Numero 2
Ottobre
2006

ArsTEXnica

Rivista italiana di TEX e LATEX

Atti del GUIT *meeting 2006*

GUIT

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

GUIT – Gruppo Utilizzatori Italiani di TEX

ArsTEXnica è la pubblicazione ufficiale del GUIT

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

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

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

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

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

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

`arstexnica@sssup.it`.

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 un quota di iscrizione annuale di 25,00 € nel caso di persone fisiche o di 50,00 € nel caso di Enti o Istituzioni.

Indirizzi

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

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

Codice ISSN 1828-2369

Stampata in Italia
Pisa: 21 Ottobre 2006

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 2, Ottobre 2006

Gruppo Utilizzatori Italiani di TEX

Editoriale

Massimiliano Dominici, Maurizio W. Himmelmann

Questo secondo numero di ArsTEXnica è già un numero speciale. Contiene infatti gli Atti del Terzo Convegno Annuale del GUIT (GUIT *meeting 2006*), che si svolge a Pisa, come nelle precedenti occasioni, il 21 Ottobre 2006, presso l'Aula Magna della Scuola Superiore Sant'Anna. Gli interventi, tutti di ottimo livello, sono rappresentativi delle numerose aree di impiego di TEX e sono indirizzati a differenti tipologie di utente.

Dopo il discorso introduttivo di Lance Carnes, direttore del PracTEX Journal, la mattina è dedicata ai temi più tradizionali. Enrico Gregorio espone, nel suo intervento, un esempio di utilizzo dei codici di categoria, allo scopo di ottenere espressioni di logica formale da un codice sorgente leggibile e intuitivo. L'intervento di Gustavo Cevolani è una rassegna dei numerosi metodi con cui è possibile produrre un libretto, un opuscolo, o un pieghevole, impiegando pacchetti di LATEX appositamente creati o uno dei programmi ausiliari che ne accompagnano la distribuzione. Lapo Mori offre una descrizione dettagliata dei problemi che si possono incontrare dovendo creare delle tabelle e delle soluzioni fornite dai numerosi pacchetti che trattano l'argomento. Sempre nel corso della mattinata, Kaveh Bazargan, della River Valley Technologies, discute su come sia possibile ottenere con TEX un allineamento tipografico alla griglia, modificandone il comportamento normale. La sessione mattutina si conclude con l'intervento di Jean-Michel Hufflen che propone la sua reimplementazione di BibTEX, MlBibTEX, che ha lo scopo di migliorarne le capacità di lavorare in un ambiente multilinguistico.

Nel pomeriggio sono previsti interventi che mettono in luce come LATEX possa essere utilizzato anche al di fuori degli ambiti tradizionali di impiego. Onofrio de Bari presenta un agile *tutorial* sull'uso di GNU Emacs per LATEX. Gli interventi di Salvatore Palma e Jerónimo Leal, invece, propongono due differenti esperienze didattiche in cui LATEX ha una parte rilevante. Nel caso di Salvatore Palma si tratta dell'uso di LATEX per produrre test interattivi di matematica in una scuola media superiore. Jerónimo Leal dà conto di un corso su edizioni critiche basato su LATEX. Gli ultimi due interventi sono dedicati al Progetto Maurolico, finalizzato all'edizione critica delle opere di Francesco Maurolico. Pier Daniele Napolitani, direttore del progetto, e Massimiliano Dominici forniscono un'introduzione generale al MAURO-TEX (il linguaggio sviluppato nell'ambito di questo progetto) e una panoramica dei suoi pregi, limiti e sviluppi futuri. Roberta Tucci descrive invece la propria esperienza concreta con MAURO-TEX per l'edizione critica di un'opera matematica.

Segue il discorso di benvenuto del Presidente del Gruppo Utilizzatori Italiani di TEX, Maurizio Himmelmann.

Discorso di benvenuto

Il terzo meeting si svolge in un periodo di grande fermento del Gruppo. Grazie a tutte le attività messe in essere in questi anni sta adesso recuperando velocemente terreno rispetto agli altri TUG europei: la costituzione di GUIT come società legalmente riconosciuta, la pubblicazione della rivista ArsTEXnica, l'organizzazione periodica di convegni sono tutti frutti del lavoro appassionato svolto in questi anni, ben testimoniato anche dal crescente numero di persone che stanno decidendo di aderire a quest'iniziativa.

In questo contesto il meeting annuale costituisce da sempre un giorno speciale, essendo l'occasione per uscire dall'anonimato della rete ed incontrarci finalmente di persona per discutere insieme di LATEX, del suo utilizzo e del suo futuro. Il livello scientifico oggi proposto è altissimo potendo vantare relatori di prestigio internazionale. Ringrazio quindi Lance Carnes direttore del PracTEX Journal e Kaveh Bazargan di River Valley Technologies, grazie al cui supporto tutti gli interventi saranno riproducibili in podcasting. Un ringraziamento anche a tutto lo staff del Comitato Organizzatore, insieme al quale abbiamo messo da parte famiglie ed impegni personali pur di fare in modo che l'organizzazione fosse all'altezza della situazione.

Infine, un ringraziamento speciale va al Prof. Giulio Bottazzi del Laboratorio di Economia e Management della Scuola Superiore Sant'Anna senza il cui supporto questa giornata non sarebbe stata possibile.

È con tutte queste premesse che vi porgo il benvenuto al GUIT *meeting 2006*.

▷ Massimiliano Dominici
Direttore di ArsTEXnica
mlgdominici@interfree.it

▷ Maurizio W. Himmelmann
Presidente del Gruppo
Utilizzatori Italiani di TEX
himmel@sssup.it

Programma del Convegno

Sessione Mattutina

09:00 – **Registrazione**

09:30 – **PracTEX: Increasing TEX's user base**
Lance Carnes, PracTEX Journal

10:00 – **Codici di categoria**
Enrico Gregorio, Università di Verona

10:30 – **Libretti in LATEX**
Gustavo Cevolani, Università di Bologna

11:00 – **Coffee break**

11:30 – **Tabelle su LATEX – pacchetti e metodi da utilizzare**
Lapo Mori, Northwestern University

12:00 – **Removing vertical stretch: mimicking traditional typesetting with TEX**
Kaveh Bazargan, River Valley Technologies

12:30 – **MlBibTEX's architecture**
Jean-Michel Hufflen, Université de Franche-Comté

13:00 – **Pausa Pranzo**

Sessione Pomeridiana

15:00 – **GNU Emacs e AUCTEX per LATEX**
Onofrio de Bari

15:30 – **Test interattivi di Matematica e Fisica on-line: il LATEX come strumento di sviluppo**
Salvatore Palma, Liceo Scientifico A. Volta

16:00 – **Esperienze didattiche**
Jerónimo Leal, Pontificia Università della Santa Croce

16:30 – **Coffee break**

17:00 – **Edizione con LATEX delle opere di Francesco Maurolico**
Massimiliano Dominici
Pier Daniele Napolitani, Università di Pisa

17:30 – **Un pacchetto LATEX per l'edizione critica di in'opera matematica**
Roberta Tucci, Università di Pisa

18:00 – **LATEX Help Desk (esperti di LATEX a vostra disposizione) e key signing party**

18:30 – **Chiusura dei lavori**

Codici di categoria

Enrico Gregorio

Sunto. TEX lavora con liste di *token* che forma via via che legge i caratteri da un documento `.tex`. Capire la procedura di *divisione in token* è importante se si vuole modificare il comportamento usuale di TEX. Sotto questo aspetto la nozione di *codice di categoria* assegnato a un carattere è fondamentale.

Di particolare interesse sono i *caratteri attivi*. Ne darò un'applicazione che sfrutta alcune delle nuove funzionalità di ε-TEX.

Abstract. TEX works with token lists formed when it reads characters from a `.tex` document. Understanding the *tokenization* procedure is important if one wants to modify the usual behavior of TEX. Under this respect the notion of *category code* attached to a character is fundamental.

Chiefly interesting are the *active characters*. I will give an application of them, which exploits some of the new features of ε-TEX.

Introduzione

Quando si lancia la composizione di un documento, all'interno di TEX arrivano solo *token*, che potremmo pensare come unità indivisibili dotate di significato. Ciascun *token* può essere sviluppabile o no; a un certo livello del programma, i *token* del primo tipo sono sviluppati e questa operazione produce una nuova lista di *token*, che a sua volta viene sottoposta al procedimento in modo ricorsivo, finché rimangono solo *token* non sviluppabili che quindi raggiungono il livello del programma dove ha luogo la composizione tipografica vera e propria.

Da un altro punto di vista, i *token* possono essere di due tipi: *caratteri* e *sequenze di controllo*, che chiamerò per brevità *simbolici*. Vediamo l'esempio classico, lo sviluppo del comando `\TeX`:

```
T\kern-.1667em\lower.5ex
  \hbox{E}\kern-.125emX\@
```

è una lista formata da 29 *token*, 24 caratteri e 5 simbolici. Una differenza fondamentale fra il primo tipo e il secondo è che i caratteri hanno un significato immutabile, una volta che raggiungono l'interno di TEX, mentre l'interpretazione dei *token* simbolici può cambiare anche dopo (ridefinendo un comando prima che sia sviluppato).

In un primo momento, questa analisi può essere sufficiente. Quando però si deve andare più in profondità, ci si accorge che la verità è un po' più complessa.

1 Codici di categoria

Analizzerò qui i *token* di tipo carattere; occorre ricordare che all'interno di TEX arriva prima di tutto un flusso di caratteri che viene sottoposto alla *divisione in token*. Per esempio, una riga nel `.tex` come

```
\author{E. Gregorio}
```

è formata da 20 caratteri; quando il flusso di caratteri viene immesso in TEX, è trasformato in 14 *token*:

```
\author • { • E • . • ␣ •
 G • r • e • g • o • r • i • o • }
```

(il simbolo • serve solo a separare visivamente un *token* dall'altro).

Che cosa permette di riunire i primi sette caratteri in un unico *token*? Per capirlo occorre dire che ogni carattere che entra nel procedimento di divisione in *token* è visto come una coppia di numeri[1]

(*carattere*,*categoria*)

dove *carattere* è un intero fra 0 e 255, mentre *categoria* è un intero fra 0 e 15. Nella situazione usuale, i caratteri precedenti sono visti come le coppie

$(92, 0)$, $(97, 11)$, $(117, 11)$, $(116, 11)$, $(104, 11)$, $(111, 11)$, $(114, 11)$, $(123, 1)$, $(69, 11)$, $(46, 12)$, $(32, 10)$, $(71, 11)$, $(114, 11)$, $(101, 11)$, $(103, 11)$, $(111, 11)$, $(114, 11)$, $(105, 11)$, $(111, 11)$, $(125, 2)$.

La trasformazione dei sette caratteri iniziali in un unico *token* dipende proprio dal fatto che il primo ha categoria 0. Non starò qui a specificare la regola precisa; basta dire che sono possibili due casi, quando TEX incontra un carattere di categoria 0:

1. se il carattere che segue non ha categoria 11, il *token* è formato dal carattere di categoria 0 e da quel carattere seguente;

2. se il carattere che segue ha categoria 11, il *token* è formato dal carattere di categoria 0 e da tutti i caratteri consecutivi di categoria 11 successivi.

1. TEX usa il codice ASCII per i caratteri da 0 a 127; sui caratteri da 128 a 255 non fa supposizioni, è questo che rende possibile usare diverse codifiche tramite il pacchetto inputenc.

La categoria assegnata a un carattere non è fissata e può essere modificata, ma solo fino a quando il carattere stesso non è ancora entrato nel meccanismo di lettura; dunque possiamo solo modificare la categoria dei caratteri che *entreranno*.

A che serve la categoria? A dire la *funzione* di ciascun carattere. Per esempio, secondo la convenzione usuale, `B` ha categoria 11, cioè la sua funzione è di rappresentare un carattere alfabetico che, se raggiunge la profondità ultima di TEX, produrrà una bella 'B' nel `.dvi`. Invece `{` ha categoria 1, cioè serve da delimitatore di apertura dei gruppi o degli argomenti. Ecco la lista completa; dopo il nome c'è un esempio, secondo le convenzioni usuali:

0: carattere di *escape* (\);

1: inizio di gruppo (`{`);

2: fine di gruppo (`}`);

3: inizio o fine di formula (`$`);

4: punto di allineamento (`&`);

5: fine riga;

6: parametro (`#`);

7: esponente (`^`);

8: pedice (`_`);

9: carattere da ignorare;

10: spazio;

11: lettera (`a`);

12: carattere 'altro' (`?`);

13: carattere attivo (`~`);

14: commento (`%`);

15: carattere non valido.

Che significa 'altro'? Che non appartiene a nessuno degli altri gruppi. Un carattere di categoria 9 non oltrepassa la fase di lettura dei caratteri; un carattere di categoria 14 esclude dalla lettura tutti i caratteri che seguono fino al primo di categoria 5 (compreso). Un carattere non valido produce un messaggio di errore e viene poi ignorato.

Un carattere di categoria 5 viene convertito in uno di categoria 10 oppure in un *token* `\par` secondo certe regole non facilissime; è la procedura che ci permette di andare a capo come ci pare e di lasciare una riga vuota per indicare la fine di un paragrafo[2].

Un carattere di categoria 10 (usualmente lo spazio e il carattere di tabulazione) può essere ignorato in certe situazioni, per esempio dopo un nome di comando oppure a inizio riga. Altrimenti funge da spazio, come è ovvio. In tutti i casi, durante la lettura, caratteri consecutivi di categoria 10 vengono ridotti a uno solo.

Un carattere attivo ha uno stato molto particolare: equivale a un comando che verrà sviluppato secondo le stesse regole dei *token* simbolici, quando non segue immediatamente un singolo carattere di categoria 0, dove TEX sta cercando un nome di comando; la stessa osservazione vale per tutti gli altri codici. Perciò `~` è un comando, così come `\~`.

Il modo di assegnare un codice di categoria a un carattere è

`\catcode`⟨*numero a 8 bit*⟩`=`⟨*numero a 4 bit*⟩

Per esempio, `plain.tex` comincia assegnando il codice 1 alla graffa aperta e il codice 2 alla graffa chiusa[3], ma lo fa in modo curioso:

```
\catcode`\{=1
\catcode`\}=2
```

Il ⟨*numero a 8 bit*⟩ può essere specificato: (1) come numero in notazione decimale (per esempio 123); (2) come numero in notazione ottale (per esempio `'173`); (3) come numero in notazione esadecimale (per esempio `"7B`); (4) come carattere ASCII (per esempio `` `\{ ``). L'ultimo modo è quello più pratico in questo caso: si scrive il carattere "accento grave", la barra rovescia e il carattere corrispondente, così non c'è da ricordare qual è il suo numero nel codice ASCII (meglio però non giocare con i caratteri da 128 a 255, per via delle differenti codifiche). Non è obbligatorio precedere il carattere con la barra rovescia, se questo carattere ha categoria 11 o 12.

Molto spesso si trova, spulciando i *forum* di discussione su TEX, il comando `\makeatletter` e il corrispondente `\makeatother`. Il loro sviluppo è

```
\catcode`\@=11
\catcode`\@=12
```

rispettivamente; è il primo comando che permette di usare `@` come se fosse una lettera (categoria 11, per essere precisi) durante la divisione in *token*. Quando un comando il cui nome contiene un `@` di categoria 11 è entrato nel meccanismo di lettura, il suo codice è immutabile. Perciò, anche dopo aver dato `\makeatother` il richiamo a quel comando sarà interpretato nel modo corretto.

Preciso meglio. Supponiamo di scrivere

```
\makeatletter
\newcommand{\@xxx}{xxx}
\newcommand{\xxx}{\@xxx}
\makeatother
```

2. Il sistema operativo dice a TEX dov'è la fine di una riga; a questo punto TEX ignora l'eventuale carattere che la indica e lo sostituisce con un carattere di categoria 5.

3. Occorre dire che iniTEX ha incorporato il codice usuale per la barra rovescia, per il 'per cento' e per i caratteri alfabetici e numerici.

Lo sviluppo del *token* `\xxx` è allora `\@xxx`, che a sua volta viene sviluppato come i tre *token* `xxx`, indipendentemente dal codice di categoria del carattere `@` al momento in cui `\xxx` viene sviluppato. Tuttavia, quando `@` ha categoria 12 non ci si può riferire direttamente al comando `\@xxx`.

2 Caratteri attivi

In qualche punto di `latex.ltx` si trova[4]

```
\catcode`\~=13
```

e, più avanti, si legge

```
\def~{\leavevmode\penalty10000 \ }
```

Come si vede un carattere attivo può essere definito allo stesso modo di un comando. L'idea è di (1) cominciare un paragrafo, se per caso si è in 'modo verticale'; (2) inibire la possibilità di spezzare una riga; (3) lasciare uno spazio.

I caratteri attivi sono molto usati da babel, per varie abbreviazioni. Bisogna usare una certa cautela, perché se scriviamo

```
\catcode`\a=13
```

non possiamo più usare nomi di comando che contengano la lettera 'a'. E non potremmo nemmeno tornare indietro, visto che il comando per l'assegnazione del codice di categoria contiene la 'a'!

Naturalmente le assegnazioni di categoria rispettano i gruppi; perciò se scriviamo

```
{\catcode`\a=13 \gdefa{xxx}}
```

dopo la fine del gruppo il carattere `a` ha ancora categoria 11. Nel gruppo abbiamo definito *globalmente* il comando `a` (quando il carattere è attivo). La definizione rimane silente, fino a che TEX incontra un carattere `a` attivo; in questo caso lo svilupperà secondo le solite regole. Faccio il solito esempio stupido.

```
\def\restorecc{\catcode`\a=11 }
{\catcode`\a=13 \gdefa{\restorecc}}
```

Quando usiamo `\restorecc`, il codice di categoria dei caratteri che compongono il corpo della definizione è già fissato. Perciò scrivendo

```
\catcode`\a=13 a
```

otterremo che il codice di categoria di `a` ridiventa 11. Infatti dopo aver fatto diventare `a` attivo, diamo il 'comando `a`' che viene sviluppato e ciò che viene eseguito alla fine è l'assegnazione al carattere `a` del codice 11.

4. Quasi mai scriverò esattamente ciò che si legge in `latex.ltx`, ma userò una forma semplificata e, agli effetti pratici, equivalente.

Ovviamente questo non è molto interessante, ma ci sono molte situazioni in cui un saggio uso di caratteri attivi facilita le cose.

Un esempio importante è il codice dell'ambiente verbatim che vedremo più avanti; è evidente che il concetto di cambio di codice di categoria è essenziale per poterlo definire. Una delle cose che in effetti cambia è il codice del carattere `‘`, per un motivo molto semplice: quando TEX incontra la successione di caratteri `?‘`, produce il carattere ¿. Invece, ponendo

```
\catcode`\‘=13
\def‘{\leavevmode\kern0pt\char`\‘}
```

il problema è risolto, perché TEX non riconosce più la legatura. La stessa cosa viene eseguita per gli altri caratteri che hanno lo stesso problema. Fra parentesi: `\char`⟨*numero a 8 bit*⟩ produce il carattere corrispondente al numero, ma solo come comando interno. Quindi non si può scrivere, per esempio,

```
\begin{tabular}[\char`\b]{cc}
```

e sperare che l'argomento opzionale sia interpretato come una lettera `b`.

3 L'ambiente verbatim

Come ho già detto, parlando di codici di categoria vengono subito in mente l'ambiente verbatim e il comando `\verb`. Prenderò in esame le parti principali della definizione dell'ambiente.

```
{\catcode`\ =\active%
\gdef\@vobeyspaces{\catcode`\ \active
  \let \@xobeysp}}
```

Si comincia un gruppo nel quale lo spazio è reso attivo, per poter dare la definizione di `\@vobeyspaces`; questo comando, quando eseguito, rende attivo lo spazio e lo definisce come `\@xobeysp`, che è un comando interno equivalente a `\nobreakspace` e al comando `~` (cioè uno spazio non impiegabile per spezzare una riga). Siccome siamo in un gruppo, la definizione deve essere *globale*, questo è il motivo di `\gdef`.

Le righe successive sono le più importanti per capire il funzionamento dell'ambiente.

```
\begingroup \catcode`|=0
\catcode`[= 1 \catcode`]=2
\catcode`\{=12 \catcode`\}=12
\catcode`\\=12
|gdef|@xverbatim#1\end{verbatim}
  [#1|end[verbatim]]
|endgroup
```

Qui viene aperto un nuovo gruppo, nel quale il carattere `|` diventa di categoria 0, `[` di categoria 1 e `]` di categoria 2. I caratteri `\`, `{` e `}` diventano

di categoria 12, quindi stampabili normalmente. Viene poi definito il comando `\@xverbatim`.

Occorre una precisazione. Ci può essere un numero qualunque di caratteri di categoria 0; se, per esempio, `\` e `|` sono di categoria 0, scrivere `\mbox` o `|mbox` è equivalente. Tanto che Knuth, nel TEXbook, usa la notazione `\mbox` per indicare quel *token* quando è stato letto da TEX. Per evitare ambiguità con il carattere di categoria 0 in uso, applicherò la convenzione che ‖nome‖ indica il comando con quel nome, almeno per un po'.

Usualmente solo `\` ha categoria 0, ma in questa particolare applicazione è necessario usare un altro carattere. Infatti il comando ‖@xverbatim‖ è un comando con *argomento delimitato*.

La definizione rigorosa e completa di che cosa sia un argomento delimitato è troppo lunga da dare; mostrerò solo un esempio. Scrivendo `\def\Sc#1/{{\scshape #1}}` (non è una definizione particolarmente utile) possiamo usare

```
\Sc Donald E. Knuth/ è un nome famoso
```

e l'argomento al comando ‖Sc‖ è tutto ciò che viene dopo il nome del comando (spazio escluso, naturalmente: viene ignorato come sempre) fino al primo carattere `/`. Per la precisione fino al primo carattere `/` di categoria 12, se non abbiamo fatto modifiche prima della definizione. Perciò scrivendo

```
\def\Sc#1/{{\scshape #1}}
\catcode`\/=11
\Sc Donald E. Knuth/ è un nome famoso
```

otterremmo un messaggio di errore, perché nella definizione `/` ha categoria 12, mentre il carattere `/` incontrato nell'uso del comando ha categoria 11. Provare, se non ci si crede. Eventuali spazi che seguono il delimitatore *non* sono ignorati, se questo delimitatore non è una sequenza di controllo. Si possono ottenere particolari effetti, perché il delimitatore può anche essere un *token* non definito.

Torniamo però al caso in esame, ‖@xverbatim‖. Tutto ciò che sta fra questo comando e la successione di caratteri

```
\end{verbatim}
```

è preso come argomento. Lo sviluppo di questo comando è l'argomento seguito da `\end{verbatim}`.

Facciamo attenzione: non ho scritto ‖end‖`{verbatim}`. Infatti il delimitatore del comando è la successione di 14 caratteri

$$\backslash_{12}\,\mathtt{e}_{12}\,\mathtt{n}_{12}\,\mathtt{d}_{12}\,\{_{12}\,\mathtt{v}_{12}\,\mathtt{e}_{12}\,\mathtt{r}_{12}\,\mathtt{b}_{12}\,\mathtt{a}_{12}\,\mathtt{t}_{12}\,\mathtt{i}_{12}\,\mathtt{m}_{12}\,\}_{12}$$

dove ho segnato accanto ai caratteri speciali la categoria che devono avere per essere riconosciuti come delimitatori in questo caso (le lettere hanno categoria 11).

Come vedremo fra poco, il comando ‖verbatim‖ che fa cominciare tutta la faccenda (ed emesso tramite ‖begin‖`{verbatim}`) rende quei caratteri di categoria 12, come richiesto. È per questo motivo che non è possibile (almeno senza ridefinire l'ambiente) lasciare uno spazio fra `\end` e `{verbatim}`.

Proseguiamo nell'analisi; ciò che segue è

```
\def\@verbatim{\trivlist \item\relax
  ...
  \let\do\@makeother \dospecials
  \obeylines
  ...}
```

Ho omesso la parte più noiosa; questo comando apre un ambiente trivlist, fa un po' di cose non troppo importanti e poi emette il comando ‖dospecials‖, dopo aver definito ‖do‖ come ‖@makeother‖.

