikogreek`[4] to `greek.polutoniko` is not a difficult task for the user who needs to compile again a legacy document.

4 The hyphenation pattern files

The three existing hyphenation pattern files for the three Greek variants to be used by the 8-bit engines are named `grahyph5.tex`, `grmhyph5.tex`,

3. With the ligature mechanism you would save all the backslashes but, with certain fonts and certain glyphs, you would notice some kerning glitches.

4. Günter Milde informed me that there exist several programs that can produce LaTeX files and that still use the `polutonikogreek` "pseudo" language, in spite of the fact that this name is being kept in the Greek language description file only for legacy documents, and its use is deprecated; among these programs there is the well known *LyX* word processor.

Table 1: Single and combined diacritics.

diacritics	◌̓	◌̔	◌̈
accents	◌́	◌̀	◌͂
combined diacritics	◌̓́	◌̔̀	◌̈͂
	◌̓̀	◌̔́	◌̈́
	◌̓͂	◌̔͂	◌̈̀
iota subscript	◌ι		

`grphyph5.tex`: the third letter in their names recalls the initial letter of the specific variant.

Since pattern files must contain only lowercase letters[5] and no expandable macros, the construction of pattern files is pretty complicated if some non ASCII glyphs have to take part in the process.

Th Greek LGR encoded fonts cannot work if only the ASCII code points are used; possibly monotonic Greek can get along with such limitations when using the ISO-8859-7 encoding[6] that is being used by a normal Greek keyboard. So for both polytonic varieties it is necessary to address the upper half plane of the 8-bit font encoding.

This requires to assign a lower case code to all diacritics, the simple ones and the combined ones; see table 1.

And since these simple and combined diacritics must appear in the pattern files as ASCII glyphs, they must be set with their lower case codes.

For monotonic greek it is necessary to set the lower case code also to the seven accented letters with the single acute accent and the iota and upsilon with just the diaeresis or with the diaeresis and acute accent. With both polytonic varieties it is necessary to set also the lower case code of approximately 120 upper plane glyphs that include all the available base letters with all combinations of allowed diacritics.

This second part was missing from the `gr*hyph5.tex` files. The new `gr*hyph6.tex` files had to be completed with the lower case codes of all the available letters. But this was not sufficient.

The actual patterns included in the `gr*hyph5.tex` files contained the diacritics only as simple ASCII characters, not as diacritic macros (forbidden in the pattern files), nor with the actual code points of the pre-composed glyphs. The upper plane ones are nor representable with ASCII codes, therefore it is necessary to use their numerical code points. The most cryptic representation of the numerical code point is the double caret lower case hexadecimal representation, where, for example, `^^ba` addresses the Greek glyph ὦ.

5. For use within the pattern files a *letter* is any glyph with a positive lower case code. Several languages assign a positive lower case code to analphabetic characters, such as the apostrophe, for example; Italian is one of these languages.

6. The ISO standard for monotonic Greek.

This double character representation, in spite of being so cryptic, is useful because it does not require spaces after control sequence names, and its two hexadecimal digits are well defined as belonging to the glyph address, and cannot be confused with the numerical digits used in the patterns to identify possible allowed or prohibited hyphen points.

Therefore the pattern `gr*hyph6.tex` files had to be enriched, compared to the existing `gr*hyph5.tex`, with the new patterns containing the double caret hexadecimal code points of the pre-composed glyphs. The `grmhyph6.tex` required very little work, because the patterns to add were relatively few. The `grphyph6.tex` file required a very attentive long editing, because the new patterns to add were several hundred ones. The `grahyph6.tex` file required several days of work, because there were about 2000 patterns to add.

5 Testing

I run some tests on monotonic, modern and ancient polytonic texts and found that several patterns were missing; not only the new ones with the pre-composed glyph double caret code points, but also the previous ones based on the ligature mechanism. The problem was very delicate with the ancient Greek variant. The difficulty arises from the fact that prefixes and suffixes are hyphenated on an etymological base; patterns become increasingly more complicated, because they create several variants for the same prefix or suffix and the same word stems where accents change position according to the word declination or conjugation.

Moreover Greek is perhaps the only language where typographical hyphenation can take place after the initial vowel and before the ending vowel. This feature implies more detailed patterns and certainly it does not simplify the whole process. Notice that when I created the hyphenation patters for Italian I worked as if it were possible to hyphenate after the initial vowel and before the ending vowel, but I set the minimum length of the first and last syllable as *two* letters; if the user needs more break points, s/he can locally set the `\lefthyphenmin` and `\righthyphenmin` counters to one, but by default they equal two. This said, the hyphenation trie for Italian contains about 340 patterns and 37 ops; for ancient Greek more than 5000 patterns are necessary end they require 122 ops. In spite of a very similar sonority, the two polytonic Greek variants require much more attention due to the multitude of diacritics and the etymological hyphenation.

I introduced the modifications I considered necessary; since I am not a Greek speaker, but I am just interested in this language that I happened to study in high school (hyphenation was not part of the syllabus...), I think I did the best I could,

but what I did is certainly very far from perfect; of course now the work in progress consists in a full control of the new Greek pattern files performance, and Dimitrios Filippou is working on that.

At the same time it seems that the UNICODE compliant pattern files created for the three Greek variants to be typeset with UNI-aware typesetting engines could be converted to LGR encoding with a suitable conversion program, similar to the ones used with the other existing UNICODE to ⟨*encoding*⟩ conversion programs. Mojca Miklavec volunteered to create such program. If this conversion program can do a good job, may be that what I did becomes automatically obsolete. In facts with a good conversion program it is possible to maintain only the UNICODE compliant pattern files, and the LGR compliant ones can be derived directly from the updated/upgraded UNICODE files.

But my work represents a feasibility proof that the now existing pattern files can be edited to become suitable for typesetting with 8-bit aware engines, *pdflatex* in particular, while, at the moment, they are not suited for this purpose.

6 Examples

Here some hyphenated words are shown with words taken from the comment lines of the `grahyph6.tex` pattern file. Supposedly they will be hyphenated correctly if the ancient Greek language is selected; but we use the same set of words to show the hyphen points also with the other two variants; of course the words for monotonic are input with a single tonic accent and/or dialitika. Here is the list.

ἀδυσώπητος ἀναίδεια ἀναμφισβήτητος
Ἀναξάνδρος ἄνοικος ἀπρόσκοπος
ἀπρόσκοπτος εἰσίν ἐκλείσαμεν ἐκληροῦτον
ἐκρυστάλλισα Ἑλλήσποντος ἐνίδρωσα
εὐσύνετος καλωσόρισα μελανόρισα
μελανόμματος μαγισαψεδάφα ξυναγείρατο
παλινάγρετος πανισδόμην προσεισπτάσσω
προϋπεξορμάω ὑοσκύαμος ὑπεκλήψομαι
χαρίσανδρος δισχίλιοι

The testing results are shown in figures 2, 3, and 4.

Comments

It may be noticed that words with an initial upper case precomposed vowel require patterns that need an hybrid approach between ASCII patterns and LGR literal Greek transformation into double caret code points; for example, the word Ἀναξάνδρος requires three patterns, the first containing only ASCII characters, `>an'a2x1an`; the second with only double caret code points, `^^82n^^882x1an`; the third in hybrid form, `>an^^882x1an`. This is due to the fact that before hyphenating, the typesetting engine internally changes the word to be hyphenated to lowercase; but since diacritics with the

αδυ-σώ-πη-τος
αναί-δεια
αναμ-φι-σβή-τη-τος
Ανά-ξαν-δρος
άνοι-κος
απρό-σκο-πος
απρό-σκο-πτος
ει-σίν
εκλεί-σα-μεν
εκλη-ρού-τον
εκρυ-σ τάλ-λι-σα
Ελ-λή-σπο-ντος
ενί-δρω-σα
ευ-σύ-νε-τος
κα-λω-σό-ρι-σα
με-λα-νό-ρι-σα
με-λα-νόμ-μα-τος
μα-γι-σα-ψε-δά-φα
ξυ-να-γεί-ρα-το
πα-λι-νά-γρε-τος
πα-νισ-δό-μην
προ-σει-σπτάσ-σω
προ-ϋ-πε-ξορ-μάω
υο-σκύ-α-μος
υπε-κλή-ψο-μαι
χα-ρί-σαν-δρος
δι-σχί-λιοι

Figure 2: Hyphenation test in monotonic Greek.

ἀ-δυ-σώ-πη-τος
ἀ-ναί-δεια
ἀ-ναμ-φι-σβή-τη-τος
Ἀ-νά-ξαν-δρος
ἄνοι-κος
ἀ-πρό-σκο-πος
ἀ-πρό-σκο-πτος
εἰ-σίν
ἐκλεί-σα-μεν
ἐκλη-ροῦ-τον
ἐκρυ-στάλ-λι-σα
Ἑλ-λή-σπον-τος
ἐνί-δρω-σα
εὐ-σύ-νε-τος
κα-λω-σό-ρι-σα
με-λα-νό-ρι-σα
με-λα-νόμ-μα-τος
μα-γι-σα-ψε-δά-φα
ξυ-να-γεί-ρα-το
πα-λι-νά-γρε-τος
πα-νισ-δό-μην
προ-σει-σπτάσ-σω
προ-ϋπε-ξορ-μά-ω
ὑ-ο-σκύ-α-μος
ὑ-πε-κλή-ψο-μαι
χα-ρί-σαν-δρος
δι-σχί-λιοι

Figure 3: Hyphenation test in polytonic modern Greek.

ἀ-δυσ-ώ-πη-τος
ἀν-αί-δει-α
ἀν-αμ-φισ-βή-τη-τος
Ἀ-νάξ-αν-δρος
ἀ-νάξ-αν-δρος
ἄν-οι-κος
ἀ-πρό-σκο-πος
ἀ-πρό-σκο-πτος
εἰσ-ίν
ἐ-κλεί-σα-μεν
ἐ-κλη-ροῦ-τον
ἐ-κρυ-στάλ-λι-σα
Ἑλ-λήσ-πον-τος
ἐλ-λήσ-πον-τος
ἐν-ί-δρω-σα
εὐ-σύν-ε-τος
κα-λωσ-ό-ρι-σα
με-λα-νό-ρι-σα
με-λαν-όμ-μα-τος
μο-γισ-α-ψε-δά-φα
ξυν-α-γεί-ρα-το
πα-λιν-ά-γρε-τος
πα-νισ-δό-μην
προσ-εισ-πτάσ-σω
προ-ϋπεξ-ορ-μά-ω
ὑ-οσ-κύ-α-μος
ὑ-πεκ-λή-ψο-μαι
χα-ρίσ-αν-δρος
δισ-χί-λι-οι

Figure 4: Hyphenation test in polytonic ancient Greek.

initial capital letter of a word do not get ligated in a single character with the diacritic over the base letter, but before the base letter, the pattern must contain the initial letter in ASCII form as if the diacritic had to be ligated; but the ligature takes place at the font level, while direct access to the upper case accented character takes place before the paragraph construction, and therefore before the hyphenation algorithm gets executed.

Even if the hyphenation results of the ancient Greek variant appears pretty unusual, it has to be so. The file I modified contained exactly those hyphen points and Dimitrios Filippou, who supervised and contributed to the creation of the `grahyph5.tex` file, knows exactly how this Greek variant has to be processed; he explained the whole operation that the Greek team brought to a conclusion in his chapter *Hyphenation patterns for ancient and modern Greek* included in the book by Apostolos Syropoulos (Filippou, 2004).

7 Differences between Latin and classic Latin

The existing Latin pattern files that I prepared some 20 years ago, dealt with modern and medieval Latin with just one set of patterns; the language description file for Latin provided differently spelled infix words such as *Praefatio* and *Præfatio*, or *novembris* and *nouembris*; and the pattern files were set up to recognize the use of the letter 'u' in place of the letter 'v' as used in modern spelling. The hyphenation rules were taken from an high school grammar, that synthetically stated that the rules are identical to those for Italian; I verified how such rules were followed in in a "professional" book, typeset in modern spelling, *Novum testamentum graece et latine*, (Merk, 1984), and I verified that such rules were actually followed in a systematic way, except very few inconsistent instances such as *re-gnum* vs. *reg-num*, both being present in different parts of the book.

8 Classic Latin hyphenation rules

Classic Latin is treated in a different way.

The rules reported by Farina and Marinone (1979) are the following:

1. The classic Latin alphabet contains only the following 23 letters: *a b c d e f g h i k l m n o p q r s t u x y z* with the corresponding upper case ones, with the exception that *V* is the upper case version of *u*. Supposedly letters *y z* appear only in words deriving from Greek. Moreover letters *j v* do not exist, so that in a correct classic spelling they should be substituted with *i u* respectively.

2. Two vowels can be divided if they do not form a diphthong, but it is preferable to avoid such divisions; for example *re-us cre-a-re gau-di-um* should remain *reus crea-re gau-dium* respectively.

3. Single consonants, the graphemes *ch ph rh th qu*, the letters *x z*, and *i u* when they have the value of a consonant, remain attached to the following vowel; for example *ro-sa me-ta mo-ri-tu-rus phi-lo-so-phus e-quus ae-qui-tas di-xi a-zy-mus ie-iu-ni-um ui-ue-re.*

4. Groups of two consonants where the first is one of *p b t d c g f* and the second one is one of *l r*, are never divided; exceptionally also *gn* is not divided; for example *du-plum pu-bli-cus pa-tres sa-cra a-grus ua-frum sta-gnum.*

5. Any other group of two or more consonants, irrespective if they are equal or different, are divided just before the last one, except when the last two consonants fall in the previous case; for example *ag-ger es-se uel-le fac-tus op-ti-mus ag-men om-nes al-tus mer-ces cer-no mul-tum tem-pus uin-dex sun-to tex-tor scrip-si ip-se cas-tus quaes-tor sanc-tus emp-tor tem-plum nos-tri cas-tra.*

6. In compound words and when an enclitic is present it is customary to divide the elements even if the previous rules are violated, unless the elements are merged in a unique joining letter; examples: *ex-i-re red-i-re ex-ta-re ob-ru-e-re abs-ti-ne-re ads-cri-be-re mul-tas-que uis-ne suap-te pae-nin-su-la quin-quen-ni-um.*

7. Words derived from Greek are divided according to the Greek rules; for example *sce-ptrum dra-chma cy-cnus i-sthmus rhy-thmus A-ri-sto-te-les.*

Such rules appear not particularly difficult to implement with patterns, but at a second look the real difficulties show up. The rules contain too many words or phrases such as "exceptionally", "except when", "it is customary", "derived from".

In facts patterns are not so intelligent as to know when there are exceptions to the rules or when some words are not of Latin origin; something can be done with prefixes or suffixes, but not all of them; patterns for words of Greek origin may be implemented if they imply the digraphs *ch ph th rh* or the letters *y z*, but a word such as *sceptrum* cannot be detected as being of Greek origin.

9 The actual patterns

In this situation it would be more convenient to create the necessary patterns by means of a huge hyphenated word list and to apply *patgen*, (Liang, 1983); even with 100 000 word lists it is impossible to be sure that wrong break points never show up.

I decided to do my best with the above rules in creating the classic Latin patterns, and leave the user the responsibility of using an exception list by means of the command `\hyphenation` that is available for this purpose.

I temporarily left the ligatures æ and œ among the possible Latin letters, but the user should refrain from using them, because they do not belong to classic Latin; in future releases of this pattern file such letters probably will be eliminated.

I followed the suggestion of Farina and Marinone (1979) to avoid diphthongs; therefore I could avoid the difficulty of deciding if *i u* play the rôle of vowels or consonants. In this way some possible break points are missed, but many errors are avoided. I followed this same philosophy also when preparing the hyphenation patterns for several other languages I created patterns for; it proved to be a wise decision and I continue to follow this philosophy: "a missed break point is better than a wrong one".

I think I did a good job with prefixes *ab abs ob circum ex para sub* and suffixes *que ne*. I could not do the same (supposedly) good job with other prefixes: for example *re* appears in a lot of words where it is not a prefix; the same with *red trans con* and so on. This is a typical application of the `\hyphenation` exception lists.

In any case I created a *shorthand* by means of character `"` that is never used in Latin; both babel and polyglossia allow the definition of shorthands, so that I defined the above character as an *active* one; an active character behaves as a TEX macro; I defined it as to insert a *compound word mark*; such mark inserts a *word boundary* and a discretionary break that, contrary to the default definition of `\-`, it allows hyphenation on both word elements. For example, without this mark the word *transierat* would get divided as *tran-si-e-rat*; but if the mark is inserted at the end of prefix "trans", as in `trans"ierat`, then the word is correctly divided as *trans-ie-rat*. Actually the correct division should be *trans-i-e-rat* but TEX refrains from breaking words leaving strings that are too short before or after the break points; in classic Latin the initial and terminal "shortness" limits are specified in two letters, so that initial or terminal syllables of one letter are forbidden.

The above shorthand should not to be used in every compound word; this means that the user should initially enter the Latin text in the source TEX file without any concern to compound words; when revising the drafts s/he might introduce the shorthand in those rare instances where compound words are not hyphenated correctly on the word boundaries. Alternatively s/he can enter the correctly hyphenated words in the argument of `\hyphenation`. Beware, though; in order to have `\hyphenation` work correctly in every situation, it is not sufficient to enter one hyphenated word; but, since Latin has both declination of nouns and adjectives and conjugation of verbs„ it would be necessary to enter all cases: 12 entries for nouns, 36 entries for adjectives, about 100 entries for verbs for all persons, modes, for the three masculine, feminine and neutral present, past and future participles, gerund, infinitive and for the active and passive diathesis; well, probably in the same document it is very unusual that all forms of a verb are present, but just in case...

With babel (version 3.9 or higher) it is possible to use the command `\babelhyphenation` that extends the funcionality of the standard LATEX `\hyphenation` command; it has the syntax:

`\babelhyphenation[`⟨*language*⟩`]{`⟨*exceptions*⟩`}`

where ⟨*language*⟩ is a language name (in our case `latin` in lower case letters, the name that babel understands) or a list of comma separated language names; if no ⟨*language*⟩ is specified, the ⟨*exceptions*⟩ are valid for all languages used in the document, otherwise they are valid only for the specified language(s).[7]

10 Testing

I test my patterns by means of a small package I wrote myself, testhyphens (Beccari, 2014); this package implements for LATEX the plain TEX code that was contributed to the TEX community by Jonathan Kew (Eijkhout, 1993). This package defines the checkhyphens environment; it contains a space delimited word list, or, even better, some real text in the language that is being tested; by running *pdflatex* on the source file, the words of this text are printed on consecutive lines with their break points.

I find it very convenient and, when a real text is entered in the checkhyphens environment I can see the hyphenation algorithm at work even when the compound word mark shorthand is used. I cannot get the complete test of all my patterns, but I can always add to that argument as many unusual words as I want and check their break points.

For example I tested with `\checkhyphens` the following text from Julius Caesar's *De bello gallico*:

> Flumen est Arar, quo per fines Haeduorum et Sequanorum in Rhodanum influit, incredibili lenitate, ita ut oculis in utram partem fluat iudicari non possit. Id Heluetii ratibus ac lintribus iunctis transibant. Ubi per exploratores Caesar certior factus est tres iam partes Heluetios id flumen traduxisse, quartam fere partem citra flumen Ararim reliquam esse,

7. This functionality is not yet available with polyglossia since it has not been upgraded by the end of August 2014.

Flu-men	u-tram	fac-tus	ter-tia	im-pe-di-tos
est	par-tem	est	ui-gi-lia	et
A-rar,	fluat	tres	cum	i-no-pi-nan-tes
quo	iu-di-ca-ri	iam	le-gio-ni-bus	ad-gres-sus,
per	non	par-tes	tri-bus	ma-gnam
fi-nes	pos-sit.	Hel-ue-tios	e	par-tem
Hae-duo-rum	Id	id	ca-stris	eo-rum
et	Hel-ue-tii	flu-men	pro-fec-tus	con-ci-dit:
Se-qua-no-rum	ra-ti-bus	tra-du-xis-se,	est	re-li-qui
in	ac	quar-tam	ad	se-se
Rho-da-num	lin-tri-bus	fe-re	eam	fu-gae
in-fluit,	iunc-tis	par-tem	par-tem	man-da-runt
in-cre-di-bi-li	trans-i-bant.	ci-tra	per-ue-nit	at-que
le-ni-ta-te,	U-bi	flu-men	quae	in
i-ta	per	A-ra-rim	non-dum	pro-xi-mas
ut	ex-plo-ra-to-res	re-li-quam	flu-men	sil-uas
o-cu-lis	Cae-sar	es-se,	trans-ie-rat.	ab-di-de-runt.
in	cer-tior	de	Eos	

FIGURE 5: The output of the checkhyphens environment on the test file taken from *De bello gallico*. Parameters `\lefthyphenmin` e `\righthyphenmin` have been set to 1; but the default language settings assume they equal 2.

> de tertia uigilia cum legionibus tribus e castris profectus est ad eam partem peruenit quae nondum flumen transierat. Eos impeditos et inopinantes adgressus, magnam partem eorum concidit: reliqui sese fugae mandarunt atque in proximas siluas abdiderunt.

The compound word mark shorthand was used in `trans"ibant` and `trans"ierat`; it actually enters the scene in the word *trans-ibant* at the end of its line. Of course with checkhypens all break points are shown; besides the break points created with the compound word mark, it is interesting to see *Hel-ue-tii, per-ue-nit* where the *u* plays the rôle of a consonant as the the letter *v* in modern spelling.

The output of the test is shown in figure 5.

11 The Latin language description file

Package babel accepts the definition of attributes: with version 3.9 they are preferably used as modifiers. Package polyglossia has always accepted option lists in the specification of some of its working languages.

I had to write again both description files `babel-latin.ldf` and `gloss-latin.ldf` in order to add the attribute/modifier `classic`. The new version of `gloss-latin.ldf` now behaves as such: if no options are specified with polyglossia to the specific command `\setotherlanguage{latin}`, the modern spelling is assumed and the shorthand is not defined. The option `babelshorthands` activates the compound word mark shorthand; the keyword `variant` set to one of `modern, medieval, classic` specifies a Latin spelling style (`modern` being the default); these variant values are mutually exclusive and it is not possible to switch style within a given document. The real reason of this constraint is that different hyphenation pattern files are used: one is for classic Latin, while the other works with both modern and medieval Latin.[8]

For babel I had to discover the undocumented trick for specifying a different pattern set within the same language description file; actually the trick is very simple and similar to the one used for the Greek description file: it amounts to associate the control sequence that addresses the language counter to a different value. After getting by this small difficulty, it was simple to integrate classic Latin in the previous `babel-latin.ldf`.

The set of files to be used with *pdflatex* are available with the 2014 distribution of the TEX system; of course I can already use these updates on my computer even when using *xelatex*, because I created them and I know how to integrate hyphen pattern files in the formats of my installation. It is a pretty delicate procedure that I am not going to explain here; in facts it is not necessary to know these inner workings of the TEX system, because such operations are automatically performed each time a user updates its installation with the TEX Live program *tlmgr* or with the MiKTEX wizard.

12 Conclusion

For what concerns Greek the actual hyphenation pattern files for 8-bit aware typesetting engines are not suited for direct literal input of the Greek text in the source files. I discovered the problem and found a solution. The three maintainers of

8. I already uploaded the new version of the `gloss-latin.ldf` but by the end of August 2014 it is not yet available with polyglossia.

the various aspects of the Greek language handling, Mojca Miklavec for the overall file handling in CTAN, Günter Milde for the Greek support to the babel package, Dimitrios Filippou for the hyphenation pattern files, are doing the rest of the work in order to have an efficient and complete set of instruments to enhance the Greek language support. I thank them all very much for what they are doing.

For what concerns classic Latin producing hyphenation pattern files by implementing the grammar rules is not particularly difficult; it requires just some attention and a lot of patience; the TEXbook (Knuth, 1996) explains everything about hyphenation patterns so that creating valid pattern files is not a terrible task.

It goes without saying that the examples set forth by Farina and Marinone (1979) work correctly. The new patterns show their effectiveness in figure 5, but there remain still some problems with certain prefixes. In facts the prefix *trans* had to be "terminated" with a compound word marker in order to have the correct break points. The weak points of my patterns with prefixes and suffixes have been discussed in section 9.

13 Acknowledgements

I acknowledge the help received from Günter Milde, Dimitrios Filippou and Mojca Miklavec who are now the three pillars of the whole Greek support to the babel system; I just happened to spot the problem that is described in this article; I gave suggestions to Mojca Miklavec; I sent some patches to Günter Milde; I reworked the pattern files for Dimitrios Filippou. But the real work, the part of the work that is still under way, is in their hands. When everything will be finished it will be their merit to have put together some typesetting instruments for the Greek language, that will be useful not only to the Greek TEX users, but to the whole TEX community.

I hope that by the time this paper is published, their work is finished and integrated into CTAN and TEX Live.

I want to express again my thanks also to Raffaella Tabacco; without her support I would not have been able to do anything in this field.

References

Beccari, C. (2013a). «`teubner.sty` – An extension to the greek option of the *babel* package». PDF document. The PDF document is in CTAN but it can be easily read by means of the terminal command `texdoc teubner-doc`.

— (2013b). «The `teubner` package –Extensions for Greek philology». PDF document. The PDF file is archived on CTAN, but it can be readily read with the terminal command `texdoc teubner`.

— (2014). «The testhyphens package». PDF document. The document is part of the TEX Live 2014 distribution. It can be read by means of the termial command `texdoc testhyphens`.

Braams, J. and Bezos, J. (2014). «Babel». PDF document. Easily readable with the terminal command `texdoc babel`.

Charette, F. and Reutenauer, F. (2011). *Polyglossia: a Babel replacement for XeLaTeX*. TUG. Easily readable with the terminal command `texdoc polyglossia`.

Eijkhout, V. (1993). «The bag of tricks». *TUGboat*, **14** (4), p. 424.

Farina, R. and Marinone, N. (1979). *Metodologia – Guida pratica alle esercitazioni di seminario e alle tesi di laurea per le discipline umanistiche*. Società editirice Internazionale, Torino.

Filippou, D. (2004). «Hyphenation patterns for ancient and modern Greek». In *TEX, XML, and digital typography*, Springer-Verlag, Berlin-Heidelberg.

Knuth, D. E. (1996). *The TEXbook*. Addison Wesley, Reading, Mass., 16th edition.

Levy, S. and Murphy, T. (2010). «Using Greek fonts with LaTeX». PDF document. This PDF document is available on CTAN and simply read by means of the terminal command `texdoc lgreekuse`.

Liang, F. (1983). *Word Hy-phen-a-tion by Comput-er*. PhD thesis, Stanford University.

Merk, A. (ed. by) (1984). *Novum testamentum graece et latine*. Sumptibus Pontificii Instituti Biblici, Romae, 10th edition.

Mylonas, C. and Witney, R. (1992). «Complete Greek with adjunct fonts». *TUGbooat*, **13** (1), p. 39–50.

Syropoulos, A. and Milde, G. (2014). «Babel support for the greek language». PDF document. The PDF file `babel-greek.pdf` is available on CTAN and easily readable by means of the terminal command `texdoc babel-greek`.

▷ Claudio Beccari
Professore emerito
Politecnico di Torino
`claudio dot beccari at gmail.com`

LilyPond: un *music engraver* integrabile con LaTeX

Tommaso Gordini, Davide Liessi

Sommario

L'incisione della musica su lastre di metallo ha rappresentato per almeno un secolo il vertice qualitativo della scrittura musicale. Oggi quest'arte è praticamente scomparsa, soppiantata dai moderni programmi di notazione musicale che, tuttavia, non riescono nemmeno ad avvicinarsi alla bellezza di una calcografia.

Perché l'incisione a mano è di tanto superiore? Perché i software attuali faticano a raggiungerne la qualità? L'incisione tradizionale è davvero scomparsa o è possibile recuperarne i princìpi?

Nelle prossime pagine cercheremo di rispondere a queste domande presentando LilyPond, un *music engraver* open source capace di realizzare prodotti particolarmente raffinati, e alcune sue possibili integrazioni con LaTeX.

Abstract

Music engraving on metal plates has been the highest point in music writing for at least a century. Nowadays this art has almost disappeared in favour of modern music notation programs. However computers are far from achieving the beauty of calcography-engraved music.

Why is hand engraving so much superior? Why are computer programs so inferior in quality? Has traditional engraving really disappeared or is it possible to recover its principles?

In this paper we try to answer these questions. We will introduce LilyPond, an open source music engraver which can achieve very refined results. We will also present some possible ways of integrating LilyPond and LaTeX.

Introduzione

Questo articolo *non* spiega come scrivere la musica con LilyPond. Gli utenti interessati in questo senso possono consultare la ponderosa documentazione del programma e le risorse elencate nella sezione A. Il codice strettamente musicale presente nei prossimi paragrafi, perciò, non verrà spiegato e presuppone una conoscenza minima della sintassi del programma.

Si può suddividere il lavoro in tre parti principali.

Nella prima si spiegano le differenze più macroscopiche tra la tipografia testuale e quella musicale, l'evoluzione della stampa musicale dalle prime testimonianze alle macchine offset, i problemi più consistenti che deve affrontare chi stampa la musica e il motivo per cui 'vecchio è bello' (sezione 1).

La seconda parte presenta LilyPond, un *music engraver* freeware capace di produrre spartiti di qualità eccellente, i suoi punti di forza, le differenze principali con altri *score writer* e alcuni editor particolarmente adatti agli scopi di questo lavoro (sezioni 2, 3 e 4).

Nella terza parte si illustrano alcune possibilità per integrare la musica prodotta da LilyPond nei documenti scritti con LaTeX (sezione 5).

Chiude questo contributo una breve descrizione dei limiti del programma, di alcuni progetti realizzati con esso e una rassegna di risorse utili (sezioni 6, 7 e A).

1 Peculiarità e problemi della tipografia musicale

Questa sezione contiene qualche cenno indispensabile per comprendere le peculiarità della stampa musicale, i suoi problemi specifici e le differenze di principio rispetto a quella del testo. Le notizie storiche sono tratte in prevalenza da (Aa. Vv., 1996) e (Mancini, 2013).

1.1 Breve storia della nota stampata

Non è questa la sede per esporre nel dettaglio lo sviluppo della musica stampata dalle origini all'avvento del computer, neppure a grandissime linee. Perciò le note seguenti si limitano a descrivere gli aspetti più tecnici dell'argomento.

Fino al terzo quarto del secolo XV la musica viene riprodotta a mano. Neppure la rivoluzione di Gutenberg riesce a intaccare immediatamente il monopolio degli amanuensi: i primi esempi di notazione musicale a stampa, inseriti in libri liturgici e trattati teorici, risalgono solo a due decenni più tardi, circa al 1475 (figura 1). Fino a questa data, l'unica possibilità per inserire frammenti musicali in un libro stampato è farlo a mano e in un momento successivo alla stampa del testo: righi e note vengono disegnati da un copista, spesso con inchiostri diversi da quelli utilizzati nel resto del libro, nello spazio, spesso insufficiente, lasciato libero dallo stampatore. Immagini il lettore i costi dell'operazione e la qualità del risultato.

Il ritardo è presto spiegato: è difficile produrre da subito con sufficiente precisione caratteri che riproducano la 'strana' forma dei simboli musicali, molto più numerosi delle lettere; è difficile districar-

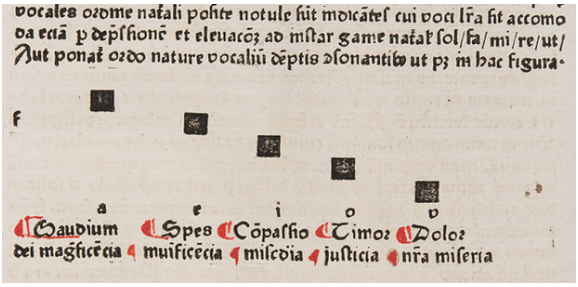


FIGURA 1: Jean Charlier de Gerson (1363–1429), *Collectorium super Magnificat*, stampato a Strasburgo nel 1473. Queste cinque note discendenti in campo bianco sono considerate tradizionalmente il primo esempio di musica stampata.

si in un sistema notazionale ancora in evoluzione e non uniforme da paese a paese. C'è poi un ostacolo inedito e temibilissimo: le note sono *sovrapposte* al rigo. Attenzione, poi, a spaziare correttamente i simboli per agevolarne la lettura. E poi c'è la questione dei due colori: rosso per il rigo e nero per le note, come ormai è tradizione, che non devono mescolarsi tra loro (figura 2).[1]

La primissima soluzione adottata è molto semplice: rigo e note sono preparati contemporaneamente, inchiostrati ciascuno con il proprio colore e impressi *in una sola volta* sul foglio. Presto, però, ci si accorge che due elementi sovrapposti richiedono altrettante fasi nel processo di stampa.

Sul versante dei caratteri mobili, l'ultimo quarto del secolo XV conosce una discreta produzione di musica liturgica stampata prevalentemente in *doppia impressione* (una per il rigo e una per le note, che talvolta continuano a essere copiate da un amanuense). L'alternativa è nota con il nome di *metodo blocco*: i caratteri, che ora riproducono sia la nota sia il relativo frammento di rigo, vengono giustapposti fino a ricomporre l'intero rigo originale, con un risultato sempre molto approssimativo, dato che praticamente *mai* i frammenti combaciano perfettamente. Sono queste imperfezioni a scatenare in tutta Europa la voglia di produrre stampe musicali sempre migliori.

Lo spartiacque è convenzionalmente individuato nell'opera del marchigiano Ottaviano Petrucci, attivo dal 1501 al 1520, che inventa un sistema completo di notazione mensurale a caratteri mobili e un metodo di impressione capaci di rivaleggiare con la perfezione e l'eleganza dei libri musicali manoscritti di fine Quattrocento. Petrucci adotta il metodo della stampa a *tripla impressione*: una per i righi, una per le note e gli altri simboli musicali, una per il testo da cantare. La difficoltà, come sempre, sta nel far combaciare esattamente i tre elementi. Ora le note vengono considerate alla stregua di normali caratteri tipografici testuali, semplici elementi con cui riempire uno spazio,

1. La pratica di stampare il rigo in rosso e le note in nero sopravvive in alcune moderne edizioni di musica gregoriana.

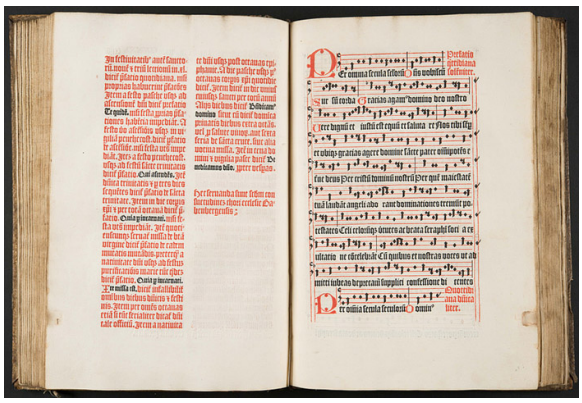


FIGURA 2: *Missale* stampato in due colori e in doppia impressione da Johann Pfeyl a Bamberga nel 1499.

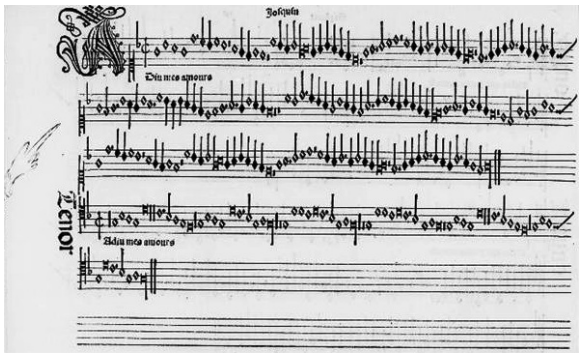


FIGURA 3: Parte del *tenor* di *Adieu mes amours*, popolare canzone profana della fine del secolo XV. La composizione è contenuta in *Harmonice Musices Odhecaton*, la prima antologia di musiche polifoniche interamente stampata con caratteri mobili da Ottaviano Petrucci a Venezia nel 1501.

perdendo il calore e la personalità emanati dalle migliori xilografie (figura 3).