Che significa tutto ciò? I caratteri speciali sono 'conservati' in una lista nel modo seguente:

```
\def\dospecials{\do\ \do\\\do\{\do\}%
  \do\$\do\&\do\#\do\^\do\_\do\%\do\~}
```

Il comando ‖do‖ non è nemmeno definito; possiamo usare questa lista per particolari azioni sui suoi elementi. Siccome viene dato

```
\def\@makeother#1{\catcode`#1=12 }
```

e ‖do‖ è stato reso equivalente a esso, il comando ‖dospecials‖ viene sviluppato eseguendo ‖do‖`\␣` che quindi diventa

```
\catcode`\ =12
```

e così via per tutti i caratteri speciali nella lista che diventano di categoria 12.

Il comando ‖obeylines‖ fa in modo che ogni fine riga emetta un comando ‖par‖. Ci sono altri comandi che trascuriamo.

Ecco ora il comando che dà inizio all'ambiente:

```
\def\verbatim{\@verbatim
  \frenchspacing\@vobeyspaces
  \@xverbatim}
\def\endverbatim{\if@newlist
  \leavevmode\fi\endtrivlist}
```

Quando TEX incontra ‖begin‖`{verbatim}`, esegue il comando ‖verbatim‖; ciò a sua volta richiama ‖@verbatim‖; una volta eseguito questo vengono sviluppati ‖frenchspacing‖, per rendere uniformi tutti gli spazi fra parole, e ‖@vobeyspaces‖, per attivare lo spazio come abbiamo visto.

Infine si esegue ‖@xverbatim‖ che trova l'argomento, lo compone tipograficamente e richiama ‖endverbatim‖ il quale chiude l'ambiente trivlist aperto in precedenza; il condizionale serve a evitare certi errori.

Ora possiamo rinunciare alla convenzione e tornare al solito modo di scrivere i comandi.

Ci rimane solo da discutere `\obeylines`. Ricordiamo che il carattere di fine riga ha categoria 5. La definizione di `\obeylines` è

```
{\catcode`\^^M=13 %
  \gdef\obeylines{\catcode`\^^M=13 }%
  \let^^M\par}
```

Una combinazione `^^`⟨*carattere ASCII*⟩ è un modo per denotare caratteri non stampabili; siccome `M` ha codice ASCII 77, la combinazione `^^M` significa il carattere di codice $77 - 64 = 13$, che è proprio il carattere di fine riga (almeno per TEX). Il compito di `\obeylines` è di renderlo attivo ed equivalente a `\par`, che è proprio l'ufficio voluto.

Va notato che questo modo di definire l'ambiente verbatim non è molto efficiente: un lungo listato potrebbe esaurire la memoria di TEX. Il pacchetto verbatim migliora la definizione, facendo una lettura di ciò che si vuole scrivere *verbatim* riga per riga. Il pacchetto fancyvrb fa molto di più.

4 Un terzo tipo di *token*?

Per completezza, occorre segnalare che TEX conosce un terzo tipo di *token*: *parametro*.

Un *token* parametro è una successione di due caratteri, il primo di categoria 6, il secondo uno dei caratteri `1...9` purché di categoria 12. Questi *token* sono però legali solo in certe situazioni: quando TEX sta leggendo i parametri di una definizione e quando sta leggendo i *token* delle sostituzioni di una definizione.

Per esempio, in una riga come

```
X #1Y
```

TEX legge cinque *token*:

```
X • ␣ • # • 1 • Y
```

e il *token* `#` produce un errore; invece in una riga come

```
\def\x#1{X #1Y}
```

TEX vede nove *token*:

```
\def • \x • #1 • { • X • ␣ • #1 • Y • }
```

Per gli scopi di questa regola, un comando di definizione è `\def`, `\gdef`, `\edef` oppure `\xdef`.

Attenzione: il concetto di *token* parametro è valido solo per la *sintassi* dei comandi di definizione. In effetti se scriviamo

```
\def\a{1}
\edef\b#1{\expandafter#\a}
```

TEX non si lamenta e un successivo `\show\b` mostra

```
> \b=macro:
#1->#1.
```

Da ciò segue che `#` è in realtà considerato come *token* unico per quanto riguarda la divisione in *token*.

5 Un problema

Scrivendo una dispensa per la parte di logica di un corso di base del primo anno, mi sono posto il problema di scrivere formule in modo un po' diverso dal solito, in modo da farle risaltare rispetto al testo e, soprattutto, per distinguere i simboli formali da quelli del linguaggio corrente; per intenderci, il simbolo dello zero deve essere distinto dal modo di denotare il numero zero.

Un trucco usato in alcuni testi di logica è di scrivere i simboli del linguaggio formale in nero. Siccome poi voglio usare la notazione polacca diretta, ogni simbolo va considerato come ordinario, con una spaziatura fissa fra uno e l'altro. Inoltre i pacchetti usuali non danno risultati molto soddisfacenti e quindi ho deciso di ricorrere al 'poor man's bold' per i simboli e al nero matematico per le lettere, riservandomi di vedere in seguito se ci siano caratteri più adatti.

Proviamo a scrivere la formula che esprime che un certo numero è divisibile per un altro:

$$\exists\, \mathbf{v}_2 = \mathbf{v}_0 \times \mathbf{v}_1\ \mathbf{v}_2$$

Il codice per scrivere questa cosa è, dopo aver caricato il pacchetto amsbsy,

```
{\pmb{\exists}}\;\mathbf{v}_{2}\;
{\pmb{=}}\;\mathbf{v}_{0}\;
{\pmb{\times}}\;\mathbf{v}_{1}\;
  \mathbf{v}_{2}
```

È pensabile di scrivere decine di cose di questo tipo? E poi riuscire a leggerle nel `.tex`? Mi sono detto che un codice come

```
\formula{E v2 s= v0 s\times v1 v2}
```

sarebbe probabilmente più leggibile. Qui '`E`' sta per ∃, '`s`' è un prefisso per un simbolo, '`v`' indica una variabile con il suo indice. Per indicare ∀ ho deciso di usare '`A`' e un prefisso '`p`' per i simboli di funzioni e relazioni che sono normalmente lettere. In più uso '`a`' e '`o`' per i connettivi 'et' e 'vel', '`i`' per 'implica', '`j`' per 'se e solo se' e '`n`' per 'non'.

Viene subito l'idea di usare caratteri attivi.

La sintassi di `\formula` dovrebbe assomigliare a un comando con argomento, ma non può esserlo veramente, perché altrimenti i codici di categoria dell'argomento sarebbero letti e non più mutabili. Il problema è noto e la soluzione anche:

```
\def\formula{\bgroup\logicactivate
  \let\next= }
```

Lo sviluppo di `\formula` apre un gruppo con `\bgroup`; il comando `\logicactivate` cambia i codici di categoria necessari e quindi rende disponibili le definizioni dei caratteri attivati; infine gli ultimi comandi ingoiano la graffa aperta. Tutto ciò che viene eseguito qui sarà annullato quando si incontra la graffa chiusa, che corrisponde al

comando \bgroup inserito da \formula. Va ricordato che \bgroup è, in certi contesti, equivalente a una graffa aperta.

Ora si tratta di definire \logicactivate:

```
\def\logicactivate{\catcode`\A=13
\catcode`\E=13
...
}
```

Dobbiamo definire i comandi; A attivo deve stare per {\pmb{\forall}}\; e vediamo subito un problemino: siccome voglio attivare anche p, non posso più dare il comando \pmb!

La soluzione è semplice: prima definisco dei comandi ausiliari e poi rendo i caratteri attivi equivalenti a questi. Mi serve poi un modo di dire \global\let senza usare caratteri proibiti.

```
{% apro un gruppo
\def\GL{\global\let}
\def\A{{\pmb{\forall}}\;}
...
\def\v#1{\mathbf{v}_{#1}\;}
\logicactivate
\GL A\A
...
\GL v\v
}
```

Se notate, nella prima parte ho usato nomi di comandi già definiti in LaTeX; tuttavia, quando il gruppo verrà chiuso, queste definizioni spariranno, ma non svanisce il significato dei caratteri quando attivati, poiché ho usato \global\let.

Rimane un problema: l'ultimo carattere inserisce una spaziatura di troppo che andrebbe tolta. Per ora non mi interessa, anche perché la soluzione definitiva non è questa.

Fin qui infatti tutto sembra funzionare: scrivendo \formula{A v0} ho quello che mi serviva (a parte le spaziature). E una ciliegia tira l'altra: ora mi viene in mente di scrivere all'interno di una formula anche *metavariabili*, cioè cose del tipo

$$\forall\, \mathbf{v}_0\ \varphi$$

dove φ sta a indicare una formula qualunque. Non è poi così difficile, dopo tutto, una volta che si capisce ciò che c'è da fare: disattivare i caratteri che abbiamo attivato, ma dentro un gruppo, cosicché ritornano attivi quando il gruppo termina. La sintassi che mi propongo è, per esempio,

```
\formula{A v0 <\varphi>}
```

Siccome dobbiamo attivare e disattivare caratteri, è bene scriversi una *lista* dei caratteri speciali, esattamente come \dospecials per verbatim, e comandi ausiliari per poter eseguire la lista in modi diversi:

```
\def\logiclist{\do\A\do\E\do\a\do\i%
  \do\j\do\n\do\o\do\p\do\s\do\v}
\def\logicactivate{\let\do\logicactive
  \logiclist\logicactive\<}
\def\logicinactivate{\let\do\logicletter
  \logiclist\logicother\<}
\def\logicletter#1{\catcode`#1=11 }
\def\logicother#1{\catcode`#1=12 }
\def\logicactive#1{\catcode`#1=13 }
```

Il comando \logicactivate *esegue* la lista dei caratteri e li attiva; poi attivo anche il carattere <. Questo carattere non va nella lista principale, perché la sua categoria usuale è 12. Il comando \logicinactivate invece fa l'operazione contraria.

Ora devo definire la funzione del carattere <; suppongo che il carattere sia già attivato e aggiungerò ai comandi precedenti

```
\def\<{\bgroup\logicinactivate
  \logicescape}
```

prima di \logicactivate e, dopo,

```
\GL <\<
```

Si tratta ora di definire \logicescape:

```
\def\logicescape#1>{#1\egroup\;}
```

Il comando considera come suo argomento tutto ciò che viene fino al carattere > (si noti la sintassi con un argomento delimitato), e lo passa invariato aggiungendo \egroup e la spaziatura. La chiusura del gruppo fa tornare attivi i caratteri giusti e la composizione della formula può continuare come prima; dunque è lecito scrivere cose del tipo

```
\formula{i <\alpha> i <\beta> <\gamma>}
```

che produce

$$\to \alpha \to \beta\ \gamma$$

Molto bene, le cose vanno alla grande. Posso sbizzarrirmi a scrivere formule in modo facile e intuitivo.

L'inghippo si vede immediatamente quando però provo a scrivere queste formule in uno degli ambienti di allineamento di amsmath: non vanno! L'ultima formula scritta dà come risultato

$$i <\alpha> i <\beta> <\gamma>$$

che non è proprio quello sperato!

Qual è il motivo? Gli ambienti di allineamento di amsmath, per esempio align, leggono tutto il contenuto dell'ambiente e poi eseguono varie composizioni per ottenere il miglior risultato possibile. Ottimo dal punto di vista tipografico, certo; pessimo per me: quando entra in azione \formula i caratteri all'interno delle graffe sono già passati per la *bocca* di TeX e quindi hanno già ricevuto la loro categoria. Ormai è troppo tardi per cambiarla.

Che fare? Arrendersi e scrivere le formule per esteso, magari inventandosi qualche abbreviazione? No: con TeX si può fare tutto!

6 ε-TEX

Da un po' di tempo, chi ha una distribuzione aggiornata del sistema TEX basata su Web2C (le più diffuse lo sono) e invoca `latex` o `pdflatex` sta in realtà usando `pdfetex`, cioè la versione di ε-TEX capace di scrivere anche in formato PDF oltre che `.dvi`.

Il progetto ε-TEX ha studiato un programma compatibile con TEX ma con alcune estensioni. Una fra queste è quella che rende possibile adattare i comandi per le formule in modo da evitare il problema con amsmath.

L'estensione di cui avevo bisogno c'è e si chiama `\scantokens`. Questo comando primitivo aggiunto da ε-TEX ha la seguente azione: legge la lista di *token* che lo segue fra parentesi graffe come se si trattasse di un *file* esterno richiamato con `\input`. Lo stesso risultato si può ottenere con TEX tradizionale scrivendo su un *file* esterno e poi leggendolo, con certe limitazioni.

I caratteri che vengono immessi tramite `\scantokens` ricevono in quel momento la categoria, perché rientrano nel meccanismo di lettura.

Niente più problemi, dunque. Quasi, a dire il vero; ma lo vedremo più avanti. La nuova definizione di `\formula` è

```
\def\formula#1{\begingroup
  \logicactivate
  \scantokens{#1}%
  \unskip\endgroup}
```

Se invece di scrivere `\;` nelle definizioni dei simboli uso un comando esplicito di spaziatura, ho anche risolto il problema della spaziatura finale. La decisione finale è stata di usare uno spazio di 0,2 em; il comando `\unskip` elimina l'ultima spaziatura.

Però c'è un altro problema, adesso: leggendo l'argomento di `\formula` e passandolo a `\scantokens`, non posso più uscire dal modo 'formula' per comporre le metavariabili. Questo all'inizio mi sorprese, ma una breve riflessione, aiutata dai messaggi di errore, mi fece capire ciò che stava succedendo.

Quando si passa una lista di *token* a `\scantokens`, questo legge uno *pseudofile* che contiene esattamente quei *token*; e infatti il messaggio riguardava il comando `\p` non definito, mentre io avevo dato `\phi` come argomento. Il problema sta nel fatto che l'apparato di lettura non vede i quattro caratteri

```
\ p h i
```

ma solo tre oggetti: il *token* `\p` seguito dai caratteri `h` e `i`. Questo perché il carattere `p` ha già ricevuto la categoria 13 quando è stato valutato l'argomento di `\scantokens` nello sviluppo di `\formula`.

Gli sviluppatori di ε-TEX hanno pensato anche a questo e hanno messo a disposizione `\detokenize`. Che cosa fa questo comando primitivo? Va seguito da una lista di *token* fra graffe ed esegue l'operazione inversa della lettura: spezza ciascun *token* nei suoi componenti iniziali. Una lettera rimane una lettera, ma un *token* simbolico come `\author` ridiventa una successione di sette caratteri che posso passare a `\scantokens` per farli leggere con le regole di assegnazione della categoria in vigore in quel momento.

Basta quindi ridefinire `\logicescape`:

```
\def\logicescape#1>{%
  \scantokens\expandafter{%
  \detokenize{#1}}%
  \endgroup\hskip.1em\relax}
```

Qui `\scantokens` va a cercare la lista che deve seguirlo e per fare questo sviluppa i *token* che seguono fino a trovare una parentesi graffa aperta; trova `\expandafter` che quindi sviluppa ciò che segue la parentesi graffa, cioè `\detokenize`. Il *token* `\expandafter` viene eliminato dal suo stesso sviluppo e `\scantokens` trova la graffa aperta; quindi legge la sequenza di caratteri ritrasformandola in *token*, ma con i codici di categoria giusti, perché `\logicescape` è stato chiamato dopo `\logicinactivate`.

C'è un altro piccolo problema quando si usa `\scantokens`. Se scriviamo `\scantokens{a}b` ci aspetteremmo che ε-TEX stampasse semplicemente 'ab', invece esce 'a b'. Questo perché, essendo i *token* letti come se venissero da un *file* esterno, a ogni 'riga' viene aggiunto il carattere di fine riga. Nel mio caso il problema non si pone, perché sto componendo in modo matematico; in altri potrebbe dare problemi che si possono curare mettendo come ultimo *token* nella lista data a `\scantokens` il comando `\noexpand`.

7 Il codice finale

Le idee di prima vengono utili anche per usare argomenti qualunque alle lettere `p` e `s`: posso usare anche per queste il ritorno ai caratteri non attivi in modo da poter scrivere anche

```
\formula{pA}
```

che non sarebbe stato concesso nella versione precedente. La lettera '`p`' va seguita da un carattere alfabetico, mentre la lettera '`s`' da un simbolo qualunque.

Ecco il codice completo, con qualche raffinamento che invito a studiare. In particolare, invece che con `\pmb` o `\mathsfb` espliciti, ho definito i comandi in termini di 'prefissi' astratti che quindi è possibile modificare facilmente. Inoltre uso invece che '`logic`' un prefisso '`lf@`' per i comandi privati. Naturalmente il tutto va circondato da `\makeatletter` e `\makeatother` oppure letto da un pacchetto (`.sty`).

Si noti che ho definito un nuovo alfabeto matematico che usa i caratteri Computer Modern

Sans Serif neri (o, comunque, quelli della famiglia `\sfdefault`). Inoltre, se è caricato il pacchetto graphicx, i simboli dei quantificatori **∀** e **∃** si ottengono ruotando le lettere **A** e **E**.

Un altro problema che ho cercato di risolvere è quello di far cooperare queste macro con altre che attivino caratteri. Usare comandi come

```
{\catcode`\A=13 \gdef A{...}}
```

può essere pericoloso; forse non con 'A', ma magari con '<' sì. La faccenda si risolve usando un altro `\scantokens`, come vedremo.

Dividerò la descrizione del codice in vari brani, dopo il codice ci sarà il commento. Il codice è presentato sotto forma di un pacchetto, formula.

Identificazione e opzioni

```
\ProvidesPackage{formula}
\newif\iflf@graph
\@ifpackageloaded{graphicx}
  {\lf@graphtrue}
  {\lf@graphfalse
   \PackageWarningNoLine{formula}{%
    It is best to load this package
    after `graphicx'}%
  }

\newif\iflf@tc
\@ifpackageloaded{textcomp}
  {\lf@tctrue
   \PackageWarningNoLine{formula}{%
    Arrow symbols can be not available
    in some fonts;\MessageBreak
    in this case use the package with
    the `notc' option}%
  }
  {\lf@tcfalse}

\DeclareOption{notc}{\lf@tcfalse}
\ProcessOptions\relax
```

Identifico il pacchetto e introduco due condizionali per verificare se sono stati caricati certi pacchetti. C'è anche un'opzione `notc` per evitare di usare i *font* nella codifica TS1; certi, infatti, non possiedono i caratteri necessari.

Preliminari

```
\DeclareMathAlphabet{\mathsfb}
  {\encodingdefault}{\sfdefault}{bx}{n}
\setbox0=\hbox{$\mathsfb{A}$}
\iflf@tc
  \def\sfbs{%
    \usefont{TS1}{\sfdefault}{bx}{n}}
  \setbox0=\hbox{\sfbs 0}
\fi
```

Se ho a disposizione graphicx e textcomp, posso definire meglio certe costruzioni; altrimenti uso un ripiego. Definisco anche due *font* particolari, il primo per comporre i simboli del linguaggio che consistono di lettere, il secondo per certi simboli che esistono nella codifica TS1.

La riga `\setbox0=\hbox{\sfbs 0}` e l'analoga per `\mathsfb` servono a far caricare i *font* subito, per evitare che il *file* `.fd` corrispondente sia letto quando i caratteri sono attivi, con gli ovvi problemi che ne risulterebbero: un `.fd` che non sia precaricato viene letto infatti al primo uso di un carattere da comporre in quel *font*.

Liste

```
\def\lf@letterlist{\do\A\do\E\do\a%
  \do\i\do\j\do\n\do\o\do\p\do\s\do\v}
\def\lf@otherlist{\do\<}

\def\lf@activate{\let\do\lf@active
  \lf@letterlist\lf@otherlist}
\def\lf@inactivate{%
  \let\do\lf@letter\lf@letterlist
  \let\do\lf@other\lf@otherlist}

\def\lf@letter#1{\catcode`#1=11 }
\def\lf@other#1{\catcode`#1=12 }
\def\lf@active#1{\catcode`#1=13 }
```

Definisco due liste, una con i caratteri di categoria 11 e una con quelli di categoria 12 da attivare in seguito. Nella seconda ce n'è solo uno, ma meglio essere previdenti: il pacchetto potrebbe essere esteso.

Poi introduco i comandi per eseguire le liste e assegnare le corrette categorie.

Comandi astratti

```
\def\lf@prefi{\pmb}
\def\lf@genI#1{%
  {\lf@prefi{#1}}\lf@space}
\def\lf@prefii{\mathsfb}
\def\lf@genII#1{%
  {\lf@prefii{#1}}\lf@space}
\iflf@tc
  \def\lf@prefiii{\sfbs}
  \def\lf@genIII#1{%
    \hbox{\lf@prefiii#1}\lf@space}
\fi
\def\lf@space{\mskip 2mu minus 1mu}
```

Definisco comandi astratti, in modo da non dover fare troppe modifiche se decido di cambiare come trattare tipograficamente i simboli. Per esempio, `\lf@space` inserisce una 'lunghezza matematica elastica' di 2 unità, che può diminuire fino a 1. Diciotto unità mu sono uno spazio quadratone (`\quad`).

Quantificatori

```
\iflf@graph
 \def\lf@rot#1{%
   \rotatebox[origin=c]{180}{#1}}
 \def\lf@A{%
```

```
   \lf@rot{$\lf@prefii{A}$}\lf@space}
 \def\lf@E{%
   \lf@rot{$\lf@prefii{E}$}\lf@space}
\else
 \def\lf@A{{\lf@prefi{\forall}}\lf@space}
 \def\lf@E{{\lf@prefi{\exists}}\lf@space}
\fi
```

Comincio a definire i nuovi comandi. Per 'esiste' e 'per ogni', prevedo due modalità diverse, se è caricato graphicx o no. Anche qui definisco un comando generico, `\lf@rot`, che può venire utile per eventuali estensioni del pacchetto.

Altri simboli

```
\def\lf@a{\lf@genI{\textstyle\bigwedge}}
\def\lf@o{\lf@genI{\textstyle\bigvee}}
\iflf@tc
  \def\lf@i{\lf@genIII{\char'031}}
  \def\lf@j{\lf@genIII{%
    \rlap{\char'030}\kern.1em\char'031}}
  \def\lf@n{\lf@genIII{\char'254}}
\else
  \def\lf@i{\lf@genI{\to}}
  \def\lf@j{\lf@genI{\leftrightarrow}}
  \def\lf@n{\lf@genI{\lnot}}
\fi
```

Proseguo nelle definizioni. Per 'implica', 'se e solo se' e 'non', uso i caratteri nella codifica TS1, se è caricato textcomp; il 'se e solo se' è ottenuto sovrapponendo la freccia verso sinistra a quella verso destra, con un piccolo spostamento. I simboli per 'e' e 'o' sono più grandi per aiutare a distinguerli.

Simboli per relazioni e funzioni

```
\def\lf@p{\bgroup\lf@inactivate\lf@pred}
\def\lf@pred#1{%
  \lf@prefii{#1}\egroup\lf@space}

\def\lf@s{\bgroup\lf@inactivate\lf@sym}
\def\lf@sym#1{%
  {\lf@prefi{#1}}\egroup\lf@space}

\def\lf@v#1{\lf@prefii{v}_{#1}\lf@space}
```

Definisco i comandi con argomenti; il primo per comporre una lettera, il secondo per un simbolo: in entrambi i casi apro un gruppo, disattivo i caratteri e chiamo la macro che esegue la composizione corretta. Nel terzo caso semplicemente compongo la lettera '**v**' con il pedice indicato, nella supposizione che sia un numero e che i numeri non vengano attivati. Posso scrivere `v0` oppure, se necessario, `v{10}`.

Uscita dal modo formula

```
\def\lf@lt{\bgroup\lf@inactivate
  \lf@escape}
\def\lf@escape#1>{%
  \scantokens\expandafter{%
  \detokenize{#1}}%
  \egroup\lf@space}
```

Definisco il comando per 'uscire' temporaneamente dal modo formula. La descrizione della macro è già stata data. Uso `\bgroup` perché così la sottoformula che compongo è considerata come un simbolo ordinario in ogni caso.

Definizione locale

```
\def\lf@doequiv{\let\\\let
  \let\A\lf@A \let\E\lf@E \let\a\lf@a
  \let\i\lf@i \let\j\lf@j \let\n\lf@n
  \let\o\lf@o \let\p\lf@p \let\s\lf@s
  \let\v\lf@v \let\<\lf@lt}
```

Definisco un comando che rende equivalenti i comandi precedenti con sequenze di controllo più brevi, nelle quali si usano solo i caratteri che vengono attivati nel modo formula. Questo perché una volta che li ho attivati non posso più usare comandi a più lettere che contengano quei caratteri! Naturalmente questo comando viene emesso dentro un gruppo e quindi le ridefinizioni che opera spariscono alla chiusura del gruppo. Il comando `\\` viene reso equivalente a `\let`.

Attivazione locale

```
\def\lf@doactivechars{%
  \\ A\A \\ E\E \\ a\a \\ i\i
  \\ j\j \\ n\n \\ o\o \\ p\p
  \\ s\s \\ v\v \\ <\<}
```

Questo è il comando che rende i caratteri attivi equivalenti ai comandi definiti prima. Il fatto che '`A`' e gli altri caratteri siano al momento di categoria 11 o 12 è irrilevante, vedremo perché.

Il comando principale

```
\DeclareRobustCommand\formula[1]{%
  \begingroup
  \lf@doequiv
  \lf@activate
  \scantokens\expandafter{%
    \lf@doactivechars}%
  \scantokens{#1}%
  \unskip
  \endgroup}
```

Questo è il comando principale, reso *robusto* per poterlo usare anche nei titoli di capitolo o sezione.

1. Si apre un gruppo, con `\begingroup` per motivi TEXnici.
2. Eseguo il comando che rende i comandi con nomi lunghi equivalenti ai comandi con nomi brevi.
3. Attivo i caratteri del modo formula.
4. Eseguo il comando `\lf@doactivechars`, ma rileggendone i *token* tramite `\scantokens`; in questo modo i caratteri dopo `\\` (che equivale a `\let`) sono attivi.

5. Leggo l'argomento, ma tramite `\scantokens`, in modo che i codici di categoria siano assegnati correttamente.

6. Elimino lo spazio esplicito dopo l'ultimo simbolo della formula e chiudo il gruppo.

Comandi ausiliari

```
\newcommand{\lvar}[1]{\lf@prefii{v}_{#1}}
\newcommand{\lrf}[1]{{\lf@prefii{#1}}}
\newcommand{\lsym}[1]{{\lf@prefi{#1}}}
```

`\makeatother`

Definisco, già che ci sono, anche abbreviazioni per poter usare i simboli anche senza abilitare il modo formula. Le parentesi graffe che sembrano in più servono a far considerare il simbolo come ordinario, in modo che non segua le sue spaziature proprie, come succederebbe senza.

8 Esempi d'uso

Ecco alcuni esempi; non occorre conoscere la logica per leggerli, basta confrontare ciò che viene composto dal codice che mostro subito dopo.

Sia **P** un simbolo di relazione binaria; l'interpretazione di **P** è una relazione di equivalenza se e solo se l'interpretazione delle seguenti tre formule è vera:

$$\forall \mathbf{v}_0\, \mathbf{P}\, \mathbf{v}_0\, \mathbf{v}_0 \tag{1}$$

$$\forall \mathbf{v}_0 \forall \mathbf{v}_1 \rightarrow \mathbf{P}\, \mathbf{v}_0\, \mathbf{v}_1\, \mathbf{P}\, \mathbf{v}_1\, \mathbf{v}_0 \tag{2}$$

$$\forall \mathbf{v}_0 \forall \mathbf{v}_1 \forall \mathbf{v}_2 \rightarrow \bigwedge \mathbf{P}\, \mathbf{v}_0\, \mathbf{v}_1\, \mathbf{P}\, \mathbf{v}_1\, \mathbf{v}_2\, \mathbf{P}\, \mathbf{v}_0\, \mathbf{v}_2 \tag{3}$$

Il codice per questa terna è

```
\begin{gather}
\formula{A v0 pP v0 v0}\\
\formula{A v0 A v1
  i pP v0 v1 pP v1 v0}\\
\formula{A v0 A v1 A v2
  i a pP v0 v1 pP v1 v2 pP v0 v2}
\end{gather}
```

Ed ecco la formula per la divisibilità:

$$\exists \mathbf{v}_2 = \mathbf{v}_0 \times \mathbf{v}_1\, \mathbf{v}_2 \tag{4}$$

che ha come codice

```
\begin{equation}
\formula{E v2 s= v0 s\times v1 v2}
\end{equation}
```

L'aritmetica di Peano al primo ordine ha i seguenti assiomi:

$$\forall \mathbf{v}_0 \neg = \mathbf{0}\, \mathbf{S}\, \mathbf{v}_0$$
$$\forall \mathbf{v}_0 \forall \mathbf{v}_1 \rightarrow = \mathbf{S}\, \mathbf{v}_0\, \mathbf{S}\, \mathbf{v}_1 = \mathbf{v}_0\, \mathbf{v}_1$$
$$\forall \mathbf{v}_0 = + \mathbf{0}\, \mathbf{v}_0\, \mathbf{v}_0$$
$$\forall \mathbf{v}_0 \forall \mathbf{v}_1 = + \mathbf{v}_0\, \mathbf{S}\, \mathbf{v}_1\, \mathbf{S} + \mathbf{v}_0\, \mathbf{v}_1$$
$$\forall \mathbf{v}_0 = \times \mathbf{0}\, \mathbf{v}_0\, \mathbf{v}_0$$
$$\forall \mathbf{v}_0 \forall \mathbf{v}_1 = \times \mathbf{v}_0\, \mathbf{S}\, \mathbf{v}_1 + \times \mathbf{v}_0\, \mathbf{v}_1\, \mathbf{v}_0$$
$$\rightarrow \bigwedge \varphi(\mathbf{0}) \forall \mathbf{v}_0 \rightarrow \varphi(\mathbf{v}_0)\, \varphi(\mathbf{S}\, \mathbf{v}_0) \forall \mathbf{v}_0\, \varphi(\mathbf{v}_0)$$

dove φ è una qualunque formula con una variabile libera e $\varphi(x)$ indica la formula che si ottiene sostituendo tutte le occorrenze della variabile libera con il termine x.

Riporto solo il codice per l'ultima formula:

```
\formula{i a <\varphi>(p0)
  A v0 i <\varphi>(v0) <\varphi>(pSv0)
  A v0 <\varphi>(v0)}
```

Una formula con una variabile libera la cui interpretazione sia 'il numero naturale denotato è primo', può essere

$$\bigwedge < \mathbf{1}\, \mathbf{v}_0 \forall \mathbf{v}_1 \rightarrow \exists \mathbf{v}_2 = \mathbf{v}_0 \times \mathbf{v}_1\, \mathbf{v}_2 \bigvee = \mathbf{1}\, \mathbf{v}_1 = \mathbf{v}_0\, \mathbf{v}_1$$

che va scritta

```
\formula{ a s< p1 v0 A v1 i
E v2 s= v0 s\times v1 v2
o s= s1 v1 s= v0 v1 }
```

La lettura della formula composta sembra impervia; ma con un po' di pazienza si fa l'abitudine alla notazione polacca. Ci sono due sottoformule collegate da un 'e': la prima dice che il numero dato è maggiore di 1; la seconda dice che, se un numero divide il numero dato, allora è 1 oppure il numero dato. Certamente però il modo di scrivere la formula in LaTeX è semplice e chiaro.

Facciamo l'ultima prova per vedere che effettivamente lo spazio finale viene cancellato:

```
\begin{gather*}
|\formula{pA}|\quad|\formula{<\psi>}|\\
|\lrf{A}|\quad|\psi|
\end{gather*}
```

produce le seguenti formule:

$$|\mathbf{A}| \quad |\psi|$$
$$|\mathbf{A}| \quad |\psi|$$

che sono effettivamente identiche.

▷ Enrico Gregorio
Dipartimento di Informatica, Università di Verona
Enrico.Gregorio@univr.it

Libretti in LaTeX

*Gustavo Cevolani**

Sommario

La prima parte dell'articolo illustra i principali metodi disponibili in LaTeX per stampare pieghevoli (§ 2), opuscoli di poche pagine (§ 3) e libri veri e propri (§ 4). La seconda parte presenta alcuni esempi di codice e il relativo risultato (§ 5), oltre a prendere in esame alcuni metodi alternativi (appendici A e B).

1 Introduzione

Un *libretto* è un documento stampato fronte-retro su fogli che vengono poi piegati e rilegati per potere esser sfogliati e letti come un normale libro. Il § 1.1 spiega brevemente come è composto un libro stampato professionalmente e quali tipi di libretti sia possibile ottenere artigianalmente. Il § 1.2 presenta brevemente i principali metodi disponibili in LaTeX per produrre questi tipi di libretto.

1.1 Nozioni di base

Un libretto è costituito da un certo numero di pagine stampate su entrambe le facciate di uno o più fogli, chiamati "segnature", che vengono poi legati assieme e rilegati con una copertina per produrre il risultato finale.

Fogli, pagine e facciate

Nel sèguito, chiameremo "foglio" il pezzo di carta fisico sul quale viene stampato il documento e "facciata" ognuno dei due lati del foglio. Quando l'ordine è rilevante, distingueremo le due facciate di un foglio indicandole con "fronte" e "retro". Chiameremo invece "pagina" l'unità logica di cui si compone il documento da stampare, che a volte viene appunto chiamata "pagina logica". Il testo del documento viene stampato su ogni pagina all'interno di una gabbia che chiameremo "blocco del testo"; l'impaginazione è l'operazione di sistemare il blocco del testo sulla superficie della pagina. Nella normale stampa non professionale, un foglio contiene solamente due pagine, una sul fronte e una sul retro, o anche una sola se stampato solo fronte. In generale, tuttavia, un singolo foglio può contenere parecchie pagine del documento; nella stampa professionale è anzi regola generale che un foglio di grandi dimensioni contenga molte pagine più piccole (si veda la figura 1).

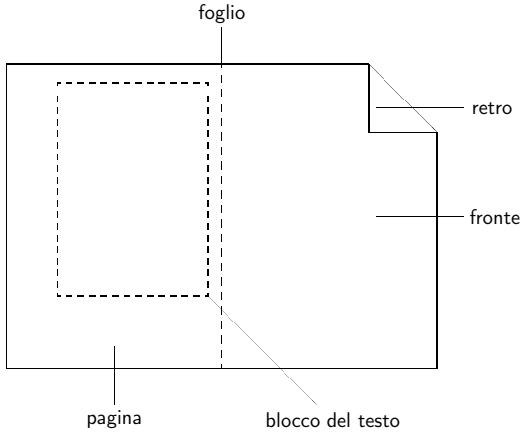


Figura 1: Un foglio con due pagine per facciata, e il relativo blocco del testo.

Formati di carta

Tutti gli esempi considerati in questo articolo prevedono l'uso di fogli di dimensioni A4, che è il formato di carta standard per la stampa quotidiana. Le istruzioni tuttavia sono generali e possono venire applicate, in linea di principio, a qualsiasi formato di carta (se si possiedono le macchine da stampa in grado di utilizzarlo).

Dato che useremo spesso espressioni quali «stampare quattro pagine A5 su un unico foglio A4», può essere utile riepilogare alcune nozioni relative alle dimensioni dei formati più comuni. Le sigle A4, A5 e simili risalgono allo standard DIN 476 — per cui spesso si usano anche sigle come "DIN-A4" o "DIN-A5" — cioè alla convenzione della *Deutsche Industrie Norm* (Norma Industriale Tedesca) che nel 1922 codificò l'attuale serie dei formati da stampa, che venne a sua volta accettata come standard internazionale ISO 216 nel 1975 (si veda la figura 2 nella pagina successiva). Il formato iniziale della serie è chiamato A0 (841×1189 mm) e ha una superficie (approssimata) di $1\,\mathrm{m}^2$. Tagliando un foglio A0 a metà lungo il lato maggiore, si ottengono due fogli in formato A1 (594×841 mm). Le dimensioni di tutti i formati minori vengono ottenute ripetendo il procedimento, cioè dimezzando (e approssimando per difetto) il lato lungo di un foglio del formato immediatamente maggiore. Quindi, per esempio, un foglio in formato A4 (210×297 mm) si ottiene dimezzando un foglio A3 (297×420 mm).

Questa serie presenta almeno due vantaggi. In-

*Desidero ringraziare Peter Wilson, autore di booklet, per aver gentilmente risposto ad alcuni dubbi e domande sul funzionamento del suo pacchetto. Grazie anche al recensore di ArsTeXnica per avermi segnalato l'esistenza del programma TeXexec di ConTeXt e per l'attenta revisione del testo. Ringrazio infine i diversi partecipanti al forum di GuIt che hanno discusso in diverse occasioni l'argomento dei «libretti in LaTeX».

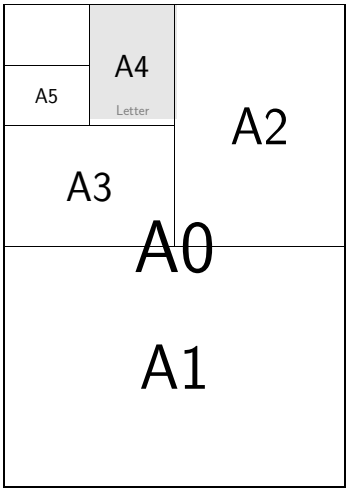


Figura 2: La serie dei formati di carta dello standard internazionale ISO, derivata dallo standard tedesco DIN, fino al formato A5. L'area ombreggiata corrisponde al formato *Letter* (216×279 mm), usato negli Stati Uniti e in Canada, qui sovrapposto al comune formato A4 (210×297 mm).

nanzi tutto, per ovvi motivi, ogni formato ha una superficie pari alla metà di quello precedente, il che permette sempre di stampare su un'unica facciata di un foglio in formato An due pagine di formato A$(n-1)$, quattro di formato A$(n-2)$ e così via. Inoltre, il rapporto fra il lato maggiore e il lato minore del foglio di ogni formato è costante e pari a $\sqrt{2}$; ciò risulta particolarmente comodo, oltre che per gli ingrandimenti e le fotocopie, quando si tratta di ridimensionare le pagine e adattarne un certo numero su fogli di formato maggiore: vedremo qualche esempio più avanti. Gli Stati Uniti e il Canada non seguono la convenzione ISO e usano il formato *Letter* (216×279 mm o $8,5 \times 11$ in), leggermente più largo e più basso del formato A4, che è anche il formato nativo di LaTeX.[1]

Segnature, legatura e rilegatura

Nella stampa non-professionale, come quella di una tesi, ogni pagina del testo viene stampata separatamente su un foglio delle sue stesse dimensioni. Le due uniche varianti sono la stampa solo-fronte (una pagina per foglio) e la stampa fronte-retro (due pagine per foglio). I fogli vengono poi uniti semplicemente impilandoli uno sull'altro, e fissandoli sul lato in diversi modi (con un punto metallico, una spirale, con la colla, ecc.).

I termini "legatura" e "rilegatura" vengono a volte usati come sinonimi. Seguendo Marchesi *et al.* (1999), chiameremo per comodità *legatura* l'operazione di mettere assieme e fissare i fogli di un libro e *rilegatura* la successiva operazione di aggiungere una copertina. Nella stampa artigianale, solitamente esiste solo la legatura, dato che la copertina è costituita da due ulteriori fogli (pagine fuori testo) aggiunte in testa e in coda al blocco del documento.

La stampa professionale prevede che le pagine di un documento vengano stampate su un certo numero di segnature, che vengono poi legate assieme e infine rilegate per produrre il libro finale. Una *segnatura* è un foglio sul quale vengono stampate fronte-retro diverse pagine del documento, in un ordine e orientamento tali per cui il foglio possa poi venir piegato e tagliato in modo da ottenere un fascicolo correttamente sfogliabile (si veda per esempio la figura 3).[2] In linguaggio tecnico, l'operazione di disporre le pagine sul foglio nell'ordine e orientamento corretto è chiamata *imposizione* (o "impostazione"). Le segnature vengono poi messe assieme in un blocco, che costituisce il *volume*, e legate assieme con vari metodi (a filo, a colla, ecc.). Infine, si aggiunge una copertina (rigida o morbida) che è un foglio di grandi dimensioni (per l'esattezza, alto quanto le pagine del volume e largo il doppio più lo spessore del volume stesso) che viene fissato al blocco delle segnature a colla o con altri metodi (si veda Marchesi *et al.* (1999)).

Esistono vari tipi di segnature. La più semplice, chiamata *in-folio*, si ottiene piegando il foglio lungo il lato maggiore e senza nessun taglio. In questo caso, l'imposizione è semplice: occorre stampare su una facciata del foglio la prima e la quarta pagina, sull'altra la seconda e la terza, in modo che la seconda sia sul retro della prima e la quarta sul retro della terza. Dopo la piegatura, il risultato è un fascicolo di quattro pagine già ordinate. Una segnatura *in quarto* (o *in-4°*) si ottiene piegando successivamente il foglio per due volte (sempre sul lato lungo) e tagliando lungo la prima piega. Il risultato è un fascicolo ordinato di otto pagine, la cui superficie è appunto un quarto di quella del foglio originale (si veda la figura 3 a fronte). Similmente, la segnatura *in ottavo* (o *in-8°*) prevede tre pieghe, sempre lungo il lato maggiore, e due tagli, lungo le prime due pieghe, e produce un fascicolo con 16 pagine. E così via:

Pieghe (n)	Tagli ($n-1$)	Pagine (2^n)	Nome
1	0	4	in-folio
2	1	8	in-4°
3	2	16	in-8°
4	3	32	in-16°
5	4	64	in-32°

1. "Letter" viene talvolta tradotto con l'italiano "quadrotta", che tuttavia sembrerebbe riferirsi a un formato di 270×440 mm.

2. "Segnatura" è anche il numero progressivo o il segno di riconoscimento stampato sulla prima pagina di ognuno dei fascicoli così ottenuti, in modo da poterli comodamente ordinare prima della legatura. Solitamente, la parte del foglio con la segnatura risulta asportata durante la rifilatura del volume. A volte si usa, per indicare la segnatura come fascicolo, il termine antiquato "quinterno", che indicava un gruppo di cinque fogli piegati in due e inseriti uno dentro all'altro.

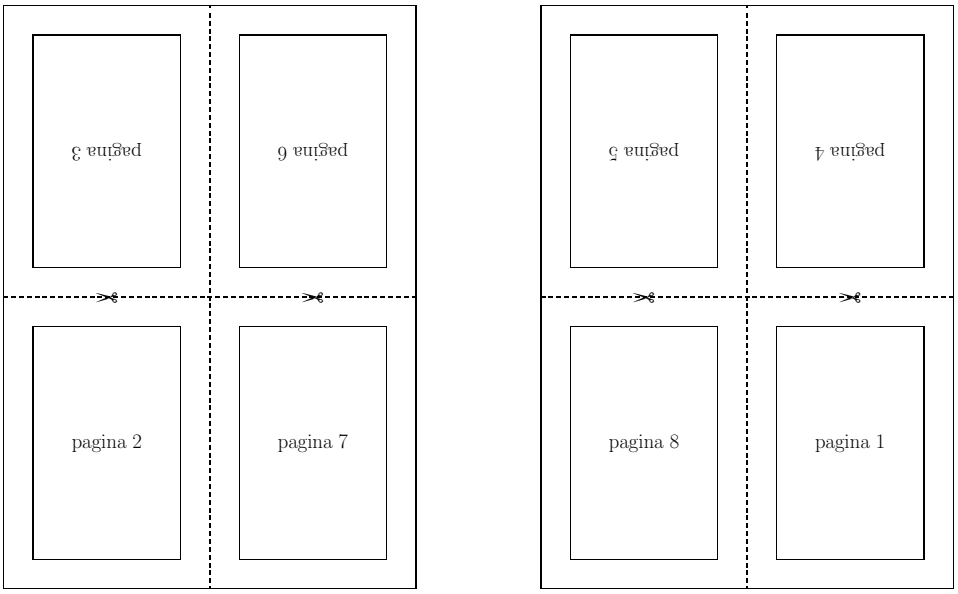


Figura 3: Le due facciate di una segnatura *in quarto*. Il foglio viene stampato fronte-retro, piegato due volte sul lato maggiore e tagliato lungo la prima piegatura; si ottengono in questo modo due nuovi fogli grandi la metà dell'originale, piegati in due, su ognuno dei quali sono stampate fronte-retro quattro pagine. Il risultato è il fascicolo già ordinato delle otto pagine del documento.

Altri metodi di imposizione e taglio generano altre segnature: per esempio, la segnatura *in sesto* prevede che il foglio venga prima piegato in due e quindi in tre, producendo un fascicolo di 12 pagine. Una semplice piegatura in tre parti genera invece un pieghevole a sei pagine (§ 2.2).[3]

Tipi di stampato

Nel sèguito, distinguiamo tre tipi fondamentali di stampato, dipendentemente dal numero e dal tipo di segnature che li compongono (si veda la figura 4 nella pagina successiva):

- Un *pieghevole* è uno stampato costituito da un'unica segnatura di quattro o sei pagine, corrispondente a un unico foglio piegato in due o tre parti e stampato su entrambe le facciate. Una semplice segnatura in-folio, per esempio, è di fatto un pieghevole di quattro pagine.

- Un *opuscolo* è uno stampato costituito da un'unica segnatura di almeno otto pagine (quindi segnature in quarto, in sesto, in ottavo, in sedicesimo, ecc.), ottenuta dalla piegatura di un foglio e da almeno un taglio.

- Un *libro* è uno stampato costituito da almeno due segnature di un qualsiasi numero di pagine.

Useremo il termine generico "libretto" per indicare tutti i suddetti tipi di stampato. Dal punto di vista pratico, la differenza principale fra i diversi tipi di libretto riguarda la loro legatura e rilegatura:

- un pieghevole, come dice la parola stessa, richiede una semplice piegatura del foglio su cui è stampato;

- un opuscolo richiede invece almeno una piegatura e un taglio, e soprattutto la legatura della segnatura con punto metallico o altro metodo;

- un libro, infine, richiede che le segnature di cui è composto vengano legate fra loro (a filo o in altro modo) e quindi il volume rilegato con copertina a parte.

Il modo di produrre in LaTeX pieghevoli, opuscoli e libri è descritto rispettivamente nei §§ 2, 3 e 4. I metodi descritti si prendono automaticamente cura dell'imposizione delle pagine sul foglio per produrre qualsiasi tipo di segnatura. Essendo finalizzati alla stampa artigianale, tuttavia, questi metodi simulano l'imposizione professionale e prevedono sempre la stampa di sole quattro pagine per foglio, due per facciata, senza bisogno di alcun taglio.[4]

3. Con la standardizzazione dei formati dei fogli, i termini "in quarto", "in ottavo", ecc. sono passati spesso a indicare l'altezza media delle pagine del libro che si ottiene usando la relativa segnatura. Nella stampa industriale odierna si usano molto spesso segnature in ottavo e questa espressione indica anche un formato di libro con altezza compresa fra i 20 e i 28 cm. In linea di principio, tuttavia, la segnatura usata è del tutto indipendente dal formato del foglio (e quindi della pagina che si ottiene).

4. Unica eccezione è l'esempio di opuscolo in ottavo descritto nel § 5.5.

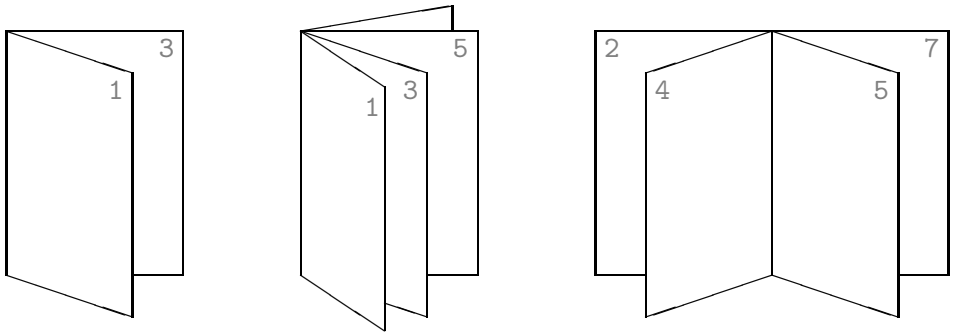


Figura 4: Da sinistra a destra, i tre tipi fondamentali di libretto: un *pieghevole* di quattro pagine (corrispondente a una segnatura in-folio), un *opuscolo* di otto pagine (corrispondente a una segnatura in quarto) e un *libro* di otto pagine (corrispondente a due segnature in-folio legate assieme).

1.2 Pacchetti, classi e altri programmi

In LaTeX, esistono sostanzialmente due strade per produrre un libretto:

- partire da un documento sorgente LaTeX e compilarlo, grazie a pacchetti o classi appositi, in modo che risulti direttamente stampato in forma di libretto;
- partire da un documento pre-esistente e modificarne, con un qualche programma esterno, ordine e disposizione delle pagine per poterlo poi stampare in forma di libretto.