In quest'epoca, infatti, c'è un altra soluzione contemporanea alla stampa a caratteri mobili: la *xilografia*, tecnica antica e già molto diffusa in Europa nel secolo XV, che dalla manualistica teorica medievale e primo-rinascimentale migra facilmente alla musica, soprattutto quella per tastiera. La 'scrittura con il legno', eccellente per riprodurre illustrazioni di soggetto vario, settore nel quale è per lo più utilizzata, presenta difficoltà non trascurabili dovute alla natura essenzialmente grafica della musica, costituita di linee, curve, simboli, spazi e angoli molto precisi (figura 4). Ma se è abile come l'istriano Andrea Antico (attivo dal 1510 e temibile concorrente del Petrucci) lo xilografo produce autentici capolavori di design editoriale. La xilografia consiste nel ricavare dalla faccia piatta di un blocco di legno (poi anche di metallo) il contenuto da riprodurre asportando con sgorbie e scalpelli le parti che non costituiscono il disegno. Questa tecnica cavalca l'onda del successo almeno fino alla metà del secolo XVI, riuscendo addirittura a competere per costi ed estetica con la stampa a caratteri mobili, almeno per un po', anche perché è economica. Non si possono correggere gli errori né

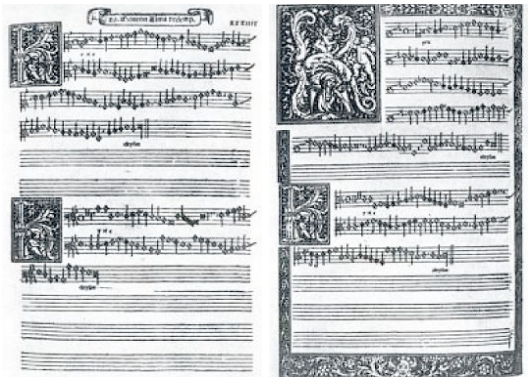


Figura 4: *Liber quindecim Missarum*, edito da Andrea Antico a Roma nel 1516. Questo *Messale*, stampato con la tecnica xilografica, rappresenta il primo esempio di libro *in-folio*.

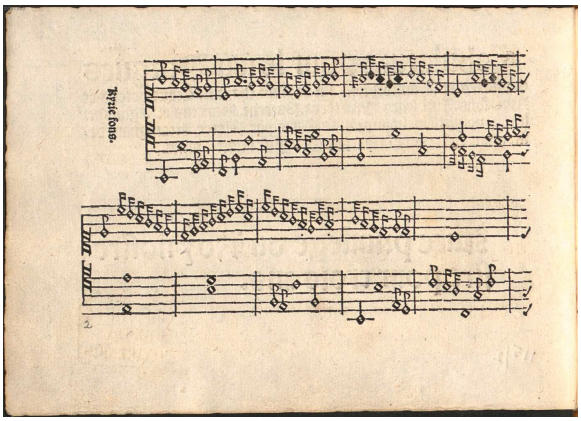


Figura 5: *Magnificat* per organo pubblicato da Pierre Attaignant con la stampa a singola impressione.

migliorare l'impaginazione strada facendo, è vero, ma se correttamente utilizzate, le matrici rimangono buone anche per edizioni successive, limitando il numero di copie da stampare a quelle effettivamente necessarie. Il che non è poco, con quel che costa la carta al tempo.

In Francia, intanto, gli stampatori riducono le tre impressioni del Petrucci a *una sola*: il nuovo metodo, adottato su larga scala dal parigino Pierre Attaingnant a partire dal 1528, supera di molte misure gli standard editoriali del Nostro e contemporaneamente avvia la xilografia musicale verso un lento declino (figura 5). Le pagine di musica si assemblano ora come normali pagine di testo, specularmente, in una volta sola. I caratteri musicali vengono realizzati come quelli testuali: dal punzone alla matrice al tipo finale ottenuto per fusione. Nascono i primi font musicali completi, con tipi per tutte le esigenze. Ciascuno rappresenta o una nota distinta per altezza e valore, unita al proprio segmento di rigo completo, oppure altri simboli musicali. Ma è difficile realizzare buoni caratteri per questo tipo di stampa: il pericolo che il rigo stampato non sia perfetto è dietro l'angolo.

Perciò, i problemi mai risolti dell'impressione singola (allineamenti imprecisi, impossibilità o quasi

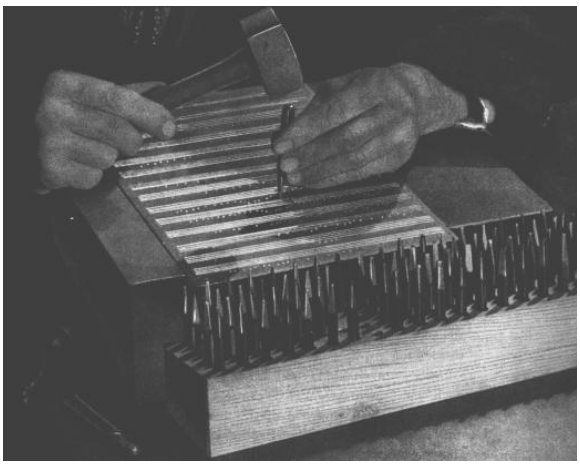


Figura 6: Maestro incisore all'opera su una lastra metallica.

di realizzare partiture ed edizioni in formato ridotto, specie se per musica per tastiera) e la tendenza all'ornamentazione virtuosistica nel frattempo affermatasi (con la selva di nuovi ed elaborati caratteri che richiede) impongono di individuare un nuovo strumento di stampa.

Vecchia di più di un secolo, nel 1575 viene 'reinventata' per la stampa musicale la *calcografia* ('scrittura con il rame'), che sembra poter risollevare la situazione (figura 6). Tecnica laboriosa, particolarmente adatta alla riproduzione in serie della musica, presenta così tanti vantaggi che sembra naturale doverla preferire alla tipografia tradizionale. Tuttavia, mentre quest'ultima continua a dominare il mercato delle edizioni economiche, per circa un secolo la prima non riesce che a recitare da comprimaria nella nicchia delle tirature di pregio. Solo a cavallo tra Sei e Settecento, con l'affermarsi della musica strumentale e delle edizioni in partitura, si impone quasi ovunque anche per la produzione su larga scala. Ma né il segno netto e precisissimo del bulino, né la flessibilità permessa in scrittura e impaginazione (ora si può riprodurre virtualmente qualunque figurazione) né la vocazione al formato ridotto e alla riproduzione della musica per tastiera straordinariamente fedele al manoscritto, né la possibilità di limitare le stampe allo stretto necessario riescono a fronteggiare un mondo musicale sempre più affamato di nuova musica e ad alte tirature. Troppo costosa la carta adatta, troppo lungo il procedimento, troppo precoce il deterioramento delle lastre.[2]

Nel frattempo, la notazione a caratteri mobili subisce qualche ritocco qua e là: a fine Seicento, a Londra si decide di arrotondare la testa delle note, fino ad allora romboidale o quadrata, e di dare anche agli altri simboli una forma più vicina a quella moderna. È il 1755 quando l'editore Breitkopf di Lipsia inventa il *tipo a mosaico*: i simboli

2. Si può vedere all'opera un maestro incisore della G. Henle Verlag su http://www.youtube.com/watch?v=345o3Wu95Qo.

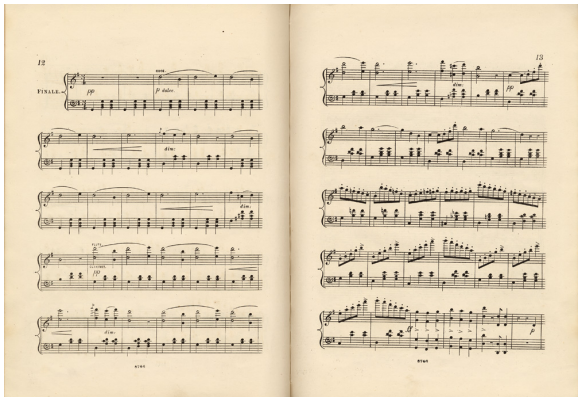


Figura 7: Inizio del Finale di *Mazeppa*, cantata di Michael William Balfe (1808–70). Stampa litografica pubblicata a Londra nel 1865.

musicali sono composti a propria volta da 'pezzi' più piccoli tutti della stessa dimensione. Qualcun altro, in Francia, si inventa di fondere caratteri in diversi corpi.

Qualche decennio più tardi, l'anno è il 1790, si tenta la strada della *stereotipia*, capace di riportare su una lastra metallica un'intera pagina di musica precedentemente composta in caratteri mobili, i quali, utilizzati in pratica una sola volta, praticamente non si logorano.

Se il rame non funziona, funzionerà la pietra, si pensa. La *litografia* ('scrittura con la pietra'), inventata nel 1796, predomina a partire dagli anni Sessanta del secolo XIX (figura 7). Le pietre calcaree trattate chimicamente e disegnate a gran velocità con speciali matite permettono tirature virtualmente illimitate: proprio quello che ci vuole. Meglio o peggio della lastra di rame? Tutto dipende dalla mano del copista: la matita litografica, infatti, non richiede la perizia dell'incisore professionista, tanto che spesso sono gli stessi compositori a riprodurre le proprie opere sulle petrose matrici. Con risultati a volte discutibili. E i calcografi? Si capisce di poter sfruttare le loro insuperabili qualità grafiche 'riportando' sulla pietra il disegno inciso sulla lastra di rame come un enorme timbro al contrario, abbreviando ulteriormente i tempi. La lastra è salva, e anche le alte tirature di qualità.

L'invenzione della macchina fotografica apre nuove strade. Poter trasferire direttamente una stampa fotografica dell'originale su pietra litografica (*fotolitografia*) o su una lastra di zinco o di altro metallo elimina con un colpo di spugna la lunga fase della copiatura. L'originale, però deve essere 'ben scritto': per farlo, si inventa un sistema a trasferibili.

Oggi la musica, come gran parte delle pubblicazioni, si stampa con il metodo *offset*, una variante della litografia messa a punto agli inizi del secolo XX. Si tratta di una stampa indiretta in cui la matrice tipografica, una lastra di vario materiale ottenuta tramite impressione fotografica, è avvolta

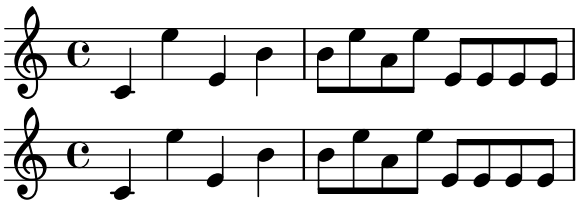


Figura 8: Differenza tra la spaziatura matematica delle note (sopra) e quella corretta da LilyPond (sotto).

intorno a un rullo e, inumidita prima e inchiostrata poi, trasferisce la stampa a un secondo rullo rivestito di caucciù il quale solo a questo punto la imprime sulla carta. Con vantaggi enormi.

Negli ultimi decenni (il primo font musicale digitale immesso nel mercato, il Sonata™, risale al 1985) l'avvento dei sistemi di videoscrittura musicale ha mutato nuovamente lo scenario. Oggi chiunque dispone di mezzi informatici con cui realizzare spartiti e partiture molto facilmente, dai software a interfaccia grafica estremamente intuitivi a quelli a input testuale più macchinosi. Estrarre parti singole è una banalità, produrre file MIDI per ascoltare quanto si è appena scritto è quasi scontato, ritoccare impaginazione e spaziatura, imitare stili e caratteri dei grandi editori è routine. Ciascuno può diventare l'editore di sé stesso e produrre musica stampata di eccellente qualità.

Il panorama è vastissimo e c'è davvero di che perderci la testa. Perché, dunque, tra i tanti programmi disponibili la nostra scelta è caduta su LilyPond?

1.2 Problemi della stampa musicale

Le osservazioni e gli esempi contenuti in questa sezione sono tratti da (The LilyPond Development Team, 2014a).

1.2.1 Note e spazi

Nella scrittura della musica, la distribuzione degli spazi deve riflettere i valori di durata delle note. Tuttavia, come si vedrà nel paragrafo 1.2.3, numerose edizioni moderne che rispettano questi valori con precisione matematica si rivelano di scarso valore tipografico. La figura 8 mostra lo stesso motivo stampato due volte ma con due spaziature diverse. La differenza è minima e un occhio inesperto non se ne accorge. Quale dei due si preferisce?

Entrambe le misure del motivetto contengono solo note da suonare con ritmo costante: la distanza fra di esse, dunque dovrebbe riflettere questa uniformità. Purtroppo, l'occhio ci inganna un po'. Infatti, non solo si accorge della distanza fra le *teste* delle note, ma prende in considerazione anche la distanza fra *gambi* consecutivi. Come risultato, le note di una sequenza 'gambo in su-gambo in giù' dovrebbero essere allontanate e quelle della sequenza opposta avvicinate; il tutto, naturalmente, a seconda di come le note sono combinate in ver-

ticale. Le due misure inferiori contengono questa correzione; in quelle superiori, invece, le note sono spaziate con le impostazioni matematiche predefinite. Un maestro incisore saprebbe certamente aggiustare lo spazio per soddisfare l'occhio.

C'è di più. Gli algoritmi di spaziatura di LilyPond considerano anche le stanghette di battuta, motivo per cui dopo l'ultima nota con il gambo in su dell'esempio inferiore c'è un piccolo spazio aggiuntivo. Un gambo in giù non avrebbe certo bisogno di questa correzione.

Dettagli cavillosi? Forse, agli occhi dei più, che non se ne intendono. Ma il musicista esperto, anche senza accorgersene, leggerà le note più facilmente e sarà più contento.

1.2.2 Interruzioni di pagina

L'interruzione di pagina in uno spartito musicale è profondamente diversa da quella di qualunque altro documento a stampa, tanto da essere uno degli elementi determinanti per scegliere un'edizione o un'altra.

L'elemento in comune ai due tipi di pubblicazione è che in entrambi i casi a un certo punto bisogna voltare pagina e questo si fa con le mani. Ciò che non sempre viene compreso (nemmeno da parte di chi stampa la musica per mestiere) è che prima di voltare pagina l'esecutore ha le mani sullo strumento e non sulla pagina da voltare, come accade leggendo un libro. Questa situazione si verifica praticamente *sempre*.

La sequenza che il musicista deve seguire in prossimità di una *voltata* (questo è il termine esatto in gergo musicale) è la seguente.

1. Staccare una mano dallo strumento.
2. Voltare la pagina.
3. Rimettere la mano sullo strumento.

Chi suona uno strumento a fiato, deve aggiungere a queste altre operazioni. Nulla di straordinario: sono gesti naturali che diventano automatici dopo un po' di esperienza, come sanno bene i musicisti.

Il problema nasce quando chi suona è costretto a compierli frettolosamente. Di solito le cause sono due, elencate in ordine crescente di gravità.

1. La pagina termina con un numero di misure di pausa insufficiente a voltare comodamente, perché il tempo della composizione è veloce.
2. Tra due pagine consecutive la musica è continua e non ci sono misure di pausa.

Naturalmente, il problema non si pone se la fine della pagina coincide con la fine del brano. Esiste anche la situazione in cui l'ultima misura della pagina va suonata e la pagina successiva comincia con un certo numero di misure di pausa, per cui o la voltata comoda sarebbe possibile o si ricadrebbe nel punto 1. In entrambi i casi, il musicista è costretto ad aggiungere una quarta fase prima o dopo delle tre appena viste: non suonare alcune delle note scritte o addirittura intere misure. Sceglierà lui se saltare qualcosa prima o dopo la voltata, a seconda di cosa è meglio far sentire, ma di certo non ha materialmente il tempo di suonare tutto. È proprio *questa* la fase da evitare.

Naturalmente, prima dello studio o della prova il musicista coscienzioso guarderà la propria parte *anche* alla ricerca delle voltate problematiche. Se ce ne sono, vorrà trovare una soluzione per non perdere nemmeno una nota.

Specie per chi in leggio è da solo e magari suona uno strumento ingombrante, però, la faccenda è seria: la corsa contro il tempo per la 'voltata perfetta' potrebbe trasformarsi in una catastrofe. Allora, ecco spuntare fotocopie doppie, leggii strabordanti di musica, mollette da bucato, collage improbabili, miniature a matita dei punti incriminati, acrobazie per continuare a suonare e voltare contemporaneamente.

Tutto questo si può evitare molto facilmente e a monte se chi stampa la musica si pone l'obiettivo di facilitare la vita del musicista anche in questi aspetti non strettamente tipografici. Cominciare una nota non è affatto banale e richiede tempo e concentrazione. Una voltata problematica può causare ansia e un attacco sbagliato per la fretta di riportarsi nelle condizioni per suonare. Se *clamorosamente* sbagliato, poi, un'intera esecuzione può uscirne compromessa.

Certo è che i grandi incisori attivi tra Otto e Novecento avevano a cuore questi problemi e la loro musica era anche funzionale all'esecuzione, oltre che bella da vedere. Se la musica da suonare richiede di voltare pagina, *bisogna* fare di tutto per mettere il musicista in grado di farlo con comodo.

Naturalmente, LilyPond è attrezzato per calcolare e ottimizzare anche questo aspetto, dove possibile.

1.2.3 Lastre–computer: 1–0

Si osservino con attenzione le figure 9 e 10. La prima riproduce un bello spartito inciso a mano su lastra nel 1950 e la seconda una moderna edizione realizzata al computer.

Le note mostrate sono identiche (si tratta dell'inizio del Preludio della prima Suite per violoncello solo di Bach) ma è il loro aspetto a essere diverso, specie se le si stampa e le si osserva da lontano. Si provi a leggerle e a suonarle entrambe e si troverà che quella incisa a mano è più piacevole da utilizzare. Possiede linee fluide e un movimento, quasi fosse un brano musicale 'vivo' e dotato di respiro proprio, mentre l'edizione più nuova appare fredda e meccanica.

È difficile individuare a colpo d'occhio che cosa renda quest'ultima diversa dalla prima. Tutto vi

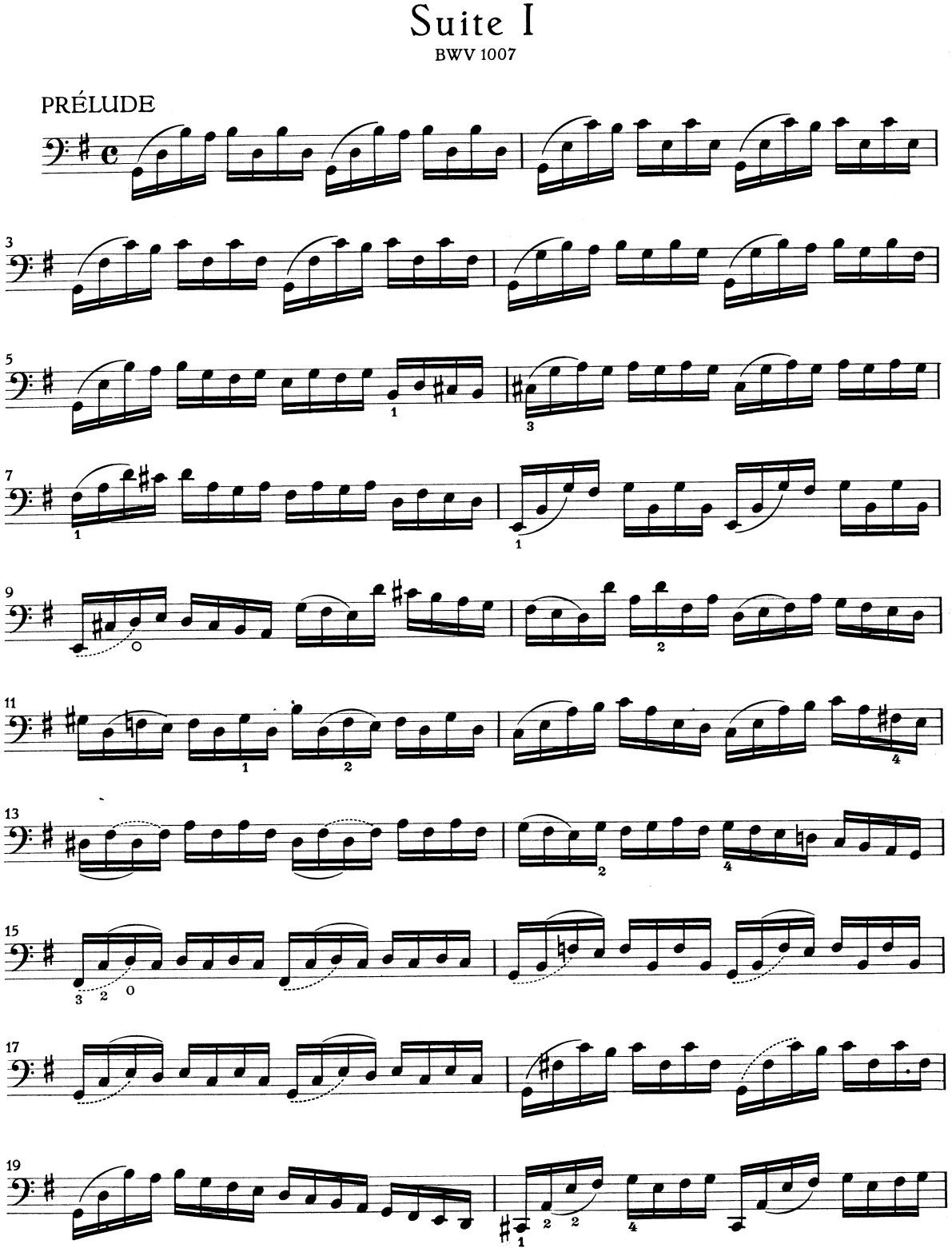


Figura 9: Preludio dalla Suite per violoncello n. 1 in sol maggiore BWV 1007 di Johann Sebastian Bach (1685–1750). © 1950 Bärenreiter-Verlag, BA 320.

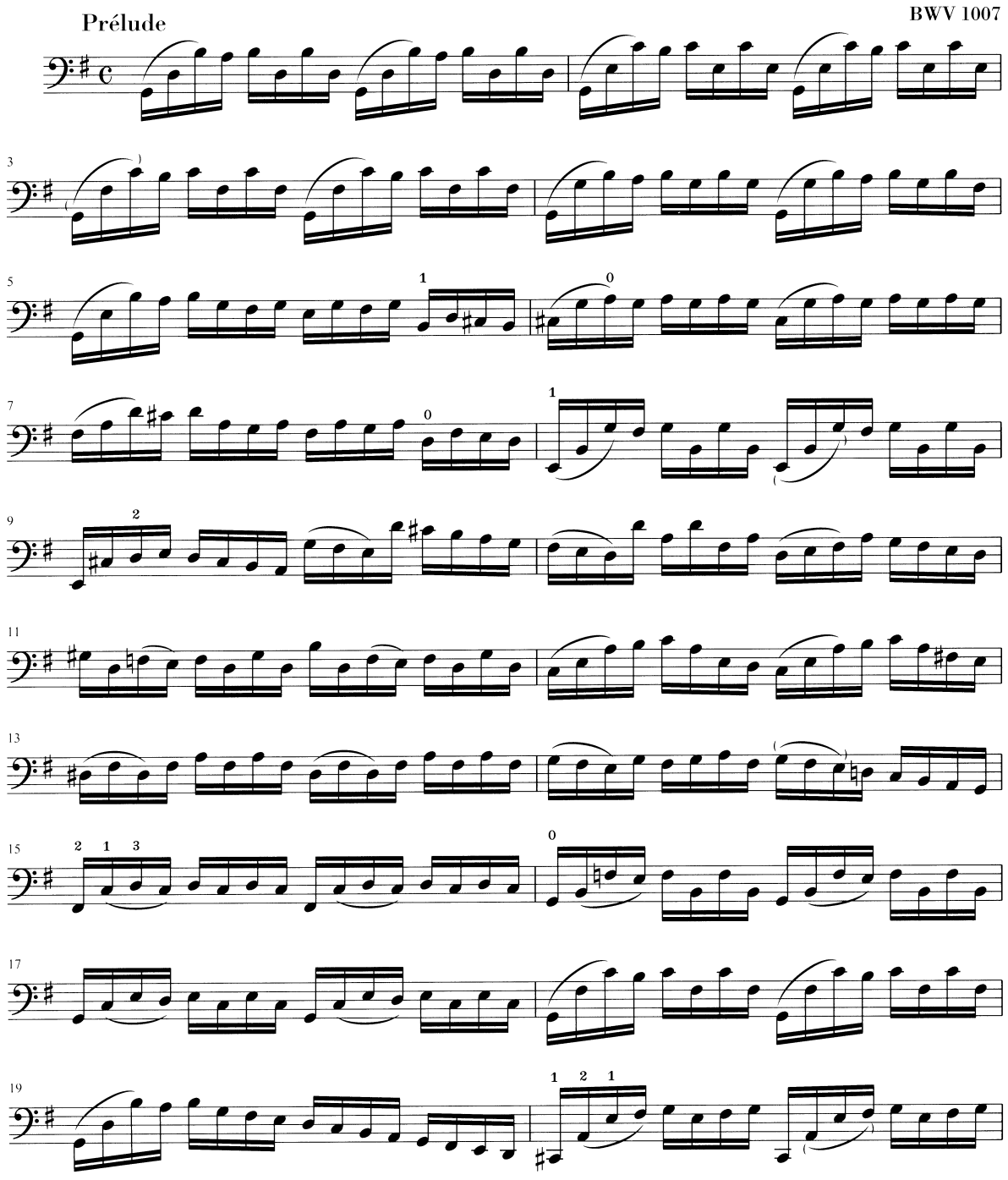


Figura 10: Preludio dalla Suite per violoncello n. 1 in sol maggiore BWV 1007 di Johann Sebastian Bach (1685–1750). © 2000 G. Henle Verlag, HN 666.

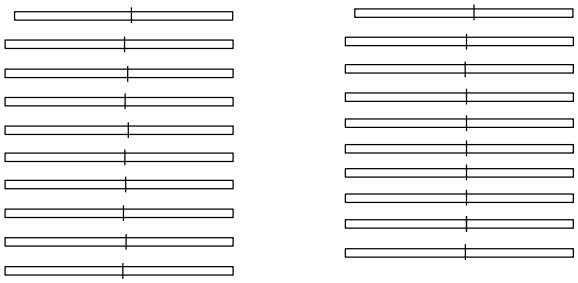


Figura 11: Posizione delle stanghette di misura nelle pagine di musica mostrate nelle figure 9 (a sinistra) e 10 (a destra).

appare ordinato e preciso, forse addirittura migliore, perché sembra omogeneo: quasi 'computerizzato', si potrebbe dire. Ma qualcosa non va. Che cosa?

La risposta sta nell'uniformità perfetta e matematica dell'edizione più nuova. Si trovino le stanghette di battuta a metà di ciascun rigo: nella parte incisa a mano la loro posizione varia naturalmente, mentre nella versione più recente esse si allineano quasi alla perfezione. La figura 11 mostra questo aspetto in modo semplificato.

Nella parte generata al computer, perfino le singole teste delle note sono allineate in colonne verticali, facendo scomparire il profilo della melodia in una griglia piuttosto rigida.

Ci sono altre differenze, ovviamente: nell'incisione le linee verticali sono tutte più spesse, le legature rimangono più attaccate alla testa delle note e l'inclinazione delle aste orizzontali appare più varia. Anche se dettagli come questi possono sembrare eccessivamente cavillosi, il risultato è una parte più facile da leggere. Nella parte generata al computer tutte le linee sono quasi identiche e se il musicista distoglie lo sguardo dalla musica per un momento, si perderà sulla pagina.

LilyPond è stato progettato per risolvere i problemi riscontrati nei programmi esistenti e per creare bella musica che imiti i migliori spartiti incisi a mano. I suoi autori l'hanno scritto proponendosi di rispondere a una domanda fondamentale: perché la maggior parte della musica scritta al computer non riesce a eguagliare la bellezza e l'equilibrio di uno spartito inciso a mano?

2 LilyPond: il *music engraving* reimplementato

2.1 Che cos'è LilyPond?

LilyPond è un software e un formato di file per il *music engraving* orientato a produrre spartiti composti secondo le regole in vigore ai tempi in cui la musica veniva incisa a mano su lastre. È un programma libero, disponibile per molti sistemi operativi e rilasciato sotto i termini della GNU General Public License. Il suo nome ('stagno delle ninfee', in inglese) si ispira al progetto Rosegarden e a una conoscente dei suoi due autori di nome Suzanne, che in ebraico significa 'giglio'.[3]

La sua storia comincia nel 1996, quando Han-Wen Nienhuys e Jan Nieuwenhuizen decidono di abbandonare il progetto MPP (MusiXTEX Pre-Processor) a cui collaboravano da un anno per creare un nuovo programma di incisione musicale liberamente basato su MusiXTEX. Due anni dopo esce la versione 1.0 e il 24 settembre 2003 la versione 2.0. Sono sempre disponibili due versioni del programma: una stabile, contrassegnata da un numero pari (2.18.2, al momento di andare in stampa) e una di sviluppo, contrassegnata da un numero dispari (2.19.14). Oggi gli autori originari sono coinvolti solo marginalmente nella vita di LilyPond, il cui sviluppo è nelle mani di un gruppo di programmatori che si occupa anche di scriverne la documentazione e tradurla nelle varie lingue.[4]

Il programma è in una fase di sviluppo molto attiva e numerose comunità di utenti e programmatori si sono formate per discuterne e migliorarne le prestazioni.

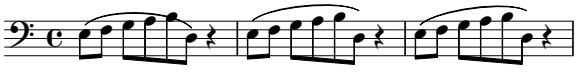
2.2 A chi va la mela d'oro?

Le osservazioni e l'esempio contenuti in questa sezione sono tratti da (The LilyPond Development Team, 2014a).

Nascendo per rispondere alla 'domanda fondamentale', LilyPond incorpora alcune funzioni sconosciute agli altri programmi, le quali rendono i suoi spartiti particolarmente attraenti: la scalatura ottica dei font (la forma dei simboli musicali varia al variare delle dimensioni del rigo), la spaziatura ottica delle note (paragrafo 1.2.1), le lineette supplementari più brevi se precedute da un accidente, la possibilità di spaziare proporzionalmente le note sono solo alcune di queste.

Prima di potersi fregiare del titolo di maestro incisore, l'artigiano alle dipendenze di una casa editrice musicale doveva avere alle spalle almeno due quinquenni di esperienza, dopo i quali si riteneva in grado di riprodurre nel modo migliore qualunque originale, decidendo autonomamente su ogni aspetto della resa grafica e creando con il proprio stile la bellezza dell'incisione. Anche LilyPond sa decidere tra le (virtualmente) illimitate soluzioni per disegnare questo o quell'elemento sul rigo ed è istruito per farlo sempre 'nel modo migliore'.

Un esempio chiarirà le idee. Si osservi il frammento seguente: quale legatura dovrebbe adottare LilyPond?



I pochi libri sull'incisione musicale in circolazione contengono talmente poche regole in proposito da

3. In inglese, *lily* significa sia 'ninfea' sia 'giglio'.
4. Buona parte della documentazione di LilyPond è disponibile anche in italiano.

vanificare anche la sola idea di scrivere un algoritmo comprensivo di tutte le possibilità. Molto meglio, si è pensato, descrivere dettagliatamente gli *obiettivi generali* della scrittura musicale e permettere a LilyPond di giudicare da sé l'attrattiva delle varie alternative. Poi, a ciascuna configurazione possibile si assegna un punteggio di 'bruttezza' sulla base di determinati parametri e si sceglie la configurazione meno 'brutta'. In questo caso LilyPond sceglie la legatura più a destra, non perché sia la più bella legatura possibile (la mano dell'incisore avrebbe fatto sicuramente di meglio) ma perché è la meno peggio delle tre. È un po' come fa TeX quando assegna i punti alla *badness* di qualche elemento del documento.

Questa tecnica è del tutto generale e viene adottata dal programma per decidere in modo ottimale su configurazione di aste orizzontali, legature di valore e punti negli accordi, interruzioni di rigo e pagina. Naturalmente, LilyPond non potrebbe compiere alcuna scelta se i suoi sviluppatori non lo migliorassero giorno dopo giorno confrontandone di continuo i risultati con gli spartiti incisi a mano, che costituiscono il termine di paragone obbligato.

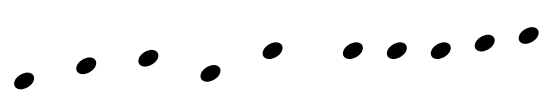
2.3 Una creatura multiforme

Le osservazioni e gli esempi contenuti in questa sezione sono tratti da (The LilyPond Development Team, 2014a).

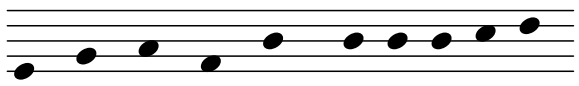
La notazione musicale è un sistema per registrare la musica che si è evoluto nel corso degli ultimi mille anni fino a raggiungere la forma attuale, che risale agli inizi del Rinascimento. Le sue applicazioni spaziano dalle melodie a una sola voce ai mostruosi contrappunti di una grande orchestra.

Quali possibilità si hanno di catturare una simile creatura e costringerla entro la gabbia di un programma per computer? La soluzione degli sviluppatori è stata di spezzare la notazione in 'pezzi' digeribili e programmabili: ogni tipo di simbolo è gestito da un *plugin*, un modulo completamente indipendente così da poter essere sviluppato e migliorato separatamente. Questi moduli sono chiamati *engraver* ('incisori') per analogia con l'artigiano che traduce le idee musicali in simboli grafici.