Il secondo metodo è l'unico possibile nei casi in cui non si possieda il sorgente del documento da impaginare (un esempio è considerato nel § 5.4). Naturalmente, è sempre possibile compilare normalmente un documento sorgente e poi trattarlo a posteriori col secondo metodo.[5] La tavola 1 a fronte riassume i pacchetti, le classi e i programmi principali disponibili in LaTeX per stampare libretti secondo i due metodi menzionati (da sorgente o da documento pre-compilato) e dipendentemente dal formato col quale si desidera stampare o da quello del documento a disposizione. Di sèguito, descriviamo brevemente le caratteristiche generali di questi programmi, il cui uso dettagliato verrà spiegato nei paragrafi seguenti.

Booklet

Il pacchetto booklet (Wilson, 2005) permette di compilare un sorgente LaTeX per produrre direttamente un libretto stampabile fronte-retro. Il risultato può essere un semplice pieghevole o un libro vero e proprio completo di segnature, di cui è possibile scegliere il formato. Booklet è stato scritto da Peter Wilson a partire dal codice di 2up (si veda l'appendice B), una vecchia (1991) raccolta di macro per TeX (poi adattata a stile per LaTeX) scritta da Timothy van Zandt con lo scopo generale di stampare più pagine su un unico foglio. Si vedano Wilson (2005) e van Zandt (1993) per le relative documentazioni.

Leaflet

La classe leaflet permette di stampare direttamente da sorgente un pieghevole di sei pagine verticali (tre per lato affiancate) su un unico foglio orizzontale di formato A4 o A3. Leaflet modifica, oltre all'impaginazione, tutto lo stile di un normale documento LaTeX (titoli delle sezioni, elenchi, ecc.). La classe è stata originariamente creata da Jürgen Schlegelmilch, quindi riscritta e sviluppata da Nieprashk *et al.* (2004).

Pdfpages

Il pacchetto pdfpages (Matthias, 2001) permette di inserire ed elaborare in un sorgente LaTeX pagine di un documento esterno in formato PDF. In particolare, può venir usato per riordinare e ridimensionare le pagine di un normale documento in modo da stamparlo come libretto. In questo caso, è possibile ottenere lo stesso risultato del pacchetto booklet da un documento pre-compilato. Booklet e pdfpages possono venir utilizzati assieme, come mostra l'esempio discusso nel § 5.4.

Altri programmi

Oltre ai pacchetti e alle classi sopra descritte, esistono alcuni programmi legati a LaTeX che permettono di rielaborare documenti in formato DVI, PS o PDF per produrre libretti. A conoscenza dell'autore, i principali programmi in questione sono i seguenti.

Dvidvi. Dvidvi[6] è un piccolo programma sviluppato da Tom Rokicki (l'autore di dvips) e Esteban Zimanyi (Rokicki e Zimanyi, 1991) per manipolare le pagine di un documento DVI e crearne uno nuovo con le pagine modificate.

PSUtils. (Duggan, 2000) è il nome di una raccolta di piccoli programmi che permettono di

5. Questo, per inciso, consente di stampare libretti anche con programmi diversi da LaTeX (come Word o OpenOffice), ammesso che si riescano a produrre documenti in formato PS o PDF.

6. ©1989-1991 *Radical Eye Software.*

Formato:	DVI	PS	PDF
da sorgente	booklet, leaflet	booklet, leaflet	booklet, leaflet
da documento pre-compilato	dvidvi	PSUtils	pdfpages (+ booklet), TeXexec, pdfbook

Tabella 1: I principali metodi per creare pieghevoli, libretti e libri in LaTeX relativi al formato del documento da produrre o del documento originale da re-impaginare.

elaborare e modificare documenti in formato PS. In particolare, il programma pstops permette di ruotare, scalare e traslare a piacere le pagine di un documento PS, mentre altri programmi più semplici, quali psbook, psnup e psselect, sono pure utili per produrre libretti e segnature.

TeXexec. TeXexec (Pragma ADE, 2005) è un programma PERL che fa parte della distribuzione di ConTeXt (Hagen, 2000), una moderna estensione del TeX alternativa a LaTeX (si veda Scarso (2005) per un'introduzione). Fra le altre funzionalità, permette di eseguire sostanzialmente tutte le operazioni di pstops delle PSUtils su documenti in formato PDF. Pur non riguardando strettamente il mondo di LaTeX, TeXexec è presente in tutte le distribuzioni di TeX e rappresenta quindi una soluzione disponibile a tutti gli utenti LaTeX.

Pdfbook. Pdfbook (Buchert *et al.*, 2005) è basato su alcuni dei programmi delle PSUtils e ne replica il funzionamento (e la sintassi) per i documenti in formato PDF. Non è tuttavia disponibile per tutti i sistemi operativi, per esempio Microsoft Windows.

Nel sèguito dell'articolo, verranno spiegati nel dettaglio soltanto i pacchetti booklet e pdfpages, e la classe leaflet. Per quanto riguarda i programmi "esterni" a LaTeX menzionati sopra, l'appendice A descrive brevemente la sintassi delle PSUtils, che è abbastanza rappresentativa anche degli altri programmi non considerati.

I sorgenti discussi nei §§ 2-4 possono venir utilizzati come altrettanti *template* per impratichirsi con i relativi pacchetti e classi.

2 Pieghevoli

Un *pieghevole* è uno stampato di poche pagine ottenuto piegando su se stesso un unico foglio di carta e senza bisogno di tagli.[7] In pratica, esistono due tipi di pieghevoli: quelli a quattro pagine (ottenuti piegando in due un unico foglio, cioè corrispondenti a una segnatura in-folio) e quelli a sei pagine (ottenuti piegando in tre un unico foglio). Il programma di un convegno, il menu di una cena, un *dépliant* pubblicitario sono esempi di pieghevole.

2.1 Pieghevoli a quattro pagine

Il metodo più diretto per produrre un pieghevole a quattro pagine è l'uso del pacchetto booklet come segue:

Modello 1: Pieghevole con booklet

```
1  \documentclass[a4paper]{article}
2
3  \usepackage{geometry}
4  \geometry{centering,nohead,nofoot}
5  \geometry{width=108.5mm,height=170mm}
6
7  \usepackage[print,1to1]{booklet}
8
9  \begin{document}
10 \setpdftargetpages
11
12 prima\clearpage
13 seconda\clearpage
14 terza\clearpage
15 quarta
16
17 \end{document}
```

In generale, conviene sempre impaginare il documento in vista del libretto che si andrà a stampare. Ciò significa che le dimensioni del blocco del testo andranno calcolate direttamente per la pagina "rimpicciolita" in modo da adattarsi alla stampa in forma di libretto. Alle righe 3-5 usiamo il pacchetto geometry (Umeki, 2002) per impostare il *layout* della pagina del pieghevole. Lo scopo finale è ottenere due pagine di formato A5 affiancate su un foglio di formato A4. Specifichiamo che il testo andrà centrato nella pagina (centering) e che non abbiamo bisogno di spazio per le testatine e per i piè di pagina (nohead,nofoot). Quindi specifichiamo le dimensioni del blocco del testo: in questo caso, sono state determinate sottraendo alle dimensioni di una pagina A5 (148.5×210 mm) due margini per lato da 2 cm ciascuno. (Naturalmente tutte queste impostazioni verranno di volta in volta personalizzate.)

Alla riga 7 usiamo il pacchetto booklet. L'opzione print dice a booklet di impostare direttamente le pagine del documento in forma di pieghevole, cioè appunto due per facciata su un foglio orizzon-

7. Un *volantino* è invece un foglio semplice (normalmente di dimensioni ridotte) adatto per la distribuzione. Dato che non richiede piegature o tagli, né altre operazioni particolari oltre alla stampa fronte-retro, non viene trattato in questa sede.

tale.[8] L'opzione 1to1 dice invece a booklet che il testo del pieghevole è già stato progettato (grazie a geometry) per essere sistemato su due pagine affiancate su un foglio A4 orizzontale. In generale, è sempre meglio progettare fin dall'inizio la pagina di un libretto, piuttosto che utilizzare le dimensioni standard e lasciare che sia booklet a tentare di calcolare il corretto ridimensionamento.

Il comando `\setpdftargetpages` (riga 10) è necessario se la compilazione produce un documento PDF (per esempio se si compila con PDFLATEX). Un comando simile, `\setdvipstargetpages`, è disponibile per produrre documenti in formato PS. Si noti che, dato che geometry imposta il layout all'inizio del documento, questi comandi vanno richiamati *dopo* `\begin{document}` affinché booklet registri i cambiamenti. Allo stesso modo, booklet va sempre richiamato dopo geometry.

Il sorgente descritto sopra produce un pieghevole correttamente stampabile fronte-retro, le cui pagine contengono l'unico testo "prima", "seconda", "terza" e "quarta" oltre al numero della pagina stessa. Un esempio leggermente più interessante è descritto nel § 5.1.

Se per qualche motivo si preferisce o si deve produrre il pieghevole utilizzando un documento precompilato, è possibile usare il pacchetto pdfpages. Supponendo di avere un documento `original.pdf` contenente le quattro pagine del pieghevole, si crea un nuovo sorgente sul seguente modello:

Modello 2: Pieghevole con pdfpages

```
1  \documentclass[a4paper]{article}
2
3  \usepackage{pdfpages}
4
5  \begin{document}
6
7  \includepdf[pages=-,signature=4,%
8              landscape]{original}
9
10 \end{document}
```

Alla riga 7 il comando `\includepdf` — che prende come argomento il nome del documento PDF — inserisce nel nuovo documento le pagine prescelte (in questo caso tutte) del documento originale. L'opzione signature serve a specificare quante pagine vanno su ogni segnatura (in questo caso 4, le uniche presenti), mentre l'opzione landscape (riga 8) specifica che il foglio deve essere orizzontale. Il risultato è simile a quello ottenuto con booklet, ma può essere necessario aggiustare al meglio le dimensioni delle pagine inserite, che vengono calcolate automaticamente da pdfpages (si veda Matthias (2001) per i vari comandi a disposizione).

8. Il fatto che print non sia l'opzione di default dipende dalla gestione dei riferimenti incrociati in documenti più complessi di un semplice pieghevole, che verrà spiegata nel § 3.

2.2 Pieghevoli a sei pagine

La classe leaflet permette di stampare direttamente da sorgente pieghevoli a sei pagine. Un esempio è il seguente:

Modello 3: Pieghevole con leaflet

```
1  \documentclass[a4paper,notumble]%
2                {leaflet}
3
4  \begin{document}
5
6  prima\clearpage
7  seconda\clearpage
8  terza\clearpage
9  quarta\clearpage
10 quinta\clearpage
11 sesta
12
13 \end{document}
```

Leaflet si prende cura automaticamente delle dimensioni e dell'ordinamento delle pagine; l'opzione di classe notumble — che fa sì che il retro non sia rovesciato, come di default, rispetto al fronte[9] — è stata usata (riga 1) solo per mostrare il risultato della compilazione nella figura 5 nella pagina successiva. Si noti la disposizione delle pagine. Leaflet inserisce automaticamente fra la seconda e la terza pagina un piccolo segno (personalizzabile) per la prima piegatura (la seconda piegatura risulta determinata dalla prima); altre opzioni di questa classe molto flessibile saranno illustrate nel § 5.2.

Alternativamente, è possibile progettare manualmente le pagine del pieghevole (per esempio con geometry) e quindi "montare" il risultato sul pieghevole stesso usando il pacchetto pdfpages. Un esempio è il seguente. Dapprima si produce, usando per esempio geometry, il documento `original.pdf` con sei pagine di dimensioni 99×210 mm (99 mm è un terzo di 297 mm, la larghezza di un normale foglio A4 orizzontale); quindi si inseriscono in un nuovo file simile a questo:

Modello 4: Pieghevole (6 pagine) con pdfpages

```
1  \documentclass[a4paper]{article}
2
3  \usepackage{pdfpages}
4
5  \begin{document}
6
7  \includepdf[pages={5,6,1,2,3,4},%
8              nup=1x3,%
9              landscape]{original}
10
11 \end{document}
```

Invece di usare l'opzione signature (valida solo per i libretti contenenti multipli di 4 pagine), si usa

9. L'orientamento della facciata sul retro rispetto a quella sul fronte dipende dal tipo di stampante usata; andrà quindi adattato alla propria stampante (ciò vale in generale per tutti gli esempi di libretto considerati).

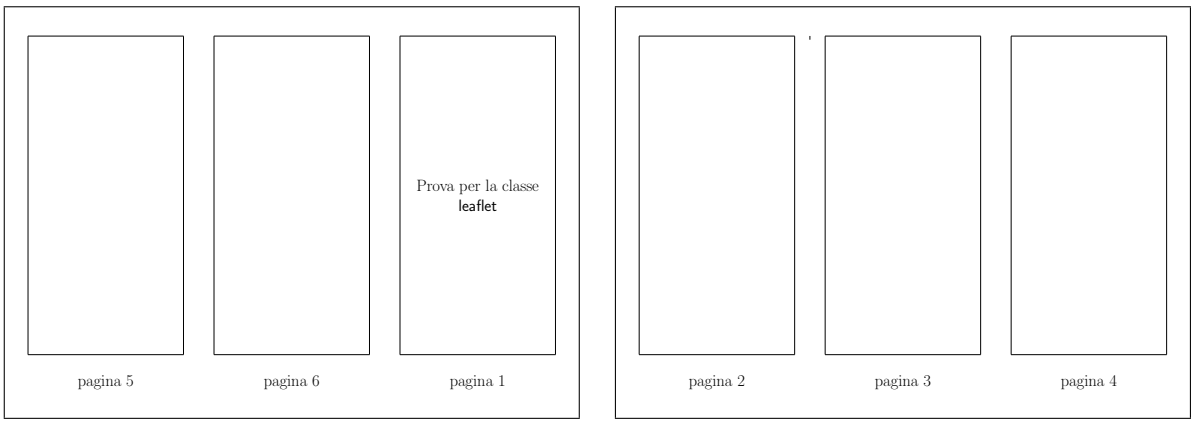


Figura 5: Fronte e retro di un pieghevole a sei pagine prodotto con la classe leaflet (si veda il codice 3 nella pagina precedente).

(riga 8) l'opzione esplicita nup, che specifica quante pagine mettere in orizzontale e quante in verticale sul foglio. Si noti che è necessario specificare esplicitamente (riga 7) l'ordine in cui le pagine devono essere inserite.

3 Opuscoli

Un *libretto* od *opuscolo* è uno stampato costituito da un'unica segnatura di poche pagine, ma più corposo di un semplice pieghevole.[10] La presenza di un'unica segnatura permette di rilegare il libretto con una semplice pinzatrice, dopo aver piegato in due la segnatura stessa. Normalmente, la copertina di un libretto è costituita dal primo foglio del volume (magari in cartoncino più pesante e/o colorato) e viene pinzata assieme al resto.

Il numero di pagine di un libretto è limitato proprio da questo metodo molto elementare di rilegatura. Infatti, con l'aumentare del numero dei fogli da piegare, aumenta velocemente l'effetto di scalatura del bordo esterno, che andrebbe poi rifilato con una taglierina. Già un libretto di poche decine di pagine presenta questo problema in modo piuttosto vistoso.

Utilizzando il pacchetto booklet è possibile produrre libretti di ogni tipo semplicemente usando il codice 1 per il pieghevole da quattro pagine. L'unica differenza è che il numero di pagine di un libretto varia e deve essere specificato. Booklet ha quattro opzioni di classe, four, eight, sixteen e thirtytwo (che è quella di default) che specificano quante pagine per segnatura contiene il libretto (4, 8, 16 o 32). Col comando `\pagespersignature` è possibile specificare un numero di pagine diverso da questi — che deve comunque essere un multiplo di 4 — secondo le proprie esigenze.

Finché le pagine dell'opuscolo sono meno di 32, è possibile usare booklet esattamente come nel caso del pieghevole. Se invece le pagine sono di più, se ne approssima il numero per eccesso al primo multiplo di 4 e lo si specifica come argomento di `\pagespersignature`. Se per esempio avessimo un documento di 33 pagine, l'opuscolo relativo dovrà contenere almeno 36 pagine (delle quali le ultime tre vuote). Si useranno quindi i seguenti comandi:

Modello 5: Opuscolo con booklet

```
1  \documentclass[a4paper]{article}
2
3  \usepackage{geometry}
4  \geometry{centering,nohead}
5  \geometry{width=108.5mm,height=170mm}
6
7  %\usepackage[noprint]{booklet}
8  \usepackage[print,1to1]{booklet}%
9   \nofiles
10 \pagespersignature{36}
11
12 \begin{document}
13 \setpdftargetpages
14
15 % <pagine 1–33>
16
17 \end{document}
```

Alle righe 7-10 richiamiamo il pacchetto booklet in modo leggermente diverso rispetto al caso del pieghevole a quattro pagine (§ 2.1). Il motivo per le righe 7 e 9 è che un opuscolo conterrà in generale, al contrario di un semplice pieghevole, riferimenti incrociati, per esempio un indice iniziale o altro. Affinché tali riferimenti risultino corretti, occorre inizialmente compilare alcune volte con l'opzione noprint di booklet (riga 7), che non ha alcun effetto visibile; infine, si compilerà *una volta sola* con l'opzione print *e* il comando `\nofiles` (riga 9), che dice a LaTeX di usare i vecchi file relativi ai riferi-

10. Spesso questi libretti vengono chiamati *brochure* (dal francese) o brossure; tuttavia, "brossura" si riferisce anche (e soprattutto) a un tipo economico, e molto diffuso, di rilegatura, in cui la copertina di cartoncino non rigido viene semplicemente incollata al volume (legato o meno). Secondo lo *Zingarelli 2003* (Zanichelli, Le Opere su Cd-Rom, Bologna) un opuscolo non contiene in genere più di 80 pagine.

menti incrociati (`.aux`,`.toc`, ecc.) senza generarne di nuovi.

Infine, alla riga 10 diciamo a booklet che l'unica segnatura di cui è composto l'opuscolo deve contenere 36 pagine; il pacchetto aggiunge automaticamente tre pagine vuote, per completare il libretto, e produce 18 facciate (36/2) che andranno poi stampate su 9 fogli (36/4).[11] Un esempio di opuscolo prodotto in questo modo è discusso nel § 5.3.

Anche nel caso di pdfpages, è sufficiente usare lo stesso sorgente per il pieghevole a quattro pagine specificando il numero di pagine della segnatura desiderata; supponendo che `original.pdf` contenga 33 pagine, si scriverà (riga 7):

Modello 6: Opuscolo con pdfpages

```
1 \documentclass[a4paper]{article}
2
3 \usepackage{pdfpages}
4
5 \begin{document}
6
7 \includepdf[pages=-,signature=36,%
8              landscape]{original}
9
10 \end{document}
```

per avere il risultato desiderato.

4 Libri

Al crescere del numero delle pagine di un documento, diventa fisicamente impossibile rilegarlo in forma di opuscolo. Il blocchetto di fogli stampati infatti diventa troppo grosso per poter esser semplicemente piegato e puntato. Diventa allora necessario suddividere le pagine in varie segnature.

Supponiamo di avere un libro di 107 pagine e volerlo rilegare *in ottavo*, cioè con segnature da 16 pagine l'una. I comandi visti sopra, sia per booklet sia per pdfpages sono sufficienti a ottenere il risultato voluto. Per controllare il risultato, possiamo già prevedere che la prima pagina verrà stampata affiancata alla sedicesima (questa sarà la prima facciata della prima segnatura), che la diciassettesima pagina sarà stampata a fianco della trentaduesima (prima facciata della seconda segnatura) e così via.

Nel caso di booklet avremo quindi:

Modello 7: Libro con booklet

```
1 \documentclass[a4paper]{article}
2
3 \usepackage{geometry}
4 \geometry{centering,nohead}
5 \geometry{width=108.5mm,height=170mm}
6
7 %\usepackage[noprint]{booklet}
8 \usepackage[print,1to1]{booklet}
9 \nofiles
10 \pagespersignature{16} % in ottavo
11
12 \begin{document}
13 \setpdftargetpages
14
15 % <pagine 1-107>
16
17 \end{document}
```

Nel caso di pdfpages avremo invece:

Modello 8: Libro con pdfpages

```
1 \documentclass[a4paper]{article}
2
3 \usepackage{pdfpages}
4
5 \begin{document}
6
7 \includepdf[pages=-,signature=16,%
8              landscape]{original}
9
10 \end{document}
```

È interessante notare una differenza nel comportamento dei due pacchetti. Booklet produce infatti, per un libro di 107 pagine in ottavo, 54 facciate da stampare su 17 fogli, che corrispondono a sei segnature da 16 pagine più una settima segnatura da sole 12 pagine (l'ultima bianca). Pdfpages, invece, produce 56 facciate da stampare su 18 fogli, corrispondenti a sette segnature da 16 pagine, l'ultima delle quali contiene in coda cinque pagine bianche.

La differenza è dovuta al modo in cui i due pacchetti completano le segnature. Entrambi infatti calcolano automaticamente il numero di pagine mancanti necessario per stampare il libro: tuttavia booklet minimizza il numero di fogli totale, producendo eventualmente segnature "monche", mentre pdfpages produce comunque le segnature nel formato richiesto riempiendo quelle incomplete con pagine bianche.

In questo caso, per stampare in ottavo — cioè, ricordiamolo, 16 pagine per segnatura — un libro di 107 pagine sono necessarie almeno sette segnature. Infatti sei segnature possono contenere solo le prime $16 \times 6 = 96$ pagine del libro: le restanti 11 richiedono una segnatura aggiuntiva. Questa settima segnatura può essere uguale alle altre, cioè contenere 16 pagine, o contenerne solo 12, lo stretto indispensabile per sistemare le 11 pagine rimaste. Nel primo caso, che è quello di pdfpages, il libro sarà infine composto da $16 \times 7 = 112$ pagine, le ultime cinque delle quali vuote. Nel secondo caso, quello di booklet, il libro conterrà invece solo 108 pagine ($(16 \times 6) + 12$), l'ultima delle quali vuota.

11. Al momento (2 novembre 2006) l'uso di `\pagespersignature` sembra talvolta causare l'aggiunta di una o più facciate spurie in coda all'opuscolo; d'altra parte, i fogli aggiuntivi così prodotti sono completamente vuoti e possono essere semplicemente eliminati (o non stampati) prima della rilegatura. Peter Wilson, autore di booklet, è al corrente del problema (comunicazione personale).

112 è infatti il primo multiplo di 16 superiore a 107, 108 il primo multiplo di 4.

5 Esempi commentati

Alcuni esempi di pieghevole (§ 5.2 e § 5.1), di opuscolo (§ 5.3) e di libro (§ 5.4) vengono presentati e discussi. Ogni esempio è corredato da una figura che mostra parte del risultato finale; vengono inoltre introdotti alcuni nuovi comandi relativi ai pacchetti già visti. Il § 5.5 spiega come ottenere un'autentica segnatura in ottavo, cioè come stampare ordinatamente sedici pagine su un unico foglio.

5.1 Una partecipazione per matrimonio

La figura 6 nella pagina seguente riproduce un biglietto di partecipazione per matrimoni più o meno standard. La partecipazione originale era stampata su un cartoncino rifilato di dimensioni 170 × 230 mm, con margini su ogni pagina di circa 15 mm. Diversamente dai normali pieghevoli a quattro pagine, una partecipazione prevede di stampare fronte-retro quattro pagine orizzontali su un foglio in verticale.

Questo risultato è stato ottenuto con booklet, con il seguente preambolo:

Esempio 1: Partecipazione per matrimonio

```
1  \documentclass[14pt]{extarticle}
2
3  \usepackage[centering,nohead,nofoot]%
4                {geometry}
5  \geometry{paperwidth=230mm,%
6            paperheight=170mm}
7  \geometry{textwidth=140mm,%
8            textheight=85mm}
9
10 \usepackage[print,1to1,landscape]%
11               {booklet}
12 \setlength{\pagesepwidth}{.4pt}
13 \setlength{\pageseplength}{3mm}
14 \setlength{\pagesepoffset}{0mm}
15
16 \pagestyle{empty}
17 \setlength{\parindent}{0pt}
18 \begin{document}
```

Le uniche differenze rispetto al codice base per il pieghevole a quattro pagine (§ 2.1) sono nelle specifiche di pagina.