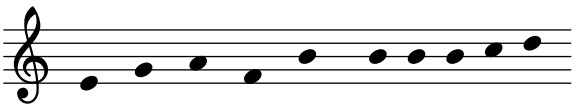
L'esempio seguente comincia con il plugin per la testa delle note, il `Note_heads_engraver`:



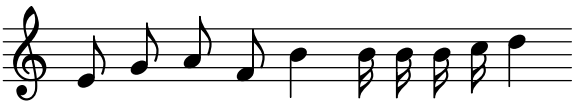
Poi, lo `Staff_symbol_engraver` aggiunge il rigo,



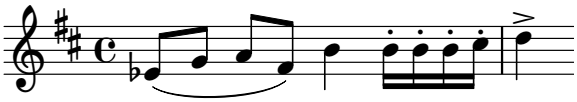
il `Clef_engraver` stabilisce un punto di riferimento per le altezze,



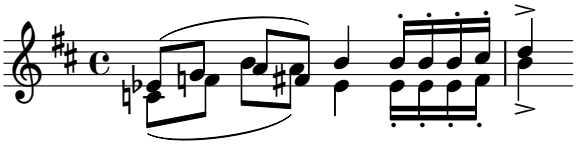
e lo `Stem_engraver` aggiunge i gambi:



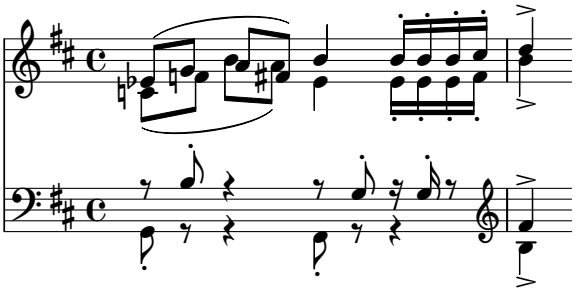
Ogni volta che compare una testa di nota nuova (o più teste, se si tratta di un accordo) lo `Stem_engraver` viene avvisato: subito crea un gambo e lo unisce alla testa della nota. Aggiungendo incisori per code, legature, accenti, accidenti, stanghette di battuta, indicazioni di tempo e armature di chiave, si ottiene una notazione completa:



Questo sistema funziona bene per la musica a una sola voce, ma per quanto riguarda la polifonia? Nella notazione polifonica, molte voci possono condividere un solo pentagramma:



In questa situazione, accidenti e pentagramma sono condivisi, ma gambe, legature, code, eccetera appartengono in modo esclusivo a ciascuna voce, perciò gli incisori devono essere raggruppati. Quelli per testa delle note, gambe, legature, eccetera, vanno in un gruppo chiamato *Voice context*; quelli per armatura di chiave, accidenti, battute, eccetera, in un gruppo chiamato *Staff context*. Nel caso della polifonia, un singolo Staff context contiene più di un Voice context. Analogamente, più Staff context possono essere messi in un singolo *Score context*. Quest'ultimo rappresenta il 'contesto notazionale' di livello superiore. Ecco un esempio ancora più complesso:



2.4 Il linguaggio di LilyPond

Il linguaggio di LilyPond non è né un linguaggio di programmazione né un linguaggio che descrive graficamente la partitura; è piuttosto un linguaggio

```
\language "italiano"
\relative {
  \key sol \major
  \time 3/4
  sol'4.\mordent re8( sol la)
  si8 <sol dod>
  <fad re'>4-. <re fad do'>-.
  <re sol si>2.\fermata
  \bar "|."
}
```

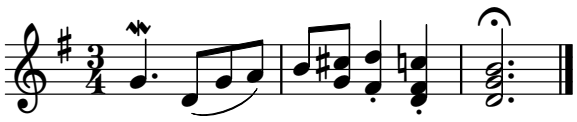


FIGURA 12: Un semplice sorgente LilyPond (in alto) e il corrispondente PDF composto (in basso).

semantico, che descrive la *musica* rappresentata nella partitura.

Un approccio di questo tipo, unito alla capacità di LilyPond di prendere autonomamente le decisioni sul piano grafico, fa sì che molte operazioni che in altri programmi rovinano l'impaginazione richiedendo un intervento manuale da parte dell'utente (come la trasposizione o il cambiamento delle interruzioni di riga) siano in LilyPond quasi indolori.

Nonostante che il programma decida, questo linguaggio prevede la possibilità di ritoccare l'aspetto della musica. separando abbastanza nettamente le istruzioni che regolano gli aspetti grafici dello spartito dalla codificazione del contenuto strettamente musicale.

La sintassi di LilyPond è estremamente semplice e intuitiva, almeno per quanto riguarda la scrittura di musica con notazione standard. La figura 12 ne mostra un esempio.

2.5 Punti di forza

Uno dei principali punti di forza di LilyPond è l'utilizzo di un formato di input testuale, il che permette di leggere *sempre* il contenuto dei sorgenti anche senza avere a disposizione il programma. Inoltre, i file LilyPond si comportano bene con i sistemi di controllo della versione, risultando quindi anche un ottimo formato da adottare per progetti a cui collaborino più persone.

Le *variabili* sono un elemento fondamentale del linguaggio di LilyPond: esse possono contenere da una singola nota alla parte completa di uno strumento fino a un'intera partitura; oppure, del semplice testo o elementi grafici o ancora gruppi di impostazioni.

LilyPond contiene anche l'interprete Guile del linguaggio di programmazione Scheme, che permette di accedere alle proprietà dei vari elementi musicali e grafici e di scrivere direttamente nel codice sorgente ulteriori funzioni per generare o manipolare la musica e per automatizzare certi compiti.

Formato testuale e meccanismo delle variabili permettono all'utente di riutilizzare il codice in grande misura. Infatti, dovendo scrivere la stessa melodia più di una volta (come accade nel caso di una partitura e delle corrispondenti parti staccate, per esempio), anziché copiarla e incollarla si può inserirla in una variabile e utilizzare quest'ultima in diversi punti del file o addirittura in file distinti, riducendo la possibilità di errori e mantenendo sincronizzate le varie parti in caso di modifiche o correzioni. O ancora, se un problema viene risolto tramite una funzione scritta in Scheme, basta mettere questa funzione in un file apposito e includerlo nel documento o nei documenti in cui la funzione è necessaria, come si farebbe con il comando `\input` di LaTeX.

Le possibilità di automazione offerte da LilyPond sono di grande valore per i compositori di musica algoritmica: non solo essi possono istruire i propri programmi in modo che generino file LilyPond, ottenendo allo stesso tempo la partitura e il file MIDI corrispondente, ma addirittura possono scrivere gli algoritmi in Scheme direttamente in un sorgente LilyPond.

Nel blog *Scores of Beauty* si trova un ottimo esempio di come si possano semplificare compiti molto faticosi generando codice LilyPond tramite programmi esterni.[5] In questo caso, con poche righe di codice Python e LilyPond si riescono a generare tutti i possibili schemi ritmici (successioni di note e pause) da due a otto tempi (ma potenzialmente lo stesso codice è utilizzabile per qualunque numero di tempi), senza bisogno di intervenire a mano nell'impaginazione.

L'approccio semantico alla musica di LilyPond è di per sé un grande punto di forza. Infatti, la sintassi del programma codifica il contenuto musicale con grande precisione, rendendolo il più possibile indipendente da come poi la musica sarà impaginata e stampata. Ciò permette da un lato di ottenere diverse rappresentazioni grafiche della stessa musica a partire da un unico file sorgente, dall'altro di ottenere in modo semplice notazioni complesse.

La figura 13 mostra all'opera contemporaneamente questi due aspetti. I tre esempi sono stati realizzati a partire dal medesimo codice LilyPond, infatti la musica è la stessa. Le leggere differenze dall'uno all'altro dipendono unicamente da diverse impostazioni generali assegnate a ciascuno e scritte in punti del file distinti dalle parti di codice strettamente musicale. Inoltre, la complessa notazione polimetrica con molti gruppi irregolari che si può osservare, in cui le note che cadono nello stesso istante sono perfettamente sincronizzate

5. http://lilypondblog.org/2013/07/creating-anything-you-can-imagine-with-finale/ http://lilypondblog.org/2013/07/programmatically-generating-lilypond-input/.

orizzontalmente, è ottenuta senza alcun bisogno di correggere manualmente la posizione delle note.

3 Altri programmi di notazione musicale

3.1 Programmi con input testuale

3.1.1 SCORE

SCORE, nato nel 1971, può considerarsi il decano dei programmi di notazione musicale. Scritto in Fortran per il sistema operativo DOS da Leland C. Smith, professore di Musica all'università di Stanford, negli Stati Uniti, negli anni '90 del Novecento è stato lo standard dell'editoria musicale, utilizzato da numerosi editori tra cui Schott Music, Schirmer/Mosel Verlag, Universal Edition e Casa Ricordi.[6]

I suoi principali punti di forza sono il completo controllo della posizione dei simboli sulla pagina e la possibilità di riprodurre notazioni arbitrarie (per esempio nuovi simboli o sistemi di scrittura inventati dal compositore). Nonostante che queste caratteristiche lo rendano piuttosto complicato da utilizzare, SCORE permette all'utente esperto di creare partiture di elevatissima qualità e di godere della stessa libertà e precisione di un incisore tradizionale.

In SCORE la musica viene inserita in modalità testuale. Il programma fornisce un'anteprima della musica in cui i simboli, di qualità ridotta, sono collocati sulla pagina con la stessa precisione che avrebbero nel documento finito (nei limiti della risoluzione dello schermo).

A differenza di LilyPond, l'approccio di SCORE alla musica è di tipo *grafico*: l'inserimento della musica avviene una pagina alla volta e riga per riga, separando le varie caratteristiche delle note (altezza, durata, articolazioni, eccetera) in altrettante fasi di input. Infatti, una delle peculiarità del programma è il metodo di inserimento a cinque stadi, descritto di seguito.

1. Stadio delle *altezze* delle note, in cui si inseriscono anche chiave, armatura di chiave, indicazione di tempo e stanghette di misura.
2. Stadio del *ritmo*, in cui si inseriscono in successione le durate delle note inserite nel primo stadio.
3. Stadio dei *simboli*, in cui si inseriscono segni di articolazione (accenti, staccati), indicazioni dinamiche (*piano*, *mezzoforte*), abbellimenti (trilli, mordenti) e tutti gli altri simboli.
4. Stadio delle *aste orizzontali*, in cui si determina come le note di valore uguale o inferiore a un ottavo sono unite tra loro.
5. Stadio delle *legature*.

6. http://scoremus.com/

Nella figura 14, che mostra il programma all'opera, sono presenti solo gli stadi delle altezze e del ritmo: per esempio, `TR` è la chiave di violino (*treble*), `K1S` indica che c'è un diesis (*sharp*) in chiave, `Q` è il valore di un quarto e `E` quello di un ottavo (*eighth*).

Ciascun elemento grafico è definito internamente da un insieme di parametri numerici che l'utente può modificare per rifinire l'impaginazione (per spostare una nota orizzontalmente o per ritoccare la forma di una legatura, per esempio).

La musica della figura 14 si può scrivere in LilyPond come segue:

```
{
  \clef treble
  \key g \major
  \time 4/4
  \relative {
    d''4. c8 b4 g fis a g b
    a c b d d, fis g g
    \bar "|."
  }
}
```

Come si può osservare, il codice LilyPond risulta *immediatamente* più comprensibile: contiene meno abbreviazioni, per indicare i valori utilizza numeri anziché lettere e non separa durata e altezza di una stessa nota in punti differenti del codice. Si noti che in entrambi i codici la scrittura dei valori delle note è abbreviata (in SCORE, `QX14` significa 14 note da un quarto; in LilyPond, una nota senza valore esplicito vale quanto la nota precedente).

Dopo la morte del professor Smith, avvenuta il 17 dicembre 2013, non è ancora chiaro quale sarà il futuro di SCORE, che rimane comunque una pietra miliare nella tipografia musicale.

3.1.2 MusiXTEX e la sua famiglia

L'idea di stampare musica con TEX nasce nel 1987 con la tesi di laurea di Andrea Steinbach e Angelika Schofer, che scrivono il pacchetto MuTEX, limitato a un singolo rigo musicale. Nel 1991, Daniel Taupin scrive MusicTEX, che permette di scrivere più righi musicali, ma produce un output di minore qualità, non potendo gestire automaticamente in modo flessible la spaziatura orizzontale.

Per risolvere i problemi di MusicTEX, nel 1997 Daniel Taupin, Ross Mitchell e Andreas Egler creano MusiXTEX, introducendo il sistema dei tre passaggi, descritto di seguito.

1. Composizione del sorgente con TEX, che produce un'impaginazione preliminare della musica.
2. Composizione con `musicflx`, che determina la migliore spaziatura orizzontale della musica.
3. Ulteriore compilazione con TEX, che tiene conto del lavoro fatto nel secondo passaggio.

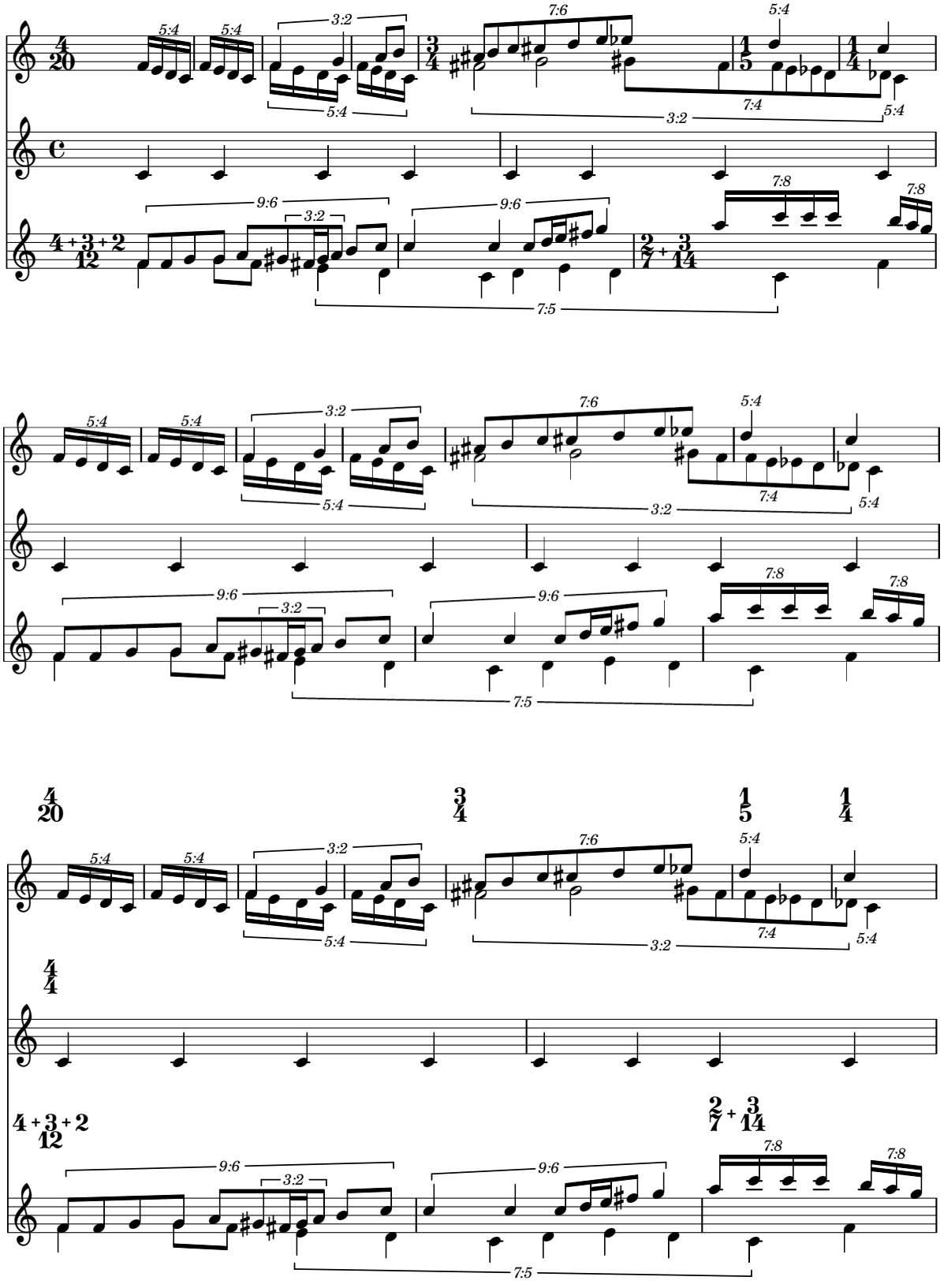


FIGURA 13: Tre realizzazioni dello stesso esempio di notazione polimetrica con gruppi irregolari nidificati prodotte da LilyPond a partire dallo stesso codice sorgente. Le leggere differenze osservabili dipendono dai diversi comandi di impostazione generale della pagina utilizzati nei tre casi.

```
TR/K1S/COM/D5/C/B4/G/M/F/A/G/B/M/A/C5/B4/D5/M/D4/F/G/G/MH;
Q./E/QX14;
```

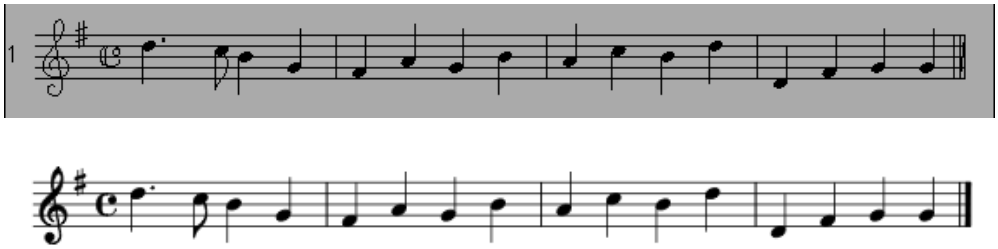


FIGURA 14: SCORE in azione. Dall'alto al basso: un input testuale; lo stesso input come appare nell'anteprima; e nel PDF composto.

Nel corso degli anni, molte persone hanno contributo in vario modo a MusiXTEX, anche dopo la morte di Daniel Taupin, avvenuta nel 2003.[7]

Il linguaggio di MusiXTEX richiede di decidere su numerosi aspetti dello spartito *mentre si scrive il codice*, tanto sul piano grafico (come la quantità di spazio tra le note) quanto su quello musicale (come la suddivisione delle aste orizzontali) tramite lunghe serie di comandi dai nomi criptici, il che rende molto noioso scriverne il codice e particolarmente difficile leggerlo. Va da sé che gran parte di queste decisioni potrebbero essere prese facilmente dal programma in modo automatico.

Proprio per questo motivo Don Simons ha scritto PMX, un preprocessore per MusiXTEX che permette di inserire la musica con un linguaggio molto più intuitivo e decide automaticamente in numerose situazioni. Successivamente, Dirk Laurie ha scritto M-Tx, un altro preprocessore che si appoggia a PMX e facilita l'inserimento delle parole nei brani di musica vocale. La sintassi più semplice richiesta da questi due programmi si sconta con una minore flessibilità nell'impaginazione, ma entrambi permettono di inserire del codice TEX (e quindi anche MusiXTEX) arbitrario nel file, rimediando così ad alcuni loro limiti. La figura 15 confronta gli input dei diversi programmi descritti in questo paragrafo.

MusiXTEX è presente su CTAN e fa parte sia della TEX Live sia di MiKTEX.

3.2 Programmi con input grafico

Gli *score writer* ('programmi di notazione musicale') oggi più diffusi, non solo tra i musicisti che si scrivono la musica da sé ma anche nelle redazioni di molte case editrici musicali, sono Finale e Sibelius, la cui interfaccia grafica del tipo WYSIWYG li rende più intuitivi all'utente inesperto di quanto non farebbe un'interfaccia testuale.[8]

Non è semplice istituire un confronto tra i primi due e LilyPond. Circoscrivendo il paragone alla notazione musicale standard inserita senza ritocchi grafici di alcun tipo, allora in genere LilyPond produce spartiti più gradevoli e tipograficamente corretti. Beninteso, con tutti e tre è possibile ottenere una qualità tipografica da pubblicazione, ma l'impresa richiede talmente tanta esperienza da essere adatta ai soli utenti esperti.

Ci sono però alcuni aspetti in cui LilyPond risulta vincente indipendentemente dai ritocchi, ed è proprio per realizzare *quei* dettagli che si ispira all'incisione tradizionale su lastre: linee con le estremità arrotondate invece che appuntite (come le terminazioni delle legature), un font con corpi ottici diversi e linee di spessore diverso a seconda della dimensione del rigo.

Per un confronto dettagliato tra LilyPond e Finale o Sibelius, si rimanda il lettore ad alcune pagine disponibili in Rete.[9]

Nel mondo del software libero e open source si cita spesso la necessità di utilizzare formati di file, possibilmente di tipo testuale, con specifiche liberamente disponibili, per evitare di legarsi a un formato di file leggibile da un solo programma (il cosiddetto *vendor lock-in*) che in futuro potrebbe essere ritirato dal mercato o diventare incompatibile con i computer delle successive generazioni. Molti ritengono remota la possibilità che un file diventi illeggibile con il passare del tempo: dopo tutto, si dice, non sembra affatto che Microsoft® abbia intenzione di abbandonare lo sviluppo di Office®.

Parlando di software per la scrittura musicale, però, questa possibilità non è così peregrina. Nel 2012, Avid, proprietaria di Sibelius, ha chiuso il proprio ufficio di Londra licenziando gli sviluppatori del programma. Sebbene la società abbia poi assunto nuovo personale e rilasciato una nuova ver-

7. http://icking-music-archive.org/software/htdocs/htdocs.html

8. Finale è un marchio registrato della MakeMusic, Inc. Sibelius è un marchio registrato della Avid Technology, Inc.

9. http://www.musicbyandrew.ca/finale-lilypond-1.html
http://bartruffle.blogspot.it/2012/09/musescore-vs-score-vs-lilypond-vs.html
http://www.denemo.org/score-writers-comparison

```
\input musixtex
\parindent10mm
\setname1{Piano}
\setstaffs12
\generalmeter{\meterfrac44}
\nobarnumbers
\startextract
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\ql l\sk\ql n\en
\bar
\Notes\ibu0f0\qb0{dgf}|\qlp i\en
\notes\tbu0\qb0g|\ibbl1j3\qb1j\tbl1\qb1k\en
\Notes\ibu0f0\qb0{cge}\tbu0\qb0g|\hl j\en
\endextract
\end
```

```
2 1 4 4 4 4 0 0
1 1 20 0.12
Piano
tt
./
% Bars 1-2
c8 g+ e g c- g+ e g | d g f g c- g+ e g Rb /
c2+ e4 g | bd4- c1 d c2 /
```

```
Style: piano
Piano: Voices MD MS; Clefs G G; Continuo
Name: Piano
Meter: 4/4
%% w120m
c2+        e4    g   | b4d-  c1 d c2        |
c8+ g+ e g c- g+ e g | d g f g    c- g+ e g |
```

```
\new PianoStaff \with {
  instrumentName = "Piano"
  \numericTimeSignature
} <<
  \new Staff {
    \clef treble
    \relative {
      c''2 e4 g
      b,4. c16 d c2
    }
  }
  \new Staff {
    \clef bass
    \relative {
      c8 g' e g c, g' e g
      d8 g f g c, g' e g
    }
  }
>>
```

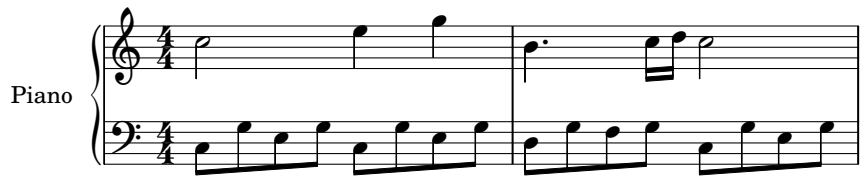


Figura 15: Incipit della Sonata per pianoforte in do maggiore K 545 di Wolfgang Amadeus Mozart (1756–91) codificato in diversi formati. Dall'alto al basso: MusiXTEX, PMX, M-Tx e LilyPond (del quale, alla fine, si dà anche la versione composta)

sione del software contenente alcuni miglioramenti, non è affatto chiaro quale sarà il suo futuro. Sibelius non è stato ancora abbandonato, ma per una casa editrice musicale è inaccettabile dipendere in modo strategico da un programma che fra pochi mesi potrebbe non esistere più.

Sostituire il programma di notazione musicale significa anche ripensare buona parte delle procedure di lavoro: si tratta quindi di un'operazione molto costosa per l'azienda e non è certo questa la sede in cui proporre soluzioni. Tuttavia, un programma come LilyPond potrebbe risultare sempre più interessante per l'industria della musica stampata.

4 Quale editor utilizzare?

I file `.ly` e `.lytex` (due estensioni con cui registrare i file LilyPond) sono semplici file di testo esattamente come i file `.tex`, perciò si possono scrivere con un editor di testi qualunque. Come accade per LaTeX, ne esistono di specializzati per LilyPond particolarmente adatti a scrivere musica 'pura' che, dati gli scopi di questo articolo, non verranno analizzati.[10]

Se il parco degli editor per LilyPond è piuttosto nutrito, e ancor più lo è quello di LaTeX, tuttavia scarseggiano i veri e propri factotum utili nel nostro caso: editor che comprendano entrambi i linguaggi (LaTeX e LilyPond) e che contemporaneamente siano capaci di lanciare `lilypond-book` sul file sorgente. L'alternativa è una sola: utilizzare un editor LaTeX per scrivere il sorgente e la riga di comando per eseguire su di esso `lilypond-book`. Qualche ricerca ha selezionato tre editor adatti ai nostri scopi.

Il primo è TeXShop, che gira solo su (Mac) OS X.[11] A partire dalla versione 2.36, infatti, questo editor per LaTeX arriva con tre motori dedicati a LilyPond scritti da Nicola Vitacolonna che permettono di lanciare sul file sorgente i programmi `lilypond`, `lilypond-book` e `convert-ly`, rendendo di fatto TeXShop un editor completo anche per scrivere la musica.[12]

Il secondo è jEdit, un potente editor di testi multipiattaforma che tramite i due plugin LaTeXTools e LilyPondTool permette di realizzare qualunque tipo di documento che contenga musica, dagli spartiti agli studi di musicologia.[13] Si noti, però, che nonostante LilyPondTool sia stato per anni uno degli strumenti più potenti per scrivere la musica con LilyPond, dal 2012 il suo autore ne ha cessato lo sviluppo. Tuttavia, è ancora disponibile per chi volesse provarlo.[14]

L'ultimo è LyX, un editor di testi utilizzabile per scrivere documenti LaTeX che dalla versione 2.0 comprende il codice LilyPond ed esegue `lilypond-book`.[15]

Probabilmente esistono altri editor di testi che potrebbero fare al caso nostro, se opportunamente 'preparati': sembra che Emacs, Vim e Kile siano tra questi.

5 Integrare LaTeX con LilyPond

Sia LaTeX sia LilyPond si prefiggono l'obiettivo di produrre documenti tipograficamente corretti ed eleganti. Preparando un testo per la stampa contenente musica, dunque, è naturale cercare di utilizzarli insieme per sfruttarne i rispettivi punti di forza.

In commercio circolano pubblicazioni contenenti musica di diversa natura: volumi di musicologia, monografie ed enciclopedie per studiosi e appassionati, partiture e spartiti per i musicisti, manuali didattici per gli studenti, solo per fare qualche esempio. Anche se ciascuno presenta problemi tipografici peculiari, le esigenze principali sono sempre due:

- poter inserire agevolmente simboli o frammenti musicali all'interno del documento, tipica del musicologo; e
- leggere uno spartito impaginato in modo che faciliti l'esecuzione, tipica del musicista.

I paragrafi seguenti propongono alcune soluzioni per soddisfare le esigenze dell'uno e dell'altro. Una tipica edizione musicale userà una combinazione di tutte queste tecniche.

5.1 Dalla parte del musicologo

5.1.1 Frammenti musicali: `lilypond-book`

Nei documenti di tipo musicologico spesso è necessario intercalare brevi frammenti musicali al testo. Naturalmente si possono produrre con LilyPond per includerne successivamente i relativi PDF come si farebbe con una normale figura, eventualmente ritagliandoli (o con le opportune opzioni di `\includegraphics` o modificando le dimensioni della pagina direttamente con LilyPond). Se i frammenti sono numerosi, però, adattarli a mano è noioso e fa perdere tempo.

Per automatizzare queste operazioni si può ricorrere al programma `lilypond-book`, incluso in LilyPond. Si tratta di un preprocessore che prende come input un file LaTeX (da registrare convenzionalmente con estensione `.lytex`) in cui ci possono essere dei comandi 'speciali' (sconosciuti a LaTeX ma compresi da `lilypond-book`) che permettono

10. http://lilypond.org/easier-editing http://en.wikipedia.org/wiki/List_of_scorewriters http://en.wikipedia.org/wiki/Comparison_of_scorewriters.
11. http://pages.uoregon.edu/koch/texshop/
12. http://users.dimi.uniud.it/~nicola.vitacolonna/software/lilypond-texshop/
13. http://www.jedit.org/
14. http://lilypondtool.blogspot.it/
15. http://www.lyx.org/

di inserire codice LilyPond nel sorgente LaTeX in una delle due forme seguenti:

- semplici frammenti più o meno lunghi scritti direttamente nel sorgente (comando `\lilypond` e ambiente lilypond); oppure
- interi file esterni (comando `\lilypondfile`).

`lilypond-book` estrae i diversi frammenti di codice musicale dal sorgente e li dà in pasto a LilyPond, il quale genera i PDF corrispondenti; contemporaneamente, genera un file `.tex` omonimo e identico all'originale, nel quale i PDF appena generati vengono inclusi al posto dei comandi speciali elencati sopra. A questo punto, basta ricomporre il file `.tex` come al solito per ottenere il documento completo degli esempi musicali.[16]

Vediamo ora un semplice esempio. Si crei una cartella di lavoro, si apra un nuovo file con l'editor LaTeX, lo si registri come `esempio.lytex` nella cartella appena creata e vi si copi il codice seguente:

```
\documentclass[a4paper]{article}

\begin{document}

Breve documento preparato
per \verb+lilypond-book+.

Possiamo inserire un frammento
musicale con l'ambiente
\verb+lilypond+:

\begin{center}
 \begin{lilypond}
  \language "italiano"
  \relative { do'4 re mi fa }
 \end{lilypond}
\end{center}

Esistono alcune opzioni
utili per l'ambiente
\verb+lilypond+: per esempio,
\verb+staffsize+ cambia
la dimensione del pentagramma
e \verb+notime+ sopprime
le indicazioni di tempo:

\begin{lilypond}[%
  staffsize=10,notime]
   { e'4 }
\end{lilypond}

\end{document}
```

Si apra un terminale, ci si porti nella cartella di lavoro e si componga il file con il comando

```
lilypond-book --output=out /
--pdf esempio.lytex
```

16. Si noti che utilizzando TeXShop (sezione 4) `lilypond-book` si occupa anche di ricomporre il documento con LaTeX. Non sappiamo se questo accada anche con gli altri editor lì descritti.

(gli spazi sono significativi; il carattere `/` indica che il comando va dato tutto su una sola riga). Al termine della composizione, nella cartella iniziale si vedrà una nuova cartella `out` contenente molti file tra cui `esempio.tex`, che basterà comporre come al solito con LaTeX per ottenere il risultato mostrato nella figura 16.

`lilypond-book` permette di inserire in un documento frammenti musicali di lunghezza arbitraria. Innanzitutto il programma determina la larghezza della colonna di testo in base alle impostazioni contenute nel preambolo del file `.lytex`; successivamente assegna la stessa misura al rigo musicale composto da LilyPond.

Se un frammento eccede la larghezza della riga, è suddiviso in più file che vengono inclusi nel file `.tex` uno sotto l'altro, permettendo di 'spezzare' la musica tra due pagine là dove serve. Tuttavia, questa modalità delega la spaziatura verticale tra i righi a LaTeX, ignorando quella (molto migliore) stabilita da LilyPond, il che può portare a risultati non ottimali, sia sul versante tipografico sia su quello pratico: in questi casi, può essere più conveniente preparare e inserire a mano i file PDF prodotti con LilyPond, oppure seguire il metodo descritto nel paragrafo 5.2.

La routine di LilyPond è piuttosto lenta, e dover ricomporre da capo tutti gli esempi musicali ogni volta potrebbe diventare un'operazione *molto* lunga. Perciò, se si modifica il file `.lytex` e si lancia nuovamente su di esso `lilypond-book`, il programma riconosce i frammenti musicali modificati e ricompone solo quelli.

La descrizione completa del programma `lilypond-book` è contenuta in (The LilyPond Development Team, 2014b), a cui si rimanda il lettore.

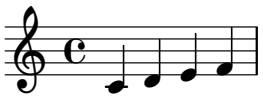
5.1.2 Frammenti musicali: `musicexamples`

Nel paragrafo 5.1.1 abbiamo visto come codice LaTeX e codice LilyPond possono convivere nello stesso file. Ma qual è il modo migliore di gestire gli esempi musicali in un documento?

La via più semplice è inserirli in altrettanti ambienti figure, sfruttandone la numerazione progressiva delle figure e il tipo di didascalia. Si può accettare la cosa in documenti non specialistici in cui gli inserti musicali compaiano solo occasionalmente, ma non in pubblicazioni musicologiche o didattiche in cui i frammenti di musica sono molti o moltissimi.

In primo luogo, se gli esempi musicali rivestono un'importanza fondamentale nel documento, allora è preferibile numerarli, gestirli ed elencarli separatamente. Inoltre, può capitare che gli esempi più lunghi debbano essere spezzati in più pagine, situazione che figure non è in grado di gestire. Infine, per ragioni tipografiche può essere utile ruotare i frammenti di musica o metterli sulla pagina in modo che il testo si disponga attorno ad essi. Si

Esempio di documento preparato per `lilypond-book`.
Possiamo inserire un frammento musicale con l'ambiente `lilypond`:



Esistono alcune opzioni utili per l'ambiente `lilypond`: per esempio `staffsize` cambia la dimensione del pentagramma e `notime` sopprime le indicazioni di tempo: 

FIGURA 16: Documento composto con `lilypond-book`.

possono risolvere questi problemi con altrettanti pacchetti per LaTeX, ma sarebbe un po' scomodo.

Ecco perché Urs Liska ha scritto il pacchetto musicexamples, che mette a disposizione in una volta sola tutti gli strumenti per soddisfare le esigenze appena descritte.[17]

I principali sono:

- l'ambiente galleggiante musicExample;
- l'ambiente musicExampleNonFloat, suo corrispondente 'statico', che può contenere più immagini distribuite su più pagine;
- il comando `\fullPageMusicExample` per inserire esempi musicali a pagina intera.

Inoltre, se sono caricati i pacchetti wrapfig, rotating, sidecap, e fltpage vengono generati comandi corrispondenti con le caratteristiche particolari di ciascuno (per esempio, `\sidewaysmusicexample`, con cui ruotare esempi musicali).[18] Il pacchetto musicexamples è ancora incompleto rispetto agli intenti dell'autore, ma nei suoi comandi di base è perfettamente funzionante e compatibile con `lilypond-book`.

L'esempio seguente mostra un semplice documento che utilizza musicexamples e `lilypond-book`, in cui figurano anche i comandi per modificare l'etichetta predefinita della didascalia e per modificare il titolo dell'elenco degli esempi musicali.

```
\documentclass[a4paper]{article}

\usepackage{musicexamples}
\setXmpCaptionLabel{%
  Esempio musicale}
\setXmpListName{%
  Elenco degli esempi musicali}

\begin{document}

\begingroup
\Urlmuskip = 0mu
\listofmusicexamples
\endgroup

\bigskip

\begin{musicExample}
  \begin{lilypond}
    \language "italiano"
    \relative { do'4 re mi fa }
  \end{lilypond}
  \caption{Un semplice
    esempio musicale.}
  \label{xmp:esempio}
\end{musicExample}

\end{document}
```

La figura 17 ne mostra il risultato.

Il pacchetto musicexamples *non* è presente in CTAN e va installato a mano nel proprio albero personale o sistemato nella cartella di lavoro.[19]

Per conoscerne tutte le potenzialità, si rimanda il lettore a (LISKA, 2013a).

5.1.3 *Simboli musicali:* `lilyglyphs`

Preparando i documenti di cui ci occupiamo in queste pagine, spesso è necessario inserire nel testo simboli musicali isolati, come se fossero normali caratteri.

La *Comprehensive LaTeX Symbol List* mostra i pochi simboli musicali disponibili per LaTeX, che messi insieme costituiscono una dotazione del tutto insufficiente, per nulla flessibile, incoerente con il font utilizzato da LilyPond: il che li rende decisamente inadatti in un contesto specialistico come il nostro.

Una soluzione ingenua è fornita dal programma `lilypond-book` (paragrafo 5.1.1) che però richie-

17. Urs Liska è uno degli utenti più attivi nel mondo di LilyPond. Insieme a Janek Warchoł, un altro utente e sviluppatore del programma, è autore del progetto openLilyLib, che raccoglie alcune risorse per LilyPond e LaTeX tra cui i pacchetti musicexamples (LISKA, 2013a) e lilyglyphs (LISKA, 2014) e i tutorial (BOOR, 2013) e (LISKA, 2013b).