Alle righe 5-6 usiamo geometry per specificare le dimensioni del foglio (che verrà stampato su un A4 e poi rifilato. Alle righe 7-8 specifichiamo le dimensioni del corpo del testo: prima dimezziamo il foglio per ottenere le dimensioni della pagina (115 × 170 mm), quindi togliamo 15 mm di margine per lato. Infine, alla riga 10, con l'opzione landscape diciamo a booklet di aspettarsi pagine in orizzontale da sistemare su fogli in verticale. (Equivalentemente, landscape può venir specificata come opzione di classe). Il segno di piegatura parte dal margine fisico del foglio (riga 14).

Il carattere usato è quello fornito dal pacchetto calligra; le due iniziali sulla prima pagina sono state inserite coi pacchetti standard graphicx e xcolor.

5.2 Un pieghevole a sei pagine

La figura 7 nella pagina successiva presenta un esempio di pieghevole a sei pagine ottenuto con la classe leaflet.[12] Rispetto al codice di prova del § 2.2, il sorgente relativo usa alcuni nuovi comandi messi a disposizione da leaflet. La porzione rilevante del codice della figura 7 è la seguente:

Esempio 2:]Pieghevole GUIT*meeting*

```
1  \documentclass[a4paper,notumble]%
2                {leaflet}
3
4  \CutLine{6} % tagli pp. 4–5
5  \CutLine{4}
6
7  %\AddToBackground{2}{% sfondo p. 2
8  \AddToBackground*{2}{% sfondo pp. 2–4
9  \includegraphics[width=.66\paperwidth]
10                 {gmlogo.jpg}}
11
12 \begin{document}
```

Alle righe 4-5 usiamo il comando `\CutLine` che inserisce una linea tratteggiata di divisione fra due pagine del pieghevole, con il simbolo delle forbici per indicare il taglio. La linea viene inserita immediatamente prima (cioè a sinistra guardando la facciata del pieghevole) della pagina indicata nell'argomento di `\CutLine`. La versione asteriscata del comando, `\CutLine*`, inserisce una semplice linea tratteggiata senza forbici.

Alla riga 9, inseriamo l'immagine `gmlogo.jpg` come sfondo della pagine 2 e 3 del pieghevole. Il comando relativo, `\AddToBackground*`, prende come primo argomento il numero della *facciata* (1 o 2) del pieghevole su cui inserire la figura, e come secondo il nome del file-immagine. Le dimensioni dell'immagine di sfondo vanno determinate manualmente. La versione non asteriscata prende invece come primo argomento il numero della *pagina* (1-6) in cui inserire la figura. Si noti la differenza fra i due comandi togliendo il commento alla riga 7 e commentando la riga 8.

Il segno della piegatura può essere eliminato con l'opzione di classe nofoldmark o personalizzato cambiando, con `\renewcommand`, i parametri `\foldmarkrule` (per lo spessore della linea) e `\foldmarklength` (per la sua lunghezza).

Infine, si noti, alle pagine 2-5 del pieghevole in figura 7 nella pagina seguente, come leaflet diminuisca le dimensioni del carattere per i titoli delle

12. Ringrazio Emanuele Vicentini del GUIT per avermi messo a disposizione il codice originale del pieghevole per il GUIT*meeting*.

Figura 6: Le due facciate di un invito di partecipazione da matrimonio, prodotte col pacchetto booklet (vedi codice 1 nella pagina precedente).

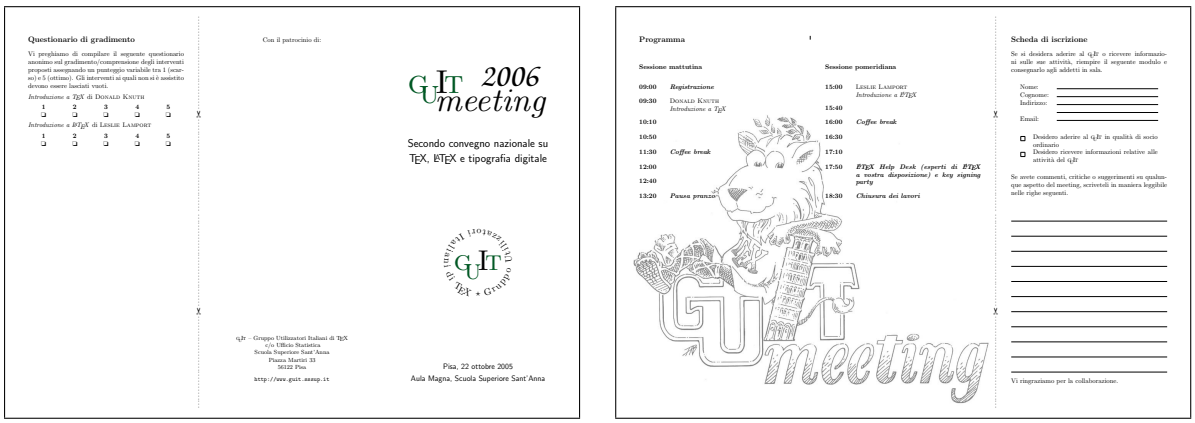


Figura 7: Le due facciate di una bozza del pieghevole per il Meeting del GUIT di quest'anno, prodotto con la classe leaflet (esempio 2 nella pagina precedente).

`\(sub)section` e sopprima il numero di paragrafo progressivo.

5.3 Un opuscolo

La figura 8 nella pagina successiva mostra la prima (fronte) e l'ultima (retro) facciata di un opuscolo. Il libretto in questione è il *De moneta* di Nicola Oresme, un trattato medioevale dedicato alla moneta e alla sua falsificazione da parte del sovrano (signoraggio). Il libro di Oresme è disponibile nella *Bibliotheca Augustana* all'indirizzo `http://www.fh-augsburg.de/%7Eharsch/Chronologia/Lspost14/Oresme/ore_mon0.html`.

Una volta impaginato con la classe book di LaTeX, su una pagina di formato A5 e con carattere 10pt, il libretto riempie 41 pagine, compresa la prima e la quarta pagina di copertina. Occorre dunque una segnatura da 44 pagine per contenerlo. Dato che la quarta di copertina deve risultare affiancata alla prima, per produrre in stampa la copertina completa, abbiamo completato "a mano" il libretto aggiungendo esplicitamente tre pagine per riempire la segnatura.

La porzione rilevante del codice relativo è la seguente:

Esempio 3: Nicola Oresme, *De moneta*

```
1  \documentclass[a4paper,10pt,twoside]%
2                {book}
3
4  \usepackage[latin]{babel}
5  \usepackage[text={104mm,147mm}]%
6              {geometry}
7
8  %\usepackage[noprint]{booklet}
9  \usepackage[print,1to1]{booklet}%
10  \nofiles
11 \pagespersignature{44}
12
13 \begin{document}
14 \setpdftargetpages
15
16 % <pagine 1--40>
17
18 % pagine 41--43:
19 \clearpage\null\thispagestyle{empty}
20 \clearpage\null\thispagestyle{empty}
21 \clearpage\null\thispagestyle{empty}
22 % pagina 44 (quarta di copertina):
23 \clearpage\thispagestyle{empty}
24 % <testo>
25 \end{document}
```

Alle righe 5-6 specifichiamo le dimensioni del blocco del testo (circa uguali a quelle standard di geometry per il formato A5); lasciamo che i margini vengano calcolati automaticamente da geometry, cioè col rapporto 2 : 3 sia in orizzontale (interno : esterno) sia in verticale (superiore : inferiore).

Alle righe 8-11 usiamo il pacchetto booklet, specificando con `\pagespersignature` che l'opuscolo sarà costituito da un'unica segnatura di 44 pagine. Dato che il libretto conterrà un indice iniziale dei capitoli, è qui essenziale compilare alcune volte con l'opzione noprint (riga 8) e fare *solo l'ultima* compilazione con l'opzione print *e* il comando nofiles (righe 9-10), che "costringe" booklet a usare il vecchio file di indice (corretto) e a generare i numeri di pagina giusti.

Contenendo solo 41 pagine, l'opuscolo è stato completato con tre pagine vuote (righe 19-21), alle quali è stata aggiunta in coda la quarta di copertina, che nella stampa risulterà affiancata alla prima pagina del libretto. La copertina dell'opuscolo è stata creata ritagliando l'immagine secondo le proporzioni del formato ISO ($1 : \sqrt{2} = 0,707$) e inserendola sullo sfondo delle due pagine col pacchetto eso-pic. Date queste proporzioni, è sufficiente richiamare il file immagine col comando `\includegraphics` del pacchetto graphicx scalandola del fattore relativo alla serie ISO: `[width=.707\paperwidth]`.

5.4 Una ristampa anastatica

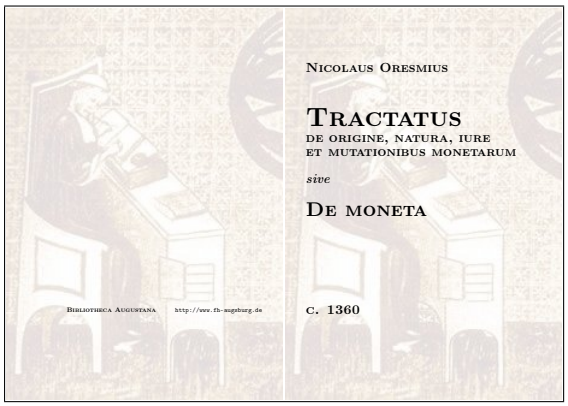
Una copia o ristampa *anastatica* di un libro consiste nella riproduzione fedele del documento originale, senza che ne venga modificata l'impaginazione, la numerazione e tutte le principali caratteristiche. Ogni pagina del libro originale viene riprodotta (originariamente con un procedimento litografico) e stampata esattamente come è, con l'unico eventuale cambiamento del formato del libro.

Un esempio di ristampa anastatica è illustrato nella figura 9 nella pagina seguente, che mostra le prime due facciate del classico libretto di Henry Hazlitt, *Economics in one lesson*. Il libro è disponibile in formato PDF all'indirizzo `http://www.economicsinonelesson.com` ed è stato ottenuto dalla scansione delle pagine di una copia originale. Il documento contiene 198 pagine; in questo caso, l'abbiamo riprodotto all'interno di un nuovo documento LaTeX, aggiungendo due pagine per il frontespizio e le informazioni editoriali e infine prevedendo una rilegatura con segnature in-16°. Il risultato è un libretto di 200 pagine suddivise in sei segnature da 32 pagine l'una più una segnatura finale in-4°, cioè da 8 pagine.

La parte rilevante del codice relativo è la seguente:

Esempio 4: Hazlitt, *Economics in One Lesson*

```
1  \documentclass[a4paper,11pt]{book}
2
3  \usepackage[text={108mm,170mm},%
4              centering]{geometry}
5  \usepackage{pdfpages}
6  %\usepackage[noprint]{booklet}
7  \usepackage[print,1to1]{booklet}
8  \pagespersignature{32} % in--16°
9
10 \begin{document}
11 \setpdftargetpages
12
```



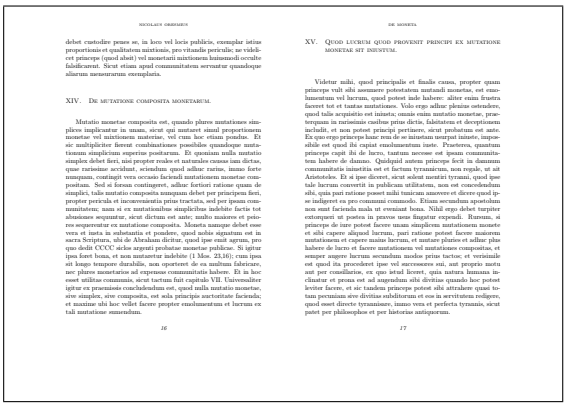


Figura 8: La prima (copertina) e l'ultima facciata di un opuscolo di 44 pagine, prodotto col pacchetto booklet (esempio 3 nella pagina precedente).

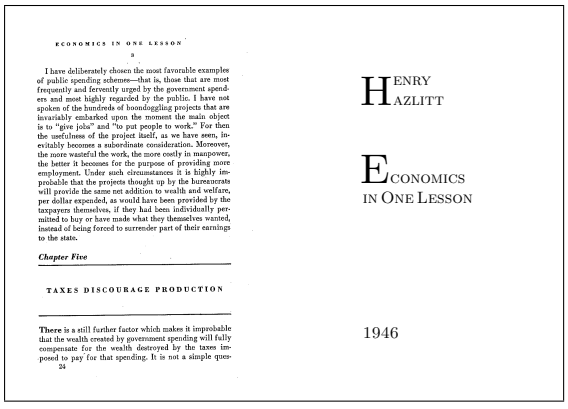


Figura 9: Le prime due facciate (fronte e retro) di una ristampa anastatica prodotta coi pacchetti booklet e pdfpages (vedi 4 nella pagina precedente).

```
13 \frontmatter
14 % Frontespizio:
15 \begin{titlepage}
16  % <autore, titolo, data>
17 \end{titlepage}
18 % Informazioni editoriali:
19  % <info>
20 \mainmatter
21 % Libro:
22 \includepdf[pages=-,noautoscale,%
23              scale=1.2]{hazlitt}
24 \end{document}
```

Alla riga 8 impostiamo una segnatura in sedicesimo; ciò significa che dovremo piegare e legare fascicoli da 32 pagine di otto fogli l'uno.

Il frontespizio, cioè la prima pagina dopo la copertina vera e propria (che verrà creata alla fine delle dimensioni adeguate per poter rilegare il libro), viene inserito alle righe 15-17; in questo caso si è usato il pacchetto lettrine per i capilettera. Le informazioni editoriali (relative alla prima edizione del libro e a quella qui ristampata) vengono poste sul retro del frontespizio (riga 19). Queste pagine "a parte" rispetto al libro vero e proprio vengono relegate nel `\frontmatter`.

Il libro originale viene inserito pagina per pagina usando pdfpages (righe 22-23); per ottenere una migliore sistemazione sulla nuova pagina si è disattivato il ridimensionamento automatico del pacchetto (noautoscale, riga 22): il fattore desiderato, ottenuto per tentativi, è stato specificato con l'opzione scale (riga 23). In questo modo si ottiene un nuovo documento di esattamente 200 pagine.

Infine, usando booklet (riga 7) si imposta l'intero documento in forma di libretto. Il risultato sono 100 facciate da stampare su 50 fogli suddivisi in sei segnature contenenti otto fogli e una settima di soli due fogli.

5.5 Una segnatura in ottavo

Tutti gli esempi finora discussi hanno sfruttato la capacità dei pacchetti booklet e pdfpages di *simulare* l'imposizione professionale, cioè la stampa di segnature di più pagine. Tuttavia, con l'eccezione dei pieghevoli (che costituiscono una segnatura di per sé), quelle prodotte non sono "vere" segnature, stampate su un unico foglio, ma appunto simulano il risultato della piegatura e del taglio della segnatura vera e propria.

A puro titolo illustrativo, mostriamo come ottenere una "vera" segnatura in ottavo stampando 16 paginette di formato A6 su un normale foglio A4, che viene poi ripiegato tre volte su se stesso e tagliato lungo le prime due piegature.

Supponiamo di avere un documento `original.pdf` di 16 pagine. Chiamiamo `rovescia.tex` il seguente sorgente:

```
1 \documentclass[a4paper]{report}
2
3 \usepackage{pdfpages}
4
5 \begin{document}
6
7 \includepdf[pages={1-4}]{original}
8 \includepdf[pages={5-12},angle=180]%
9             {original}
10 \includepdf[pages={13-16}]{original}
11
12 \end{document}
```

Compilando `rovescia.tex`, pdfpages inverte di 180 gradi (testa-piedi) le otto pagine centrali del documento originale: otteniamo così `rovescia.pdf` che è identico a `original.pdf` a parte per le otto pagine rovesciate.

A questo punto scriviamo un nuovo sorgente come segue:

```
1 \documentclass[a4paper,landscape]%
2                {report}
3
4 \usepackage{pdfpages}
5
6 \begin{document}
7
8 \includepdf[pages={11,6,7,10,
9                   14,3,2,15,
10                  9,8,5,12,
11                  16,1,4,13},
12                  nup=4x2]
13                  {rovescia}
14
15 \end{document}
```

che, sempre usando pdfpages, riordina le pagine di `rovescia.pdf` (righe 8-11) e le inserisce su un foglio A4 orizzontale (riga 1), otto per facciata (riga 12).

Un esempio del risultato è riportato in figura 10, dove qè stata usata l'opzione frame di pdfpages per sottolineare la divisione delle pagine. L'ordine delle pagine e il loro orientamento (dritte o rovesciate) va calcolato a mano; il modo migliore per farlo è quello empirico: si prende un foglio A4, lo si piega per tre volte sul lato maggiore, quindi si scrive il numero di pagina progressivo su ognuna delle pagine così ottenute; infine si riapre il foglio che mostra sulle due facciate il fronte e il retro della segnatura desiderata.

A Libretti con le PSUtils

Alcuni dei programmi delle PSUtils permettono di replicare tutti gli esempi finora discussi lavorando con documenti in formato PS. In particolare, il programma pstops è in grado di eseguire tutte le operazioni necessarie per produrre qualsiasi tipo di segnatura e quindi di libretto. Tuttavia, spesso conviene rivolgersi a programmi più limitati ma più semplici (come psselect, psbook e psnup) per velocizzare il lavoro.

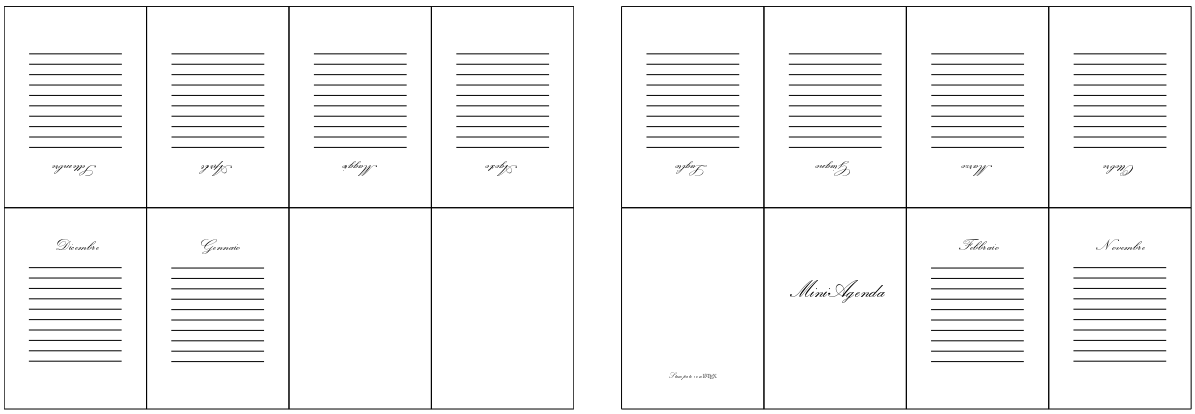


Figura 10: Le due facciate di segnatura in ottavo prodotta con pdfpages (vedi § 5.4). Stampandola fronte-retro su un foglio A4, piegandolo tre volte sul lato maggiore e tagliando lungo le prime due pieghe, si ottiene un'agendina tascabile in formato A6.

Pieghevoli

Supponendo che il documento originale sia `original.ps`, si useranno i comandi:

```
psbook original.ps riordina.ps
```

che dispone nel corretto ordine le quattro pagine originali; e

```
psnup -2 riordina.ps libretto.ps
```

che sistema due pagine per lato del foglio producendo il risultato voluto.

Un modo per produrre un pieghevole a sei pagine, a parte eventuali ridimensionamenti, è il seguente:

```
psselect -p5,6,1,2,3,4
         original.ps riordina.ps
```

che dispone le pagine nell'ordine voluto (§ 2.2); e

```
psnup -3 riordina.ps pieghevole.ps
```

che imposta tre pagine per facciata del foglio finale.

Opuscoli

Dato che gli opuscoli sono costituiti da una sola segnatura, i comandi per produrli sono gli stessi relativi ai pieghevoli: psbook riordina le pagine del documento originale e psnup (con l'opzione -2) le monta a due a due su un foglio orizzontale.

Ancora una volta, lo stesso risultato si può ottenere con pstops, ma più scomodamente dato che occorre differenziare il trattamento per le pagine che andranno sul fronte e quelle che andranno sul retro dei fogli stampati (Duggan, 2000, pstops, p. 2).

Libri

L'opzione `-s` di psbook permette di specificare il numero di pagine di ogni segnatura, e quindi di stampare libri veri e proprio formati da un certo numero di segnature.

Per esempio, la seguente riga di comando specifica una segnatura in quarto da otto pagine:

```
psbook -s8 original.ps riordina.ps
```

Come al solito, per avere il libretto finale da stampare occorre impostarlo con psnup:

```
psnup -2 riordina.ps libretto.ps
```

Imposizioni generiche con pstops

Il programma pstops permette di manipolare le pagine di un documento PS in una notevole varietà di modi. In particolare, permette contemporaneamente di traslare, ruotare e scalare (in quest'ordine) ogni pagina indipendentemente dalle altre e impostarle su un numero a piacere di fogli. Di conseguenza, pstops permette da solo di eseguire tutte le operazioni necessarie per produrre libretti di ogni tipo, comprese "vere" segnature come quella del § 5.5 (si veda per esempio Nermander (2005)).

La sintassi di pstops è la seguente:

```
pstops [specifiche] [file1.ps] [file2.ps]
```

dove `file1.ps` è il documento originale e `file2.ps` è il libretto da ottenere. Il preambolo `[specifiche]` è a sua volta così formato:

```
[bloccopagine]:[[pagina]comandi]
```

Il documento da processare (`file1.ps`) viene suddiviso da pstops in *blocchi*: `bloccopagine` specifica quante pagine sono contenute in ogni blocco (almeno una, al massimo il numero di pagine del documento). Per ogni `pagina` di ogni blocco è possibile eseguire una serie di `comandi`. Si noti tuttavia che pstops numera le pagina del documento a partire da 0: quindi per indicare pagina 1 occorrerà scrivere 0, per la 2 scrivere 1, e così via.

I `comandi` che ci interessano sono sostanzialmente tre:

- `L|R|U` ruota la pagina indicata a sinistra (L, senso anti-orario), destra (R, senso orario) o testa-piedi (U). La rotazione avviene attorno all'angolo in basso a sinistra della pagina.

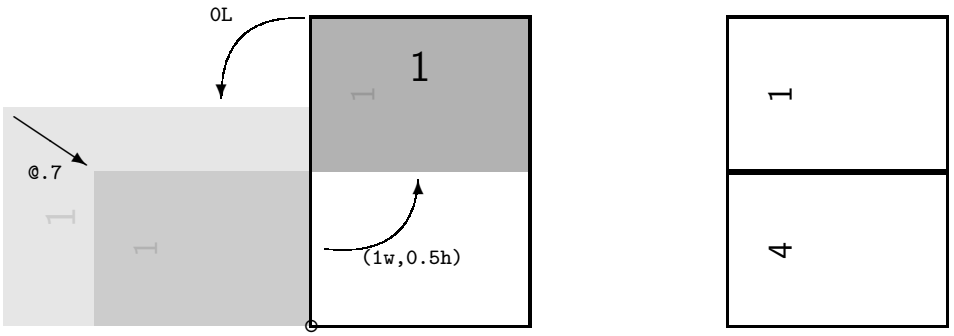


Figura 11: Illustrazione degli effetti del comando `pstops 4:0L@.7(1w,0.5h)`, parte dei comandi per produrre un pieghevole di quattro pagine (vedi appendice A). Sulla sinistra la pagina originale, sulla destra la facciata che si vuole ottenere. Ogni freccia mostra una delle tre operazioni di rotazione, scalatura e traslazione della prima pagina.

- `@fattore` ridimensiona la pagina scalandola del `fattore` indicato, che può essere qualsiasi frazione (per esempio: `@.5` per un dimezzamento di entrambi i lati della pagina).

- `(deltax,deltay)` trasla la pagina in orizzontale e in verticale della quantità indicata, che può essere espressa in termini assoluti (per esempio `(1cm,0)`) o in termini relativi alle dimensioni del foglio originale (per esempio `(1w,1h)`, dove `w` e `h` indicano rispettivamente la larghezza e l'altezza del foglio).