18. In realtà si tratta di un aiuto fornito dal pacchetto newfloat, utilizzato da musicexamples per creare i diversi ambienti.

19. Basta scaricare i due file `.sty` elencati su `https://github.com/openlilylib/musicexamples`.

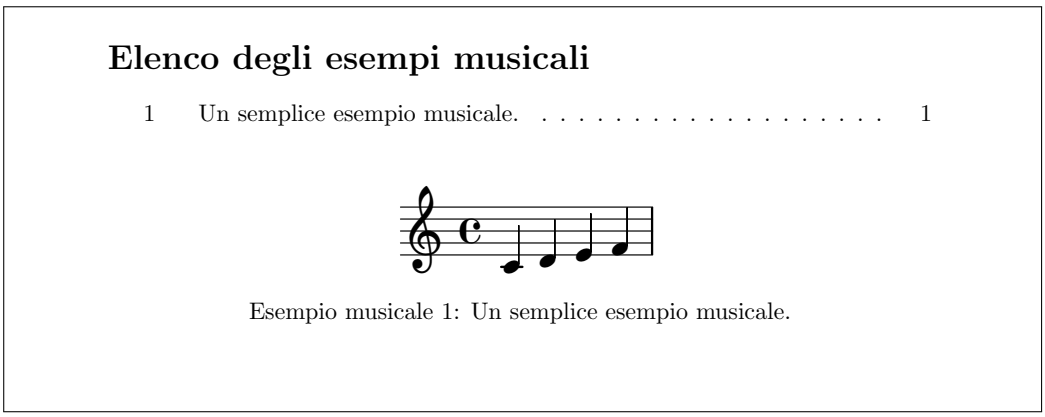


FIGURA 17: Esempio di alcune potenzialità del pacchetto musicexamples.

de di regolare a mano gli spazi caso per caso e di ricomporre il documento a ogni nuovo simbolo musicale inserito (e singole note o segni di alterazione potrebbero essere *molto* frequenti nel testo). I tempi dell'operazione, come si può immaginare, si allungherebbero un po' troppo.

Soccorre l'utente il pacchetto lilyglyphs, scritto da Urs Liska, che definisce i comandi per inserire nel documento simboli isolati del font Emmentaler/Feta,[20] predefinito di LilyPond, e alcune combinazioni predefinite di simboli già calibrati per essere messi in linea nel testo. I glifi singoli sono presi dal font OpenType e inseriti come tali; quelli complessi vengono precomposti in formato PDF e successivamente inclusi nel documento. Va da sé che il pacchetto richiede di utilizzare XeLaTeX o LuaLaTeX.

L'esempio seguente mostra lilyglyphs in azione.

```
\documentclass[a4paper]{article}
\usepackage{fontspec}
\usepackage{lilyglyphs}

\begin{document}

Con il pacchetto \verb+lilyglyphs+
si possono inserire ad esempio
delle note \halfNoteDotted{},
dei segni di alterazione
\flatflat{} e delle chiavi
\clefGInline{}.
Inoltre, si possono inserire
facilmente delle indicazioni
di tempo \lilyTimeCHalf{},
anche non standard
\lilyTimeSignature{7 + 5}{8},
e delle indicazioni dinamiche con
la tradizionale forma delle
lettere, come \lilyDynamics{mp}
e \lilyDynamics{fff}.

\end{document}
```

Se ne può vedere il risultato nella figura 18.

Tramite alcuni script in Python forniti con il pacchetto, si possono definire nuove combinazioni di simboli a partire dal corrispondente codice LilyPond. La documentazione di lilyglyphs, a cui si rimanda il lettore, le descrive nel dettaglio (LISKA, 2014).

Si noti che il pacchetto lilyglyphs è presente in CTAN e fa parte della TeX Live, ma *non* di MiKTeX, dove va installato *a mano* come indicato nella documentazione.

5.2 Dalla parte del musicista

Nel paragrafo 5.1.1 abbiamo accennato al fatto che l'impaginazione dei frammenti musicali su più righe prodotti da `lilypond-book` può non essere quella ottimale per l'esecuzione. La cosa non ha molta rilevanza nei documenti musicologici, dove la musica è inserita in virtù della propria importanza, cioè come semplice corredo esemplificativo del testo principale.

Ma se la musica va suonata, il musicista *deve* poter contare sulle caratteristiche che rendono gli spartiti di LilyPond eccellenti per l'esecuzione, come la regolazione della lunghezza ottimale dei righi, la spaziatura tra di essi e la posizione delle interruzioni di pagina. Tutto questo con `lilypond-book` si perde.

Come agire, allora? Semplice: utilizzando i due programmi al massimo delle rispettive potenzialità. LilyPond decide su composizione e formattazione della musica, LaTeX sulle questioni testuali. Le pagine di musica in formato PDF verranno incluse nel documento tramite il comando `\includepdf` del pacchetto pdfpages, che di qui in avanti si dà per caricato. Rispettando gli accorgimenti esaminati nelle prossime righe, si otterranno i risultati migliori.

Innanzitutto, qualche consiglio preliminare.

1. Gli spartiti devono essere pronti *prima* di cominciare a scrivere il documento LaTeX e mes-

20. Come si vede, il nome del font musicale predefinito di LilyPond oscilla tra due tradizioni casearie molto diverse: dato che non si può stabilire con certezza quale delle due prevalga, non si sbaglia indicandolo con i nomi di entrambe. Chi volesse saperne di più, può consultare https://lists.gnu.org/archive/html/lilypond-devel/2009-05/msg00212.html.

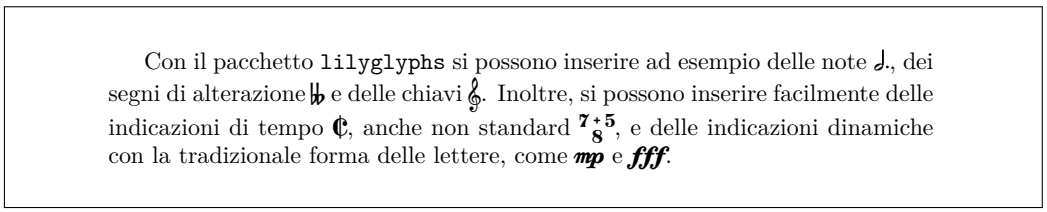


Figura 18: Alcune potenzialità del pacchetto lilyglyphs.

si preferibilmente nella stessa cartella che ne contiene i sorgenti.

2. Si raccomanda di produrre *un solo* PDF per ciascun brano di musica da inserire.

La numerazione delle pagine è lasciata a LaTeX, perciò va disattivata nei file LilyPond scrivendo

```
\paper { print-page-number = ##f }
```

A questo punto, si ripristina la numerazione anche delle pagine musicali scrivendo nel preambolo del documento

```
\includepdfset{pagecommand=%
   \thispagestyle{plain}}
```

che numererà ciascuna pagina del PDF incluso in basso al centro. Se plain non piace, si imposti un altro stile, anche personale, avendo cura che non interferisca con la musica.

Si ricordi che per impostazione predefinita il comando `\includepdf` inserisce nel documento solo la prima pagina di un PDF esterno. Per includerle tutte, basta scrivere

```
\includepdf[pages=-]{⟨nome del file⟩}
```

Di solito si vuole che i brani musicali compaiano nell'indice: i comandi `\chapter` e `\section` non sono adatti allo scopo perché prima del titolo stampano un numero. Siccome gli spartiti possiedono già il proprio titolo inserito da LilyPond, ecco come fare. Immaginando di dover inserire il brano intitolato *Composizione* e di volerne mandare il titolo nell'indice ma *senza che LaTeX stampi alcun titolo* prima della musica, basta scrivere

```
\addcontentsline{toc}{section}%
   {Composizione}
\includepdf[pages=-]%
   {Composizione.pdf}
```

Queste inclusioni di PDF convivono tranquillamente con i comandi di sezionamento di LaTeX, il che è particolarmente utile se al brano in questione va premessa un'informazione testuale di qualche tipo. Se la sezione porta lo stesso titolo del pezzo di musica, `\addcontentsline` non serve più.

Dato che i comandi di sezionamento producono titoli numerati sia nel corpo del documento, sia nell'indice, e dato che i titoli mandati nell'indice con `\addcontentsline` *non* sono numerati, come non lo sono i titoli dei brani musicali, si potrebbe voler uniformare il tutto omettendo la numerazione in tutto il documento. Le strade sono due: o si scrive una cosa del tipo

```
\section*{Composizione}
\addcontentsline{toc}{section}%
   {Composizione}
Testo introduttivo...
\includepdf[pages=-]%
   {Composizione.pdf}
```

o si aggiunge nel preambolo

```
\setcounter{secnumdepth}{-1}
```

e si leva l'asterisco dai comandi di sezionamento.

La soluzione descritta in questo paragrafo si ispira a (Boor, 2013).

6 Limiti e futuro di LilyPond

LilyPond non è un programma perfetto. Lo testimoniano quasi mille segnalazioni di bug ancora aperte, di cui più di cento avanzate nel solo 2014.[21] Grazie ai molti miglioramenti apportati al programma nel corso degli anni, con tutta probabilità molte di esse oggi non sono più valide. Tuttavia, rimangono ancora attuali alcuni problemi di lunga data: per esempio, il famigerato bug numero 34, che interessa la sincronizzazione degli abbellimenti all'inizio di un brano. Vi si può rimediare con un semplice accorgimento, ma i nuovi utenti di LilyPond rimangono invariabilmente disorientati.

Attualmente, l'obiettivo principale degli sviluppatori di LilyPond è sostituire la versione 1 di Guile con la versione 2. La prima, infatti, è considerata obsoleta tanto che a breve verrà eliminata da molte distribuzioni di Linux in favore di quella nuova. Se LilyPond non sarà aggiornato in tempo, non potrà più essere utilizzato.

Per il resto, il normale sviluppo di LilyPond consiste nel risolvere i bug individuati e nel razionalizzare e rifinire alcune parti del programma. David Kastrup, l'unico sviluppatore di LilyPond a tempo pieno, sostenuto nel proprio lavoro da donazioni degli utenti, da molti mesi si dedica proprio a semplificare alcune parti interne del software. Molti dei cambiamenti da lui apportati non sono direttamente visibili agli utenti, pur avendo un grande

21. https://code.google.com/p/lilypond/issues/list

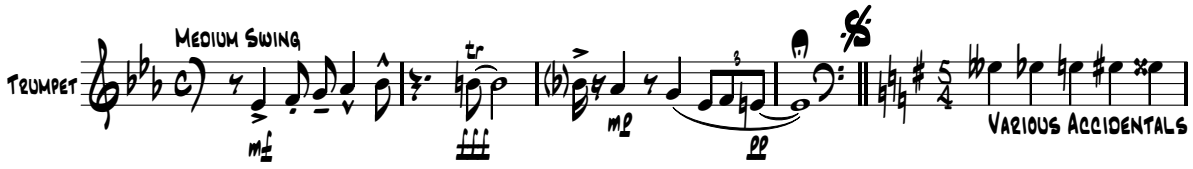


Figura 19: Il font LilyJAZZ.

impatto sul funzionamento interno, mentre altri portano alcune importanti novità nella sintassi di LilyPond, rendendola più intuitiva e coerente.

Una recente modifica operata da Mike Solomon ha migliorato notevolmente gli algoritmi di spaziatura di LilyPond: affinché gli oggetti grafici non collidessero, in precedenza ciascuno di essi era racchiuso in un singolo rettangolo e la spaziatura veniva calcolata tra questi rettangoli; ora, invece, il profilo reale degli oggetti è approssimato con le cosiddette *skylines*, il che comporta un grande risparmio di spazio e un maggiore equilibrio nell'impaginazione.

Un altro utente e sviluppatore di LilyPond, Janek Warchoł, di recente ha cominciato a ristrutturare gli algoritmi di allineamento degli oggetti, rendendoli più flessibili.

Per anni, un limite 'storico' di LilyPond è stata la quasi impossibilità di utilizzare per i simboli musicali font diversi dall'Emmentaler/Feta. Il primo tentativo di superarlo è stata la creazione del font Gonville, disegnato da Simon Tatham, che per poter essere utilizzato, però, richiede di fare una copia della cartella dei file che compongono LilyPond e di sostituire in essa i file di Emmentaler/Feta con quelli di Gonville.[22]

A esso è seguito LilyJAZZ, disegnato da Torsten Hämmerle, che, come rivela il nome, è un font in stile jazz (figura 19), più facile da utilizzare, non richiedendo di modificare l'installazione del programma.[23]

Negli ultimi mesi Nathan Ho, Joram Berger, Abraham Lee e Alexander Kobel hanno messo a punto una serie di funzioni che hanno reso più semplice utilizzare font musicali alternativi, inclusi alcuni font di edizioni storiche e di altri programmi musicali (come SCORE) e soprattutto il font Bravura di Daniel Spreadbury, che utilizza il nuovo standard Standard Music Font Layout (SMuFL), una specifica che permette di mappare tutti i simboli musicali convenzionali secondo le direttive Unicode. Le figure 20 e 21 mostrano alcuni di questi caratteri. Perciò, un futuro in cui LilyPond possa utilizzare in modo naturale diversi font musicali è ormai a portata di mano.

22. http://www.chiark.greenend.org.uk/~sgtatham/gonville/
23. http://lilypondblog.org/2013/09/lilypond-and-lilyjazz/

7 Progetti realizzati con LilyPond

Nonostante le sue indiscutibili qualità, LilyPond viene praticamente ignorato dalle grandi case editrici musicali. Infatti, è utilizzato in prevelenza da singoli professionisti della musica che realizzano le proprie edizioni preparandole per l'esecuzione e la pubblicazione.[24]

Tra essi figurano compositori come Kieren MacMillan e Mike Solomon e piccoli editori come Nicolas Sceaux e Adoro Music Publishing. Con LilyPond sono state realizzate anche edizioni critiche, tra le quali si ricorda quella dei *Lieder* di Oskar Fried,[25] curata da Alexander Gurdon e Urs Liska e preparata per la pubblicazione dallo stesso Urs Liska e da Janek Warchoł, e l'edizione dell'opera *Enea nel Lazio* di Tommaso Traetta, che costituisce la tesi di dottorato di Luca Rosetto Casel.

Si citano, inoltre, la nuova edizione del libro dei canti liturgici (*Liedboek*) delle chiese riformate dei Paesi Bassi e del Belgio fiammingo: si tratta di una delle più importanti e consistenti pubblicazioni realizzate con LilyPond, contando circa 1100 brani tirati in tre edizioni e 140 000 copie nel 2013.[26]

Per concludere, è interessante notare che questi progetti di respiro più o meno ampio possono ricadere positivamente sullo sviluppo di LilyPond: non è infrequente, infatti, che le modifiche apportate al programma o le nuove funzioni scritte per risolvere alcuni problemi inediti vengano inserite stabilmente nelle sue successive versioni.

A Risorse online

Questa sezione raccoglie le più importanti risorse su LilyPond presenti in Rete.

LilyPond Sito ufficiale di LilyPond. Contiene tutto ciò che serve per installare e utilizzare LilyPond.
http://www.lilypond.org/

Mailing list La mailing list degli utenti di LilyPond è il luogo principale per chiedere aiuto agli esperti. È richiesta la registrazione.
https://lists.gnu.org/mailman/listinfo/lilypond-user

Scores of Beauty È il blog ufficiale di LilyPond. Non si tratta di un botta e risposta (per questo

24. http://lilypond.org/productions.html
25. http://lilypondblog.org/2013/10/oskar-fried-complete-songs/
26. http://www.liedboek.nl/

c'è la mailing list) ma di una una raccolta di piccole trattazioni su diversi aspetti e problemi del programma.
http://lilypondblog.org/

The LilyPond Snippet Repository È una vasta collezione di esempi inviati dagli utenti di LilyPond, liberamente copiabili e utilizzabili nei propri documenti.
http://lsr.di.unimi.it/LSR/Search

openLilyLib È una collezione di risorse per chi è interessato a incidere musica con LilyPond e a comporre documenti di argomento musicologico con LaTeX. Qui si trovano alcuni tutorial e le versioni più aggiornate dei pacchetti lilyglyphs e musicexamples.
http://www.openlilylib.org/

Operation LilyPond È una serie di 25 videotutorial creati da Ben Lemon e pensati per l'utente principiante.
http : / / benlemon . me / blog / music / lilypond/operation-lilypond/

Fonti delle illustrazioni

Riferimenti bibliografici

Aa. Vv. (1996). *Enciclopedia della musica*. Le garzantine. Garzanti, Milano.

Boor, J. (2013). *Songbooks with LilyPond and LaTeX*. URL http://openlilylib.org/public/tutorials/createSongbooks.pdf.

Liska, U. (2013a). *musicexamples*. URL http://openlilylib.org/public/manuals/musicexamples_0-2.pdf. Versione 0.2.

— (2013b). *Plain Text files in Music*. URL http : / / openlilylib . org / public / tutorials/plaintextworkflows.pdf.

— (2014). *lilyglyphs*. URL http://www.ctan.org/pkg/lilyglyphs. Versione 0.2.3.

Mancini, F. (2013). *Metodologie e tecniche per l'editoria musicale*. URL http://www.fabiomancini.altervista.org/Universita/MetodologieTecnicheEditoriaMusicale.pdf.

The LilyPond Development Team (2014a). *Essay*. URL http://www.lilypond.org/doc/v2.18/Documentation/essay.pdf. Versione 2.18.2.

— (2014b). *Usage*. URL http://www.lilypond.org/doc/v2.18/Documentation/usage.pdf. Versione 2.18.2.

▷ Tommaso Gordini
Padova
illinguista1972 at gmail dot com

▷ Davide Liessi
Conegliano (TV)
davide dot liessi at gmail dot com

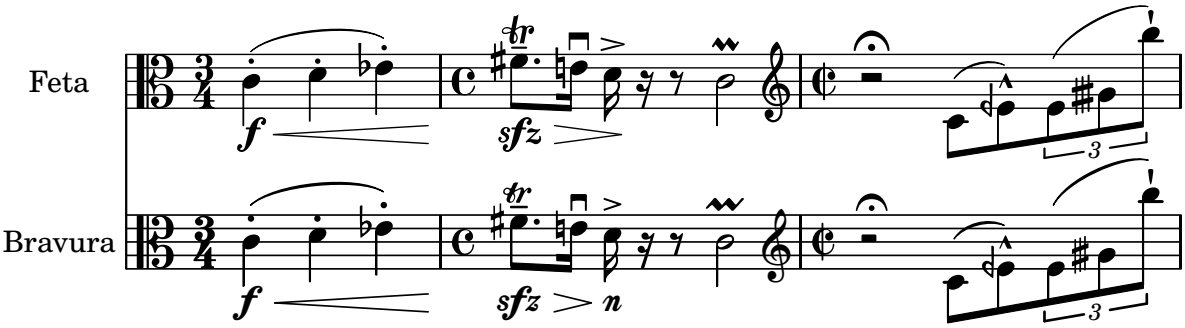


Figura 20: Confronto tra il font Emmentaler/Feta (in alto) e il Bravura (in basso).

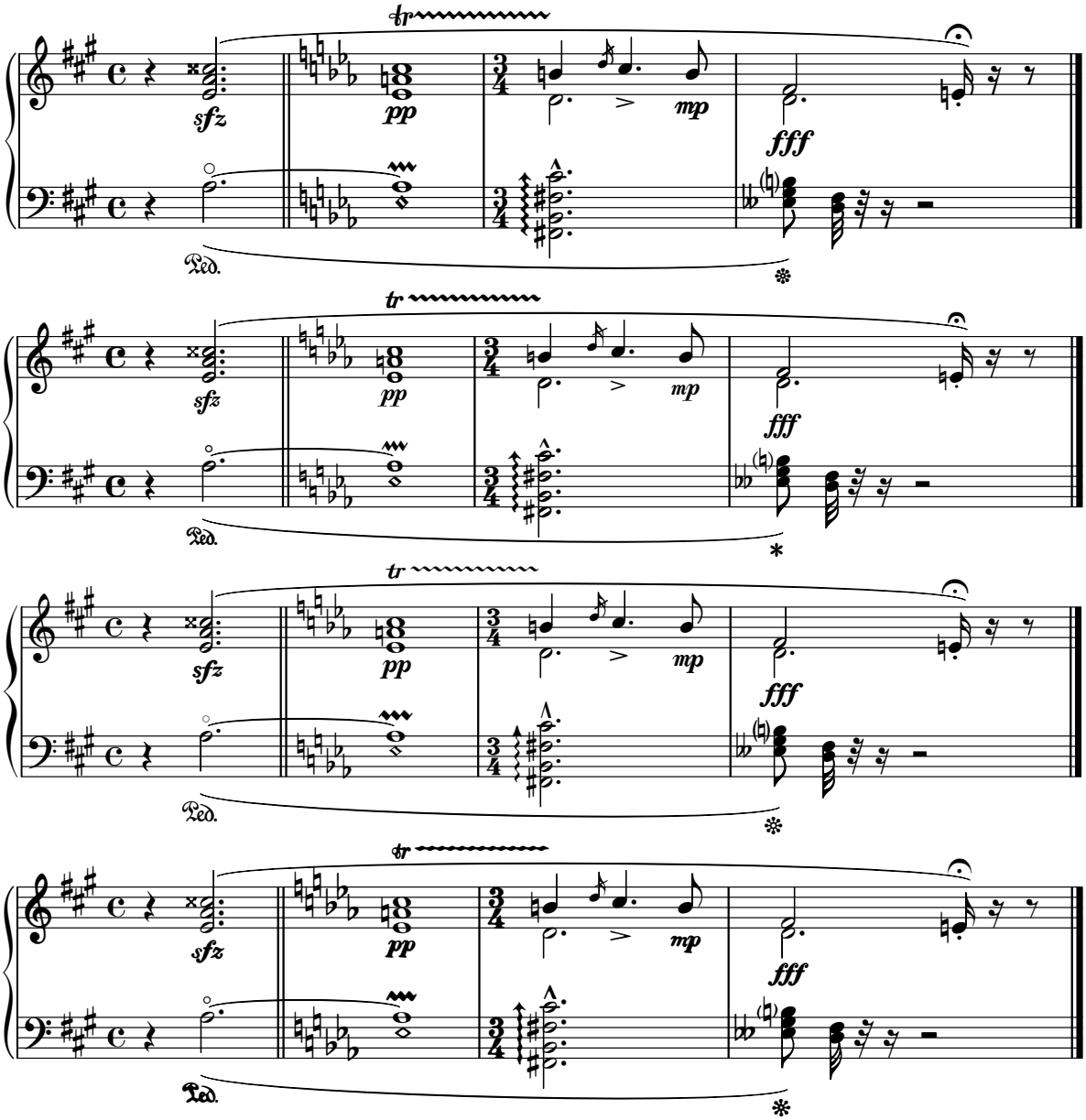


Figura 21: Confronto tra alcuni font utilizzabili in LilyPond grazie al lavoro di Abraham Lee. Dall'alto al basso: Emmentaler/Feta, Scorlatti (copia del font utilizzato da SCORE), Gonville, Ross (copia del font scelto da Ted Ross nel libro *The Art of Music Engraving and Processing*).

XML-TEI e Tagged PDF

Luigi Scarso

Sommario

In questo articolo mostriamo come gestire in ConTEXt un documento in formato XML per produrre una rappresentazione in formato PDF e come l'utilizzo dei tag permetta di strutturare il documento PDF in modo analogo al documento XML. Viene presentato un originale foglio di stile per il completo embedding del documento XML sorgente e come analizzare il PDF tramite luaTEX per estrarre il documento.

Abstract

In this paper we show how to manage an XML document in ConTEXt to produce a PDF output and how the tags of the PDF can be used to give a structure to the PDF document. We also show a novel stylesheet for the complete embedding of the XML source into the PDF and how to analyze the final PDF with luaTEX to extract the document.

1 Introduzione

Con il termine *workflow* si intende una descrizione di un processo che, possibilmente per passi successivi, trasforma uno o più *input* in uno o più *output*. La definizione è evidentemente adattabile a numerosi contesti, e nell'ambito della tipografia digitale è solitamente usato per descrivere come ottenere un documento finale (che nella maggior parte delle volte è in formato PDF) a partire da uno o più documenti sorgenti, attraverso varie fasi che trasformano i sorgenti in forme intermedie via via simili a quella finale. Ad esempio, nel caso di una rivista, i documenti sorgenti possono essere gli articoli, le foto e le inserzioni pubblicitarie ed il workflow descriverebbe allora come trasformare i testi per il software di impaginazione, la corretta dimensione delle immagini e delle inserzioni, e come marcare i componenti intermedi così ottenuti per il corretto ordine nella rivista finale. In ambito umanista il caso tipico è quello che presenta un solo documento sorgente (normalmente una edizione cartacea) e un solo output, anche in questo caso PDF in quanto, oltre ad offrire elevate qualità tipografiche, supporta l'archiviazione a lungo termine ed è usufruibile su una gran numero di dispositivi. Il problema in questo caso è come passare dal documento cartaceo al PDF: la scansione delle pagine è solamente una parte della soluzione perché, benché sia il metodo più economico per preservare le informazioni (anche tipografiche), ha l'evidente svantaggio di non rendere fruibile il documento sorgente per attività di elaborazione del testo. Una alternativa è tradurre il testo originale in un formato elettronico che sia facilmente adattabile ad un generico workflow, e questo è precisamente il compito della *Text Encoding Initiative* (TEI), un consorzio che comprende università ed enti di ricerca ed il cui compito è mantenere una *applicazione* XML per la codifica di testi (specialmente dell'area umanista, scienze sociali e linguistica) in forma elettronica. In breve, in Anonymous (2014) TEI specifica e descrive un formato XML per la *struttura* del testo e alcune metainformazioni (ad esempio le revisioni, i dati relativi all'edizione originale ecc.) ma *non* le proprietà tipografiche — in sostanza è il duale della scansione. Un workflow che ha come ingresso un formato XML quasi sempre prevede dei passi intermedi che utilizzano XSLT o XQUERY (altre applicazioni XML, cfr. (Clark, 1999), (Kay, 2007) e Florescu *et al.* (2010)) per trasformare il documento in una forma adatta; in questo caso è lo stesso TEI che mette a disposizione presso `http://www.tei-c.org/Tools/Stylesheets/index.xml` i file XSLT 2.0 (noti come "fogli di stile" o *stylesheets*) per trasformare un documento XML-TEI in una moltitudine di output: csv, docbook, docx, dtd, epub3, epub, fo, html5, html, ibooks, json, latex, lite, oddhtml, odt, p4, rdf, relaxng, tcp, txt, e xlsx, seguendo quanto scritto in `http://www.tei-c.org/release/doc/tei-xsl`. Si noti che non è direttamente previsto il formato PDF. In pratica esiste un solo programma (in questo ambito chiamato *processore*) open source in grado di utilizzare i fogli di stile XSLT 2.0: Saxon (disponibile presso `http://saxon.sourceforge.net`), che è un programma Java e richiede il runtime adeguato.

Considerando l'ambito TEX, praticamente esistono due tipi di workflow:

- il primo, delineato sopra, utilizza un processore XSLT per tradurre un documento XML-TEI in formato pdfLATEX o XELATEX e quindi usa il motore di impaginazione corrispondente per il PDF. In ambito LATEX probabilmente è l'unico workflow usato;
- il secondo utilizza un processore scritto in TEX (non necessariamente completamente XSLT compatibile) per elaborare direttamente il file XML.

È da notare che il set di elementi/attributi descritti nelle *guidelines* è veramente molto ricco: è naturale che un autore utilizzi un sottoinsieme

piuttosto ristretto (ad esempio, solo gli elementi relativi ad una edizione critica) e che quindi il relativo foglio di stile implementi un workflow XML che traduca solo questi elementi. Si tratta in sostanza di distinguere tra workflow che accetta un *qualsiasi* XML-TEI contro un workflow specifico per un preciso testo, il quale normalmente risulta più gestibile rispetto ad uno generico.

Ecco un frammento dell'opera di William Shakespeare *Romeo and Juliet* (di cui più avanti daremo i dettagli) in formato XML-TEI; sono le prime battute del primo atto:

```
<div1 type="act" n="1" org="uniform"
                    sample="complete">
  <head>ACT I</head><lb ed="F1" n="2" />
<div2 type="scene" n="1" org="uniform"
                     sample="complete">
<head>SCENE I</head>
<stage type="setting">
Verona. A public place.
</stage>
<lb ed="F1" n="3" />
<stage type="entrance">
Enter SAMPSON and GREGORY,
<lb ed="F1" n="4" />
of the house of Capulet,
armed with swords and bucklers.</stage>
<lb ed="G" /><lb ed="F1" n="5" />
<sp who="sam.">
<speaker>Sam.</speaker><p>
Gregory, o' my word, we'll not
<lb ed="G" />carry coals.
<lb ed="G" />
<lb ed="F1" n="6" /></p></sp>
<sp who="gre.">
<speaker>Gre.</speaker>
<p>No, for then we should be colliers.
<lb ed="G" />
<lb ed="F1" n="7" /></p></sp>
<sp who="sam.">
<speaker>Sam.</speaker>
<p>I mean, an we be in choler, we'll
<lb ed="G" />draw.
<lb ed="G" />
<lb ed="F1" n="8" /></p></sp>
<sp who="gre.">
<speaker>Gre.</speaker><p>
Ay, while you live, draw your neck
<lb ed="G" />
<lb ed="F1" n="9" />out o' the collar.
<lb ed="G" />
<lb ed="F1" n="10" />
</p></sp><sp who="sam.">
<speaker>Sam.</speaker>
<p>I strike quickly, being moved.
<lb ed="G" />
<lb ed="F1" n="11" />
</p></sp><sp who="gre.">
<speaker>Gre.</speaker>
<p>But thou art not quickly moved to
<lb ed="G" />strike.
<lb ed="G" n="10" />
<lb ed="F1" n="12" /></p></sp>
<sp who="sam.">
<speaker>Sam.</speaker>
<p>A dog of the house of Montague moves me.
<lb ed="G" /><lb ed="F1" n="13" /></p></sp>
<sp who="gre.">
<speaker>Gre.</speaker>
<p>To move is to stir; and to be valiant
<lb ed="G" />is to stand:
<lb ed="F1" n="14" />
therefore, if thou art moved, thou
<lb ed="G" />runn'st away.
<lb ed="G" /><lb ed="F1" n="15" />
</p></sp><sp who="sam.">
<speaker>Sam.</speaker>
<p>A dog of that house shall move me
<lb ed="G" />to stand: <lb ed="F1" n="16" />
I will take the wall of any man or
<lb ed="G" />maid of Montague's.
```

Nella figura 1 c'è una possibile presentazione.

Infine, nella quasi totalità dei casi non interessa se il workflow è *reversibile*, ovvero se dall'uscita sia possibile riscostruire l'ingresso ed in quale misura. Infatti nei PDF prodotti con pdfLaTeX o XeLaTeX la struttura logica appare evidente all'utente ma non è codificata *dentro* il PDF in modo da poter essere utilizzabile da altri programmi (i *bookmark* contengono soltanto le informazioni di struttura, non tabelle, liste ecc).

Nelle prossime sezioni vedremo come affrontare un workflow in XML con ConTeXt-MkIV, e se le osservazioni precedenti hanno risposta soddisfacente.

2 XML in ConTeXt

ConTeXt-MkIV ((Hagen, a),(Hagen, b)) utilizza esclusivamente luaTeX come motore di impaginazione (*engine*)[1] ed implementa un linguaggio per trasformare documenti XML o più precisamente per selezionare gli elementi di un documento per mapparli poi in macro TeX (Hagen, c). Questo linguaggio *non* è una implementazione di XSLT ed utilizza la libreria interna LPEG (Ierusalimschy, 2009) ed una specifica *Parsing Expression Grammar* (o PEG, cfr. (Grune e Jacobs, 2007)) per il parsing di un generico documento XML. Il parser risultante è piuttosto performante, cosa non insolita se si pensa che deve convertire gli elementi *solo* verso ConTeXt.

Come accennato in precedenza il documento TEI scelto è l'opera di William Shakespeare *Romeo and Juliet*, W. G. Clark, W. Aldis Wright, Ed., The Globe Shakespeare, New York, Nelson Doubleday, Inc. disponibile per il download presso http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.03.0053.

1. Il termine è un po' impreciso in quanto, a volte, in un workflow basato su TeX, con motore di impaginazione si intende ConTeXt o LaTeX, cioè il macro-formato più che l'engine stesso.

ACT I

SCENE I

Verona. A public place.

Enter SAMPSON and GREGORY,
of the house of Capulet, armed with swords and bucklers.

Sam. Gregory, o' my word, we'll not
carry coals.

Gre. No, for then we should be colliers.

Sam. I mean, an we be in choler, we'll
draw.

Gre. Ay, while you live, draw your neck
out o' the collar.

Sam. I strike quickly, being moved.

Gre. But thou art not quickly moved to
strike.

Sam. A dog of the house of Montague moves me.

Gre. To move is to stir; and to be valiant
is to stand:
therefore, if thou art moved, thou
runn'st away.

Sam. A dog of that house shall move me
to stand:
I will take the wall of any man or
maid of Montague's.

5

Figura 1: Le battute iniziali del primo atto dell'opera di W. Shakespeare *Romeo and Juliet*.

Come primo passo è stato deciso di modificare il file per renderlo compatibile con l'ultima versione P5 delle specifiche TEI (Anonymous, 2014); la modifica più importante probabilmente è stata il corretto nome dell'attributo `id` (deve essere `xml:id`) ma a parte questo pochi altri elementi sono stati modificati. Questo implica una sostanziale robustezza del formato TEI: documenti relativamente "vecchi" possono essere facilmente aggiornati.