Per ogni blocco, occorre specificare le opzioni per ogni pagina del blocco: se queste vengono divise dal segno +, le pagine vengono montate sullo stesso foglio nella posizione specificata; se vengono separate dalla virgola , vengono invece disposte su fogli diversi (sempre nella posizione specificata.)

Come esempio, la seguente riga di comando permette di produrre un pieghevole di quattro pagine a partire dal documento `original.ps`:

```
pstops 4:3L@.7(21cm,0cm)+
       0L@.7(21cm,14.85cm),
       1L@.7(21cm,0cm)+
       2L@.7(21cm,14.85cm)
        original.ps libretto.ps
```

Equivalentemente, il seguente comando usa dimensioni relative alla larghezza (`w`) e all'altezza (`h`) della pagina invece che dimensioni assolute:

```
pstops 4:3L@.7(1w,0)+
       0L@.7(1w,0.5h),
       1L@.7(1w,0.5h)+
       2L@.7(1w,0.5h)
        original.ps libretto.ps
```

La figura 11 illustra una fase del processo di impostazione, relativa alla prima pagina del pieghevole, e dovrebbe aiutare a capire meglio la sintassi di `pstops`.

B Altri pacchetti LaTeX

Descriviamo brevemente altre soluzioni disponibili in LaTeX per produrre libretti. La lista dei seguenti pacchetti e classi è tratta soprattutto da Williams (2006, Topical: "Office" Applications, Leaflets).

2up

Il pacchetto 2up (van Zandt, 1993) è distribuito sia come insieme di macro per TeX (`2up.tex`) sia come pacchetto LaTeX (`2up.sty`). È il "progenitore" di booklet. Chiamato twoup da Williams (2006).

2in1

Il pacchetto 2in1 (Saxena, 2001) imposta due pagine verticali fianco a fianco su un foglio orizzontale; non sembra funzionare con PDFLaTeX. Chiamato twoinone da Williams (2006).

faltblat

La classe faltblat ("pieghevole" in tedesco) produce un pieghevole a sei pagine simile a quello prodotto da leaflet; tuttavia, oltre ad avere come unica documentazione un *template* scritto in tedesco, non funziona altrettanto bene. Fa parte della raccolta di classi e pacchetti di Ingo Klöckl, *LaTeX Tips and Tricks* (Klöckl, 2000).

twoup

Il pacchetto twoup (Hansen, 2005) è una sorta di "assistente" all'uso dei pacchetti booklet e 2up o all'uso delle PSUtils. Richiamandolo assieme a uno dei pacchetti suddetti, si prende cura di calcolare al meglio le dimensioni della pagine da impostare; compilando il sorgente con dvips, invece, scrive nel relativo file `.log` le opzioni con cui usare psnup (eventualmente con psbook e psselect) per ottenere un libretto. Chiamato twoupltx da Williams (2006).

Conclusioni

Il pacchetto booklet permette di stampare con facilità libretti di ogni tipo, senza doversi preoccupare

dell'imposizione delle segnature. Specialmente se usato assieme a geometry e con l'opzione 1to1 non richiede la manipolazione delle pagine del documento e non dà quindi problemi di scalatura e ridimensionamento del testo. Rappresenta quindi, a parere dell'autore, la migliore soluzione per stampare documenti progettati fin dall'inizio in forma di libretto.

Il pacchetto pdfpages è molto comodo per ottenere in fretta un libretto da un documento già compilato. Inoltre è più generale di booklet e permette qualsiasi genere di imposizione (per esempio booklet non è in grado di produrre la segnatura in ottavo dell'esempio 5.5). Richiede tuttavia di ridimensionare le pagine del documento originale, con conseguenti possibili problemi legati alla scalatura del testo.

Considerazioni simili valgono per i programmi delle PSUtils. Per gli utenti di ConTEXt, TEXexec rappresenta probabilmente la soluzione più naturale. Data la diffusione dei formati PS e soprattutto PDF, e la possibilità di trasformare documenti DVI in entrambi questi formati, il programma dvidvi appare ormai obsoleto. Non essendo disponibile per tutti i sistemi operativi, pdfbook non costituisce una soluzione generale per la stampa di libretti in LATEX.

Riferimenti bibliografici

Buchert, R., Eldering, J. e Aivazian, T. (2005). «Pdfbook». CTAN. `support/pdfbook`.

Duggan, A. J. C. (2000). «Psutils». `http://www.tardis.ed.ac.uk/~ajcd/psutils/index.html`. CTAN: `support/psutils`.

Hagen, H. (2000). «ConTEXt the manual». `www.pragma-ade.com`.

Hansen, M. L. (2005). «The twoup package». CTAN. `macros/latex/contrib/twoup`.

Klöckl, I. (2000). «LATEX tips and tricks». CTAN. `info/examples/ltt`.

Marchesi, G., Paolini, M. e Sucato, V. (1999). *Rilegare i libri*. De Vecchi Editore, Milano.

Matthias, A. (2001). «The pdfpages package». CTAN. V. 0.3e.

Nermander, P. (2005). «How to make impositions with pstops». Scribus Public Wiki. `http://wiki.scribus.net/index.php/How_to_make_impositions_with_pstops`.

Niepraschk, R., Schmidt, W. e lein, H. G. (2004). «The document class leaflet». CTAN.

Pragma ADE (2005). «TEXexec explained». `http://www.pragma-ade.com/pragma-ade/texexec.htm`.

Rokicki, T. e Zimanyi, E. (1991). «dvidvi». CTAN. `dviware/dvidvi`.

Saxena, S. (2001). «`2in1.sty`». CTAN. `macros/latex/contrib/misc`.

Scarso, L. (2005). «ConTEXt: un moderno sistema di *packages* basato su TEX». In *Atti del Secondo convegno nazionale su TEX, LATEX, e tipografia digitale (*GUITMeeting 2005*)*. Pisa, pp. 7–23.

Umeki, H. (2002). «The geometry package». CTAN. V. 3.2.

van Zandt, T. (1993). «Two-up printing for generic TEX». CTAN. `macros/generic/2up`.

Williams, G. (2006). «The TEX catalogue online». `http://texcatalogue.sarovar.org/`.

Wilson, P. (2005). «Printing booklets with LATEX». CTAN. V. 0.7a.

▷ Gustavo Cevolani
Dipartimento di Filosofia
Università di Bologna
`g.cevolani@email.it`

Tabelle su LaTeX 2ε
pacchetti e metodi da utilizzare

Lapo F. Mori

Sommario

Lo scopo del presente articolo è fornire gli strumenti per creare e formattare correttamente tabelle utilizzando LaTeX 2ε. Tale obiettivo è conseguito analizzando i problemi tipici incontrati durante la creazione di tabelle e le possibili soluzioni; si pone particolare attenzione ai pacchetti da usare nelle varie circostanze. Per i singoli casi si riportano esempi e si rimanda ai manuali dei pacchetti suggeriti, ove necessario.

Premessa

Le tabelle sono uno degli oggetti che vengono usati più frequentemente in documenti scientifici. Di fianco ai comandi standard LaTeX esistono numerosi pacchetti che permettono di personalizzare le tabelle e di superare le limitazioni dei primi. La documentazione sulle tabelle è molto frammentata perché i singoli pacchetti agiscono solo alcuni aspetti e dunque gli autori sono spesso costretti a cercare informazioni su diverse guide.

Il presente articolo spiega come utilizzare LaTeX 2ε ed i pacchetti disponibili per creare e formattare tabelle cercando di affrontare ogni loro aspetto. La filosofia seguita è quella di non approfondire i vari temi nei dettagli (per i quali si rimanda ai manuali dei rispettivi pacchetti) ma di trattare il più vasto numero di argomenti e di indicare le soluzioni che l'autore ritiene migliori. La scelta dei contenuti deriva in primo luogo dall'esperienza dell'autore ed in secondo dai numerosi interventi riguardanti le tabelle sul forum del GuIt (Gruppo Utilizzatori Italiani di TeX),[1] che resta sempre un valido riferimento per tutte le tematiche trattate nel presente documento.

Il testo presume che il lettore conosca già i rudimenti di LaTeX 2ε, ovvero che abbia letto una delle numerose guide di base disponibili gratuitamente in rete (Flynn (2005); Oetiker *et al.* (2000)) oppure un libro (Beccari (1991); Diller (1999); Goossens *et al.* (1995); Grätzer (1999, 2000); Hahn (1993); Higham e Griffiths (1997); Knuth (1992); Kopka e Daly (1995); Lamport (1994)). Ogniqualvolta si cita un pacchetto, non si fornisce una descrizione completa del suo funzionamento, per la quale si rimanda al relativo manuale,[2] ma si analizzano le opzioni più importanti e se ne suggerisce l'utilizzo.

Il testo si concentra sulla preparazione delle tabelle, non affrontando problemi ad essa associati quali ad esempio il posizionamento degli oggetti flottanti o la formattazione della didascalia (Caucci e Spadaccini (2005)). Per questi argomenti, che sono in comune a tutti gli oggetti flottanti (tra i quali ad esempio le figure), si rimanda a Mori (2005).

1 Tabelle generiche

1.1 Regole generali

La formattazione delle tabelle dovrebbe fondarsi sulle seguenti regole Fear (2005):

1. non usare mai righe verticali;
2. evitare righe doppie;
3. inserire le unità di misura nell'intestazione della tabella (invece che nel corpo);
4. non usare virgolette per ripetere il contenuto di celle.

Per capire l'importanza di queste regole si confrontino le tabb. 1 e 2.

1.1.1 Formattazione del codice

LaTeX non richiede che le colonne (ovvero `&`) siano allineate, tuttavia è consigliabile che lo siano per facilitare le modifiche nel sorgente. A titolo di esempio, la tabella 3 può essere ottenuta con questo codice

```
\begin{table}[tp]%
\caption{Carico massimo e tensione nominale.}
\label{aggiungi}\centering%
\begin{tabular}{clccc}
\toprule%
$D$ &            & $P_u$     & $\sigma_N$   \\
(in)&            & (lbs)     & (psi)
\\\otoprule%
5   & test 1     & 285       & 38.00        \\
    & test 2     & 287       & 38.27        \\
    & test 3     & 230       & 30.67
\\\midrule
10  & test 1     & 430       & 28.67        \\
    & test 2     & 433       & 28.87        \\
    & test 3     & 431       & 28.73
\\\bottomrule
\end{tabular}
\end{table}
```

1. Il forum del GuIt è raggiungibile al seguente indirizzo `http://www.guit.sssup.it/forum/`.

2. 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, ma in generale esso si trova in una sottocartella di `/texmf/doc`.

Tabella 1: Tabella non formattata correttamente secondo le regole generali.

D	P_u	u_u	β	G_f
5 in	269.8 lbs	0.000674 in	1.79	0.04089 psi · in
10 in	421.0 lbs	0.001035 in	3.59	"
20 in	640.2 lbs	0.001565 in	7.18	"

Tabella 2: Tabella formattata correttamente secondo le regole generali.

D (in)	P_u (lbs)	u_u (in)	β	G_f (psi · in)
5	269.8	0.000674	1.79	0.04089
10	421.0	0.001035	3.59	0.04089
20	640.2	0.001565	7.18	0.04089

Tabella 3: Tabella senza celle multiriga.

D (in)		P_u (lbs)	σ_N (psi)
5	test 1	285	38.00
	test 2	287	38.27
	test 3	230	30.67
10	test 1	430	28.67
	test 2	433	28.87
	test 3	431	28.73

oppure con il seguente

```
\begin{table}[tp]%
\caption{Carico massimo e tensione nominale.}
\label{aggiungi}\centering%
\begin{tabular}{clccc}
\toprule%
$D$&&&$P_u$&&$\sigma_N$\\
(in)&&(lbs)&(psi)\\\otoprule%
5&test 1&285&38.00\\
&test 2&287&38.27\\
&test 3&230&30.67\\\midrule
10&test 1&430&28.67\\
&test 2&433&28.87\\
&test 3&431&28.73\\\bottomrule
\end{tabular}
\end{table}
```

1.2 Comandi standard

1.2.1 Creazione di una tabella

Gli strumenti di base offerti da LaTeX per la creazione di tabelle e matrici sono `tabular`, `tabular*` e `array`. L'ambiente `array` può essere usato solo in modalità matematica; tutti e tre creano una minipage. La sintassi per questi ambienti è

```
\begin{array}[pos]{cols}
righe
\end{array}
```

```
\begin{tabular}[pos]{cols}
righe
\end{tabular}
```

```
\begin{tabular*}{width}[pos]{cols}
righe
\end{tabular*}
```

dove il significato degli argomenti è il seguente (Kopka e Daly (1995)):

`pos` Posizione verticale. Può assumere i seguenti valori:

- `t` la linea in alto (*top*) è allineata con la *baseline* del testo;
- `b` la linea in basso (*bottom*) è allineata con la *baseline* del testo;
- **nessuno** quando non viene specificata nessuna opzione la tabella è centrata alla *baseline* del testo;

`width` Può essere usato solo con `tabular*` e definisce la larghezza totale della tabella. Quando viene usato, l'argomento `cols` deve contenere da qualche parte `@{\extracolsep{\fill}}`.

`cols` Definisce l'allineamento ed i bordi di ogni colonna. Può assumere i valori:

- `l` la colonna è allineata a sinistra (*left*);
- `r` la colonna è allineata a destra (*right*);
- `c` la colonna è centrata (*center*);
- `p{wth}` la colonna è giustificata ed è larga `wth` (il testo è inserito in un `parbox` largo `wth`);
- `*{num}{form}` il formato `form` è ripetuto `num` volte; ad esempio `*{3}{|l|}` equivale a scrivere esplicitamente `|l|l|l|`.

Oltre al formato delle colonne, si può specificare cosa deve venire interposto tra due colonne con i seguenti simboli:

- `|` disegna una linea verticale (sconsigliato);
- `||` disegna due linee verticali (sconsigliato);
- `@{text}` inserisce il testo `text` in ogni linea della tabella in mezzo alle colonne dove appare. Tale comando elimina lo spazio che viene automaticamente inserito tra le colonne. Se si vuole inserire uno spazio tra `text` e le colonne adiacenti, questo deve essere inserito esplicitamente con il comanda `\hspace{}`. Il comando `\extraspace\fill` in un ambiente `tabular*` dilata lo spazio tra le colonne dove compare in modo da far assumere alla tabella la larghezza predefinita (si veda ad esempio la tab. 12 ottenuta con il

Tabella 4: Esempio di utilizzo dei comandi standard LaTeX `multicolumn` e `cline`.

	Provino	Rugosità R_a (nm)
A	anello	385
	piastra	397
B	anello	376
	piastra	390

codice a pag. 36). Per eliminare lo spazio che viene normalmente inserito tra due colonne è possibile mettere il comando vuoto `@{}`.

`righe` contiene il contenuto delle celle della tabella per ogni riga che viene terminata dal comando `\\`. Ogni riga contiene la sequenza del contenuto per ogni colonna delimitato dal simbolo `&`; ogni riga contiene lo stesso numero di celle (ovvero di simboli `&`)[3] che deve essere uguale a quello dichiarato dalla definizione `cols`.

`\hline` può apparire prima della prima riga, oppure dopo la fine di una riga `\\` e disegna una riga orizzontale larga come tutta la tabella.

`\cline{n-m}` disegna una riga orizzontale dalla sinistra della colonna `n` fino alla destra della colonna `m`.

`\multicolumn{num}{col}{text}` combina le seguenti `num` colonne in una singola cella della larghezza delle corrispondenti celle inclusi gli spazi tra colonne. L'argomento `col` deve contenere un simbolo di posizionamento `l`, `r` o `c`.

Un esempio di utilizzo di questi comandi è riportato nella tab. 4 che è generata dal seguente codice:

```
\begin{tabular}{llc}
\hline%
\multicolumn{2}{c}{Provino}
        & Rugosità $R_a$\\
    &                   & (nm)\\\hline%
A   & anello            & 385\\\cline{2-3}
    & piastra           & 397\\\hline
B   & anello            & 376\\\cline{2-3}
    & piastra           & 390\\\hline
\end{tabular}
```

Dato che il comando `@{...}` inserisce il testo contenuto come argomento senza lo spazio tra le colonne, può essere utilizzato senza alcun argomento se si vuole eliminare tale spazio. Ad esempio in alcuni casi può essere desiderabile eliminare lo spazio a sinistra della prima colonna ed a destra dell'ultima; la tab. 5 è ottenuta con il seguente codice

```
\begin{tabular}{@{}>{\bfseries}lp{6cm}@{}}
...
\end{tabular}
```

Si confronti il risultato con quello della tab. 9.

1.2.2 Parametri per lo stile della tabella

Ci sono alcuni parametri che controllano lo stile delle tabelle ed a cui LaTeX assegna un valore di default. Questi comandi possono essere modificati globalmente nel preambolo oppure localmente all'interno di un ambiente.

`\tabcolsep` è metà della larghezza dello spazio inserito tra le colonne degli ambienti `tabular` e `tabular*`.

`\arraycolsep` è metà della larghezza dello spazio inserito tra le colonne degli ambienti `array`.

`\doublerulesep` è lo spazio tra le doppie linee (`\hline\hline`).

La modifica di questi parametri va effettuata con il comando `\setlength` per il quale si rimanda ad una qualunque guida di base (Beccari (1991); Diller (1999); Flynn (2005); Goossens *et al.* (1995); Grätzer (1999); Hahn (1993); Higham e Griffiths (1997); Kopka e Daly (1995); Lamport (1994); Oetiker *et al.* (2000)).

1.2.3 Tabella flottante

Normalmente le tabelle vanno trattate come oggetti flottanti (Fear (2005); Kopka e Daly (1995); Lamport (1994); Mori (2005)). In questo caso l'ambiente `tabular` deve essere inserito nell'ambiente `table` che:

- Permette di generare l'indice delle tabelle con il comando `\listoftables` (analogo al `\listoffigures` per le figure).
- Permette di creare la didascalia preoccupandosi di:
  - assegnare il giusto nome all'ambiente; esso dipende dalla lingua impostata da `babel`: in italiano è "tabella" mentre in inglese è "table";
  - assegnare il numero alla tabella.
- Permette di assegnare alla tabella un `\label` con cui richiamarla nel testo.

Per centrare una tabella flottante[4] deve essere usato il comando `\centering` invece dell'ambiente `center` poiché quest'ultimo inserisce uno spazio verticale supplementare indesiderato (Fairbairns (2006); Trettin (2005)).

Ad esempio la tab. 5 è ottenuta con il seguente codice

3. Le celle possono anche essere vuote.

4. Si noti che lo stesso vale anche per qualunque altro oggetto flottante.

Tabella 5: Tabella senza spazio orizzontale a sinistra della prima colonna ed a destra dell'ultima.

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

```
\begin{table}[tp]
\caption{...} \label{...} \centering\small
\begin{tabular}{...}
...
\end{tabular}
\end{table}
```

1.3 Celle multiriga

Analogamente al comando `\multicolumn` per avere celle su più colonne, esiste il comando `\multirow` per avere celle su più righe. Tale comando richiede il pacchetto multirow.

`\multirow` può essere utilizzato in due modi differenti:

`\multirow{row}*{text}` crea una cella contenente il testo `text` che si estende su `row` righe ed ha larghezza non definita;

`\multirow{row}{larg}*{testo}` crea una cella contenente il testo `text` che si estende su `row` righe ed ha larghezza pari a `larg`.

Un esempio di tabella con celle multiriga è riportato in tab. 6 ottenuta con il seguente codice:

```
\begin{tabular}{clcc}
\toprule%
\multicolumn{2}{c}{$D$}          & $P_u$
& $\sigma_N$\\
\multicolumn{2}{c}{(in)}         & (lbs)
& (psi)\\\otoprule%
\multirow{3}*{5}    & test 1     & 285       &
38.00\\\cmidrule(l){2-4}
    & test 2                     & 287
& 38.27\\\cmidrule(l){2-4}
    & test 3                     & 230
& 30.67\\\midrule
\multirow{3}*{10}   & test 1     & 430       &
28.67\\\cmidrule(l){2-4}
    & test 2                     & 433
& 28.87\\\cmidrule(l){2-4}
    & test 3                     & 431
& 28.73\\\bottomrule
\end{tabular}
```

Si confronti tale tabella con l'analoga tab. 3 dove non si fa uso del comando `\multirow`.

1.4 Miglioramenti estetici

L'ambiente `tabular` offerto da LaTeX ha una resa tipografica non del tutto soddisfacente a causa del troppo poco spazio tra le linee orizzontali (ottenute con `\hline`) e il testo delle celle.

Per risolvere questo problema i pacchetti booktabs e ctable offrono i comandi `\toprule`, `\midrule` e `\bottomrule` da usarsi al posto di `\hline`. In particolare `\toprule` e `\bottomrule` sono da usarsi rispettivamente per la prima e l'ultima riga ed hanno uno spessore maggiore delle linee per le altre righe, ottenute con `\midrule`. In analogia a `\cline` viene offerto `\cmidrule`. Tali comandi intervengono sullo spazio verticale prima e dopo la riga e sullo spessore della stessa; in particolare `\toprule` e `\bottomrule` sono spesse e con uno spazio maggiore in basso e in alto rispettivamente, mentre `\midrule` è sottile e con spazio superiore ed inferiore uguali. Si confrontino le tabb. 2 e 7.

Tabella 6: Esempio di utilizzo dei comandi standard LaTeX `multicolumn` e `multirow`.

	D (in)	P_u (lbs)	σ_N (psi)
5	test 1	285	38.00
	test 2	287	38.27
	test 3	230	30.67
10	test 1	430	28.67
	test 2	433	28.87
	test 3	431	28.73

Di default lo spessore delle linee è 0.08 em per `\toprule` e `\bottomrule` e 0.05 em per `\midrule`. Se si vuole modificare lo spessore localmente (ovvero ad una sola riga) è sufficiente inserire il nuovo spessore tra parentesi quadre dopo il comando della linea; ad esempio il comando

```
\midrule[0.08em]
```

modifica localmente lo spessore di una `\midrule`. Se si vuole modificare globalmente (ovvero in tutto il documento) lo spessore delle linee, si deve intervenire sulle due lunghezze `\heavyrulewidth` (controlla le linee spesse, ovvero `\toprule` e `\bottomrule`) e `\lightrulewidth` (controlla le linee sottili, ovvero `\midrule`). Ad esempio in questo articolo le linee spesse sono larghe 0.1 em grazie al comando

```
\setlength{\heavyrulewidth}{0.1em}
```

In questo articolo la riga che segue l'intestazione delle tabelle ha sempre lo stesso spessore di `\toprule` e `\bottomrule` ma centrata in verticale

Tabella 7: Tabella ottenuta con le righe standard di LaTeX (`hline`).

D (in)	P_u (lbs)	u_u (in)	β	G_f (psi · in)
5	269.8	0.000674	1.79	0.04089
10	421.0	0.001035	3.59	0.04089
20	640.2	0.001565	7.18	0.04089

rispetto alla riga superiore ed inferiore (caratteristica tipica delle righe `\midrule`). Per ottenere questo risultato è stata definita una nuova riga con il seguente comando

```
\newcommand{\otoprule}{\midrule[\heavyrulewidth]}
```

2 Allineamento del testo nelle celle

Di default con LaTeX è possibile creare colonne con quattro tipi di allineamento (vedi il par. 1.2.1): allineato a sinistra (`l`), allineato a destra (`r`), centrato (`c`) e giustificato con larghezza predefinita (`p{wth}`). Ulteriori tipi di allineamento orizzontale del testo possono essere utilizzati grazie a pacchetti specifici.

2.1 Pacchetto array

Il pacchetto array, definisce nuove opzioni per l'allineamento delle colonne di ambienti `array` e `tabular`:

`m{wth}` definisce una colonna giustificata di larghezza `wth` le cui celle sono centrate verticalmente analogamente a `\parbox[c]{wth}`;

`b{wth}` definisce una colonna giustificata di larghezza `wth` le cui celle sono allineate in basso analogamente a `\parbox[b]{wth}`;

`>{ins}` può essere usato prima di un comando `l`, `r`, `c`, `p`, `m` o `b` ed inserisce `ins` prima del contenuto della cella;

`<{ins}` può essere usato dopo un comando `l`, `r`, `c`, `p`, `m` o `b` ed inserisce `ins` dopo il contenuto della cella.