Processare un file XML con ConTEXt non differisce molto da usare un processore XSLT: per prima cosa creiamo il file

`Perseus_text_1999.03.0053-2.6.0.tex`

il quale carica il file di stile

`s-tei-Perseus-text-1999-03-0053-style`

e poi elabora il file XML

`Perseus_text_1999.03.0053-2.6.0.xml`

ovvero:

```
\loadmarkfile{%
s-tei-Perseus-text-1999-03-0053-style}
\starttext
\xmlprocessfile{RandJ}
   {Perseus_text_1999.03.0053-2.6.0.xml}{}
\stoptext
```

Lo stylesheet definisce le caratteristiche tipografiche come il font (TeX Gyre Schola), il layout di pagina ed il setup dei due soli livelli di sezionamento usati (in corrispondenza degli analoghi livelli nel documento XML). I file rilevanti per il mapping degli elementi XML sono però

`s-tei-Perseus-text-1999-03-0053-basics.lua`

e

`s-tei-Perseus-text-1999-03-0053-basics.mkiv`

```
%D \module
%D [file=s-tei-Perseus-text-1999-03-0053-style,
%D  version=2014.07.01,
%D  title=Romeo and Juliet,
%D  subtitle=,
%D  author=Luigi Scarso,
%D  date=\currentdate,
%D  copyright={Luigi Scarso}]

\registerctxluafile%
{s-tei-Perseus-text-1999-03-0053-basics}
{1.001}

\loadmarkfile%
{s-tei-Perseus-text-1999-03-0053-basics}

\registerctxluafile%
{s-tei-Perseus-text-1999-03-0053-style}
{1.001}

\setuppapersize[A4,landscape][A4,landscape]
\starttypescript[randj]
 \definetypeface [randj]%
           [rm] [serif] [schola] [default]
 \definetypeface [randj]%
           [ss] [sans]  [schola] [default]
 \definetypeface [randj]%
           [tt] [mono]  [schola] [default]
 \definetypeface [randj]%
           [mm] [math]  [modern] [default]
\stoptypescript
\setupbodyfont[randj,10pt]

\setuplayout
  [width=middle,
   height=middle,
   backspace=3cm,
   cutspace=1cm,
   rightmargin=0cm,
   leftmargin=2cm,
   margindistance=0cm,
   footer=1cm,
   header=\zeropoint]
\setuppagenumbering
 [alternative=singlesided,
  location={footer,middle}]
\setuphead[chapter]
          [style=\tfc,
           align=middle,
           number=no,
           expansion=yes,
           before=,
           after=\blank,
           page=yes]
\setuphead[section]
          [style=normal,
           align=middle,
           number=no,
           expansion=yes,
           before={\blank[2*line]},
           after={\blank[line]},
           page=no]
\setupinteraction[state=start,
          color=black,
          contrastcolor=black,
          style=normal,
          title={Romeo and Juliet},
          author={Luigi Scarso}]

\placebookmarks[chapter][all][force=yes]

\mainlanguage[en]
```

Il file MkIV `*basics.mkiv` mostra il meccanismo utilizzato da ConTEXt per l'XML: si definisce un ambiente `setups` che descrive come mappare gli elementi XML in ambienti ConTEXt:

```
\startxmlsetups xml:tei:Perseus:text:1999:03:
0053-setups
\xmlsetsetup{#1}{*}{xml:tei:*}
\stopxmlsetups
```

e lo si registra per attivarlo:

```
\xmlregistersetup{xml:tei:Perseus:text:
1999:03:0053-setups}
```

Il significato della macro `\xmlsetsetup{#1}{*}{xml:tei:*}` è il seguente: l'elemento *e* (`*` significa "tutti gli elementi") del file `#1` associato a questo setup è processato dal `setups` `xml:tei:`*e* — che deve essere definito dall'utente. Ad esempio l'elemento `TEI` è processato da

```
\startxmlsetups xml:tei:TEI
   \xmlfunction{#1}{tei:TEI}
\stopxmlsetups
```

il suo figlio `teiHeader` è:

```
\startxmlsetups xml:tei:teiHeader
   \xmlfunction{#1}{tei:teiHeader}
\stopxmlsetups
```

La macro `\xmlfunction{#1}{#2}` equivale alla funzione Lua `xml.functions["#2"](#1)`, dove `#1` è l'elemento del documento; ad esempio `\xmlfunction{#1}{tei:TEI}` vista prima chiama la funzione `xml.functions["tei:TEI"](#1)` con `#1` l'elemento `TEI` del documento, cioè la radice.

Ogni elemento del documento è mappato in questo modo: in pratica ogni elemento del documento chiama una funzione Lua definita nel file Lua `*basics.lua`. Ecco la sezione del file Lua relativa agli elementi `TEI` e `teiHeader`:

```
local xml_func = xml.functions or {}

local function TEI(t)
   local att = {['elementname']='TEI'}
   context.startelement({"document"},att)
   context.setupelementuserproperties(
                        {'document'},att)
   lxml.flush(t)
   context.stopelement()
end
local function teiHeader(t)
   local att = {['elementname']='teiHeader'}
   context.startelement({"metadata"},att)
   context.setupelementuserproperties(
                       {'metadata'}, att)
   context([[{\sc]])
   lxml.flush(t)
   context([[}]])
   context.blank()
   context.stopelement()
end

xml_func["tei:TEI"]  = TEI
xml_func["tei:teiHeader"] = teiHeader
```

Tralasciando per il momento `startelement`, `setupelementuserproperties` e `stopelement` che vedremo nella prossima sezione, la funzione `TEI(t)` con `lxml.flush(t)` semplicemente abilita l'elaborazione degli elementi figli; `teiHeader(t)` è più o meno simile, ma racchiude i figli dentro un gruppo `{\sc ..}` per attivare il maiuscoletto del font. Come si può vedere, quando si è sulla "parte" Lua è possibile comunicare con la "parte" TEX tramite la funzione `context(<tex-string>)`: in ConTEXt è una funzione parecchio usata, ed è specializzata per gestire le macro al punto che è possibile scrivere interi documenti solamente in Lua (ad esempio la funzione Lua `context.blank()` equivale alla macro `\blank`, e così per ogni macro di ConTEXt: naturalmente in alcuni casi TEX rimane più semplice e chiaro rispetto alla controparte Lua).

Tralasciamo gli altri elementi, in quanto riteniamo di aver mostrato le parti principali del processore XML. In Hagen (c) sono descritte le varie tecniche per selezionare/filtrare nodi usando determinate condizioni, ma sostanzialmente la complessità non è differente dal linguaggio XSLT. Anche la relazione tra elemento XML ed elemento tipografico richiede la conoscenza della sintassi di ConTEXt, ma anche in questo caso la complessità è simile a quella di LATEX. Nella prossima sezione vedremo invece una caratteristica peculiare di ConTEXt, i PDF con tags.

3 Tagged PDF

La guida di riferimento del PDF 1.7 (disponibile presso Adobe) dedica il capitolo 10 alla descrizione di come sia possibile inserire informazioni sulla struttura logica del documento sorgente all'interno del file PDF. Il livello più semplice è il *contenuto evidenziato* (Marked Content) dove una serie di operatori PDF associa un tag ad uno stream di contenuto (o parte di esso): ad esempio la coppia `BDC EMC` usa la sintassi *tag properties* `BDC ... EMC` ed è usata in ConTEXt per marcare una parte del testo, ad esempio il titolo di una sezione:

```
/sectiontitle <</MCID 0>>BDC
BT
/F1 17.21541 Tf 1 0 0 1 353.324 528.9996 Tm
[<002F0060001C004B001C>30<006900420062006800
5400320060006200510040>25<00A4>]TJ
ET
EMC
```

Ad un livello superiore esiste la struttura logica, un set gerarchico di elementi struttura ciascuno rappresentato da un dizionario. Gli elementi struttura e i contenuti visibili e associati sono memorizzati in parti differenti del PDF ma ciascuno ha un puntatore verso l'altro, in modo ad esempio da poter associare un titolo ad una pagina. In pratica, la struttura logica permette di definire una struttura ad albero, dove i figli sono terminali (per esempio Marked Content) o nodi non terminali — cioè altri elementi struttura. A questo livello *non* vengono stabiliti nomi degli elementi struttura, per cui l'utente può utilizzare i nomi più convenienti. Adobe ha introdotto un ulteriore livello, il Tagged PDF, che si basa sui due precedenti ma dove fissa il nome degli elementi struttura e la loro semantica (i *tags*). Purtroppo i tags definiscono un documento piuttosto povero dal punto di vista semantico, ma nella struttura logica è previsto un dizionario (*RoleMap*) che mappa il nome degli elementi struttura definito da un utente con i tags del Tagged PDF: questo implica che un utente che fornisca una RoleMap automaticamente definisce un Tagged PDF.

Per utilizzare Tagged PDF in ConTEXt bisogna abilitarli con

```
\setupstructure[state=start,method=auto]
\setuptagging[state=start]
```

in quanto per default sono disabilitati. Una volta abilitati, gli elementi struttura vengono automaticamente aggiunti con i comandi di sezionamento, liste, tabelle e figure.

ConTEXt definisce un proprio insieme di elementi struttura, e fornisce la relativa RoleMap, quindi può produrre un Tagged PDF — in effetti, può produrre PDF/A-1a cioè Tagged PDf validi per l'archiviazione digitale. Anche se questo insieme è migliore di quello definito da Adobe ovviamente non è l'insieme di tags usati per esempio da TEI: ovviamente è possibile modificare il codice ConTEXt per usare i tag di TEI con una opportuna RoleMap, ma qui proponiamo una soluzione differente.

Al livello di struttura logica per un elemento struttura è definito il dizionario (opzionale) UserProperties il cui contenuto è gestito dall'applicazione utente. Esso contiene un array di coppie chiave/valore che è possibile utilizzare in questo modo:

1. si fissa la chiave `elementname` con il nome dell'elemento XML.
2. se un elemento XML ha un attributo `a` con valore `v` si fissa la chiave `a` con valore `v` — in pratica si copia l'attributo.

Ad esempio l'elemento TEI `sp` che marca il discorso di un attore e che ha come attributo `who` viene mappato nell'elemento `p` di ConTEXt in questo modo:

```
local function sp(t)
   local at = t.at
   local att = {['elementname']='sp'}
   att.who = at.who or nil
   context.startelement({"p"},att)
   context.setupelementuserproperties({'p'},
                                      att )
   lxml.flush(t)
   context.stopelement()
end
```

Ecco ad esempio come viene salvato nel PDF l'elemento `<sp who='chorus'>...</sp>`:

```
209 0 obj
<< /A << /O /UserProperties /P [
<< /V (sp) /N (elementname) >>
<< /V (chorus.) /N (who) >> ] >>
/Type /StructElem /S /p
/Pg 201 0 R /K 208 0 R /P 207 0 R >>
```

È evidente che il nome `elementname` è arbitrario ed in generale va verificato che non sia in conflitto con altri attributi — in questo specifico caso non esistono conflitti.

Il passo finale è il seguente: è possibile verificare la struttura di un Tagged PDF ? La risposta è positiva: con l'attuale versione di LUATEX il modulo Lua `epdf` fornisce un insieme di funzioni di basso livello per leggere un file PDF e quindi per "navigare" attraverso i suoi oggetti. Naturalmente richiede una conoscenza piuttosto approfondita del formato PDF, ma le specifiche ADOBE sono chiare e liberamente disponibili. Inoltre utilizzando funzioni di basso livello il relativo programma sarà meno soggetto alle conseguenze degli aggiornamenti delle librerie. L'autore, tuttavia, essendo parte del team di sviluppo, ha notato che con l'ultimo aggiornamento della libreria `poppler` che è la base di `epdf` sono stati aggiunti nuovi metodi per gli elementi struttura e quindi ha aggiunto tali metodi alla libreria `epdf` (la modifica riguarda per il momento solo la versione sperimentale, ma è probabile che a breve verrà estesa a quella canonica).

Costruire un programma che legga la struttura di un PDF è leggermente più semplice se si dispongono dei nuovi metodi per gli elementi struttura; in questo caso il programma salva le informazioni relative alle UserProperty di un elemento struttura e le presenta in formato XML. Nel caso del documento TEI di *Romeo and Juliet* i primi tag dentro il PDF sono:

```
<TEI>
  <teiHeader type="text" >
   <fileDesc>
    <titleStmt>
     <title>   Romeo and Juliet
     </title>
     <author>  William Shakespeare
     </author>
     <editor role="editor" > W. G. Clark
     </editor>
     <editor role="editor" > W. Aldis Wright
     </editor>
     <sponsor>  Perseus Project,
                Tufts University
     </sponsor>
     <principal>  Gregory Crane
     </principal>
     <respStmt>
      <resp>  Prepared under the
              supervision of
      </resp>
      <name>   Lisa Cerrato
      </name>
      <name>   William Merrill
      </name>
      <name>   Elli Mylonas
      </name>
      <name>   David Smith
      </name>
     </respStmt>
     <funder>  NSF, NEH: Digital
    Libraries Initiative, Phase 2
     </funder>
     <respStmt xml:id="CEW.ed" >
      <name>  CEW
      </name>
      <resp>  ed.
      </resp>
     </respStmt>
```

```
    </titleStmt>
    <publicationStmt>
```

da confrontare con quello originale:

```
<TEI>
 <teiHeader type="text" > <!-- status="new" -->
  <fileDesc>
   <titleStmt>
    <title>Romeo and Juliet</title>
    <author>William Shakespeare</author>
    <editor role="editor">W. G. Clark</editor>
    <editor role="editor">W. Aldis Wright
    </editor>
    <sponsor>Perseus Project, Tufts University
    </sponsor>
     <principal>Gregory Crane</principal>
     <respStmt>
      <resp>Prepared under the supervision of
      </resp>
      <name>Lisa Cerrato</name>
      <name>William Merrill</name>
      <name>Elli Mylonas</name>
      <name>David Smith</name>
     </respStmt>
  <funder n="org:DLI2">NSF, NEH: Digital
    Libraries Initiative, Phase 2</funder>
  <!-- Revision -->
  <respStmt xml:id="CEW.ed"><name>CEW</name>
  <resp>ed.</resp></respStmt>
  </titleStmt>

      <publicationStmt>
```

Il codice è disponibile presso: https://github.com/mlgdominici/TEI-TeX/tree/master/ctx/Perseus_text_1999.03.0053-2.6.0

4 Conclusioni

L'elaborazione di file XML è piuttosto diretta in ConTEXt e non si appoggia a processori esterni. Questo significa ridurre la dipendenza da componenti esterne, utilizzando però un linguaggio di trasformazione che non è XSLT, sicuramente più diffuso e per il quale sono disponibili processori solidi relativamente a buon mercato. La scelta quindi dipende da altri vincoli imposti dal workflow utilizzato. Invece sono interessanti le prospettive che si aprono con i Tagged PDF, in particolare la possibilità di interrogare i PDF come se fossero documenti XML. Va detto che normalmente un PDF presenta diverse parti compresse per ottimizzare lo spazio, quindi l'interrogazione nel caso di un documento lungo diventa inefficiente se prima non si salva l'XML "embedded". Nel caso di PDF tipo "forms" che presentino i dati in forma tabellare in genere su 2 o 3 pagine, rinunciare alla compresssione può essere fattibile e questo approccio ai Tagged PDF risulta più promettente: ad esempio nel caso di estrazioni di report da un database, un Tagged PDF con queste caratteristiche si presta bene ad essere utilizzato da programmi che analizzano report — magari per pubblicare dati sul Web con una presentazione completamente differente. Un altro ambito interessante è la scheda tecnica di un prodotto industriale: le caratteristiche sono per lo più in forma tabellare e non è particolarmente difficile tradurle in XML — probabilmente in questi casi il problema pricipale è scegliere un XML adeguato.

Riferimenti bibliografici

Adobe. «Document Management – Portable Document Format – Part 1: PDF 1.7, First Edition». http://www.adobe.com/devnet/pdf/pdf_reference.html.

Anonymous (2014). *TEI P5: Guidelines for Electronic Text Encoding and Interchange. Version 2.6.0. Last updated on 20th January 2014, revision 12802.* TEI Consortium, eds. URL http://www.tei-c.org/Guidelines/P5/.

Clark, J. (1999). «XSL transformations (XSLT) version 1.0». W3C recommendation, W3C. http://www.w3.org/TR/1999/REC-xslt-19991116.

Florescu, D., Simeon, J., Boag, S., Chamberlin, D., Fernandez, M. e Robie, J. (2010). «XQuery 1.0: An XML query language (second edition)». W3C recommendation, W3C. http://www.w3.org/TR/2010/REC-xquery-20101214/.

Grune, D. e Jacobs, C. (2007). *Parsing Techniques: A Practical Guide.* Monographs in Computer Science. Springer. URL http://books.google.it/books?id=05xA_d5dSwAC.

Hagen, H. (a). «? context». http://www.pragma-ade.nl/general/manuals/what-is-context.pdf.

— (b). «CONTEXT MKII CONTEXT MKIV». http://www.pragma-ade.nl/general/manuals/mk.pdf.

— (c). «Dealing with XML in ConTEXt-MkIV». http://www.pragma-ade.nl/general/manuals/xml-mkiv.pdf.

Ierusalimschy, R. (2009). «A text pattern-matching tool based on parsing expression grammars». *Softw. Pract. Exper.*, **39** (3), pp. 221–258. URL http://dx.doi.org/10.1002/spe.v39:3.

Kay, M. (2007). «XSL transformations (XSLT) version 2.0». W3C recommendation, W3C. Http://www.w3.org/TR/2007/REC-xslt20-20070123/.

▷ Luigi Scarso
luigi dot scarso at gmail dot com

Lo stile tipografico: teoria e pratica

Claudio Vincoletto

Sommario

Vengono presi in esame alcuni processi per elaborare la sintesi di uno stile tipografico, attingendo da abitudini del passato ed esplorando limiti e possibilità di tecniche consolidate. Si propone lo studio di antichi modelli, non solo per perfezionare il gusto nei confronti di una buona tipografia, ma come stimolo di riflessione e sviluppo di nuove e originali soluzioni.

Abstract

Some processes are considered to develop the synthesis of a typographic style, drawing from past habits and exploring the limits and possibilities of established techniques. It is proposed the study of ancient models to improve the taste of a good typography, as a stimulus for reflection and development of new and original solutions.

1 Prologo

> Chi resta a lungo sporto sopra un precipizio sente il desiderio di tuffarvisi, per mettere fine alla tentazione delle vertigini. (C.R. Maturin, *Melmoth the Wanderer*, 1820)

Fine ultimo di chi usa il sistema TEX è quello di ottenere una buona impostazione tipografica. Ma quali sono i principi che permettono di valutare una composizione come buona o per lo meno efficace, se non funzionale? Il grado di libertà nel comporre tipograficamente un testo digitale è elevatissimo. Se applicata in modo indiscriminato, tale libertà può arrivare a confondere il processo di lettura fino a renderlo incomprensibile.

Di solito i vari motori di composizione del sistema TEX (pdfTEX, XƎTEX, LuaTEX), non vengono usati direttamente da chi compone testi, bensì mediante un sistema di macro che è anche un linguaggio di markup, ossia LATEX nelle sue varie incarnazioni (pdfLATEX, XƎLATEX, LuaLATEX) oppure ConTEXt. Ciò implica che l'utente venga guidato negli aspetti formali dello scritto, senza che siano esplicitati i criteri che rendono identificabile una buona tipografia. I risultati, grazie a modelli preconfezionati, sono spesso eccellenti; e questo, per l'utente, nella maggior parte dei casi risulta essere più che sufficiente.

Non appena il neofita comincia ad addentrarsi nei meccanismi del sistema TEX, apprende che ogni aspetto della tipografia digitale può essere controllato. Un tale approccio può condurre al disastro, se non guidato da precise competenze, esattamente come accade a chi utilizza in modo disinvolto i *word processor*. Si cercherà quindi di ricostruire un percorso, attraverso il quale vengano evidenziati alcuni criteri base dell'impaginazione editoriale, prendendo spunto da alcuni lavori del passato. Per facilitare tale compito è possibile delimitare da subito il campo dell'indagine, affermando che per una composizione tradizionale i principi da tenere presente sono essenzialmente due: il formato della pagina è determinato dalle dimensioni del foglio di carta, quello della gabbia del testo dalla grandezza dei caratteri in metallo.

2 Strumenti

> Non v'è passione al mondo, più diabolicamente impaziente, di quella di colui che, rabbrividendo sull'orlo d'un precipizio, medita di gettarvisi. (E.A. Poe, *The imp of the Perverse*, 1845)

Per capire come procedere in modo spedito e con sufficiente padronanza, è necessario esaminare con attenzione alcuni aspetti chiave che motivavano le scelte del tipografo del passato.

Gli antichi mastri cartai non seguivano delle misure standard per la fabbricazione della carta. Ogni bottega produceva su commissione, anche in diversi formati, ma questi non si discostavano molto dalle dimensioni di un foglio di pergamena, che fungeva da modello. D'altronde, ogni foglio non poteva essere più esteso della pelle di animale (pecora, capra, vitello) da cui era ricavato.

Tuttavia, la carta veniva ripiegata più volte, così da ottenere le proporzioni desiderate. Tale considerazione non ci consente di delimitare il campo delle lunghezze che determinano il rettangolo della pagina, perché in seguito alla legatura, questa veniva ulteriormente rimpicciolita con la rifilatura.

Così, come per i bibliografi, ci atterremo alla misurazione in millimetri del profilo della pagina, con la precedenza del lato lungo: *altezza* × *larghezza*.

Diverse sono le ragioni che regolano l'area di stampa. I caratteri tipografici erano fusi in blocchi di metallo (quelli di maggiori dimensioni, intagliati nel legno), da una matrice che ne definiva con regolarità la componente verticale. Il corpo del carattere è la misura fondamentale che ci consente di associare in famiglie i diversi glifi, nonché stabilire relazioni con famiglie differenti. È ancora il corpo

Figura 1: Dimensioni del carattere.

del carattere, distribuito in righe che si succedono longitudinalmente, a definire l'altezza della gabbia del testo.

Per rilevare la dimensione del corpo è sufficiente disporre di una buona lente d'ingrandimento e di una riga graduata, in scala metrica e con risoluzione di almeno 0,5 mm. Anche disponendo di un tipometro, ci si renderà conto di quanto la misurazione più accurata sia sempre un'approssimazione, soprattutto per i caratteri più piccoli. Occorre quindi misurare il punto più alto e quello più basso dei glifi, che generalmente coincidono con lo spazio che intercorre fra la linea dei glifi ascendenti, *occhio superiore*, e quella dei glifi discendenti, *occhio inferiore* (figura 1, fra le righe rosse).

Nei tipi in metallo erano necessari ulteriori spazi, per ragioni di manifattura, sopra e sotto tali estremi, e tali contrografismi venivano indicati come *spalla superiore* e *spalla inferiore* (intervallo fra le righe nere). Tale proprietà è conservata in alcuni caratteri digitali, come spazio per segni paragrafematici, diacritici o punteggiatura, ma può essere ignorata nella misurazione del corpo, considerato il modesto valore che viene aggiunto (Carter, 1969, p. 23).

Sempre nella figura 1, si possono notare altri due parametri verticali, utili al compositore, l'altezza della *x*, *occhio medio* (riga verde), e la *linea di base* (riga blu). La distanza fra righe di base successive viene definita *avanzamento di riga*, che corrisponde alla dimensione del corpo sommata a uno spazio interlineare. Per approfondire questo concetto, si rimanda all'ottima argomentazione fornita dalla guida ufficiale del GUIT (Beccari *et al.*, 2014, § 18).

Le proporzioni dell'area di stampa corrisponderanno perciò, per il segmento verticale, all'intervallo fra gli ascendenti della prima riga e i discendenti dell'ultima, e, per il segmento orizzontale, alla larghezza di ogni riga. La prima misura sarà determinata dal valore del corpo moltiplicato per il numero delle righe (a cui andranno eventualmente aggiunte le relative interlinee); anche la larghezza della riga risponderà per convenzione a un multiplo del valore del corpo o di una frazione di esso.

Per definire i margini, cioè il bianco della pagina, non costituiranno un problema il *bianco di testa*, superiore, quello *di piede*, inferiore, e quello *di taglio*, esterno; il *bianco di cucitura*, invece, quello interno e di difficile rilevazione, sarà deducibile dalle altre misure.

Infine, qualora non fosse possibile la misurazione diretta del volume che intendiamo riprodurre, è possibile intraprendere il lavoro a partire da un'immagine digitale, a patto che vi sia un riferimento su cui fare appiglio, come nella figura 5.

Lo strumento che ci sembra più appropriato per l'analisi dell'immagine è *Graphoskop* (Gurrado e Lestingi, 2009), un plugin di estensione del programma open source *ImageJ*, destinato a studi paleografici, che permette di rilevare dati quantitativi di tipo metrologico. Si rimanda alla relativa documentazione per le modalità d'uso. È raccomandabile però, prima di avviare il lavoro, riscalare i pixel dell'immagine in rapporto alle unità di misura analogiche. Tracciare quindi sull'immagine un segmento di una lunghezza conosciuta, selezionando il pulsante , e aprire la finestra `Analyze/Set Scale` per impostarne i valori adeguati (figura 2).

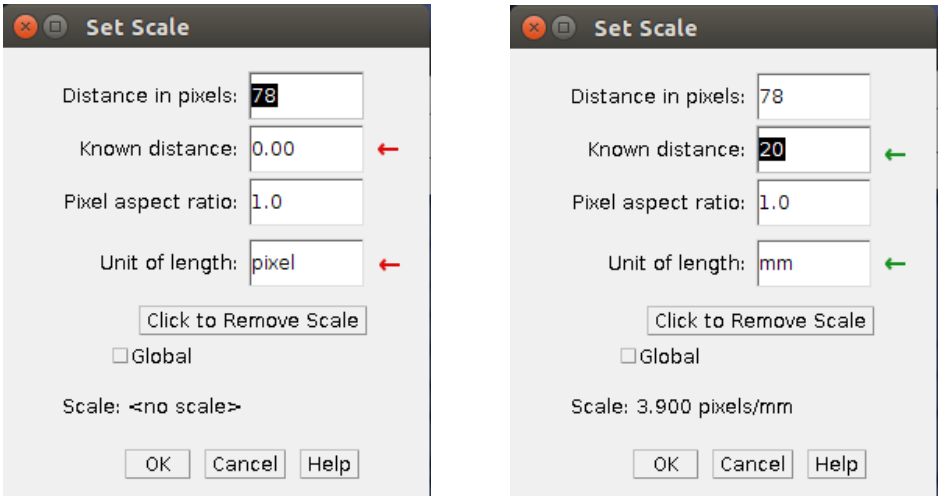
3 Applicazioni

> Non vi siete mai trovato in cima a una torre, oppure sull'orlo di un precipizio, combattuto contemporaneamente tra la tentazione di precipitarvi nel vuoto e un senso di terrore completamente opposto?
> (P. Merimée, *Lokis*, 1869)

Come primo esempio si vuole trasporre in digitale una pagina tratta da un libro stampato a Parigi nel 1905, per i tipi della *Societé d'éditions littéraires et artistiques*. L'autrice di *Claudine s'en va* è Colette, che si firma nel frontespizio con lo pseudonimo del marito, Willy. Non si tratta di una prima edizione, ma di una ristampa del tutto conforme all'originale e successiva di solo qualche anno. La scelta del testo è unicamente motivata dal fatto che l'autore di questo articolo ne possiede una copia in casa.

Sfogliando l'opera a caso, ci focalizziamo sulla pagina 89: a una rilevazione spicciola il foglio misura 175 mm × 110 mm, mentre l'area stampata 128 mm × 77 mm. I margini rifilati sembrano essere del tutto equivalenti.

L'area geografica e l'epoca indicati dallo stampatore ci suggeriscono che il corpo del carattere utilizzato sia in punti Didot. Oggi l'unità di misura più usata nei vari programmi di *desktop publishing* è il punto PostScript (`bp`, nel mondo di TEX), introdotto nella metà degli anni Settanta del XX secolo, ma diffusosi solo un decennio più tardi. Prima, per i caratteri in metallo, si usavano almeno due diverse unità di misura: il punto Didot (`dd`), stabilito da François-Ambroise Didot nel 1783 e utilizzato unicamente nella sua stamperia, sarà adottato ufficialmente in Francia solo a partire dal 1840, in Germania dal 1879 e poi nell'Europa intera; il punto di tradizione anglosassone (`pt`, che è il punto di default di TEX), diffuso soprattutto negli Stati Uniti, diverrà uno standard solo nel 1879.



(a) Campi da modificare. (b) Campi modificati.

FIGURA 2: Impostazioni di *ImageJ*, per equiparare pixel e misure reali.

Con la lente e il righello si può facilmente desumere che il carattere utilizzato in questa pagina è di 10 punti Didot. Il disegno moderno qui impiegato richiede una generosa interlinea e, considerato che le righe presenti nella pagina sono 23, si può facilmente calcolare un avanzamento di linea pari a 15 punti, per un totale di 340 punti Didot (127,83 mm) ossia 10 punti + (22 linee × 15 punti). Si consideri che alla prima riga non va aggiunto alcun avanzamento.

Tra i font più adatti e disponibili liberamente per riprodurre questo stile, scegliamo *Old Standard* di Alexey Kryukov, anche se differisce minimamente da quello usato e solo in alcuni glifi. Come motore di composizione useremo LuaLaTeX, per la capacità di gestire pure i font che non sono presenti nella distribuzione TeX Live, oltre che supportare efficacemente il pacchetto relativo alla microtipografia di cui si parlerà più avanti.

La classe che ci sembra più versatile per comporre la pagina è memoir, perché permette di controllare anche gli aspetti più minimali del testo, oltre a essere ottimamente documentata. Immediatamente, saranno necessari altri pacchetti: polyglossia, per la sillabazione (della lingua francese); fontspec, per la selezione del carattere (con opzioni che regolano le usuali sostituzioni di TeX e l'automatismo delle legature, ma anche un abbondante spazio fra le parole, com'era in voga all'epoca); microtype, per la gestione della microtipografia. Il nostro documento inizierà quindi con questo preambolo:

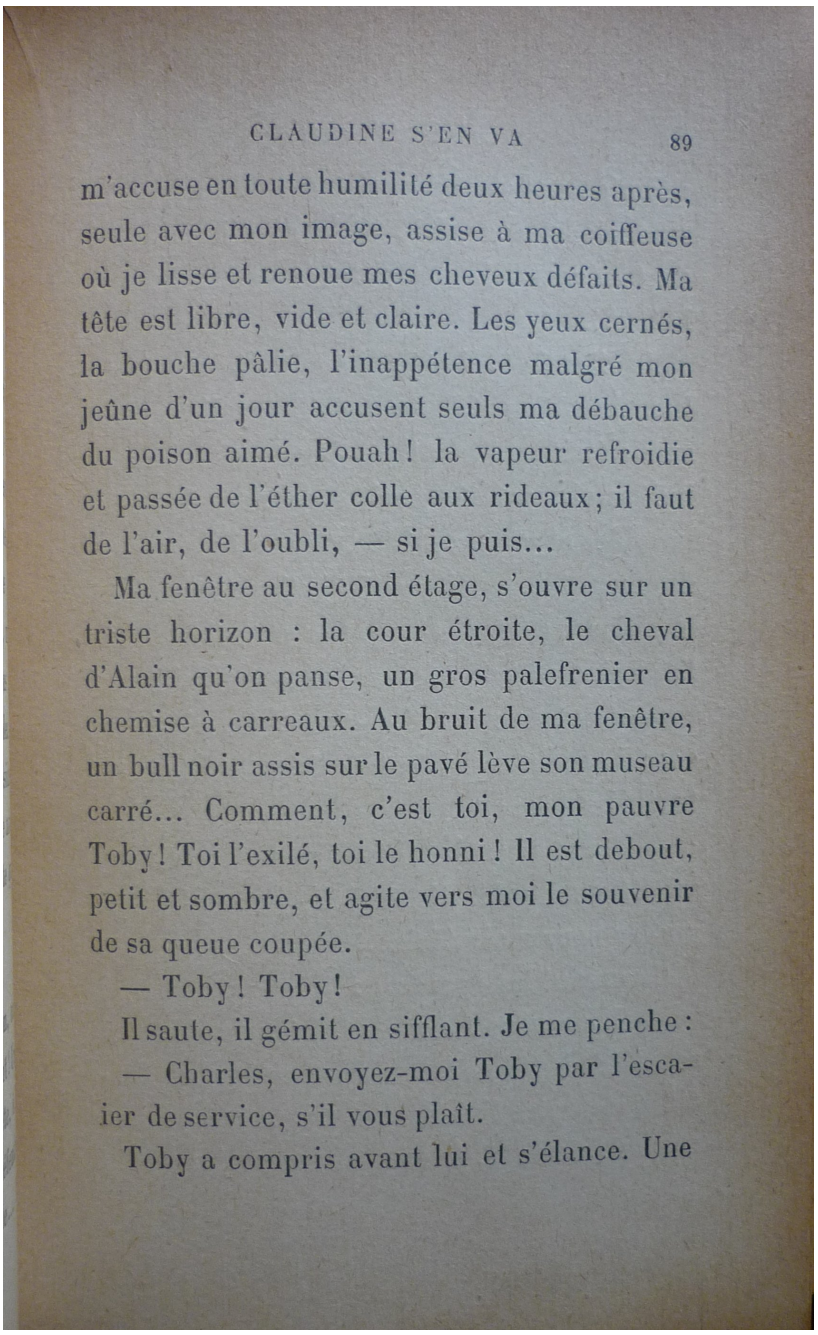
```
\documentclass[10pt,a5paper]{memoir}

\usepackage{polyglossia}
\setmainlanguage{french}

\usepackage{fontspec}
\setmainfont[Renderer=Basic,Ligatures=TeX,%
  WordSpace=1.6]{OldStandard}

\usepackage[tracking,letterspace=0]{microtype}
```

Sebbene memoir offra più varianti nella preferenza di corpi del carattere, rispetto alle classi standard di LaTeX, queste non sono sufficienti per il nostro progetto. Difatti, per disporre di caratteri in punti Didot, occorre ridefinire il file `.clo`, fornito con le macro di base. Basteranno due soli corpi, quello normale, per il testo, e quello delle note, per le testatine. Da notare come qui vengano impartite istruzioni anche per l'avanzamento di riga.

```
\makeatletter

\renewcommand{\normalsize}{%
  \@setfontsize\normalsize{10dd}{15dd}%
  \abovedisplayskip 10dd \@plus2dd \@minus5dd
  \abovedisplayshortskip \z@ \@plus3dd
  \belowdisplayshortskip 6dd \@plus3dd \@minus3dd
  \belowdisplayskip \abovedisplayskip
  \let\@listi\@listI}
\normalsize
\renewcommand\footnotesize{%
  \@setfontsize\footnotesize{8dd}{12dd}%
  \abovedisplayskip 6dd \@plus2dd \@minus4dd
  \abovedisplayshortskip \z@ \@plus1dd
  \belowdisplayshortskip 3dd \@plus1dd \@minus2dd
  \def\@listi{\leftmargin\leftmargini
              \topsep 3dd \@plus1dd \@minus1dd
              \parsep 2dd \@plus1dd \@minus1dd
              \itemsep \parsep}%
  \belowdisplayskip \abovedisplayskip
}

\makeatother
```

Quindi i comandi per memoir, che regolano le dimensioni della pagina (in questo esempio coincidono con le dimensioni della carta), l'area del testo (la lunghezza è fornita in righe di testo, la larghezza in righe tipografiche, ossia in Cicero, pari a 12 punti Didot), i margini e la distanza delle testatine.

```
\setstocksize{175mm}{110mm}
\settrimmedsize{175mm}{110mm}{*}
\setlength{\trimtop}{\stockheight}
\addtolength{\trimtop}{-\paperheight}
```



CLAUDINE S'EN VA 89

m'accuse en toute humilité deux heures après, seule avec mon image, assise à ma coiffeuse où je lisse et renoue mes cheveux défaits. Ma tête est libre, vide et claire. Les yeux cernés, la bouche pâlie, l'inappétence malgré mon jeûne d'un jour accusent seuls ma débauche du poison aimé. Pouah ! la vapeur refroidie et passée de l'éther colle aux rideaux ; il faut de l'air, de l'oubli, — si je puis...

Ma fenêtre au second étage, s'ouvre sur un triste horizon : la cour étroite, le cheval d'Alain qu'on panse, un gros palefrenier en chemise à carreaux. Au bruit de ma fenêtre, un bull noir assis sur le pavé lève son museau carré... Comment, c'est toi, mon pauvre Toby ! Toi l'exilé, toi le honni ! Il est debout, petit et sombre, et agite vers moi le souvenir de sa queue coupée.

— Toby ! Toby !

Il saute, il gémit en sifflant. Je me penche :

— Charles, envoyez-moi Toby par l'escalier de service, s'il vous plaît.

Toby a compris avant lui et s'élance. Une

Figura 3: Originale di *Claudine s'en va*.

CLAUDINE S'EN VA 89

m'accuse en toute humilité deux heures après, seule avec mon image, assise à ma coiffeuse où je lisse et renoue mes cheveux défaits. Ma tête est libre, vide et claire. Les yeux cernés, la bouche pâlie, l'inappétence, malgré mon jeûne d'un jour, accusent seuls ma débauche du poison aimé. Pouah ! La vapeur refroidie et passée de l'éther colle aux rideaux ; il faut de l'air, de l'oubli — si je puis...

Ma fenêtre, au second étage, s'ouvre sur un triste horizon : la cour étroite, le cheval d'Alain qu'on panse, un gros palefrenier en chemise à carreaux. Au bruit de ma fenêtre, un bull noir assis sur le pavé lève son museau carré... Comment c'est toi, mon pauvre Toby ! Toi l'exilé, toi le honni ! Il est debout, petit et sombre, et agite vers moi le souvenir de sa queue coupée.

— Toby ! Toby !

Il saute, il gémit en sifflant. Je me penche :

— Charles, envoyez-moi Toby par l'escalier de service, s'il vous plaît.

Toby a compris avant lui et s'élance. Une

FIGURA 4: Interpretazione di *Claudine s'en va*.

```
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
\settrims{.5\trimtop}{\trimedge}
\settypeblocksize{23\baselineskip}{17cc}{*}
\setlrmargins{*}{*}{1}
\setulmargins{*}{*}{1}
\setheaderspaces{*}{9dd}{*}
\checkandfixthelayout[lines]
```

Le testatine a loro volta presentano: in posizione centrata, il titolo del romanzo; a margine, il numero della pagina. Il titolo è composto in maiuscolo, con un corpo di dimensioni ridotte e una modesta spaziatura fra i caratteri. Questa peculiarità è ottenibile con microtype, dichiarando un'istruzione contestuale. Seguono i comandi di memoir, relativi alla disposizione delle testatine.

```
\DeclareMicrotypeSet*[tracking]{headings}
  { encoding = *,
    shape    = sc}
\SetTracking[ context = testata,%
  spacing = {830,,} ]%
  { encoding = *, shape = sc }{ 130 }

\makepagestyle{claudine}
\makeevenhead{claudine}{\footnotesize\thepage}%
  {\microtypecontext{tracking=testata}\scshape%
  \footnotesize CLAUDINE S'EN VA}{}
\makeevenfoot{claudine}{}{}{}
\makeoddhead{claudine}{}%
  {\microtypecontext{tracking=testata}%
  \scshape\footnotesize CLAUDINE S'EN VA}%
  {\footnotesize\thepage}
\makeoddfoot{claudine}{}{}{}
\pagestyle{claudine}
```

Infine, alcuni comandi per controllare lo spazio che intercorre fra i paragrafi e il rientro della prima riga del paragrafo (che coincide con la grandezza del corpo utilizzato). La spaziatura d'uso francese dopo il punto fermo risulta già attivata con la selezione della lingua. Definito il preambolo, si procede con il testo vero e proprio.

```
\parskip=0pt
\parindent=10dd

\begin{document}
```

Nelle figure 3 e 4 vengono messi a confronto modello e copia, per valutare sia il processo di trasformazione che altri accorgimenti da mettere eventualmente a punto.

4 Approfondimenti

> Aveva provato all'improvviso quello che prova un viaggiatore il quale, perduto fra le montagne, durante un'ascensione, sebbene abbia udito la guida dirgli sottovoce: "Non guardi alla sua sinistra", non tiene conto del consiglio e scorge bruscamente, a filo del piede, un picco, un abisso dalle profondità vertiginose, velate di nebbia e che sembra ricambino il suo sguardo e lo invitino al precipizio. (A. de Villiers de l'Isle-Adam, *L'Eve Future*, 1886)

Nell'affrontare qualcosa di più complesso, prendiamo in esame la pagina 36 di un altro volume che abbiamo fra le mani: *Sacros. Concilii Tridentini Canones et Decreta, item Declarationes Cardinalium Concilii interpretum...* Lugduni. Sumptibus Claudii Landrii. 1621.

Questa pubblicazione risale a un periodo in cui le officine tipografiche non disponevano di standard condivisi per la misurazione del corpo dei caratteri. Ossia, la dimensione non era definita in punti riconducibili a una precisa unità di misura, bensì mediante un nome convenzionale che rimandava allo scopo e alla tipologia di stampa cui il carattere era destinato. Ad esempio, per grossi volumi adibiti a funzioni religiose, era preferibile una famiglia di caratteri piuttosto grande e leggibile da una certa distanza; questa veniva denominata, a seconda del paese in cui veniva forgiata, come Canon, Double-canon, Corale, Doble Parangona, Kanon, Parys Romeyn, e il valore del corpo oscillava dai 16.5 mm ai 19 mm. Ogni bottega fondeva i propri tipi e le matrici venivano confezionate per esigenze strettamente personali.

Per ovviare a confusioni terminologiche, spesso i bibliografi preferiscono riferirsi a una misurazione che tiene conto delle venti righe di testo stampato (questo criterio è usato anche per i manoscritti), secondo la modalità definita da Robert Proctor e successivamente perfezionata da Konrad Haebler. Ipoteticamente, l'inconsueta dicitura `20 100 x 2.2 : 3.2 mm` significa che il carattere qui presente misura 100 mm in venti righe non spaziate (`20 100`), l'altezza media delle lettere minuscole senza ascendenti o discendenti corrisponde a 2.2 mm (`x 2.2`) e l'altezza delle maiuscole è di 3.2 mm (`: 3.2`) (Vervliet, 2008, p. 3). Si ritiene perciò preferibile adottare i millimetri come unità di misura per tutti gli stampati anteriori al 1830, anni in cui cominciano a essere maggiormente diffuse e condivise le misurazioni in punti. Quindi, solo quando si sarà sicuri di avere a che fare con un volume proveniente dalle officine di Truchet, Fournier, Didot, ovvero coloro che introdussero il concetto di punto tipografico e ne determinarono lo sviluppo, sarà consigliabile adottare le loro rispettive misurazioni.

Per la nostra trasposizione, operiamo come prima, ridefinendo il corpo del carattere principale in Cicero, e quello delle note a margine in Petit Romain. Non bisogna scordare di impostare polyglossia per la sillabazione della lingua latina.

```
\documentclass[10pt,a5paper]{memoir}

\usepackage{polyglossia}
\setmainlanguage{latin}

\makeatletter

% dimensioni Cicéro / Pica
\renewcommand{\normalsize}{
  \@setfontsize\normalsize{4.1mm}{4.25mm}}
```

```
\normalsize
% dimensioni Petit Romain / Long Primer
\renewcommand\footnotesize{
  \@setfontsize\footnotesize{2.8mm}{2.8mm}}

\g@addto@macro\@marginparreset\footnotesize
\g@addto@macro\@marginparreset\raggedright

\makeatother
```

Come si è potuto notare, sono state introdotte anche alcune varianti alle macro di base che regolano il formato delle note a margine. Nel nostro esemplare, il carattere usato nelle note si presenta rimpicciolito e il testo disposto a bandiera.

Attualmente non è disponibile un carattere libero identico a quello usato nell'originale. Quello che gli assomiglia maggiormente è l'*EB Garamond* di Georg Duffner che viene distribuito anche in formato OpenType, in grado cioè di contenere molte funzioni tipografiche avanzate. Per sfruttarne le potenzialità, faremo uso di alcune specifiche del pacchetto fontspec. Eccone un estratto conciso:

```
\usepackage{fontspec}
 \defaultfontfeatures{%
  RawFeature={%
   %+liga% legature, modalità attivata di default
   ,+calt% controllo contestuale di glifi s lunga
   ,+clig% legature contestuali corsive is e es
   ,+hlig% legature storiche st, ct e sp
   ,+cv04% forma alternata della & corsiva
   ,+cv05% forma alternata della v corsiva
   ,+cv06% forma alternata delle virgolette
   ,+ss02% variante che cambia v e u
   }%
  }

 \setmainfont{EB Garamond}
```

Gli svolazzi vengono introdotti solo per i titoli correnti in corsivo, come accade nel testo originale. Nei piedini viene indicata, come si usava un tempo, la prima parola della pagina successiva, grazie al pacchetto fwlw.

```
\makepagestyle{concilium}
\makeevenhead{concilium}{\thepage}{\itshape%
  \addfontfeature{RawFeature=+swsh}%
  Concil. Trid. cum Declarat. \&\ Remiss.}{}
\makeevenfoot{concilium}{}{}{\usebox\NextWordBox}
\makeoddhead{concilium}{}{\itshape%
  \addfontfeature{RawFeature=+swsh}%
  Sessio V. De Reform.}{\thepage}
\makeoddfoot{concilium}{}{}{\usebox\NextWordBox}
\pagestyle{concilium}
```

Per gli altri aspetti del testo, il codice non si discosta molto dall'esempio precedente, ad eccezione delle istruzioni relative alla posizione delle note a margine, alla misurazione in millimetri, alla distanza delle testatine dal testo, al rapporto fra i margini.

```
\setstocksize{181mm}{110mm}
\settrimmedsize{181mm}{110mm}{*}
\setlength{\trimtop}{\stockheight}
\addtolength{\trimtop}{-\paperheight}
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
\settrims{.5\trimtop}{\trimedge}
\settypeblocksize{34\baselineskip}{77mm}{*}
\setlrmargins{*}{24mm}{*}
\setulmargins{14mm}{*}{*}
\setmarginnotes{1.5mm}{11mm}{\onelineskip}
\setheaderspaces{*}{0mm}{*}
\setheadfoot{\onelineskip}{\onelineskip}
\checkandfixthelayout[lines]

\parskip=0pt
\parindent=4.1mm
\frenchspacing

\begin{document}
```

Il discorso attinente alle titolazioni e ai frontespizi risulta essere molto più articolato, così come quello relativo ad altri aspetti della composizione, per cui se ne rimanda ad altra sede la discussione. Quel che premeva nel nostro caso era solo fornire degli esempi concreti di progettazione del testo, a partire da stampe preesistenti.

5 Epilogo

> Oltre alle pulsioni di conservazione che costringono alla ripetizione, ce ne potrebbero benissimo essere altre che spingono verso l'evoluzione e la produzione di nuove forme. (S. Freud, *Jenseits des Lustprinzips*, 1920)

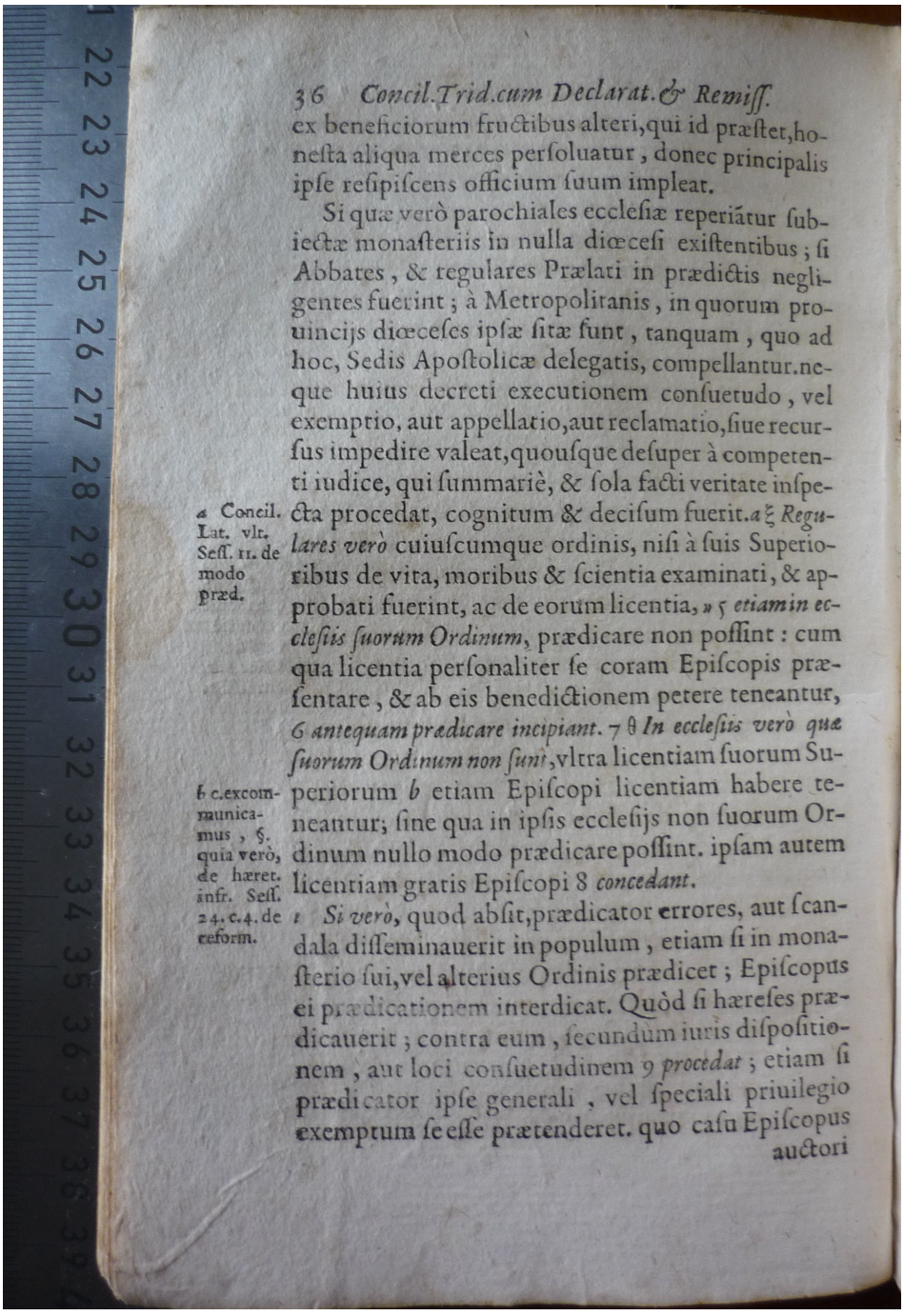
Era necessario il vago sfoggio di epigrafi ottocentesche all'inizio di ogni parte del testo? Forse gli autori qui citati non ebbero alcun timore di riciclare quanto scritto dai predecessori? Fecero tutti riferimento a un sentire comune e condiviso, tipico della loro epoca? Furono per caso precursori di quanto poi formalizzato in modo compiuto nell'opera freudiana in cui si fa più urgente l'istanza filosofica di Nietzsche? Niente di tutto questo.

Recuperare quanto sia funzionale, in un'arte poco flessibile come quella tipografica, è la regola. Questo accade in ogni campo, anche in quelli più creativi, come si richiede all'invenzione letteraria, senza che ne venga compromessa l'originalità complessiva. Perché questa viene determinata dal contesto in cui interagiscono più fattori, che a loro volta definiscono l'intero sistema. Rendersi conto dei meccanismi che hanno generato ottimi risultati non significa copiare ottusamente un modello, bensì indagare sulle strategie messe a punto per risolvere un dato problema, squisitamente estetico, nel limite delle tecnologie disponibili in un certo momento storico.

Non un punto di arrivo per definire uno stile tipografico, ma un viaggio nel tempo di cui tale percorso è solo l'inizio.

Riferimenti bibliografici

Arseneau, D. (1995). *The fwlw package*. Leggibile con il comando `texdoc fwlw` nella distribuzione TEX Live.



36 *Concil. Trid. cum Declarat. & Remiss.*
ex beneficiorum fructibus alteri, qui id præstet, honesta aliqua merces persoluatur, donec principalis ipse resipiscens officium suum impleat.

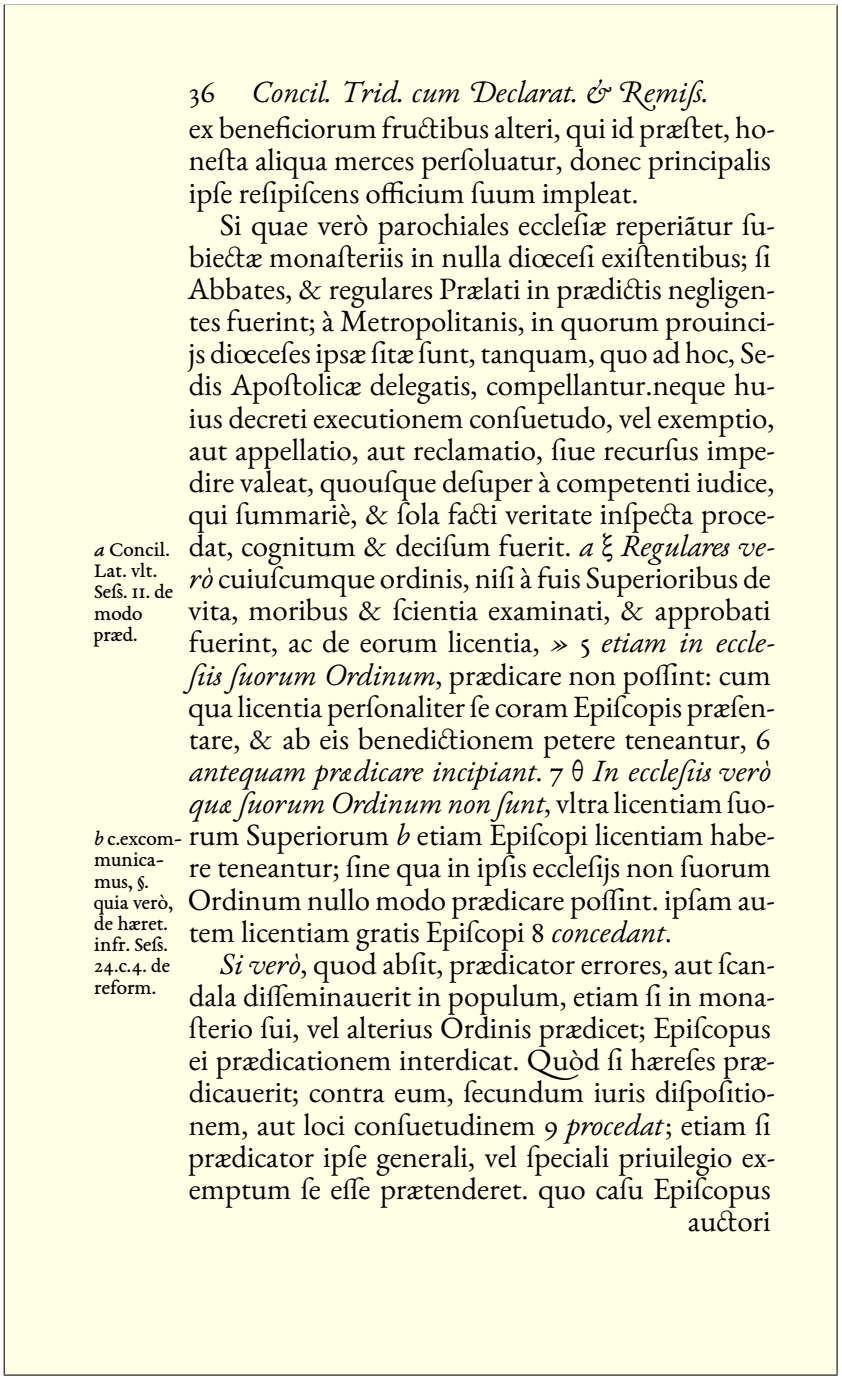
Si quæ verò parochiales ecclesiæ reperiãtur subiectæ monasteriis in nulla diœcesi existentibus; si Abbates, & regulares Prælati in prædictis negligentes fuerint; à Metropolitanis, in quorum prouincijs diœceses ipsæ sitæ sunt, tanquam, quo ad hoc, Sedis Apostolicæ delegatis, compellantur. neque huius decreti executionem consuetudo, vel exemptio, aut appellatio, aut reclamatio, siue recursus impedire valeat, quousque desuper à competenti iudice, qui summariè, & sola facti veritate inspecta procedat, cognitum & decisum fuerit. *a* ξ *Regulares verò* cuiuscumque ordinis, nisi à suis Superioribus de vita, moribus & scientia examinati, & approbati fuerint, ac de eorum licentia, » ς *etiam in ecclesiis suorum Ordinum*, prædicare non possint: cum qua licentia personaliter se coram Episcopis præsentare, & ab eis benedictionem petere teneantur, 6 *antequam prædicare incipiant.* 7 θ *In ecclesiis verò quæ suorum Ordinum non sunt*, vltra licentiam suorum Superiorum *b* etiam Episcopi licentiam habere teneantur; sine qua in ipsis ecclesijs non suorum Ordinum nullo modo prædicare possint. ipsam autem licentiam gratis Episcopi 8 *concedant.*

ι *Si verò*, quod absit, prædicator errores, aut scandala disseminauerit in populum, etiam si in monasterio sui, vel alterius Ordinis prædicet; Episcopus ei prædicationem interdicat. Quòd si hæreses prædicauerit; contra eum, secundùm iuris dispositionem, aut loci consuetudinem 9 *procedat*; etiam si prædicator ipse generali, vel speciali priuilegio exemptum se esse prætenderet. quo casu Episcopus auctori

a Concil. Lat. vlt. Sess. 11. de modo præd.

b c. excommunicamus, §. quia verò, de hæret. infr. Sess. 24. c. 4. de reform.

Figura 5: Originale dei *Concilii Tridentini Canones et Decreta.*



36 *Concil. Trid. cum Declarat. & Remiſs.*
ex beneficiorum fructibus alteri, qui id præſtet, honeſta aliqua merces perſoluatur, donec principalis ipſe reſipiſcens officium ſuum impleat.

Si quae verò parochiales eccleſiæ reperiãtur ſubiectæ monaſteriis in nulla dicœceſi exiſtentibus; ſi Abbates, & regulares Prælati in prædictis negligentes fuerint; à Metropolitanis, in quorum prouincijs dicœceſes ipsæ ſitæ ſunt, tanquam, quo ad hoc, Sedis Apoſtolicæ delegatis, compellantur.neque huius decreti executionem conſuetudo, vel exemptio, aut appellatio, aut reclamatio, ſiue recurſus impedire valeat, quouſque deſuper à competenti iudice, qui ſummariè, & ſola facti veritate inſpecta procedat, cognitum & deciſum fuerit. *a* ξ *Regulares verò* cuiuſcumque ordinis, niſi à ſuis Superioribus de vita, moribus & ſcientia examinati, & approbati fuerint, ac de eorum licentia, » 5 *etiam in eccleſiis ſuorum Ordinum*, prædicare non poſſint: cum qua licentia perſonaliter ſe coram Epiſcopis præſentare, & ab eis benedictionem petere teneantur, 6 *antequam prædicare incipiant*. 7 θ *In eccleſiis verò quæ ſuorum Ordinum non ſunt*, vltra licentiam ſuorum Superiorum *b* etiam Epiſcopi licentiam habere teneantur; ſine qua in ipſis eccleſijs non ſuorum Ordinum nullo modo prædicare poſſint. ipſam autem licentiam gratis Epiſcopi 8 *concedant*.

Si verò, quod abſit, prædicator errores, aut ſcandala diſſeminauerit in populum, etiam ſi in monaſterio ſui, vel alterius Ordinis prædicet; Epiſcopus ei prædicationem interdicat. Quòd ſi hæreſes prædicauerit; contra eum, ſecundum iuris diſpoſitionem, aut loci conſuetudinem 9 *procedat*; etiam ſi prædicator ipſe generali, vel ſpeciali priuilegio exemptum ſe eſſe prætenderet. quo caſu Epiſcopus auctori

a Concil. Lat. vlt. Seſs. 11. de modo præd.

b c.excommunicamus, §. quia verò, de hæret. infr. Seſs. 24.c.4. de reform.

Figura 6: Trasformazione dei *Concilii Tridentini Canones et Decreta.*

Beccari, C. *et al.* (2014). *Introduzione all'arte della composizione tipografica con LaTeX*. URL http://www.guitex.org/home/it/guide-per-iniziare/la-guida-guit.

Carter, H. (1969). *A view of early typography up to about 1600*. Oxford University Press.

Charette, F. (2014). *Polyglossia: a Babel replacement for XeLaTeXand LuaLaTeX*. Leggibile con il comando `texdoc polyglossia` nella distribuzione TeX Live. L'attuale versione è affidata alla manutenzione di Arthur Reutenauer.

De Vinne, T. (1900). *The Practice of typography*. Century Company. URL https://archive.org/details/cu31924016925863.

Duffner, G. (2014). «EB Garamond». URL http://www.georgduffner.at/ebgaramond/.

Gurrado, M. e Lestingi, G. (2009). «Graphoskop». URL http://www.palaeographia.org/graphoskop/index.htm.

Kryukov, A. (2010). «Old Standard». URL http://openfontlibrary.org/en/font/old-standard.

Pégourié-Gonnard, M. (2013). *A guide to LuaLaTeX*. Leggibile con il comando `texdoc lualatex` nella distribuzione TeX Live.

Rasband, W. *et al.* (2012). «ImageJ». URL http://rsb.info.nih.gov/ij/. Una distribuzione estesa del programma con svariati plugin, *Fiji*, si può trovare qui: http://fiji.sc/Fiji.

Robertson, W. e Hosny, K. (2014). *The fontspec package*. Leggibile con il comando `texdoc fontspec` nella distribuzione TeX Live.

Schlicht, R. (2013). *The microtype package*. Leggibile con il comando `texdoc microtype` nella distribuzione TeX Live.

Vervliet, H. (2008). *The Palaeotypography of the French Renaissance: Selected Papers on Sixteenth-century Typefaces*. Numero v. 1 in Library of the Written Word. Brill Academic Pub.

Wilson, P. (2013). *The memoir class*. Leggibile con il comando `texdoc memoir` nella distribuzione TeX Live. L'attuale versione è affidata alla manutenzione di Lars Madsen.

▷ Claudio Vincoletto
Torino
claudio dot vincoletto at gmail dot com

Espressioni regolari in LaTeX

Enrico Gregorio

Sommario

Con expl3, l'ambiente di programmazione per LaTeX3, e il pacchetto l3regex possiamo usare espressioni regolari come in Perl e altri linguaggi.

Abstract

With expl3, the programming environment for LaTeX3, and the l3regex package we can use regular expressions like in Perl e other languages.

1 Introduzione

Cominciamo con il solito *toy problem*. Vogliamo stampare in neretto tutte le 'a' di una parola data come argomento a una macro. Questo è abbastanza facile con una macro ricorsiva

```
\makeatletter
\def\bolda#1{\bolda@aux#1a\bolda@aux}
\def\bolda@aux#1a#2\bolda@aux{%
  #1%
  \if\relax#2\relax
   \expandafter\@gobble
  \else
    \expandafter\@firstofone
  \fi
  {\textbf{a}\bolda@aux#2\bolda@aux}%
}
\makeatother
\bolda{aiutare}
\bolda{aiuola}
```

aiutare **a**iuol**a**

Più complicato sarebbe stampare tutte le vocali, maiuscole e minuscole comprese, in neretto. Si può fare, usando macro di van der Laan (van der Laan, 1993).

```
\def\fifo#1{%
  \ifx\ofif#1\ofif\fi
  \process{#1}\fifo
}
\def\ofif#1\fifo{\fi}
\def\loc#1#2{%locate #1 in #2
  \def\locate##1#1##2\end{%
    \ifx\empty##2\empty
      \foundfalse
    \else
      \foundtrue
    \fi
  }%
  \locate#2.#1\end
}
\newif\iffound
\def\boldvowels#1{%
  \def\process##1{%
    \uppercase{\loc##1}{AEIOU}%
    \iffound\textbf{##1}\else##1\fi
  }%
\fifo #1\ofif
}

\boldvowels{aiutare}
\boldvowels{Aiuola}
```

aiutare **A**iuol**a**

Vediamo subito qualcosa di più maneggevole.

```
% \usepackage{xparse,l3regex}
\ExplSyntaxOn
\NewDocumentCommand{\boldvowels}{m}
 {
  \manual_boldvowels:n { #1 }
 }
\tl_new:N \l_manual_boldvowels_tl
\cs_new_protected:Nn \manual_boldvowels:n
 {
  \tl_set:Nn \l_manual_boldvowels_tl { #1 }
  \regex_replace_all:nnN
   { ([AEIOUaeiou]) }
   { \c{textbf}\cB\{\1\cE\} }
   \l_manual_boldvowels_tl
  \tl_use:N \l_manual_boldvowels_tl
 }
\ExplSyntaxOff

\boldvowels{aiutare Aiuola}
```

aiutare **A**iuol**a**

Due fatti sono indiscutibili: il primo è che questo codice è centinaia di volte meno efficiente dell'altro, il secondo è che questo codice è infinite volte più potente. Il pacchetto l3regex (The LaTeX team, 2014b) fa parte di expl3 (The LaTeX team, 2014a), il livello di programmazione di LaTeX3. È ancora classificato sperimentale, ma è ormai piuttosto stabile.

2 Introduzione a l3regex

Vediamo in dettaglio il codice, sistemando prima i dettagli. I comandi `\ExplSyntaxOn` e `\ExplSyntaxOff` servono a entrare nell'ambiente di programmazione di LaTeX3 e a uscirne. Il comando a livello utente passa il controllo a una funzione interna; è buona norma farlo perché spesso permette di evitare duplicazioni di codice. Le funzioni per eseguire sostituzioni con espressioni regolari richiedono poi che il testo da esaminare sia contenuto in una variabile *token list*, che quindi definiamo. Quin-

di la parte più importante, cioè la funzione interna, del tipo `protected` perché contiene assegnazioni.

Infatti, l'argomento viene memorizzato nella variabile e quindi entra in azione la funzione davvero importante: `\regex_replace_all:nnN`, come dice la sua firma, richiede tre argomenti:

```
\regex_replace_all:nnN
  ⟨regex search⟩
  ⟨regex replace⟩
  ⟨token list variable⟩
```

Il terzo argomento è una variabile di tipo *token list* e non ne parliamo più; questa variabile, dopo il massaggio eseguito dall'istruzione `\regex_replace:nnN` verrà usata alla fine. Il primo e il secondo argomento sono invece due *espressioni regolari*. Chiunque abbia un minimo di esperienza al riguardo riconoscerà nella prima una *classe*, cioè un gruppo di caratteri che verranno cercati nel contenuto della variabile; qui sono le vocali, cioè

```
[AEIOUaeiou]
```

Le parentesi intorno alla classe servono a "ricordare" il carattere trovato in modo che possa essere usato nella sostituzione. Infatti l'espressione regolare per la sostituzione contiene `\1`, che sta per la sottoespressione che costituisce un *match*. Se nell'espressione di sostituzione si fosse usato `\1\1`, le vocali sarebbero raddoppiate.

L'analisi dell'espressione di sostituzione è un po' più complicata e la rimandiamo a dopo. Per ora descriveremo le principali caratteristiche delle espressioni regolari. Lo standard POSIX è rispettato il più possibile, quindi

- `.` rappresenta un carattere qualsiasi, più precisamente un *token* qualsiasi;
- ogni carattere alfabetico rappresenta sé stesso;
- ogni cifra rappresenta sé stessa;
- ogni carattere non alfabetico preceduto dalla barra rovescia rappresenta sé stesso (per esempio, `\?` rappresenta `?` e `\.` rappresenta il punto);
- `\d` rappresenta una cifra qualsiasi, `\D` qualsiasi token che non sia una cifra;
- `\s` rappresenta uno spazio, `\S` qualsiasi token che non sia uno spazio;
- `\w` rappresenta una lettera, una linea bassa `_` oppure una cifra, mentre `\W` qualsiasi altro token.

Ci sono altre scorciatoie, più che altro per compatibilità con lo standard, ma che sono meno importanti per l'uso con LaTeX. Si possono naturalmente esprimere classi:

- `[...]` indica una classe 'positiva';
- `[^...]` indica una classe 'negativa', per esempio `[^AEIOUaeiou]` rappresenta ogni token che non sia una vocale;
- `[:`⟨*nome*⟩`:]` indica una classe predefinita, i nomi sono gli stessi dello standard POSIX;
- `[:^`⟨*nome*⟩`:]` indica il complementare della classe predefinita;
- `[x-y]` indica un *intervallo* per esempio `[a-z]` indica le lettere minuscole.

Molto importanti sono gli operatori di ripetizione che sono identici allo standard e che vengono trasformati da 'ingordi' (*greedy*) in 'pigri' (*lazy*) con l'aggiunta di `?`:

- `?` e `??` indicano nessuna o una occorrenza;
- `*` e `*?` indicano zero o più occorrenze;
- `+` e `+?` indicano una o più occorrenze;
- `{`⟨n⟩`}` indica esattamente ⟨n⟩ occorrenze (la versione pigra non ha senso);
- `{`⟨n⟩`,}` e `{`⟨n⟩`,}?` indicano ⟨n⟩ o più occorrenze;
- `{`⟨n_1⟩`,`⟨n_2⟩`}` e `{`⟨n_1⟩`,`⟨n_2⟩`}?` indicano almeno ⟨n_1⟩ e non più di ⟨n_2⟩ occorrenze.

Si possono denotare punti particolari:

- `\b` indica inizio o fine di una parola, cioè un punto che è preceduto da un token `\w` e seguito da un token `\W` o viceversa; l'inizio e la fine del testo sono considerati come se appartenessero alla classe `\W`;
- `^` o `\A` indicano l'inizio del testo;
- `$`, `\Z` o `\z` indicano la fine del testo.

È possibile usare `(...)`, `(?:...)` e `(?|...)` per 'catturare' (o no) token da usare poi nell'espressione di sostituzione con `\1`, `\2` e così via esattamente come nello standard POSIX.

Attenzione! Un'importante deviazione dallo standard, inevitabile con TeX, è che gli spazi nelle espressioni regolari sono *ignorati*. Uno spazio va espresso con `\␣` (o con `\s`).

Un'altra possibilità è quella di adoperare il prefisso `\u`; precisamente, se `\l_manual_foo_tl` è una variabile token list (può anche essere globale o costante, ovviamente), con `\u{l_manual_foo_tl}` possiamo indicare il valore della variabile in un'espressione regolare. Per esempio, se volessimo cercare

```
\begin{minipage}[t]{\textwidth}
```

invece di una complicata espressione come

```
\c{begin}\cB.minipage\cE\[t\]
  \cB.\c{textwidth}\cE.
```

possiamo più semplicemente assegnare quel complesso a una variabile con

```
\tl_set:Nn \l_manual_foo_tl
 {\begin{minipage}[t]{\textwidth}}
```

e specificare `\u{l_manual_foo_tl}` nell'espressione regolare.

3 Novità in l3regex

La sostanziale differenza con il trattamento delle espressioni regolari in Perl o `sed` è che in LaTeX abbiamo *token* e non semplici successioni di caratteri. Per fare un esempio facile, l'espressione regolare `foo` non troverà nulla se il testo da esaminare contiene `abc\foo`, perché `\foo` è un singolo oggetto agli occhi di TeX che ha poco a che vedere, internamente, con il nome della macro. Analogamente, se ci interessa trovare con `\{` una 'vera' parentesi graffa e non una che sia, per qualche motivo, un carattere stampabile.

Andando un po' più in profondità, i confronti sono eseguiti tramite `\if`, che ignora il codice di categoria. Per ottenere risultati più precisi che considerano anche la duplice natura dei caratteri in TeX sono disponibili prefissi che indicano che tipo di token si sta effettivamente cercando o si vuole sostituire. I prefissi sono tutti della forma `\cX`, dove `X` è una lettera tra `CBEMTPUDSLOA` con le seguenti corrispondenze, dove useremo le solite convenzioni riguardo ai caratteri speciali:

- `\cC` per una sequenza di controllo;
- `\cB` per `{`;
- `\cE` per `}`;
- `\cM` per `$`;
- `\cT` per `&`;
- `\cP` per `#`;
- `\cU` per `_`;
- `\cD` per `^`;
- `\cS` per uno spazio;
- `\cL` per una lettera;
- `\cO` per una 'non lettera';
- `\cA` per un carattere attivo.

Con il prefisso `\c` seguito da un argomento tra graffe contenente caratteri si indica quella precisa sequenza di controllo. I prefissi possono essere combinati anche con le classi; per esempio, l'espressione

```
[\c0\d \c[LO][A-F]]
```

corrisponde alla specifica delle cifre esadecimali in TeX: sono le cifre, con codice di categoria 12, oppure le lettere maiuscole ABCDEF con categoria 11 oppure 12 (si veda l'esercizio 24.1 del TeXbook).