Se si hanno colonne di sola matematica, può essere conveniente definirlo nel preambolo della tabella piuttosto che inserire in ogni cella i delimitatori di testo matematico (`$...$`). Ad esempio è possibile definire nuovi tipi di colonna matematica

```
\newcolumntype{A}{>{$}c<{$}}
\newcolumntype{Q}{>{$}l<{$}}
\newcolumntype{V}{>{$}r<{$}}
```

per avere colonne rispettivamente centrate, allineate a sinistra o a destra di testo matematico. Se si vuole avere matematica in formato `displaystyle` è sufficiente aggiungere il relativo comando alla definizione della nuova colonna

```
\newcolumntype{A}{>{$\displaystyle}c<{$}}
\newcolumntype{Q}{>{$\displaystyle}l<{$}}
\newcolumntype{V}{>{$\displaystyle}r<{$}}
```

Tabella 8: Tabella con testo matematico ottenuta con il pacchetto `array`.

$\int \cos x\,dx$	$\sin x + c$
$\int e^x dx$	$e^x + c$
$\int \sec^2 x\,dx$	$\tan x + c$

Ad esempio il codice

```
\newcolumntype{Q}{>{$\displaystyle}l<{$}}
\newcolumntype{A}{>{$}c<{$}}
\begin{tabular}{QA}
\toprule%
\int\cos x\,dx      & \sin x + c\\\midrule
\int e^x dx         & e^x + c\\\midrule
\int \sec^2 x\,dx   & \tan x + c\\\bottomrule
\end{tabular}
```

produce la tab. 8.

Per ulteriori dettagli sul pacchetto array si consiglia la lettura del manuale e di Gregorio (2005).

2.1.1 Formattare il testo di una colonna

I comandi `>{ins}` e `<{ins}` possono anche essere utilizzati per formattare il font di determinate colonne: è possibile utilizzare i comandi LaTeX `\upshape`, `\itshape`, `\slshape`, `\scshape`, `\mdseries`, `\bfseries`, `\rmfamily`, `\sffamily` e `\ttfamily` (Lamport (1994)). Ad esempio il codice

```
\begin{tabular}{>{\bfseries}l p{6cm}}
\toprule Forza   & Una forza è una grandezza
fisica che si manifesta nell'interazione di
due o più corpi materiali che cambia lo stato
di quiete o di moto dei corpi stessi. \\\midrule
Momento polare  & Il momento polare di una
forza rispetto ad una determinata origine è
definito come il prodotto vettoriale tra il
vettore posizione (rispetto alla stessa
origine) e la forza.\\\bottomrule
\end{tabular}
```

produce la tab. 9 dove la prima colonna ha testo grassetto.

2.1.2 Formattare il testo di una riga

Attualmente non esistono comandi o pacchetti che permettano di formattare righe, è però possibile definire un nuovo comando (Fairbairns (2006)) che sfrutta le funzionalità di array appena discusse. Si introducono queste definizioni

```
\newcolumntype{+}{>{\global\let%
    \currentrowstyle\relax}}
\newcolumntype{^}{>{\currentrowstyle}}
\newcommand{\rowstyle}[1]{\gdef%
    \currentrowstyle{#1}%
#1\ignorespaces
}
```

e poi si inserisce `+` a sinistra della prima colonna e `^` a sinistra di ogni altra nella definizione delle colonne di `tabular`. In questo modo si rende disponibile il nuovo comando

```
\rowstyle{...}
```

Tabella 9: Tabella con formattazione automatica di una colonna ottenuta con il pacchetto `array`.

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

che può essere inserito all'inizio di una riga per definire la formattazione del testo contenuto nelle sue celle. Ad esempio il codice

```
\begin{tabular}{+l^c^c^c}
\toprule\rowstyle{\bfseries}%
Grandezza & Simbolo & Unità di misura
    & Valore \\\otoprule%
...
\end{tabular}
```

produce la tab. 10 dove la prima riga ha font grassetto.

2.2 Pacchetto tabularx

Il pacchetto tabularx utilizza gli stessi argomenti di `tabular*` ma modifica la larghezza di certe colonne piuttosto che lo spazio tra le colonne al fine di coprire la larghezza definita dall'utente. Le colonne che possono essere stirate sono indicate dal comando di allineamento `X`. Tale pacchetto richiede il pacchetto array. Ad esempio i comandi

```
\begin{tabularx}{\textwidth}{>{\bfseries}lX}
\toprule Forza   & Una forza è una grandezza
fisica che si manifesta nell'interazione di
due o più corpi materiali che cambia lo stato
di quiete o di moto dei corpi stessi. \\\midrule
Momento polare  & Il momento polare di una
forza rispetto ad una determinata origine è
definito come il prodotto vettoriale tra il
vettore posizione (rispetto alla stessa
origine) e la forza.\\\bottomrule
\end{tabularx}
```

producono la tab. 11; si noti che nella analoga tab. 9 la larghezza della seconda colonna è stata definita dall'utente, mentre nella tab. 11 l'utente definisce la larghezza dell'intera tabella (in questo caso pari a `\textwidth`) e il programma calcola in automatico la larghezza per la seconda colonna. Si confronti anche il risultato con la tab. 12 ottenuta con l'ambiente `tabular*`: in questo caso per avere la larghezza definita dall'utente (`\textwidth` nell'esempio), viene stirato lo spazio tra le due colonne. Per ottenere la tabella è stato usato il seguente codice

```
\begin{tabular*}{\textwidth}[tb]%
    {>{\bfseries}l@{\extracolsep{\fill}}p{6cm}}
...
\end{tabular*}
```

Di default le colonne `X` sono colonne `p{wth}`, ovvero con testo giustificato, in cui la larghezza `wth` viene determinata automaticamente dal programma. È tuttavia possibile definire nuove colonne che siano allineate in modo diverso con il comando `\newcolumntype`. Ad esempio

```
\newcolumntype{Y}{>{\raggedright%
    \arraybackslash}X}
```

definisce colonne allineate a sinistra,

```
\newcolumntype{W}{>{\raggedleft%
    \arraybackslash}X}
```

definisce colonne allineate a destra,

```
\newcolumntype{Z}{>{\centering\arraybackslash}X}
```

definisce colonne centrate. Ad esempio il codice

```
\begin{tabularx}{\textwidth}{YXZW}
\toprule Cella con testo allineato a sinistra
    & 1 & 2 & 3\\\midrule
4 & Cella con testo giustificato & 5
    & 6\\\midrule
7 & 8 & Cella con testo centrato
    & 9\\\midrule
10 & 11 & 12 & Cella con testo allineato
    a destra\\\bottomrule
\end{tabularx}
```

produce la tab. 13 che è larga quanto il testo (`\textwidth`).

Di default le colonne `X` sono colonne `p{wth}` corrispondenti ad un `\parbox[t]`, ovvero con le celle allineate in alto. È tuttavia possibile definire nuove colonne in cui le celle abbiano un differente allineamento orizzontale con il comando `\tabularxcolumn`. Ad esempio

```
\renewcommand{\tabularxcolumn}[1]%
    {>\arraybackslash}m{#1}}
```

definisce colonne allineate in verticale al centro,

```
\renewcommand{\tabularxcolumn}[1]%
    {>\arraybackslash}b{#1}}
```

definisce colonne allineate in verticale in basso. Il codice

```
\renewcommand{\tabularxcolumn}[1]%
    {>{\arraybackslash}m{#1}}
\begin{tabularx}{\textwidth}{YXZW}
\toprule Cella con testo allineato a sinistra
    & 1 & 2 & 3\\\midrule
4 & Cella con testo giustificato & 5
    & 6\\\midrule
7 & 8 & Cella con testo centrato %
    & 9\\\midrule
10 & 11 & 12 & Cella con testo allineato a
    destra\\\bottomrule
\end{tabularx}
```

produce la tab. 14(a), mentre lo stesso codice con il comando

Tabella 10: Utilizzo del pacchetto array per formattare il testo di una riga.

Grandezza	**Simbolo**	**Unità di misura**	**Valore**
Rigidezza in direzione z	k_z	N/m	2276
Rigidezza in direzione r	k_r	N/m	3414
Peso del corpo	P	N	35

Tabella 11: Tabella ottenuta con il pacchetto `tabularx`.

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

```
\renewcommand{\tabularxcolumn}[1]%
    {>{\arraybackslash}b{#1}}
```

produce la tab. 14(b).

Il pacchetto tabularx risulta anche utile quando si vogliono produrre delle tabelle in cui alcune colonne hanno la stessa larghezza, ma non si ha la necessità di definirla a priori. Infatti, se sono presenti più colonne `X`, la loro larghezza risulta uguale. Ad esempio la tab. 15 è ottenuta con il seguente codice

```
\newcolumntype{K}{>{\centering%
    \arraybackslash$}X<{$}}
\begin{tabularx}{.7\textwidth}{*{7}{K}}
...
\end{tabularx}
```

Normalmente la larghezza delle colonne viene calcolata automaticamente dal programma, tuttavia tabularx permette di assegnare la larghezza ad alcune colonne. Questo può essere fatto a patto che la somma della larghezza delle colonne `X` rimanga invariata. Se ad esempio si vuole che una colonna sia larga la metà di un'altra e si hanno due sole colonne `X`, la prima deve essere larga $^2/_3 \cdot x$ e la seconda $^4/_3 \cdot x$ in modo che siano nella giusta proporzione e che la loro somma sia pari a $2 \cdot x$ (dato che le colonne sono due); per ottenere questo è possibile utilizzare il comando

```
{>{\hsize=0.66\hsize}X>{\hsize=1.34\hsize}X}
```

La tab. 16 è ottenuta aggiungendo questa riga al codice della tab. 11.

2.3 Pacchetto tabulary

Il pacchetto tabulary permette di ottenere risultati equivalenti a quelli ottenuti con il pacchetto tabularx in tab. 13. Invece di dover definire l'allineamento della colonna con il comando `\newcolumntype` come mostrato nel par. 2.2, il pacchetto tabulary mette a disposizione i comandi `R` (allinea a destra), `C` (centra), `L` (allinea a sinistra) e `J` (giustificato).

Ad esempio la tab. 13 può essere alternativamente ottenuta con il codice

```
\begin{tabulary}{\textwidth}{LJCR}
\toprule Cella con testo allineato a sinistra
    & 1 & 2 & 3\\\midrule
4 & Cella con testo giustificato & 5 & 6\\
    \midrule
7 & 8 & Cella con testo centrato & 9\\\midrule
10 & 11 & 12 & Cella con testo allineato
    a destra\\\bottomrule
\end{tabulary}
```

2.4 Andare a capo manualmente

Quando in una cella deve essere inserito un testo lungo, è necessario che venga spezzato su più righe. I pacchetti appena presentati (tabularx e tabulary) risolvono la questione automaticamente e dunque il loro utilizzo è consigliato. Nei casi in cui non è possibile fare uso di tali pacchetti è sempre possibile ricorrere a soluzioni manuali. Se ad esempio si vuole andare a capo solo in alcune celle di una colonna, è possibile inserire nelle celle un `parbox` oppure una `minipage`.

L'utilizzo di questi comandi risulta conveniente anche quando si voglia decidere a priori la larghezza di una colonna non giustificata.[5] Se ad esempio si vuole avere una colonna con testo allineato a destra larga esattamente 5 cm, è possibile inserire tutte le celle dentro ambienti `minipage` con il seguente comando

```
{>{\bfseries}l>{\begin{minipage}[t]{5cm}%
\raggedleft\arraybackslash}%
l<{\end{minipage}\arraybackslash}}
```

Con tale comando è possibile ottenere la tab. 17 a partire dal codice della tab. 11.

2.5 Modificare l'allineamento orizzontale di alcune celle

Per modificare l'allineamento solo di alcune celle, è possibile inserire il testo in una *box* (ad esempio con il comando `\makebox`). Ad esempio la tab. 18 è ottenuta con il codice seguente:

```
\begin{tabular}{lp{3cm}}
\hline
```

5. Se infatti si tratta di una colonna con testo giustificato è sufficiente utilizzare una colonna di tipo `p{wth}`.

Tabella 12: Tabella ottenuta con l'ambiente `tabular*`.

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

Tabella 13: Tabella ottenuta con il pacchetto tabularx ridefinendo nuovi tipi di allineamento orizzontale per le colonne.

Cella con testo allineato a sinistra	1	2	3
4	Cella con testo giustificato	5	6
7	8	Cella con testo centrato	9
10	11	12	Cella con testo allineato a destra

Tabella 14: Tabelle ottenuta con il pacchetto `tabularx` e ridefinendo l'allineamento verticale delle celle per averle centrate (a) ed allineate in basso (b).

(a) Celle centrate in verticale

Cella con testo allineato a sinistra	1	2	3
4	Cella con testo giustificato	5	6
7	8	Cella con testo centrato	9
10	11	12	Cella con testo allineato a destra

(b) Celle allineate in basso

Cella con testo allineato a sinistra	1	2	3
4	Cella con testo giustificato	5	6
7	8	Cella con testo centrato	9
10	11	12	Cella con testo allineato a destra

Tabella 15: Tabella ottenuta con il pacchetto tabularx per avere colonne della stessa larghezza.

α	$\sin\alpha$	$\cos\alpha$	$\tan\alpha$	$\csc\alpha$	$\sec\alpha$	$\cot\alpha$
-2π	0	1	0	∞	1	∞
$-7\pi/4$	$\sqrt{2}/2$	$\sqrt{2}/2$	1	$\sqrt{2}$	$\sqrt{2}$	1
$-3\pi/2$	1	0	∞	1	∞	0
$-5\pi/4$	$\sqrt{2}/2$	$-\sqrt{2}/2$	-1	$\sqrt{2}$	$-\sqrt{2}$	-1
$-\pi$	0	-1	0	∞	-1	∞
$-3\pi/4$	$-\sqrt{2}/2$	$-\sqrt{2}/2$	1	$-\sqrt{2}$	$-\sqrt{2}$	1
$-\pi/2$	-1	0	∞	-1	∞	0
$-\pi/4$	$-\sqrt{2}/2$	$\sqrt{2}/2$	-1	$-\sqrt{2}$	$\sqrt{2}$	-1
0	0	1	0	∞	1	∞

Tabella 16: Tabella ottenuta con il pacchetto `tabularx` controllando la proporzione tra la larghezza delle colonne: la prima colonna è larga la metà della seconda.

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

Tabella 17: Tabella con una colonna di larghezza predefinita ed allineamento a destra.

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

Tabella 18: Modifica dell'allineamento di alcune celle.

Testo	a sinistra
Testo	al centro
Testo	a destra

```
Testo & a sinistra\\
\hline
Testo & \makebox[3cm][c]{al centro}\\
\hline
Testo & \makebox[3cm][r]{a destra}\\
\hline \end{tabular}
```

2.6 Allineare i numeri alla virgola

2.6.1 Comandi standard

Per allineare i numeri alla virgola è possibile utilizzare il comando `@{,}` e mettere le unità su una colonna ed i decimali su un'altra. Ad esempio la tab. 19, può essere ottenuta con il seguente codice

```
\begin{tabular}{c r @{,} l}
\toprule
Espressione & \multicolumn{2}{c}{Valore}\\
\otoprule $\pi$             &     3 & 1416\\
\midrule  $\pi^{\pi}$       &    36 & 46\\
\midrule  $\pi^{\pi^{\pi}}$ & 80662 & 7\\
\bottomrule
\end{tabular}
```

2.6.2 Pacchetto dcolumn

Se non si vogliono spezzare i numeri su due colonne, è possibile utilizzare il dcolumn che mette a disposizione un nuovo tipo di colonna

```
D{sep-in}{sep-out}{prima.dopo}
```

il primo argomento è il carattere usato nel documento `.tex` per indicare la separazione delle cifre decimali (di solito il punto . o la virgola ,), il secondo quello che si vuole nel documento composto (in italiano la convenzione (Cevolani (2006)) è la virgola , ma talvolta vengono usati il punto in basso . o in alto ·), il terzo è il numero di cifre a sinistra (`prima`) e a destra (`dopo`) della virgola. I numeri vengono allineati rispetto al separatore e, nel caso che il terzo argomento sia negativo, il separatore sarà al centro della colonna. Se le colonne hanno dei titoli, è necessario inserirli all'interno di comandi `\multicolumn{1}{c}{...}`. Ad esempio la tab. 19, può essere ottenuta anche con il seguente codice

```
\begin{tabular}{cD{.}{,}{5.4}}
\toprule
Espressione
& \multicolumn{1}{c}{Valore}\\\otoprule
$\pi$               &     3.1416
\\\midrule
$\pi^{\pi}$         &    36.46
\\\midrule
$\pi^{\pi^{\pi}}$   & 80662.7
\\\bottomrule
\end{tabular}
```

Se non si vogliono inserire i tre argomenti per ogni colonna, è possibile definire un nuovo tipo di colonna, ad esempio con il comando

```
\newcolumntype{d}[1]{D{.}{,}{#1}}
```

dove `d` ha un solo argomento che definisce il numero di cifre decimali.

Gregorio (2005) mostra un interessante utilizzo di `\newcolumntype` per rendere il separatore dei decimali dipendente dalla lingua impostata con `babel`: virgola (`,`) per l'opzione `italian` e punto (`.`) per l'opzione `english`.

Tabella 19: Esempio di tabella con allineamento alla virgola.

Espressione	Valore
π	3,1416
π^{π}	36,46
$\pi^{\pi^{\pi}}$	80662,7

Tabella 20: Esempio di tabella con allineamento alla virgola.

Espressione	Valore
π	3,142
π^{π}	36,460
$\pi^{\pi^{\pi}}$	80662,700

2.6.3 Pacchetto rccol

Un pacchetto più sofisticato per l'allineamento dei numeri di una colonna al separatore delle cifre decimali è rccol. Questo pacchetto permette di ottenere gli stessi risultati di dcolumn ma in aggiunta permette di approssimare i numeri e di aggiungere automaticamente zeri in modo che tutti abbiano lo stesso numero di cifre dopo la virgola. Le colonne da allineare sono definite con il comando

```
R[sep-in][sep-out]{prima}{dopo}
```

dove gli argomenti hanno lo stesso significato di quelli per dcolumn. L'argomento dopo in questo caso definisce quale sia l'approssimazione da applicare ai numeri; ad esempio ponendo tale parametro pari a 3, 3.1416 diventa 3.142 e 80662.7 diventa 80662.700. Tale parametro assegna al numero la precisione 10^{-n} (dove n è il valore di dopo) e dunque può assumere anche valori negativi; ad esempio ponendo tale parametro pari a -1, 3.1416 diventa 0 e 80662.7 diventa 80660.

Ad esempio il codice

```
\begin{tabular}{cR[,][,]{5}{3}}
\toprule
Espressione
& \multicolumn{1}{c}{Valore}\\\otoprule
$\pi$                & 3,1416
\\\midrule
$\pi^{\pi}$          & 36,46
\\\midrule
$\pi^{\pi^{\pi}}$    & 80662,7
\\\bottomrule
\end{tabular}
```

produce la tab. 20.

3 Tabelle grandi

Soprattutto nelle appendici, può capitare che le dimensioni di una tabella siano superiori a quelle dell'intera pagina in altezza, in larghezza o in entrambe. Nel caso che la tabella sia troppo lunga è possibile

- ridurre la dimensione del font (vedi par. 3.1);
- scalarla (vedi par. 3.3);
- spezzarla su più pagine (vedi par. 3.4);

Nel caso che la tabella sia troppo larga è possibile

- ridurre la dimensione del font (vedi par. 3.1);
- ruotarla (vedi par. 3.2);
- scalarla (vedi par. 3.3);
- spezzarla su più pagine (vedi par. 3.4);

Ovviamente per ogni caso, è anche possibile applicare più soluzioni contemporaneamente.

3.1 Ridurre la dimensione del font

Per ridurre la dimensione del font all'interno di una tabella è sufficiente mettere il comando della dimensione font dentro l'ambiente `table` (o un analogo). Ad esempio il codice

```
\begin{table}[tp]\footnotesize
\caption{...} \label{...} \centering
\begin{tabular}{...}
...
\end{tabular}
\end{table}
```

produce la tab. 21. Si confronti il risultato con la tab. 9.

Nel caso che la tabella non sia flottante, ovvero che non sia inserita in un ambiente `table` (o un analogo), la dichiarazione della dimensione del font deve essere fatta fuori dall'ambiente `tabular` e poi alla fine deve essere ripristinato il font usato nel testo. Al contrario, quando la dichiarazione è inserita in un ambiente `table` (o un analogo), vale solo all'interno di tale ambiente e quindi non è necessario ripristinare la dimensione normale del font alla fine.

3.2 Ruotare tabelle

I metodi per ruotare le tabelle sono applicabili anche ad ogni altro oggetto flottante; un ottimo riferimento per l'argomento è Voss (2003).

3.2.1 Pacchetto graphicx

Il pacchetto graphicx mette a disposizione il comando

```
\rotatebox[options]{angle}{oggetto}
```

che consente di ruotare qualunque oggetto (o box) e quindi può anche essere usato per le tabelle. `options` è un argomento opzionale che permette di modificare l'origine della rotazione (vedi Voss (2003)), `angle` è l'angolo di rotazione (per le tabelle sarà tipicamente 90°) e `oggetto` è l'oggetto che deve essere ruotato (nel caso in questione sarà tipicamente l'ambiente `tabular`).

Tabella 21: Tabella con font di dimensione inferiore (`footnotesize`) rispetto al resto del testo (`normalsize`).

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

3.2.2 Pacchetto rotating

Il pacchetto rotating mette a disposizione comandi che servono a ruotare specificamente oggetti flottanti e quindi anche tabelle. In primo luogo è possibile definire il verso positivo per gli angoli con i comandi

`clockwise` (default) misura l'angolo in senso antiorario per compatibilità con Goossens *et al.* (1995), dove gli angoli positivi sono sempre misurati in senso antiorario.

`counterclockwise` come il precedente ma con angoli positivi se in senso orario.

Il pacchetto rotating mette a disposizione gli ambienti `rotate`, `turn`, `sideways` e `sidewaystable`.[6]

L'ambiente rotate.

Questo ambiente non lascia spazio verticale per l'oggetto ruotato e per questo motivo non è utilizzabile con le tabelle perché esse sovrascriverebbero il testo precedente.

```
\begin{rotate}{angle}
...
\end{rotate}
```

L'ambiente turn.

Questo ambiente, a differenza di `rotate`, aggiunge spazio verticale per l'oggetto ruotato ed ha dunque lo stesso effetto del comando `\rotatebox` (vedi il par. 3.2.1).

```
\begin{turn}{<degree>}
...
\end{turn}
```

L'ambiente sideways.

Questo ambiente è un particolare ambiente `turn`: ruota di 90° e lascia spazio verticale per l'oggetto ruotato; in questo caso non è necessario indicare l'angolo. Dato che normalmente le tabelle larghe devono essere ruotate di 90°, questo è l'ambiente più adatto.

```
\begin{sideways}
\begin{tabular}
...
\end{tabular}
\end{sideways}
```

6. Esiste anche `sidewaysfigure` che è l'analogo di `sidewaystable` per le figure.

L'ambiente sidewaystable.

Questo ambiente effettua la stessa rotazione di `sideways` ma deve essere usato per tabelle flottanti per le quali si va a sostituire all'ambiente `table`. La sintassi è

```
\begin{sidewaystable}
\caption{...}
\begin{tabular}
...
\end{tabular}
\end{sidewaystable}
```

A differenza degli ambienti precedenti, `sidewaystable` occupa una intera pagina. Dato che tale ambiente è flottante, se necessario, LaTeX riempie la pagina che lo precede con il testo che si trova dopo tale ambiente.

Ad esempio il codice

```
\begin{sidewaystable}[p]\small
\caption{...} \label{...} \centering
\renewcommand{\tabularxcolumn}[1]%
    {>{\arraybackslash}m{#1}}
\newcolumntype{W}{>{\centering\arraybackslash}X}
\begin{tabularx}{\textheight}{lccWWWWW}
...
\end{tabularx}
\end{sidewaystable}
```

produce la tabella riportata in fig. 1; in questo esempio, oltre a `sidewaystable`, si utilizza anche il pacchetto `tabularx`, con il quale di definisce la colonna di tipo `W` (testo centrato orizzontalmente) e si allinea al centro il testo in verticale, ed inoltre si riduce la dimensione del font con il comando `\small`.

3.2.3 Pacchetto lscape

Il pacchetto lscape definisce l'ambiente `landscape` che ruota il suo contenuto di 90°. Tale ambiente è stato scritto per funzionare con il pacchetto `longtable` e dunque il suo contenuto può anche estendersi su più pagine.