Tipicamente si userà `\cB.` per indicare in un'espressione di ricerca una graffa aperta o `\cC.` per una qualsiasi sequenza di controllo. Nelle espressioni di sostituzione invece faremo seguire al prefisso il carattere che effettivamente vogliamo. Si capisce dunque perché nell'esempio abbiamo

```
\c{textbf}\cB\{\1\cE\}
```

che significa: 'il comando `\textbf`, una graffa aperta, ciò che hai trovato, una graffa chiusa'. Perciò la 'A' di 'Aiuola' verrà sostituita da `\textbf{A}` e così per gli altri caratteri che corrispondono all'espressione di ricerca, cioè le vocali. Se avessimo adoperato semplicemente `\\textbf\{\1\}` il risultato sarebbe stato completamente diverso, perché avremmo sostituito semplici stringhe di caratteri (per la precisione di categoria 12) e non i token che ci servono. Ecco la prova:

```
% \usepackage{xparse,l3regex}
\ExplSyntaxOn
\NewDocumentCommand{\oops}{m}
 {
  \manual_oops:n { #1 }
 }
\tl_new:N \l_manual_oops_tl
\cs_new_protected:Nn \manual_oops:n
 {
  \tl_set:Nn \l_manual_oops_tl { #1 }
  \regex_replace_all:nnN
   { ([AEIOUaeiou]) }
   { \\textbf\{\1\} }
   \l_manual_oops_tl
  \tl_use:N \l_manual_oops_tl
 }
\ExplSyntaxOff

\oops{abc}
```

```
\textbf{a}bc
```

L'esistenza di `\regex_replace_all:nnN` suggerisce quella di `\regex_replace_once:nnN` che infatti esiste e ha la stessa sintassi, ma esegue solo la sostituzione della prima occorrenza dell'espressione di ricerca. Entrambe non fanno nulla se la ricerca non ha successo.

Ci sono parecchie altre funzioni, qualcuna la vedremo. Per ora va menzionata `\regex_match:nnTF` che ha la sintassi

```
\regex_match:nnTF
  ⟨regex⟩
  ⟨token list⟩
  ⟨true⟩
  ⟨false⟩
```

e che, a differenza delle due descritte, prende come primo argomento un'espressione regolare e come secondo un qualsiasi testo. Se l'espressione regolare ha una corrispondenza nel testo, sarà eseguito il codice ⟨*true*⟩, altrimenti il codice ⟨*false*⟩. Come al solito in expl3, sono definite anche le varianti `\regex_match:nnT` e `\regex_match:nnF` per evitare di dover specificare un argomento vuoto. Per esempio, se volessimo emettere un avviso nel caso una variabile *token list* non ha corrispondenze con l'espressione regolare, potremmo dire

```
% \usepackage{xparse,l3regex}
% \ExplSyntaxOn
\cs_generate_variant:Nn
  \regex_match:nnF { nV }
\regex_match:nVF
 { [AEIOUaeiou] }
 \l_manual_test_tl
 { \msg_warning:nn { test } { novowels } }
%\ExplSyntaxOff
```

avendo definito l'opportuno messaggio di avviso.

Ecco un'applicazione più seria. Vogliamo un'espressione regolare per riconoscere se una sequenza di caratteri soddisfa la definizione di numero decimale secondo TEX:[1]

```
\regex_const:Nn \c_manual_number_regex
 {\A \s* (\+|\-)* \cO\d* \.? \cO\d* \s? \Z}
```

dove, con `\A` e `\Z` diciamo che vogliamo esaminare tutta la sequenza di caratteri che può cominciare con qualsiasi numero di spazi e terminare con uno spazio; inoltre può avere un numero qualsiasi di caratteri `+` e `-`, seguiti da zero o più cifre (di categoria 12), poi al più da un punto decimale e quindi altre cifre. In realtà la sintassi di TEX dà la possibilità di usare la virgola, ma è bene evitarlo, perché la virgola è usatissima come separatore di liste di token nelle opzioni. Se si vuole ammettere anche la virgola, basta sostituire `\.` con `(\.|\,)`, come per i segni iniziali facoltativi. Come si vede, è un'espressione regolare del tutto standard (a parte gli spazi che però danno maggiore leggibilità).

Ecco vista una nuova proprietà di l3regex: la possibilità di definire variabili e costanti di tipo `regex`. Le funzioni del pacchetto hanno tutte la variante in cui il primo argomento è di tipo `N` e richiede una variabile o costante di tipo `regex`. Così, per vedere se un argomento della nostra macro è un numero decimale, possiamo usare

```
\regex_match:NnTF \c_manual_unit_regex
 { #1 }
 { a~number }
 { not~a~number }
```

Sono analogamente disponibili

- `\regex_replace_once:NnN` e
- `\regex_replace_all:NnN`.

L'argomento che contiene l'espressione di sostituzione non può mai essere una variabile o costante di tipo `regex`; il motivo è che, al momento, non sembra utile. Se casi d'uso dovessero emergere, non sarà difficile aggiungere le varianti. La funzione per definire una costante l'abbiamo vista; per definire e impostare variabili si adoperano

```
\regex_new:N \l_manual_vara_regex
\regex_set:Nn \l_manual_vara_regex
 { ⟨regex⟩ }
\regex_clear:N \l_manual_vara_regex
\regex_new:N \g_manual_varb_regex
\regex_gset:Nn \g_manual_varb_regex
 { ⟨regex⟩ }
\regex_gclear:N \g_manual_varb_regex
```

dove la prima variabile è da usare localmente, la seconda globalmente. Lo scopo è di avere a disposizione espressioni regolari di ricerca di uso frequente; il risparmio è nel fatto che l'espressione è già predigerita. Un esempio è nella tabella 1, in cui si mostra il risultato al terminale del comando `\regex_show:N \c_manual_number_regex`.

4 Altri esempi

Gli esempi che presenteremo sono forse banali, ma dovrebbero dare l'idea delle potenzialità del pacchetto.

Primo problema: dividere una stringa in parti uguali, cominciando da destra; per esempio, la stringa `1234567` divisa a due a due deve diventare `1,23,45,67`. Ecco il codice, con solo la parte 'interna': l'interfaccia utente è facile da scrivere.[2]

```
% \usepackage{l3regex}
\ExplSyntaxOn
\tl_new:N \l_manual_string_tl
\cs_new_protected:Nn \manual_split:nn
 {
  \tl_set:Nn \l_manual_string_tl { #2 }
  % we need to start from the end,
  % so we reverse the string
  \tl_reverse:N \l_manual_string_tl
  % add a comma after any group of #1
     tokens
  \regex_replace_all:nnN
   { (.{#1}) }
   { \1\, }
   \l_manual_string_tl
  % if the length of the string is a
  % multiple of #1 a trailing comma is
  % added, so we remove it
  \regex_replace_once:nnN
   { \,\Z }
   { }
   \l_manual_string_tl
  % reverse back
  \tl_reverse:N \l_manual_string_tl
 }
```

1. Si veda http://tex.stackexchange.com/questions/138737/.

2. Si veda http://tex.stackexchange.com/questions/171007/.

Tabella 1: Esempio di sviluppo di un'espressione regolare

```
% \regex_const:Nn \c_manual_number_regex
%  { \A \s* (\+|\-)* \c0\d* \.? \c0\d* \s? \Z }
% \regex_show:N \c_manual_number_regex

> Compiled regex variable \c_manual_number_regex:
+-branch
  assertion: anchor at start (\A)
  Match, repeated 0 or more times, greedy
    char code 32
    char code 9
    char code 10
    char code 12
    char code 13
  ,-group begin
  | char code 43
  +-branch
  | char code 45
  `-group end, repeated 0 or more times, greedy
  Match, repeated 0 or more times, greedy
    categories 0, class
      range [48,57]
  char code 46, repeated between 0 and 1 times, greedy
  Match, repeated 0 or more times, greedy
    categories 0, class
      range [48,57]
  Match, repeated between 0 and 1 times, greedy
    char code 32
    char code 9
    char code 10
    char code 12
    char code 13
  assertion: anchor at end (\Z).
```

```
\manual_split:nn { 2 } { 1234567 }
\tl_use:N \l_manual_string_tl\par
\manual_split:nn { 3 } { aaabbbccc }
\tl_use:N \l_manual_string_tl\par

\ExplSyntaxOff
```

```
1,23,45,67
aaa,bbb,ccc
```

Per ovviare al problema dell'inizio, la stringa viene rovesciata e dopo ogni gruppo di n caratteri (dove n rappresenta il primo argomento) viene aggiunta la virgola con l'espressione `(.{`⟨n⟩`})` che cattura il *match* e può adoperarlo nell'espressione di sostituzione. Il caso in cui la lunghezza della stringa è un multiplo di n produce una virgola in più che rimuoviamo ancorando la seconda espressione regolare alla fine del testo con `\Z`. Per finire, rovesciamo di nuovo la stringa.

Naturalmente questo si potrebbe ottenere anche con una macro ricorsiva, tuttavia con l3regex è possibile controllare in anticipo se la stringa contiene solo caratteri permessi, per esempio cifre.

Un'altra applicazione che evita complicate definizioni con caratteri attivi è quella di caricare parti di documenti scritte secondo le convenzioni di Markdown. Per semplicità, supporremo che siano usate solo `*parola*` per il corsivo e `**parola**` per il neretto. L'idea è di adoperare catchfile per assegnare a una variabile *token list* l'intero contenuto del file e su questa variabile operare con `\regex_replace_all:nnN`. In realtà `\CatchFileDef` non è necessaria, perché la funzione è già stata incorporata in expl3 come `\tl_set_from_file:Nnn`.

Questo è il file di esempio

```
A **test** with longer **bold face
also split** across lines.

Now also *italic*.
```

e questo è il codice completo

```
% \usepackage{xparse,l3regex}

\ExplSyntaxOn
\NewDocumentCommand{\mdinput}{m}
 {
  \manual_mdinput:n { #1 }
 }

\tl_new:N \l__manual_mdfile_tl

\cs_new_protected:Nn \manual_mdinput:n
 {
  \tl_set_from_file:Nnn
     \l__manual_mdfile_tl
```

```
    { } { #1 }
   \regex_replace_all:nnN
    { \*\*([^\*]*?)\*\* }
    { \c{textbf}\cB\{ \1 \cE\} }
    \l__manual_mdfile_tl
   \regex_replace_all:nnN
    { \*([^\*]*?)\* }
    { \c{emph}\cB\{ \1 \cE\} }
    \l__manual_mdfile_tl
   \tl_use:N \l__manual_mdfile_tl
  }

\ExplSyntaxOff

\mdinput{test.md}
```

A **test** with longer **bold face also split** across lines.
Now also *italic*.

Si potrebbe migliorare imponendo nell'espressione di ricerca controlli che vietino `\par` nel testo tra asterischi. Ovviamente il codice è inefficiente se i testi da importare sono molto lunghi.

5 Emulare la sostituzione di argomenti

Supponiamo di voler simulare un *here file*.[3] L'idea è di avere una struttura del tipo

```
\tofile{⟨file⟩}{⟨argomenti⟩}<<EOF
⟨codice⟩
<<EOF
```

dove ⟨*codice*⟩ contiene `#1`, `#2` e così via, da sostituire con gli argomenti specificati e ⟨*file*⟩ è il nome del file su cui scrivere; per esempio, vorremmo

```
\tofile{test.c}{{One}{Two}{Three}}<<EOF
/* #1, #2 */
#include <stdio.h>
main() {
  printf("#3",\n);
}
<<EOF
```

dove si noti il carattere `#` da scrivere così com'è. La stringa finale è arbitraria e va ripetuta alla fine per indicare dove si deve smettere.

L'idea è abbastanza semplice: gli argomenti sono memorizzati in una variabile *sequence*; il codice viene letto una riga alla volta (sorvolo sulla complicazione tecnica) e sulla riga vengono operate le necessarie sostituzioni con

```
\regex_replace_all:nnN
  { \# i }
  { ⟨i-esimo argomento⟩ }
  \l_manual_line_tl
```

Lento, ma efficace. Naturalmente, prima di leggere le righe, i caratteri speciali vengono tutti neutralizzati.

Nella tabella 2 si può leggere il codice che è interessante anche per altri usi di funzioni di expl3; di seguito è riportato il risultato.

6 Un pacchetto basato su l3regex

La possibilità di adoperare espressioni regolari ha subito suscitato l'idea di un nuovo pacchetto per *rappezzare* comandi e risolvere certe difficoltà che sorgono con etoolbox. Già xpatch è basato su expl3 per la procedura di decisione sul tipo di macro da rappezzare; infatti ce ne sono ben sette, perché occorre distinguere fra quelle definite con `\newcommand`, con `\DeclareRobustCommand` e `\newrobustcmd`; per le prime due è diverso il caso di parole di controllo (come `\foo`) o simboli di controllo (come `\?`). Il nuovo pacchetto si chiama regexpatch.

La verifica si esegue sull'espansione di primo livello della macro e con espressioni regolari è più facile controllarne la *firma*. Per esempio, la firma della macro `\foo` definita con `\newcommand` e un argomento facoltativo è

```
\A\\@protected@testopt\ \\foo\ \\\\
```

e il nome della macro può essere memorizzato in una variabile *token list*, da specificare con il prefisso `\u`. Sembra complicato, ma è solo questione di pazienza, perché le firme sono davvero univoche, una volta che l'espansione di primo livello della macro è trasformata in semplice stringa con `\token_get_replacement_spec:N`.

Ma questa semplificazione non è tutto. Si può usare l3regex anche per il rappezzo! E quindi fornire anche una versione che modifica tutte le occorrenze di una certa lista di token con altre. L'esempio è quello del pacchetto semantic che non è compatibile con LuaLaTeX; la modifica richiede di sostituire in `\mathligsoff` (che è un comando robusto) e in `\@addligto` ogni occorrenza di `\mathcode` con `\Umathcodenum` che compare una volta nel primo e tre nel secondo. Si può fare con etoolbox con

```
\makeatletter
\expandafter\patchcmd
  \csname mathligsoff \endcsname
  {\mathcode}{\Umathcodenum}
  {}{}
\patchcmd\@addligto
  {\mathcode}{\Umathcodenum}{}{}
\patchcmd\@addligto
  {\mathcode}{\Umathcodenum}{}{}
\patchcmd\@addligto
  {\mathcode}{\Umathcodenum}{}{}
\makeatother
```

e xpatch renderebbe solo più facile il primo rappezzo. Con regexpatch il codice diventa

3. Si veda http://tex.stackexchange.com/questions/155358.

Tabella 2: Codice per un *here file*.

```
\documentclass{article}
\usepackage{xparse,l3regex}

\ExplSyntaxOn
\NewDocumentCommand{\tofile}{m}
 {
  \manual_tofile:n { #1 }
 }

\iow_new:N \g_manual_write_stream
\tl_const:Nn \c_manual_specials_tl
 { \  \\ \{ \} \$ \& \# \^ \_ \% \~ }
\tl_new:N \l_manual_argument_tl
\tl_new:N \l_manual_line_tl
\seq_new:N \l_manual_arguments_seq
\int_new:N \l_manual_arguments_int

\cs_new_protected:Npn \manual_dospecials:
 {
  \tl_map_inline:Nn \c_manual_specials_tl
   {
    \char_set_catcode_other:N ##1
   }
 }

\cs_new_protected:Npn \manual_tofile:n #1
 {
  \group_begin:
  \tex_endlinechar:D `\^^J
  \iow_open:Nn \g_manual_write_stream { #1 }
  \manual_tofile_aux:nw
 }

\cs_new_protected:Npn \manual_tofile_aux:nw #1
     #2 ~%
 {% #1 is the list of arguments,
  % #2 is the terminator
  \tl_set:Nn \l_manual_eof_tl { #2 }
  \tl_trim_spaces:N \l_manual_eof_tl
  \seq_set_split:Nnn \l_manual_arguments_seq
   { } { #1 }
  \int_set:Nn \l_manual_arguments_int
   { \seq_count:N \l_manual_arguments_seq }
  \manual_dospecials:
  \manual_readline:w
 }

\cs_new_protected:Npn \manual_readline:w #1 ^^J
 {
  \str_if_eq:VnTF \l_manual_eof_tl { #1 }
   {
    \iow_close:N \g_manual_write_stream
    \group_end:
   }
   {
    \manual_replace_write:n { #1 }
    \manual_readline:w
   }
 }
```

```
\cs_new_protected:Npn \manual_replace_write:n #1
 {
  \tl_set:Nn \l_manual_line_tl { #1 }
  \int_step_inline:nnnn
   { 1 }
   { 1 }
   { \l_manual_arguments_int }
   {
    \tl_set:Nx \l_manual_argument_tl
     {
      \seq_item:Nn \l_manual_arguments_seq
       { ##1 }
     }
    \regex_replace_all:nnN
     { \# ##1 }
     { \u{l_manual_argument_tl} }
     \l_manual_line_tl
   }
  \iow_now:NV \g_manual_write_stream
   \l_manual_line_tl
 }

\cs_generate_variant:Nn \iow_now:Nn { NV }
\cs_generate_variant:Nn \str_if_eq:nnTF { V }

\ExplSyntaxOff

\begin{document}
\tofile{test.c}{{One}{Two}{Three}}<<EOF
/* #1, #2 */
#include <stdio.h>
main() {
  printf("#3",\n);
}
<<EOF

\end{document}
```

test.c

```
/* One, Two */
#include <stdio.h>
main() {
  printf("Three",\n);
}
```

```
\makeatletter
\xpatchcmd\mathligsoff
  {\mathcode}{\Umathcodenum}{}{}
\xpatchcmd*{\@addligto}
  {\mathcode}{\Umathcodenum}{}{}
\makeatother
```

che è certamente più comodo. La versione con l'asterisco è quella che 'cambia tutto'.

I comandi `\xpatchcmd`, `\xpretocmd` e `\xapptocmd` sono volutamente gli stessi usati da xpatch, perché l'idea è di sostituire i rappezzi basati su etoolbox con quelli basati su l3regex, ma il pacchetto è ancora in versione sperimentale dal momento che lo è ancora l3regex. C'è anche un comando nuovo, `\regexpatchcmd` che negli argomenti di ricerca e sostituzione vuole appunto espressioni regolari.

Per esempio, supponiamo di voler eliminare dal comando `\chaptermark` della classe book il malefico `\MakeUppercase`. La definizione è

```
\def\chaptermark#1{%
  \markboth{\MakeUppercase{%
    \ifnum\c@secnumdepth>\m@ne
      \if@mainmatter
        \@chapapp\ \thechapter. \ %
      \fi
  \fi
  #1}}{}}
```

e l'ovvio rappezzo sarebbe

```
\makeatletter
\xpatchcmd{\chaptermark}
  {\MakeUppercase}
  {\@firstofone}
  {}{}
\makeatother
```

perché così eliminiamo anche la coppia di graffe, a spese di un'espansione in più. Volendo eliminare anche le graffe, il rappezzo sarebbe più complicato che riscriversi la macro. Con `\regexpatchcmd` si può fare meglio:

```
\makeatletter
\regexpatchcmd{\chaptermark}
  {\c{MakeUppercase}\cB.(.*?)\cE.}
  {\1}
  {}{}
\makeatother
```

e vale la pena di capire che cosa succede. L'espressione di ricerca si combina con `\MakeUppercase` seguito da una graffa aperta, da qualsiasi token (con ricerca pigra) e una graffa chiusa. I token tra le graffe sono ricordati e infatti l'espressione di sostituzione dice proprio di lasciare solo ciò che c'è tra le graffe.

Un esempio fittizio di qualcosa che non è possibile eseguire con etoolbox. Supponiamo di avere una macro `\abc` il cui comportamento deve cambiare leggermente nell'appendice, nella quale si deve modificare l'uso di uno degli argomenti. È possibile, con `\patchcmd`, avere `#1` in uno degli argomenti di ricerca o sostituzione, ma non se `\patchcmd` appare nell'argomento di un altro comando; questo è un problema se si deve eseguire un rappezzo in `\AtBeginDocument`. Si potrebbe semplicemente eseguire il rappezzo dopo `\appendix`, ma una buona programmazione vuole il codice tutto nel preambolo. Ecco come:

```
\newcommand{\abc}[2]{#1\ #2}
\newcommand{\patchabc}{%
  \regexpatchcmd{\abc}
    {\cP.1}{\cP\#2}
    {}{}%
}
\xapptocmd{\appendix}{\patchabc}{}{}
```

In altre parole, definiamo `\abc` e anche `\patchabc` che ha come unico scopo quello di rappezzare `\abc`. Con `\xapptocmd` diciamo quando eseguire il rappezzo; si potrebbe evitare il comando ausiliario, ma in questo modo sembra più pulito. Il rappezzo può andare nell'argomento di `\newcommand` perché non si usa mai un `#` esplicito, ma solo la sua rappresentazione in un'espressione regolare.

Un altro esempio è un rappezzo all'ambiente `rubric` di CurVe[4] per togliere la riga di 'continuazione'; l'ambiente è basato su `longtable` e occorreva togliere tutto quanto va da `\endfirsthead` (escluso) a `\endhead` (incluso). Con `\regexpatchcmd` è semplicissimo:

```
\regexpatchcmd{\rubric}
  {\c{endfirsthead}.*\c{endhead}}
  {\c{endfirsthead}}
  {}{}
```

dove non occorre la ricerca pigra perché `\endhead` compare solo una volta. È chiaro che avere a disposizione funzioni molto potenti non esime dallo studio della macro da rappezzare.

Il pacchetto è in grado di eseguire facilmente altri compiti che sarebbero molto complicati con altri metodi e, in sostanza, sarebbero risolubili solo riscrivendosi le macro. La funzione `\xpatchparametertext` è in grado di modificare il *parameter text* di una macro. Un caso d'uso è legato a babel per il ceco e lo slovacco.[5] Con una di queste lingue, il carattere `-` è un prefisso (come `"` per l'italiano o il tedesco) e questo ha come conseguenza che le macro `\cline` e `\cmidrule` non funzionano, perché quando si dà `\cline{1-2}`, la macro separa i due numeri cercando un carattere `-` di categoria 12, mentre ne trova uno di categoria 13. La definizione è alla riga 5185 di `latex.ltx`

4. http://www.guitex.org/home/en/forum/5-tex-e-latex/68612.
5. http://tex.stackexchange.com/questions/111999.

```
\def\cline#1{\@cline#1\@nil}
\def\@cline#1-#2\@nil{%
  \omit
  ...
}
```

e ce n'è una simile in booktabs per `\cmidrule`. Occorre sostituire il carattere di categoria 12 con lo stesso di categoria 13; il secondo rappezzo ovviamente richiede booktabs:

```
\makeatletter
\xpatchparametertext\@cline
  {-}{\cA-}
  {}{}
\xpatchparametertext\@@@cmidrule
  {-}{\cA-}
  {}{}
\makeatother
```

Ovviamente dovrebbe essere babel a occuparsi di questo; visto che non lo fa, possiamo facilmente correre ai ripari.

Riferimenti bibliografici

THE LATEX TEAM (2014a). «expl3». CTAN.

— (2014b). «l3regex». CTAN.

VAN DER LAAN, K. (1993). «Syntactic sugar». *TUGboat*, **14** (3), pp. 310–318.

▷ Enrico Gregorio
Dipartimento di Informatica, Università di Verona
Enrico dot Gregorio at univr dot it

MenùTEX: una piattaforma per realizzare menù

Claudio Fiandrino

Sommario

L'articolo presenta MenùTEX, una piattaforma che realizza semplici menù stampati su foglio A4 in formato 2×3. Per la redazione del documento, la piattaforma mette a disposizione un'interfaccia grafica attraverso cui gli utenti accedono ad informazioni presenti su un database.

Abstract

The article presents MenùTEX, an architecture which aims to print simple menus on A4 paper in a 2×3 format. To create the final document, the architecture provides a graphical interface that enables the users to access information stored in a database.

1 Introduzione

Un menù è un documento più semplice rispetto ad una tesi o a un articolo o a una semplice lettera. Recentemente, mi è stato chiesto se ci fosse un metodo migliore rispetto all'utilizzo dei comuni software commerciali per realizzare semplici menù da stampare in formato 2×3 su foglio a4 e la scelta è caduta subito su LATEX. Perché? In primo luogo garantisce una qualità decisamente superiore rispetto ad altri software. Inoltre, il sorgente di un documento è codice, perciò può essere preparato con una *programmazione ad hoc*. Da qui è nato MenùTEX.

Il concetto di *programmazione ad hoc* è sicuramente il pilastro fondamentale della piattaforma. Rispetto ad altri tipi di documenti, un menù è sostanzialmente composto da un testo semplice, poche righe ognuna delle quali contiene generalmente poche parole. Queste righe possono cambiare giorno per giorno, tuttavia le combinazioni non sono infinite e il loro contenuto può ripetersi nel tempo. Un'organizzazione efficiente di questo tipo di informazione richiede un database. È quindi importante che le informazioni presenti sul database siano utilizzabili di volta in volta nel documento LATEX. Per questo motivo la piattaforma MenùTEX è composta da un'interfaccia grafica che permette agli utenti di organizzare e selezionare le informazioni su un database e redigere il documento da stampare con LATEX sulla base delle scelte effettuate.

I menù presi in considerazione sono molto semplici, con una doppia scelta di primi e secondi piatti più eventuali variazioni. Sono adatti, ad esempio, a ritiri di gruppi sportivi. Tuttavia, non si pensi che la piattaforma funzioni proprio per la relativa semplicità del progetto. L'idea base può essere tranquillamente estesa in altri contesti, ad esempio la realizzazione sistematica di badge per eventi come conferenze.

L'articolo è così strutturato. La sezione 2 illustra l'ambiente di lavoro usato per lo sviluppo e necessario per l'utilizzo della piattaforma. La sezione 3 analizza ad alto livello l'interfaccia grafica mentre la sezione 4 prende in esame il nucleo di MenùTEX, ossia la generazione e la compilazione dei documenti LATEX. Infine, la sezione 5 conclude il lavoro.

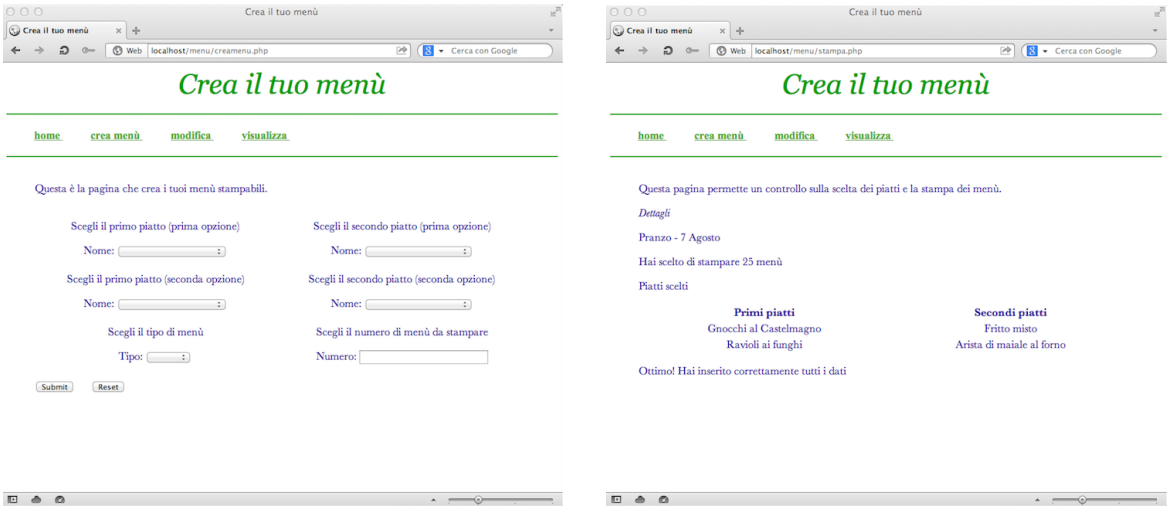
2 L'ambiente di lavoro

Il sistema operativo utilizzato per lo sviluppo è Mac OS X. Il funzionamento del sito per l'interfaccia grafica avviene in locale grazie all'applicazione open source MAMP. L'acronimo del nome è generato dalle seguenti iniziali: **M**ac OS X (il sistema operativo), **A**pache (la piattaforma server Web più comune e supportata dalla maggioranza dei sistemi operativi), **M**ySQL (la piattaforma per la gestione dei database) e **P**HP (il linguaggio di programmazione "server-side" più diffuso per Web secondo `http://w3techs.com/`). I file web dell'interfaccia grafica sono stati posizionati in `/Applications/MAMP/htdocs/`.

Lo sviluppo dell'interfaccia grafica di MenùTEX in forma Web è stata pensata con i seguenti obbiettivi. Primo in ordine di importanza, non si vincolano gli utenti a questo piuttosto che quel sistema operativo. MAMP è pensato per uno specifico sistema operativo, ma esistono simili applicazioni per Linux e Windows (LAMP e WAMP rispettivamente). Tutte queste applicazioni supportano intrinsecamente piattaforme di gestione dei database e la portabilità richiede variazioni al software minime. Non è stato necessario sviluppare un apposito editor perché i parametri da selezionare possono essere inseriti tramite qualsiasi browser installato sulla propria macchina. Inoltre, la piattaforma funziona in locale, ma può essere "convertita" a sito Web remoto vero e proprio con pochissime modifiche.

Per ottenere il risultato finale con LATEX si è usata la distribuzione MacTEX, pensata appositamente per il sistema operativo Mac OS X; in particolar modo, per il funzionamento di MenùTEX sono strettamente necessari la classe standalone Scharrer (2012) e i pacchetti:

- Ti*k*Z Tantau (2013);



(a) La pagina di input.

(b) La pagina di conferma e stampa.

FIGURA 1: L'interfaccia grafica.

- pdfpages MATTHIAS (2013);
- background MEDINA (2014).

3 L'interfaccia grafica

L'interfaccia grafica è lo strumento che permette agli utenti di gestire e selezionare le informazioni presenti sul database per la stampa.

Sviluppata con codice HTML e PHP, l'interfaccia si compone di quattro pagine principali:

- la home page;
- la pagina per selezionare le informazioni dal database;
- la pagina per immettere o eliminare informazioni dal database;
- la pagina per visualizzare le informazioni presenti nel database.

Ai fini dell'articolo prenderemo in analisi soltanto la seconda. Tale pagina non solo permette di selezionare le informazioni dal database, ma attiva anche la procedura di stampa.

La figura 1a riporta l'aspetto della pagina di selezione delle informazioni. L'utente può selezionare con classici menù a tendina le opzioni per i primi e secondi piatti. Queste informazioni vengono reperite dal database. Inoltre, è anche necessario scegliere quale tipo di menù stampare; in queso caso le opzioni sono soltanto due: *Pranzo* o *Cena*. Infine, l'utente deve digitare nel riquadro testuale il numero di menù da stampare.

Il bottone "Reset" permette di annullare tutte le opzioni selezionate. Se invece l'utente ha inserito tutti i parametri, può continuare attraverso il bottone "Submit". Una volta compiuta questa operazione, dietro le quinte viene effettuato un controllo per verificare la presenza e la correttezza di tutti i parametri inseriti. Se il controllo ha esito positivo, si viene indirizzati verso la pagina di conferma e in automatico parte la procedura di stampa. La pagina di conferma è mostrata in figura 1b.

4 Il backend: usiamo LaTeX

Come si è visto, l'interfaccia grafica permette all'utente di selezionare una serie di parametri che dovranno far parte di un documento LaTeX. Si analizza ora la metodologia con cui MenùTEX gestisce questi parametri. È importante notare che l'utente finale non si accorge delle operazioni che si andranno a prendere in esame. Ogni passaggio, dalla redazione alla stampa finale, viene svolto rigorosamente in background.

4.1 La preparazione

Avendo un vincolo preciso sul numero di menù da stampare per foglio (6) e sapendo che il numero complessivo di menù (n) può potenzialmente cambiare ogni giorno, si è pensato di operare dividendo il flusso di lavoro in due passi. Nel primo passo, si crea un documento di n pagine in cui ogni pagina è un singolo menù. A tale proposito, la classe standalone è senza dubbio lo strumento migliore. Nel secondo passo, utilizzando il pacchetto pdfpages si può facilmente raggruppare nel formato 2×3 le pagine del primo documento.

Redigere il documento standalone

La figura 2 mostra una pagina del documento standalone con i parametri di ingresso riportati in figura 1b. La "pagina-base" è stata realizzata con Ti*k*Z caratterizzando ogni etichetta come nodo e posizionando i vari nodi con l'aiuto della libreria `positioning`.

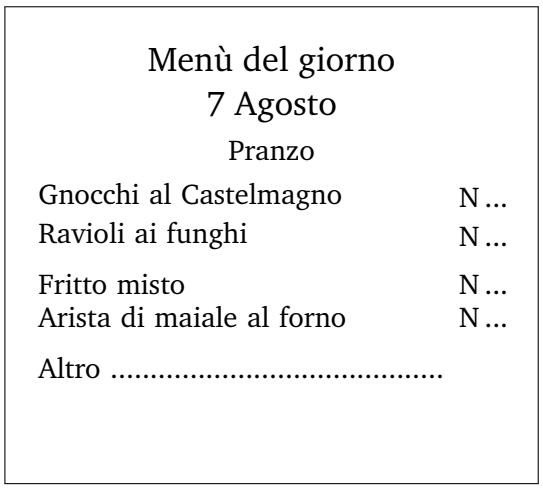


FIGURA 2: Una pagina del documento standalone.

La scelta del pacchetto non è casuale. In primo luogo, nonostante l'aspetto attuale del menù sia decisamente minimalistico, la sua implementazione con Ti*k*Z può permettere l'aggiunta di ulteriori elementi grafici in maniera semplice. In secondo luogo, la classe standalone ha un'opzione tikz per scontornare in modo ottimale figure composte con-Ti*k*Z. Infine, Ti*k*Z fornisce un costrutto `\foreach` perciò si possono creare n pagine uguali utilizzando come idea base il seguente pseudo-codice:

```
\documentclass[tikz]{standalone}

\begin{document}
\foreach \pagina in {1,...,<n>}{
 \begin{tikzpicture}
 codice figura
 \end{tikzpicture}
}
\end{document}
```

Si noti che l'opzione tikz di standalone carica automaticamente Ti*k*Z.

Prima di redigere il documento, per prima cosa è necessario creare un nuovo file. Tale file viene posizionato in una cartella temporanea chiamata `file-menu` nel solito percorso `/Applications/MAMP/htdocs/`. Il comando PHP utilizzato è:

```
$fh = fopen("./file-menu/$nf.tex", "w")
or die("can't open file");
```

dove `$nf` è una variabile che contiene il nome del file senza estensione, `$fh` è il nome del file handler che si utilizzerà successivamente e l'opzione `"w"` specifica che il file viene aperto in scrittura. Tuttavia, per essere completamente certi di poter scrivere sul nuovo file, è buona regola specificarne i permessi:

```
chmod("./file-menu/$nf.tex",777);
```

PHP mette anche a disposizione una funzione `tempnam()` per creare file temporanei con nomi univoci. Tuttavia, volendo creare una copia di archivio per ogni menù stampato, è preferibile procedere assegnando a `$nf` un nome significativo oltre che univoco. MenùTEX usa la data nel formato giorno-mese-anno più una stringa per distinguere se il menù sia "Pranzo" oppure "Cena".

A questo punto si può redigere il documento. Considerato il limitato numero di righe di codice, si è proceduto riga per riga salvando in una variabile temporanea la stringa e scrivendola sul file con l'aiuto del file handler. Ad esempio:

```
$strData = "\usepackage[T1]{fontenc}\n";
fwrite($fh, $strData);
```

Un'attenzione particolare deve essere prestata agli elementi che l'utente deve selezionare come parametri. Ad esempio, il codice per descrivere la prima opzione del primo piatto è:

```
$strData = "\\node[text width=7.2cm,
 align=left,
 anchor=south west]
 (scelta-1) at (0.25,4.5)
 {{$primo_piatto_op1}};\n";
fwrite($fh, $strData);
```

Occorre prestare attenzione al fatto che in PHP il carattere `\` deve essere "protetto" in maniera simile a quanto avviene con LaTeX. Inoltre, è anche necessario raddoppiare le parentesi graffe: analogamente con il caso precedente, quelle interne "proteggono" il contenuto. Senza tale operazione, PHP scriverebbe sul file soltanto il valore della variabile `$primo_piatto_op1` senza le parentesi graffe (generando di conseguenza un errore). Questa variabile, come ogni parametro di input, rappresenta il valore selezionato dall'utente e viene trasferito dalla pagina di inserimento dei parametri a quella di stampa e conferma attraverso il metodo POST.

Il codice 2 mostra il sorgente del documento standalone completo necessario a generare le pagine illustrate nella figura 2 di esempio.

Redigere il documento finale

Completato il documento standalone, si può procedere alla redazione del documento finale dopo avere creato il nuovo file e specificato i permessi con operazioni del tutto analoghe a quanto descritto in precedenza. La figura 3 mostra una pagina di esempio.

Come accennato, per ottenere la disposizione 2×3 è conveniente caricare il pacchetto pdfpages. Il comando base `\includepdf` permette di assegnare all'opzione nup il valore 2x3. Quindi:

```
$strDataN = "\includepdf[pages=-,
 nup=2x3,
 offset=-0cm 0cm,
 delta=0.4cm 0cm]
 {{$nf.pdf}} \n";
fwrite($fh_n, $strDataN);
```

```
\documentclass[tikz,border=2pt]{standalone}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[italian]{babel}
\usepackage{charter}
\usepackage{tikz}
\usetikzlibrary{positioning}

\tikzset{posizione/.style={align=left,anchor=south west}}

\begin{document}
\foreach \num in {1,...,25}{
\begin{tikzpicture}[font=\LARGE]
\useasboundingbox (0,0) rectangle (9,8);
\path (0,0) rectangle (9,8);

% INTESTAZIONE
\node (menu) at (4.5,7.25){Men\'{u} del giorno};
\node[below = 0cm of menu](date) { 7 Agosto};
\node[below = 0.075cm of date,font=\Large] (pranzo) {Pranzo};

% PIATTI
\begin{scope}[font=\Large,minimum height=1ex,text height=2ex]
% PRIMI
\node[text width=7.2cm,posizione] (scelta-1) at (0.25,4.5){Gnocchi al Castelmagno};
\node[text width=1cm,posizione,right= 0cm of scelta-1] (numero-1) {N\,...};
\node[text width=7.2cm,align=left,below= 0cm of scelta-1] (scelta-2)
{Ravioli ai funghi};
\node[text width=1cm,posizione,right= 0cm of scelta-2] (numero-2) {N\,...};
% SECONDI
\node[text width=7.2cm,posizione,below=0.2cm of scelta-2] (scelta-3) {Fritto misto};
\node[text width=1cm,posizione,right= 0cm of scelta-3] (numero-3) {N\,...};
\node[text width=7.2cm,align=left,below= 0cm of scelta-3] (scelta-4)
{Arista di maiale al forno};
\node[text width=1cm,posizione,right= 0cm of scelta-4] (numero-4) {N\,...};
% ALTRO
\node[text width=7.2cm,posizione,below=0.3cm of scelta-4] (altro)
{Altro .....................................};
\end{scope}

\end{tikzpicture}
}
\end{document}
```

Codice 2: Il documento standalone

Figura 3: Una pagina del documento completo.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[italian]{babel}
\usepackage{charter}

\usepackage{pdfpages}
\usepackage{tikz}
\usetikzlibrary{calc}
\usepackage[contents={}]{background}

\newcommand{\pageindicators}{%
\tikz[remember picture,overlay]{%
\draw[ultra thin, dash pattern=on 9.5pt off 9pt]
  (current page.north)--(current page.south)
  ($(current page.north west)!0.6675!(current page.west)$)--
  ($(current page.north east)!0.6675!(current page.east)$)
  ($(current page.south west)!0.6675!(current page.west)$)--
  ($(current page.south east)!0.6675!(current page.east)$);
  }%
}

\AddEverypageHook{%
\pageindicators%
\BgMaterial}%

\begin{document}
\includepdf[pages=-,nup=2x3, offset=-0cm 0cm, delta=0.4cm 0cm]
{Pranzo-7-08-2014-single.pdf}
\end{document}
```

Codice 3: Il documento finale.

Ovviamente, è necessario caricare il file pdf del sorgente `.tex` creato in precedenza. Non specificando l'estionesione del file direttamente all'interno della variabile `$nf`, quest'ultima può essere riutilizzata senza problemi.

Per agevolare il taglio dei vari menù di ogni singola pagina, si è pensato di introdurre dei segmenti di delimitazione. Ancora una volta, Ti*k*Z è di aiuto con un nodo speciale, `current page`, il quale identifica completamente la pagina che si compone. Ad esempio, per accedere al punto in alto a sinistra del foglio, è sufficiente specificare `current page.north west`.

Per poter *aggiungere* elementi alla pagina corrente generalmente si ricorre alle opzioni remember picture e overlay. Tali opzioni hanno un inconveniente: richiedono una doppia compilazione. In seguito si ritornerà su questo punto. Il nodo `current page` inoltre, identifica di volta in volta la pagina corrente. Tuttavia, i segmenti devono essere presenti in tutto il documento perciò si è usato il pacchetto background, sviluppato appositamente per questo genere di necessità. In genere si procede in questo modo. Per prima cosa si raggruppano in una macro gli elementi da inserire in ogni pagina (`\pageindicators`). Successivamente si utilizza la macro appena definita come parametro:

```
\AddEverypageHook{%
\pageindicators%
\BgMaterial}%
```

Il codice 3 mostra il sorgente del documento finale completo necessario a generare le pagina di esempio riportata in figura 3.

4.2 La compilazione

Per ottenere i documenti mostrati nelle figure 2 e 3 è ovviamente necessario compilare i sorgenti. Il primo requisito in questa fase è l'obbligo di dover compilare due volte il sorgente del documento finale in quanto si sono usate le opzioni remember picture e overlay in una figura Ti*k*Z. Inoltre, si deve prestare attenzione affinché il documento standalone sia compilato *prima* del documento finale.

PHP permette l'esecuzione di programmi esterni con diverse funzioni. Nel caso specifico, poiché il motore di composizione pdflatex può essere eseguito da terminale, si è scelto di usare la funzione `shell_exec()`.

Per semplicità, si è scelto di creare un piccolo script bash capace non solo di compilare i sorgenti nell'ordine esatto, ma anche di rimuovere i file ausiliari e i sorgenti dopo il processo di compilazione. Il codice dello script è il seguente:

```
#!/bin/bash

cd ./file-menu
# doppia compilazione
for file in *.tex; do
 /usr/local/texlive/2014/bin/
 x86_64-darwin/pdflatex "$file";
done
for file in *.tex; do
```

```
 /usr/local/texlive/2014/bin/
 x86_64-darwin/pdflatex "$file";
done
# rimozione file ausiliari e sorgenti
rm *.log
rm *.aux
rm *.tex
```

Come primo passo si accede alla cartella in cui i sorgenti sono posizionati e, per ogni file con estensione `.tex`, si lancia il processo di compilazione. Questa scelta si rivela funzionale perché i sorgenti vengono rimossi una volta terminato il processo. In questo modo si è sicuri che per ogni menù che si genera, la cartella `file-menu` inizialmente *non* contiene nessun sorgente. Il rispetto del vincolo di precedenza nella compilazione, invece, viene garantito dietro le quinte attraverso la scelta dei nomi dei due sorgenti.

Si noti che è stato necessario specificare *per intero* il percorso del motore di composizione, pena il non funzionamento dello script. Per non specificare completamente l'intero percorso e quindi avere una versione semplificata dello script:

```
#!/bin/bash

cd ./file-menu
# doppia compilazione
for file in *.tex; do
 pdflatex "$file";
done
for file in *.tex; do
 pdflatex "$file";
done
# rimozione file ausiliari e sorgenti
rm *.log
rm *.aux
rm *.tex
```

sarebbe necessario configurare il file `.bashrc` in modo opportuno affinché i percorsi dei motori di composizione siano riconosciuti.

5 Conclusione

L'articolo ha illustrato MenùTEX, una piattaforma che realizza semplici menù stampati su foglio a4 in formato 2×3. Si è focalizzata l'attenzione sull'architettura, evidenziando l'interazione tra l'interfaccia grafica e il backend, ossia come avviene la creazione dinamica dei sorgenti LaTeX con parametri impostati dall'utente e la loro successiva compilazione. Il backend è completamente nascosto all'utente che quindi non deve necessariamente avere conoscenze LaTeX specifiche per poter operare.

MenùTEX è oggi praticamente utilizzato dal ristorante Tre Verghe d'Oro (`http://www.treverghe.it/`), perciò si può considerare a tutti gli effetti un esempio concreto di applicazione LaTeX in un contesto commerciale.

6 Ringraziamenti

Voglio ringraziare il revisore anonimo per i preziosi suggerimenti che contribuiranno sicuramente al miglioramento della piattaforma. Un doveroso ringraziamento va a Claudio Beccari, il direttore di ArsTEXnica e a tutta la redazione per il prezioso lavoro e la passione che dedicano alla rivista.

Riferimenti bibliografici

Matthias, A. (2013). *The pdfpages Package*. URL `http://www.ctan.org/pkg/pdfpages`. Consultabile con: `texdoc pdfpages`

Medina, G. (2014). *The background package*. URL `http://www.ctan.org/pkg/background`. Consultabile con: `texdoc background`

Scharrer, M. (2012). *The standalone Package*. URL `http://www.ctan.org/pkg/standalone`. Consultabile con: `texdoc standalone`

Tantau, T. (2013). *The TikZ and PGF Packages*. URL `http://www.ctan.org/pkg/pgf`. Consultabile con: `texdoc tikz`

▷ Claudio Fiandrino
`claudio dot fiandrino at gmail dot com`

La produzione di una rivista: ArsTEXnica

Massimiliano Dominici

Sommario

Nell'ultimo anno è stata intrapresa la riorganizzazione della rivista ArsTEXnica. Dopo la ristrutturazione dei compiti della redazione, è la volta degli strumenti di produzione ad essere oggetto di revisione.

Abstract

During the past year ArsTEXnica has gone through a process of reorganization. After the structure of editorial board it's now the turn of the build process to be revised.

1 Introduzione

Nell'ultimo numero di ArsTEXnica è stato annunciato un programma di rinnovamento della rivista (Beccari *et al.*, 2014) e ne sono state indicate le tappe:

1. ristrutturazione della redazione (già completata al momento dell'annuncio);
2. riscrittura degli strumenti per la produzione della rivista;
3. rivisitazione della veste grafica.

In questo articolo ci occuperemo del secondo punto che, benché ancora in fase di sperimentazione, può dirsi sostanzialmente compiuto. Il terzo punto è ancora in fase di elaborazione.

Nell'articolo citato venivano indicati i problemi del sistema di produzione fino allora impiegato e individuate alcune linee guida per ovviare ad essi. È opportuno riportare qui di seguito una sintesi di quell'esposizione.

1.1 Portabilità

Il primo obbiettivo che la redazione si è posta nel ridisegno del sistema di produzione della rivista è stato il massimo grado di portabilità e la facilità di installazione, possibilmente senza dover fare ricorso a niente altro che alla propria distribuzione di TEX.

Il vecchio sistema si basava su alcuni script per la shell `bash`. Era stato utilizzato su sistemi GNU/Linux e probabilmente era in grado di funzionare su sistemi operativi MacOS X e Windows in ambiente Cygwin. Un pregio di questo sistema consisteva nel fatto che, tranne per il programma `barcode` necessario per generare il codice a barre sulla quarta di copertina, le dipendenze erano ridotte alla presenza di `bash` e delle utilità di base dei sistemi Unix. Difetto principale: la necessità di installare l'ambiente Cygwin su Windows.

Nelle linee guida si affermava la necessità di sostituire agli script `bash` l'uso di un linguaggio di programmazione vero e proprio, con il duplice vantaggio di avere un sistema funzionante su ogni sistema operativo e un ambiente di programmazione più confortevole. I candidati erano stati individuati in Lua, Python, Ruby. Alla fine la scelta è caduta su Lua per le considerazioni fatte nel primo capoverso di questo paragrafo: una distribuzione TEX, infatti, contiene già un interprete Lua sotto forma del programma `texlua`. Rimane `barcode` come dipendenza esterna ed è quindi allo studio la possibilità di sostituirlo con il pacchetto pst-barcode.

1.2 Efficienza

Il vecchio script non usava un vero e proprio sistema di automatizzazione dello sviluppo. L'intero script veniva eseguito per intero ogni volta che veniva lanciato, ricompilando anche gli oggetti che non erano stati modificati. Nel caso dei singoli articoli la compilazione seguiva l'intero ciclo (`pdflatex`, `bibtex`, `pdflatex` di nuovo, due volte).

La soluzione individuata comportava l'uso di un sistema di automatizzazione dello sviluppo (scritto nello stesso linguaggio di programmazione dello script principale, ovvero `lake` per Lua, SCons per Python o `rake` per Ruby) abbinato a `latexmk` come programma di automazione dedicato per LATEX.

Successive sperimentazioni hanno condotto l'autore alla considerazione che `latexmk` era sufficiente agli scopi, senza ricorrere a un sistema di automazione generale. Infatti molti dei compiti che nello script originale venivano eseguiti ogni volta, potevano o essere eseguiti una sola volta in fase di inizializzazione della cartella di lavoro o essere separati e selezionati con una precisa opzione da passare allo script di compilazione. Questa decisione, inoltre, prosegue nella direzione di ridurre al minimo le dipendenze esterne al sistema TEX già installato.

1.3 Gestione dei conflitti

Nel vecchio sistema il corpo dei singoli articoli era incluso nel file principale (dopo che erano stati inclusi separatmente i pacchetti caricati e i comandi definiti dall'autore) e compilato all'interno di questo. Per quanto fosse stata posta una particolare attenzione nel tentare di gestire i possibili conflitti

tra articoli, nell'esperienza dei due incaricati della produzione che si sono susseguiti finora (Gianluca Pignalberi e l'autore di questo articolo), non c'è stata una singola uscita in cui non si presentassero incompatibilità, a volte evidenti fino al punto di bloccare la compilazione, a volte più sottili e subdole, introducendo errori di composizione non sempre immediati da individuare. La risoluzione di tali conflitti è stata spesso laboriosa, sempre fastidiosa e costosa in termini di tempo speso nella fase di debug. Nei casi più insidiosi, tale risoluzione non c'è proprio stata, perché l'errore è stato scoperto troppo tardi.

È stato deciso che la soluzione migliore sarebbe stata compilare i singoli articoli separatamente e includerli in un secondo tempo come PDF. Questa soluzione, ispirata da un articolo di Thierry Bouche su TUGboat (Bouche, 2006), era già stata presa in considerazione anni fa e poi scartata perché nel processo si perdevano irrimediabilmente i collegamenti ipertestuali. Fortunatamente nel frattempo è comparso il pacchetto pax (Oberdiek, 2012) che permette, tramite uno script Perl (`pdfannotextractor`), di salvare e reinserire i collegamenti ipertestuali.

Se da una parte l'uso di pax semplifica notevolmente la gestione degli articoli, dall'altra introduce un paio di problemi. Innanzitutto pax funziona solo con `pdflatex` (o, usando qualche accorgimento nel preambolo, con `lualatex`) ma non con `xelatex`. Questa limitazione, però, è valida solo per il file che deve includere i PDF, non per i singoli PDF da includere, che possono essere prodotti con qualsiasi programma (anche non appartenente al sistema TEX), quindi di fatto non ha alcuna incidenza nel nostro caso. Il secondo inconveniente è che vengono introdotte alcune dipendenze: Perl, Java e una precisa versione della libreria PDFBox. Inoltre potrebbe essere necessario attuare qualche aggiustamento per l'uso su Windows.

Compilare autonomamente i singoli articoli e poi includerli come PDF comporta vantaggi e svantaggi. Ogni problema di compilazione derivante da conflitti nell'uso di pacchetti o nella (ri)definizione di comandi è sostanzialmente eliminato. E questo è un vantaggio. Tuttavia, poiché l'autore virtualmente è libero di fare ciò che vuole senza che questo incida nella compilazione, c'è il rischio, che comunque si correva anche in passato, di introdurre incongruenze stilistiche. Sarà l'esperienza a decidere sull'opportunità di introdurre eventualmente delle "norme editoriali" a cui autori, redattori e revisori di bozze possano fare riferimento.

2 Descrizione del processo di produzione

Come premessa a questo paragrafo va fatto notare che il sistema è ancora in fase di sviluppo e di sperimentazione, sebbene possa reputarsi abbastanza stabile nella sua architettura. Il funzionamento e il nome di alcune macro e funzioni, però, potrebbe essere soggetto a modifiche, in particolare per quanto riguarda la gestione dei metadati.

Nel paragarafo verranno esaminati solo i punti salienti del processo, senza dare una descrizione dettagliata del funzionamento generale della classe.

Il nuovo sistema di produzione è composto da due script `lua` (`AT_new_issue.lua` e `AT_build.lua`), la classe arstexnica e una serie di template per il file master e per gli elementi fissi della rivista: pagine di copertina, colophon, eventualmente altro materiale, se con il tempo si renderà necessario.

2.1 Preparazione di un nuovo numero

Ogni nuova uscita viene inizializzata tramite lo script `AT_new_issue.lua` (riportato nelle figure 1 e 2), corrispondente grosso modo al vecchio `AT_nuovo_numero`. A differenza del vecchio script, che leggeva i dati relativi al nuovo numero a partire da un file di configurazione, al nuovo script tali dati devono essere passati tramite opzioni in fase di compilazione: questo garantisce che il sistema funzioni anche in assenza del file di configurazione. Tuttavia è in fase di aggiunta anche la vecchia funzionalità, molto comoda ogni volta che si vuole semplicemente inizializzare il numero successivo all'ultimo compilato.

Un tipico uso dello script è il seguente:

```
texlua AT_new_issue --year 2014 --issue 18
```

oppure, usando l'alternativa breve per i nomi delle opzioni:

```
texlua AT_new_issue -y 2014 -i 18
```

Le opzioni sono dichiarate tramite la libreria Lua `alt_getopt`, distribuita con TEX Live.

Lo script compie le seguenti operazioni:

1. crea una cartella usando il numero della rivista come parte del nome (nel nostro esempio `Numero_18`);
2. crea una struttura all'interno di questa cartella, con sottocartelle preposte a contenere gli elementi del numero da compilare (articoli, materiale aggiuntivo, sommari, voci bibliografiche, PDF compilati);
3. copia nella struttura così ottenuta i template del file principale e delle varie pagine ausiliarie (copertina e colophon);
4. copia un albero TDS con i file `latex/bibtex` necessari per la compilazione della rivista[1];

1. A rigor di logica non sarebbe necessario, perché tutti i numeri potrebbero attingere a un unico albero centrale, ma non ho ancora deciso se è meglio questa soluzione o assicurarsi che per ogni uscita sia presente la precisa configu-

```
kpse.set_program_name('luatex')

require "alt_getopt"

-- Source (adapted):
-- http://kracekumar.com/post/53685731325/cp-command-implementation-and-benchmark-in-python-go
function cp(source, dest)
    -- body
   lfs.mkdir(dest)
   for filename in lfs.dir(source) do
      if filename ~= '.' and filename ~= '..' then
 local source_path = source .. '/' .. filename
 local attr = lfs.attributes(source_path)
 -- print(attr.mode, path)
 if type(attr) == "table" and attr.mode == "directory" then
    local dest_path = dest .. "/" .. filename
    -- print(dest_path)
    lfs.mkdir(dest_path)
    -- print(lfs.mkdir(dest_path))
    cp(source_path, dest_path)
 else
    local f = io.open(source_path, "rb")
    local content = f:read("*all")
    f:close()
    local w = io.open(dest .. '/' .. filename, "wb")
    -- print(dest .. '/' .. filename, w, f)
    w:write(content)
    w:close()
 end
      end
   end
end

local long_opts = {
   year = "y",
   issue = "i",
}

local short_opts = "y:i:"

local anno_uscita, mese_uscita, numero_uscita

local texmf = "texmf-ars"

optarg,optind = alt_getopt.get_opts (arg, short_opts, long_opts)
for k,v in pairs (optarg) do
   if k == "y" then
      anno_uscita = v
   end
   if k == "i" then
      numero_uscita = v
   end
end

if tonumber(numero_uscita) % 2 == 0 then
   mese_uscita = "Ottobre"
else
   mese_uscita = "Aprile"
end

local nuova_uscita = "Numero_" .. numero_uscita
local config_file = nuova_uscita .. "/arstexnica.cfg"
```

FIGURA 1: `AT_new_issue`, lo script che prepara un nuovo numero (prima parte).

```
local issn_online = 18282369
local issn_stampa = 18282350

local function genera_ean(issn)
   local s1 = string.sub(issn, 1, -2) .. "00"
   local s2 = string.sub(anno_uscita, -1) .. string.format("%04d", numero_uscita)
   local ean = "977" .. s1 .. " " .. s2
   return ean
end

lfs.mkdir(nuova_uscita)
lfs.mkdir(nuova_uscita .. "/articoli")
lfs.mkdir(nuova_uscita .. "/output")
lfs.mkdir(nuova_uscita .. "/output/pdf")
lfs.mkdir(nuova_uscita .. "/output/bib")
lfs.mkdir(nuova_uscita .. "/output/abstracts")
cp("repo/" .. texmf, nuova_uscita .. "/" .. texmf)
local cfg = io.open(config_file, 'w')
cfg:write([[\ATsetup{]] .. "\n")
cfg:write("  year = " .. anno_uscita .. ",\n")
cfg:write("  month = " .. mese_uscita .. ",\n")
cfg:write("  number = " .. numero_uscita .. ",\n")
cfg:write("  mode = online,\n")
cfg:write([[}]])
cfg:close()

print(genera_ean(issn_online))
print(genera_ean(issn_stampa))

os.execute("barcode -E -e ean -u cm -g 4x1.5 -b '" .. genera_ean(issn_online)
           .. "' -o " ..nuova_uscita .. "/issn_online.eps")
os.execute("barcode -E -e ean -u cm -g 4x1.5 -b '" .. genera_ean(issn_stampa)
           .. "' -o " ..nuova_uscita .. "/issn_print.eps")
```

Figura 2: `AT_new_issue`, lo script che prepara un nuovo numero (seconda parte).

5. scrive in un file di configurazione i dati del numero da compilare (anno, mese, numero e modalità di compilazione online/stampa);

6. genera il codice a barre della versione online e di quella a stampa e le copia nella cartella di lavoro.

2.2 La compilazione

La compilazione avviene tramite lo script Lua `AT_build.lua`. Lo script, oltre a impostare il percorso in cui il compilatore potrà trovare i file della classe, consente di scegliere tra la compilazione della versione online e di quella a stampa.

Il responsabile dell'impaginazione dovrà collocare, man mano che gli arrivano, gli articoli nell'apposita cartella (fantasiosamente denominata `articoli`), ciascuno nella propria sottocartella. Questi verranno poi inclusi nel file principale tramite il comando `\IncludeArticle` che prende come unico argomento il percorso del file principale dell'articolo (senza estensione) relativo alla cartella `articoli`. Per esempio:

```
\IncludeArticle{Editoriale/editoriale18}
```

Le pagine ausiliarie vanno incluse con il comando `\IncludeSuppl`, sostanzialmente una copia del precedente che esegue qualche compito in meno, ma in compenso permette di specificare, tramite un argomento opzionale, una particolare modalità di compilazione:

```
\IncludeSuppl[prima]{copertina/prima}
```

dove l'argomento opzionale indica che la pagina in questione deve essere compilata in modalità "prima di copertina", attivando quindi una serie di funzionalità che di solito sono inibite e viceversa.

Queste due macro fanno internamente entrambe riferimento alla macro `\AT@build@pdf` che rappresenta il fulcro del sistema di compilazione. Quando, infatti, il compilatore raggiunge questa istruzione, se lanciato con l'opzione `-shell-escape`, entra nella modalità di esecuzione di comandi esterni, si porta nella cartella del file da compilare, lancia `latexmk` e immediatamente dopo

razione con cui è stata compilata. In realtà, per assicurarsi di una cosa del genere sarebbe necessario disporre della precisa versione di tutti i pacchetti usati in quella determinata occasione.

```
kpse.set_program_name('luatex')

local loader_file = "luatexbase.loader.lua"
local loader_path = assert(kpse.find_file(loader_file, "lua"),
                           "File '"..loader_file.."' not found")
require(loader_path)
require "alt_getopt"

local long_opts = {
   mode = "m",
}

local short_opts = "m:"
local prev_mode = ""

local cfg = io.open("arstexnica.cfg", "r")
local lines = {}
for line in cfg:lines() do
   if(string.find(line, "mode")) then
      if (string.find(line, "online")) then
 prev_mode = "online"
      else
 prev_mode = "print"
      end
      break
   end
end

local mode = prev_mode

optarg,optind = alt_getopt.get_opts (arg, short_opts, long_opts)
for k,v in pairs (optarg) do
   if k == "m" then
      mode = v
   end
end

if prev_mode ~= mode then
   local cfg = io.open("arstexnica.cfg", "r")
   local lines = {}
   local restOfFile
   for line in cfg:lines() do
      if(string.find(line, prev_mode)) then
 lines[#lines + 1] = string.gsub(line, prev_mode, mode)
 restOfFile = cfg:read("*a")
 break
      else
 lines[#lines + 1] = line
      end
   end
   cfg:close()

   local cfg = io.open("arstexnica.cfg", "w")
   for i, line in ipairs(lines) do
      cfg:write(line, "\n")
   end
   cfg:write(restOfFile)
   cfg:close()
end


os.execute("export TEXMFHOME=`pwd`/texmf-ars:`kpsewhich -var-value=TEXMFHOME`" ..
      " && echo $TEXMFHOME && " ..
      "pdflatex -shell-escape arstexnica.tex")
```

FIGURA 3: `AT_build`, lo script di compilazione.

```
\newcommand\AT@build@pdf[2][]{%
  \AT@divide@path#2|
  \AT@build@hook
  \immediate\write18{cd \AT@subdirectory/\AT@article@path &&
                     latexmk -pdf -pdflatex='pdflatex -shell-escape
                        '\%O' \string\\PassOptionsToClass{#1}{arstexnica}\string\\input{'\%S'}'
                     \AT@article@name &&
                     pdfannotextractor \AT@article@name.pdf}%
}
%
\newcommand\IncludeSuppl[2][]{%
  \def\AT@build@hook{}
  \def\AT@subdirectory{suppl}
  \AT@build@pdf[#1]{#2}
  \IfFileExists{\AT@subdirectory/#2.pdf}{%
    \includepdf[pages=-, pagecommand={},]
                {\AT@subdirectory/#2.pdf}
  }{\typeout{Articolo #2: Qualcosa \'e andato storto...}}
}
%
\newcommand\IncludeArticle[1]{%
  \def\AT@build@hook{
    \immediate\write\AT@matedata{["\AT@article@name"]=\AT@open@group}
    \immediate\write\AT@matedata{["startpage"]=\thepage,}
  }
  \def\AT@subdirectory{articoli}
  \AT@build@pdf[injournal]{#1}
  \xdef\@currentHref{title.\theHtitle}%
  \makeatletter
  \InputIfFileExists{\AT@subdirectory/#1.lst}{%
    \typeout{Carico articolo #1}%
  }{\typeout{Articolo #1: Qualcosa \'e andato storto...}}
  \makeatother
  \thispagestyle{plain}
  \IfFileExists{\AT@subdirectory/#1.pdf}{%
    \includepdf[pages=-, pagecommand={},
                addtotoc={1,ATtitle,-1,\AT@toc@header,ATtitle-\theATtitle}]
                {\AT@subdirectory/#1.pdf}
  }{\typeout{Articolo #1: Qualcosa \'e andato storto...}}
```

Figura 4: Le macro usate per la compilazione e l'inclusione dei singoli articoli.

`pdfannotextractor`. Al termine di queste operazioni `\IncludeArticle` o `\IncludeSuppl` includono il PDF ottenuto, se tutto è andato a buon fine, usando la macro `\includepdf` del pacchetto pdfpages (Matthias, 2013). Nel caso in cui sia necessario anche generare una voce per l'indice, questa viene specificata tramite l'opzione `addtotoc` di `\includepdf`.

È da notare che `latexmk` entra in funzione realmente solo se il file in questione è stato modificato (o un altro file da cui esso dipende, per esempio la classe stessa). Se non ci sono state modifiche avviene soltanto l'inclusione del PDF con un notevole risparmio di compilazioni. Il codice delle tre macro in questione è riportato nella figura 4.

Questo sistema però comporta un problema. Il PDF così compilato contiene le proprie testatine e i piè di pagina. Teoricamente sarebbe possibile comunicare al momento della compilazione il numero di pagina iniziale (e i dati per le testatine), ma questo vanificherebbe un po' l'uso di `latexmk` perché costringerebbe a ricompilare porzioni del documento che, a parte il numero di pagina, non hanno subito modifiche. Molto più semplice fare in modo che i singoli articoli siano compilati con testatine e piè di pagina in bianco e questi siano riapplicati poi dal documento generale. A questo scopo, in fase di compilazione del singolo articolo viene passata l'opzione injournal (che attiva anche il codice per scrivere i metadati) e poi, al momento dell'inclusione, viene disattivata l'opzione di `\includepdf` che inibisce la scrittura di testatina e piè di pagina. Ovvero riportiamo il controllo di questi due elementi al documento generale togliendolo a quello incluso.

2.3 Metadati

ArsTEXnica non è prodotta solo ed esclusivamente per la stampa, ma anche per la diffusione online tramite il sito web del Gruppo Utilizzatori Italiani

di TEX, dove compaiono i dati principali dei singoli articoli (titoli, autori, sommari) e un collegamento ai PDF di ciascun articolo e dell'intera rivista. Inoltre lo stesso sito distribuisce un database bibliografico degli articoli comparsi sulla rivista e periodicamente vengono inviati alla redazione di TUGboat i dati del contenuto della rivista per la pubblicazione dei sommari.

Tutto ciò fa sì che sia estremamente conveniente avere un sistema che, almeno parzialmente, estragga questi dati automaticamente. Il vecchio script lo faceva in maniera abbastanza efficiente, ma alcune cose andavano ripensate. In particolare sarebbe bene immagazzinare i metadati in un formato che ne permetta usi differenti. Per fare un esempio, con il vecchio script i nomi degli autori erano disponibili solo nel file `bibtex`, scritto direttamente a partire da istruzioni della classe arstexnica e non potevano essere usati direttamente per la creazione della pagina web.

Nel nuovo sistema, che può giovarsi di Lua come linguaggio di programmazione, è stato naturale organizzare questi dati in una tabella. Così sarà possibile scrivere funzioni ausiliarie che accedano a questa tabella e usino i dati per i loro scopi.

Le funzioni di scrittura dei metadati usano le primitive di TEX per la gestione di input e output (o le macro del nucleo di LATEX che su di esse si basano). Di solito questo non comporta grossi problemi, ma nel nostro caso c'è un ostacolo: una parte dei dati da scrivere non è immediatamente a disposizione del documento generale, ma solo dei singoli articoli che non comunicano direttamente con il processo di compilazione principale. La soluzione trovata è stata quella di fare in modo che i singoli articoli scrivano su un file esterno la propria porzione di dati; questo file è poi letto dal processo principale di compilazione e i dati sono inseriti nella tabella generale dei metadati.

2.4 Compiti ausiliari

Lo script di compilazione conterrà anche delle funzioni ausiliarie per eseguire compiti di manipolazione dei metadati (creazione delle voci bibliografiche, creazione di un file che contenga tutti i sommari pronti per la consegna al TUGboat, generazione di frammenti di HTML per il sito web) e riorganizzazione dei prodotti di compilazione (copia dei PDF in una apposita cartella). Queste funzioni non sono state ancora implementate, ma esistono già le fondamenta per poterlo fare in maniera abbastanza rapida e semplice.

3 Sviluppi futuri

La costruzione del nuovo sistema di produzione di ArsTEXnica è giunta a buon punto, ma non è completata, infatti il numero di ArsTEXnica che state leggendo è ancora prodotto con il vecchio sistema. Probabilmente sarà l'ultimo. In generale sul versante degli script Lua c'è ancora da lavorare.

- Alcune funzioni accessorie del vecchio script non sono state ancora reimplementate, altre andranno probabilmente riviste.
- Mancano ancora i dovuti controlli in caso di corruzione del file dei metadati.
- Deve essere possibile indicare a `latexmk` che la compilazione va effettuata anche se tutti i file sono aggiornati (flag `-g`) e possibilmente a livello di singolo articolo oltre che globalmente.

Sul versante della classe, invece, un elenco parziale dei compiti da eseguire è il seguente.

- Rendere personalizzabile la stringa che controlla la compilazione dei singoli articoli (permettendo ad esempio di cambiare il motore di composizione o le altre opzioni passate a `latexmk`).
- Ripulire e riorganizzare il codice, tenendo conto anche dell'obbiettivo finale di rendere disponibile su CTAN (e quindi eventualmente su TEX Live e MikTEX) almeno la porzione dedicata agli autori. Questo comporta che si rifletta su quale grado di modularità conferire al codice nello strutturare il `dtx`.

Nei prossimi mesi, quindi, oltre a completare le parti mancanti e a testare il sistema in maniera intensiva, si cercherà di individuare nuove funzionalità da aggiungere e soprattutto si comincerà a rivedere il layout della rivista, a partire dal formato. Una volta prese le debite decisioni su come organizzare il progetto, lo sviluppo del codice avverrà sul repository Github del GUIT: `https://github.com/GuITeX`.

Riferimenti bibliografici

Beccari, C., Dominici, M. e Pignalberi, G. (2014). «Ars si rinnova». *ArsTEXnica*, (17), pp. 12–17.

Bouche, T. (2006). «A pdfLATEX-based automated journal production system». *TUGboat*, **27** (1), pp. 45–50. Proceedings of EuroTEX 2006.

Matthias, A. (2013). *The pdfpages Package*. TUG. Leggibile con `texdoc pdfpages` nella distribuzione TEX Live.

Oberdiek, H. (2012). *README for project pax*. TUG. Leggibile con `texdoc pax` nella distribuzione TEX Live.

▷ Massimiliano Dominici
`mlgdominici at gmail dot com`

Questa rivista è stata stampata
presso Centro Stampa e Riproduzione S.r.l.,
Via di Pietralata, 157 – 00158 Roma,
su carta vellum white 80 g.
Copertina: vellum white 260 g.

ArsTEXnica – Call for Paper

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 entro e non oltre il 14 Marzo 2015. 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 d'esempio (.tex).

Gli articoli potranno trattare di qualsiasi argomento inerente al mondo di LATEX e non dovranno necessariamente essere indirizzati ad un pubblico esperto. In particolare tutorial, 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 `http://www.guitex.org/arstexnica`, 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@guitex.org`.

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 18, Ottobre 2014

40018
9 771828 236001