3.3 Scalare tabelle

Il pacchetto graphicx offre due comandi che possono essere utilizzati per scalare le tabelle: `\scalebox` e `\resizebox`.

Il commando

```
\scalebox{h-scale}[v-scale]{argument}
```

scala `argument` (che in questo caso sarà una tabella) di `h-scale` in orizzontale e `v-scale` in verticale; nel caso di tabelle si vuole usare lo stesso fattore

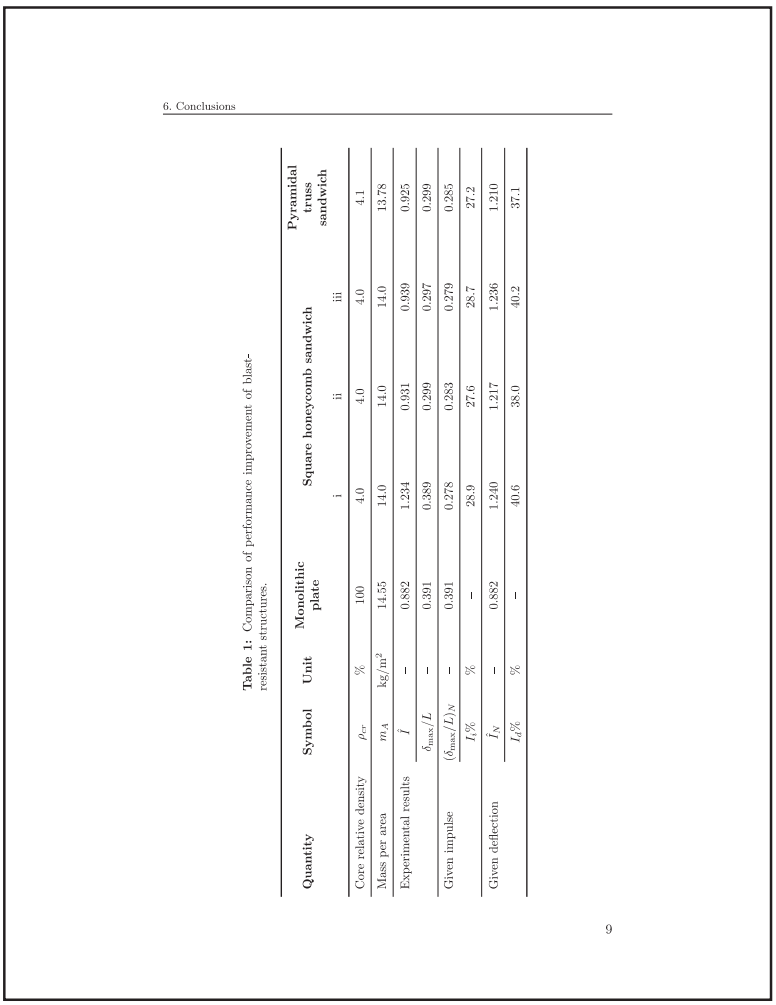


Figura 1: Esempio dell'ambiente `sidewaystable`.

di scala in orizzontale ed in verticale ed è possibile farlo dichiarando uno solo tra `h-scale` e `v-scale` ed assegnando `!` all'altro.

Il commando

```
\resizebox{width}{height}{argument}
```

scala `argument` (che in questo caso sarà una tabella) in modo da ottenere larghezza `width` e altezza `height`; anche in questo caso è possibile assegnare `!` per non modificare il rapporto tra altezza e larghezza della tabella. Un utilizzo tipico di questi comandi consiste nello scalare la tabella in modo che occupi tutta la larghezza delle righe; tale effetto può essere ottenuto ad esempio con

```
\begin{table}[tb]
\caption{...} \label{...} \centering
\resizebox{\textwidth}{!}{%
\begin{tabular}{...}
...
\end{tabular}}
\end{table}
```

Ad esempio le tabb. 14(a) e 14(b) sono realizzate con lo stesso codice della tab. 13 ma sono ridimensionate con `\resizebox` in modo da avere la larghezza pari al 65% della larghezza delle righe:

```
\resizebox{0.65\textwidth}{!}{...}
```

3.4 Tabelle su più pagine

L'ambiente `tabular` deve sempre essere contenuto in una pagina: se è più grande, le parti che sono all'esterno vengono tagliate e si riceve un errore `Overfull vbox`. Esistono diversi pacchetti che permettono di superare questa limitazione e di ripartire tabelle su più pagine.

3.4.1 Pacchetto supertabular

Il pacchetto supertabular offre l'omonimo ambiente `supertabular` che si comporta come il normale `tabular` ma controlla la lunghezza della tabella ad ogni riga: quando la lunghezza supera `textheight`, viene inserito automaticamente l'argomento opzionale `tabletail` ed il comando `\end{tabular}` ed inizia la tabella su una nuova pagina inserendo l'argomento opzionale `tablehead`. L'argomento `tabletail` può essere usato per inserire alla fine di ogni pagina in cui è presente la tabella "continua sulla pagina successiva" mentre `tablehead` per "continua dalla pagina precedente". Il pacchetto supertabular tratta la parte di tabella che si trova su una pagina come un oggetto a sé stante e dunque la larghezza delle colonne può variare tra

le diverse pagine a meno che non si usino colonne di larghezza fissa.

3.4.2 Pacchetto xtab

Il pacchetto xtab presenta circa le stesse funzioni di supertabular ma ne corregge alcuni difetti dunque se ne consiglia l'utilizzo. Offre anche la possibilità di avere una didascalia differente per l'ultima pagina (`\tablelasthead`).

3.4.3 Pacchetto longtable

Il pacchetto longtable costruisce la tabella a pezzi durante la prima compilazione e poi utilizza le informazioni che scrive nel file `.aux` per decidere dove spezzare la tabella nelle successive passate. Così facendo le colonne hanno la stessa larghezza su tutte le pagine. Il pacchetto richiede dunque successive compilazioni del codice (a differenza di supertabular) ed inoltre presenta numerose incompatibilità con altri pacchetti.

3.4.4 Pacchetto lscape

Il pacchetto lscape è stato creato per ruotare di 90° l'ambiente `supertabular` in modo da formattare tabelle larghe e lunghe.

Per tutti i pacchetti citati si rimanda ai rispettivi manuali che contengono esaurienti esempi del loro utilizzo. Per `longtable` si consiglia anche la lettura di Gregorio (2005).

4 Tabelle colorate

Quando si vuole evidenziare parte di una tabella, può risultare molto utile colorarne lo sfondo. Il pacchetto colortbl permette di colorare lo sfondo di celle, righe e colonne di ambienti `tabular`; permette anche di colorare le linee (ad esempio `\hline`) ma tale argomento non viene affrontato perché in documenti scientifici non dovrebbero mai essere usate linee colorate. Il pacchetto colortbl richiede la presenza dei pacchetti color e array. Alternativamente al pacchetto color, può essere usato il pacchetto xcolor (Daly (1998)) tuttavia di seguito si riportano solo esempi per il primo.

4.1 Colorare le colonne

Il pacchetto offre il comando `\columncolor` che deve essere usato esclusivamente all'interno del comando `>{...}` (vedi par. 2.1). La sintassi è

```
\columncolor[clrmodel]{color}[left overhang]%
    [right overhang]
```

I primi due argomenti sono comandi standard del pacchetto `color` (Carlisle (1999)), usati come nel comando `\color` mentre gli ultimi due argomenti sono lunghezze.

`clrmodel` dichiara il modello di colore da utilizzare; quelli disponibili sono `rgb`, `cmyk`, `gray` e `named`.

`color` dichiara il colore da utilizzare; tale definizione dipende dal modello di colore scelto. `rgb` (Red Green Blue) richiede una lista di tre numeri compresi tra 0 ed 1 e separati tra loro da una virgola; ognuno di essi dà la rispettiva componente del colore (rossa, verde e blu). `cmyk` (Cyan Magenta Yellow Black) richiede una lista di quattro numeri compresi tra 0 ed 1 e separati tra loro da una virgola; ognuno di essi dà la rispettiva componente del colore (azzurro, porpora, giallo e nero). `gray` richiede un solo numero compreso tra 0 ed 1 indicante il livello di grigio; nelle tabelle scientifiche si preferisce utilizzare i grigi al posto del colore quindi questo comando risulta particolarmente conveniente. `named` permette di richiamare i colori in base al nome; tale nome deve essere noto al driver utilizzato oppure definito dall'utente con il comando

```
\definecolor{nome}{clrmodel}{color}
```

dove `nome` è il nome assegnato dall'utente al colore, `clrmodel` e `color` hanno lo stesso significato che hanno in `\columncolor`.

`left overhang` e `right overhang` indicano quanto il colore della colonna si allontana dal testo a sinistra e destra rispettivamente. È consigliabile non utilizzare questi due argomenti opzionali in quanto in questo caso il colore copre tutto lo spazio riservato a `\tabcolsep`.

Si consiglia di utilizzare il comando `\newcolumntype` (vedi par. 2.1) definendo nuove colonne che abbiano il colore desiderato. Ad esempio definendo

```
\newcolumntype{K}{\columncolor[gray]%
    {0.8}\raggedright}
```

è possibile ottenere la tab. 23 con il codice

```
\begin{tabular}{ccccU}
\hline%
$D$ & $P_u$  & $u_u$  & $\beta$ & $G_f$  \\
(in) & (lbs)  & (in)
& &$(\text{psi}\cdot\text{in})$ \\\hline%
5  &269.8  & 0.000674 & 1.79 & 0.04089 \\\hline%
10 &421.0  & 0.001035 & 3.59 & 0.04089 \\\hline%
20 &640.2  & 0.001565 & 7.18 & 0.04089 \\\hline%
\end{tabular}
```

Si confronti il risultato con la tab. 7. Si noti che per le righe orizzontali non è possibile utilizzare i comandi offerti dal pacchetto ctable in quanto questi lasciano spazio verticale sopra e sotto ogni cella e dunque lo sfondo colorato risulterebbe interrotto; si veda al proposito la tab. 22.

4.2 Colorare le righe

Per colorare le righe viene offerto il comando `\rowcolor` che ha la stessa sintassi di `\columncolor` e deve essere posizionato all'inizio di una riga. Se in una tabella sono contemporaneamente presenti `\columncolor` e `\rowcolor`, quest'ultimo ha la precedenza. Ad esempio la tab. 24

Tabella 22: Tabella con una colonna colorata ottenuta con il pacchetto colortbl.

D (in)	P_u (lbs)	u_u (in)	β	G_f (psi · in)
5	269.8	0.000674	1.79	0.04089
10	421.0	0.001035	3.59	0.04089
20	640.2	0.001565	7.18	0.04089

Tabella 23: Tabella con una colonna colorata ottenute con le righe orizzontali del pacchetto ctable.

D (in)	P_u (lbs)	u_u (in)	β	G_f (psi · in)
5	269.8	0.000674	1.79	0.04089
10	421.0	0.001035	3.59	0.04089
20	640.2	0.001565	7.18	0.04089

è ottenuta con lo stesso codice della tab. 2 con l'aggiunta del comando

```
\rowcolor[gray]{.8}
```

Talvolta il colore sullo sfondo delle righe non serve ad evidenziare ma a separare le righe (funzione analoga a quella delle righe orizzontali). In questo caso risulta particolarmente conveniente il pacchetto xcolor che offre comandi per colorare in automatico tutte le righe pari e tutte le righe dispari di una tabella. In tal caso deve essere caricata l'opzione table del pacchetto con il comando

```
\usepackage[table]{xcolor}
```

ed il comando da utilizzare immediatamente prima dell'ambiente `tabular` è

```
\rowcolors{row}{odd-row color}{even-row color}
```

dove

`row` indica il numero della prima riga da colorare,

`odd-row color` indica il colore da applicare alle righe dispari (vuoto significa nessun colore),

`even-row color` indica il colore da applicare alle righe spari (vuoto significa nessun colore).

Ad esempio la tab. 25 è identica alla tab. 7 ma, al posto delle righe orizzontali (`\hline`), utilizza lo sfondo grigio per le righe dispari. Il codice utilizzato è

```
\rowcolors{2}{gray!35}{}
\begin{tabular}{ccccc}
...
\end{tabular}
```

4.3 Colorare singole celle

Per colorare lo sfondo di singole celle viene offerto il comando `\cellcolor` che funziona come `\columncolor` and `\rowcolor`, ed ha precedenza su entrambi questi; `\cellcolor` può essere posizionato ovunque nella cella a cui va applicato.

Tabella 24: Tabella con una riga colorata ottenuta con il comando `\rowcolor`.

D (in)	P_u (lbs)	u_u (in)	β	G_f (psi · in)
5	269.8	0.000674	1.79	0.04089
10	421.0	0.001035	3.59	0.04089
20	640.2	0.001565	7.18	0.04089

Tabella 25: Tabella con righe dispari colorate ottenuta con il pacchetto xcolor.

D (in)	P_u (lbs)	u_u (in)	β	G_f (psi · in)
5	269.8	0.000674	1.79	0.04089
10	421.0	0.001035	3.59	0.04089
20	640.2	0.001565	7.18	0.04089

Ad esempio la tab. 26 è ottenuta con lo stesso codice della tab. 21 con la sola aggiunta del comando

```
\cellcolor[gray]{0.8}
```

5 Generazione automatica di tabelle

Spesso si ha la necessità di inserire in documenti LATEX delle tabelle con dati elaborati da altri programmi, ad esempio di statistica, di matematica oppure fogli di calcolo. Le tabelle in LATEX si prestano bene ad essere generate automaticamente da altri programmi in quanto esse sono costruite con un numero limitato di comandi. Tipicamente, infatti, le celle di una stessa riga della tabella sono semplicemente separate da `&` e le righe terminate da \\ (vedi il par. 1.2.1).

5.1 Stata

Un'interessante applicazione di questo concetto è stata presentata per il programma di statistica Stata[7]. Gini (2004) spiega come generare automaticamente con Stata tabelle LATEX con righe colorate (vedi il par. 4) e tabelle in cui alcune celle sono evidenziate con il grassetto (ad esempio quelle contenenti il valore massimo o minimo di una serie di dati).

5.2 Fogli di calcolo

Esistono numerosi software gratuiti, tra cui Excel2LaTeX[8] e Spreadsheet2LaTeX[9], che generano codice LATEX pronto per essere compilato a partire da una tabella formattata nel foglio di calcolo, con

7. Stata è un marchio registrato di StataCorp LP.
8. Excel2LaTeX è disponibile all'indirizzo `ftp://ftp.tex.ac.uk/tex-archive/support/excel2latex/xl2latex.zip`.
9. Spreadsheet2LaTeX è disponibile all'indirizzo `http://pegasus.rutgers.edu/~elflord/unix/software/spreadsheet2latex/`.

Tabella 26: Tabella con una cella colorata con il comando `\cellcolor`.

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione (rispetto alla stessa origine) e la forza.

colori, righe orizzontali e verticali. In generale è comunque semplice scrivere macro che permettano di produrre il codice LaTeX a partire da un foglio di calcolo qualunque.

6 Specialità

6.1 Stili per le linee

Come detto nel par. 4, il pacchetto colortbl permette di modificare il colore delle linee orizzontali e verticali, tuttavia tale modifica dovrebbe essere evitata in documenti scientifici e dunque si lasciano al lettore gli approfondimenti a tale riguardo.

Il pacchetto arydshln permette di avere linee tratteggiate. I comandi `\hdashline` e `\cdashline` sono gli equivalenti di `\hline` e `\cline`.

6.2 Note dentro tabelle

Il comando LaTeX standard per le note (`\footnote`) non funziona nelle tabelle perché l'ambiente `tabular` non permette al comando di posizionare testo a piè di pagina. Per aggirare questo problema ci sono molte soluzioni e pacchetti (Fairbairns (2006)); di seguito si riportano le migliori.

In generale possono presentarsi due casi:

a) si vuole che la nota segua la numerazione delle altre note presenti nel testo;

b) si vuole una numerazione particolare (ad esempio con lettere) per le note di una tabella.

La soluzione per il caso (a) è fornita dal pacchetto footnote. Tale pacchetto mette a disposizione l'ambiente `savenotes` che permette di inserire note anche nell'ambiente `tabular` e le posiziona a piè di pagina. Se si vuole permettere la presenza delle note in tutte le tabelle del documento è sufficiente inserire nel preambolo il comando

```
\makesavenoteenv{tabular}
```

Ad esempio la tab. 27 è ottenuta con il seguente codice

```
\begin{savenotes}
\begin{table}[..] \caption{...}
\label{...}\centering
\begin{tabular}{...} \toprule
...
Momento polare  & Il momento polare di una forza
rispetto ad una determinata origine è definito
come il prodotto vettoriale tra il vettore
posizione \footnote{Rispetto alla stessa
origine.} e la forza.\\\bottomrule
\end{tabular}
\end{table}
\end{savenotes}
```

La soluzione per il caso (b) è offerta dal pacchetto ctable. L'ambiente `ctable` offerto dall'omonimo pacchetto prevede il comando `\tmark` che posiziona il simbolo della nota (in questo caso la numerazione di default è con le lettere minuscole) e dal comando `\tnote{...}` che contiene il testo della nota. Le note sono posizionate subito sotto la tabella e non a piè di pagina. Ad esempio la tab. 28 è ottenuta con il seguente codice

```
\begin{ctable}%
[caption = ...,label = ...]{...}%
{\tnote{Rispetto alla stessa origine.}}%
{...\\\midrule
Momento polare  & Il momento polare di una forza
rispetto ad una determinata origine è definito
come il prodotto vettoriale tra il vettore
posizione\tmark\ e la forza.\\\bottomrule}
\end{ctable}
```

6.3 Caselle barrate

Talvolta la prima cella in alto a sinistra di una tabella contiene due argomenti: il primo descrive il contenuto della prima colonna e il secondo quello della prima riga. Tale soluzione, sconsigliabile in documenti scientifici, è ottenibile con il comando `\backslashbox` fornito dal pacchetto slashbox. Ad esempio il codice

```
\begin{tabular}{|l|>{$}c<{$}|>{$}c<{$}|}\hline
\backslashbox{Funzione}{Argomento} &0&\pi/2\\
\hline $\sin$ & 0 & 1\\
\hline $\cos$ & 1 & 0\\
\hline
\end{tabular}
```

produce la tab. 29.

Il pacchetto slashbox disegna una figura con due label ai due lati di una riga obliqua e poi posiziona la figura nella cella della tabella. Il pacchetto usa la modalità `picture` di LaTeX che ha molte restrizioni e non produce figure di alta qualità. Per migliorare la qualità è possibile caricare il pacchetto pict2e.

Riferimenti bibliografici

Beccari, C. (1991). *LaTeX, Guida a un sistema di editoria elettronica.* Hoepli.

Carlisle, D. P. (1999). *Packages in the 'graphics' bundle.* `ftp://tug.ctan.org/`

10. Rispetto alla stessa origine.

Tabella 27: Tabella con nota che segue la numerazione delle note nel testo e viene posizionata a piè di pagina (pacchetto footnote).

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione[10] e la forza.

Tabella 28: Tabella con note con una numerazione propria e posizionate subito sotto la tabella (pacchetto ctable).

Forza	Una forza è una grandezza fisica che si manifesta nell'interazione di due o più corpi materiali che cambia lo stato di quiete o di moto dei corpi stessi.
Momento polare	Il momento polare di una forza rispetto ad una determinata origine è definito come il prodotto vettoriale tra il vettore posizione[a] e la forza.

[a] Rispetto alla stessa origine.

Tabella 29: Tabella con una cella barrata ottenuta con il pacchetto slashbox.

Funzione \ Argomento	0	$\pi/2$
sin	0	1
cos	1	0

pub/tex-archive/macros/latex/required/graphics/grfguide.pdf.

Caucci, L. e Spadaccini, M. (2005). *Gestione di Figure e Tabelle con LATEX*. http://www.guit.sssup.it/downloads/fig_tut.pdf.

Cevolani, G. (2006). «Norme tipografiche per l'italiano in LaTeX». *ArsTEXnica*, (1), pp. 29–42.

Daly, P. (1998). *Graphics and Colour with LATEX*. http://tex.loria.fr/graph-pack/grf/grf.pdf.

Diller, A. (1999). *LATEX Line by Line: Tips and Techniques for Document Processing*. John Wiley & Sons.

Fairbairns, R. (2006). *The UK TEX FAQ*. http://www.ctan.org/tex-archive/help/uk-tex-faq/letterfaq.tex.

Fear, S. (2005). *Publication quality tables in LATEX*. http://www.ctan.org/tex-archive/macros/latex/contrib/booktabs/booktabs.pdf.

Flynn, P. (2005). *A beginner's introduction to typesetting with LATEX*. http://www.tug.org/tex-archive/info/beginlatex/.

Gini, R. (2004). *Generazione automatica di tabelle con LATEX e Stata*. http://www.guit.sssup.it/downloads/dispensa_gini.pdf.

Goossens, M., Mittelbach, F. e Samarin, A. (1995). *The LATEX Companion*. Addison-Wesley.

Grätzer, G. (1999). *First Steps in LATEX*. Springer Verlag.

(2000). *Math into LATEX*. Birkhauser.

Gregorio, E. (2005). *LATEX, Breve guida ai pacchetti di uso più comune*. http://profs.sci.univr.it/~gregorio/breveguida.pdf.

Hahn, J. (1993). *LATEX for Everyone: A Reference Guide and Tutorial for Typesetting Documents Using a Computer*. Prentice Hall.

Higham, D. e Griffiths, D. (1997). *Learning LATEX*. Society for Industrial and Applied Mathematics.

Knuth, D. (1992). *The TEXbook*. Addison-Wesley.

Kopka, H. e Daly, P. (1995). *A Guide to LATEX – Document Preparation for Beginners and Advanced Users*. Addison-Wesley.

Lamport, L. (1994). *LATEX: A Document Preparation System, User's Guide and Reference Manual*. Addison-Wesley.

Mori, L. F. (2005). «Scrivere la tesi di laurea con LaTeX 2ε». In GUIt*meeting 2005*. GUIt (Italian TEX Users Group), Pisa, Italy.

Oetiker, T., Partl, H., Hyna, I. e Schlegl, E. (2000). *Una (mica tanto) breve introduzione a LaTeX 2ε*. http://www.ctan.org/tex-archive/info/lshort/italian/itlshort.pdf.

Trettin, M. (2005). *Elenco dei "peccati" degli utenti di LaTeX 2ε*. http://www.tug.org/tex-archive/info/l2tabu/italian/l2tabuit.pdf.

Voss, H. (2003). *Rotating Text, Tabulars and Images*. http://perce.de/LaTeX/rotating.pdf.

▷ Lapo F. Mori
Department of Mechanical Engineering
Northwestern University
Evanston 60208 IL, USA
mori@northwestern.edu

Removing vertical stretch – mimicking traditional typesetting with TEX

Kaveh Bazargan, CV Radhakrishnan

Abstract

One of TEX's advantages over traditional typesetting systems is the mechanism to stretch horizonal and vertical glue as needed, in order to aid paragraph building and pagination. But all TEX operators involved in day-to-day page make-up know that this inbuilt intelligence is often 'too clever' and frustrating for the user. TEX will not do what you want it to do, and only over many years can operators gain the knowledge that allows them to make just the right change to the source code in order to coerce TEX to produce the desired result.

Recently we have been experimenting with removing vertical stretchability, with promising results. Our approach is to round off the height of all vertical material, including floats and displayed equations, to be an integral number of the leading of the main text. One advantage is that this allows true 'grid' setting in double column text.

1 Some things we have always wanted

It is useful to look at some things we have always wanted to do in TEX but found difficult.

1.1 More control over glue

Anyone involved with 'real world' pagination of TEX and LATEX files is aware of the frustration that TEX's 'glue' can generate. TEXies have long learned to accept this limitation, and developed tricks to work efficiently. However, some apparently simple tasks are still mysteriously complex to the non-TEXie. For example, in figure 1 suppose that two lines have to move from the first to the second column. Logic would imply that two lines from the second column would have to move to the next page. But in TEX this rarely happens. For instance the glue around the displayed equation might shrink or stretch instead.

1.2 Grid setting

One of the most frequent complaints about TEX when setting double column text is that normally, the lines of text are not set on a grid. In other words, the baselines of one column do not align with those of the other. This is another consequence of the inherent stretchability of TEX's glue mechanism.

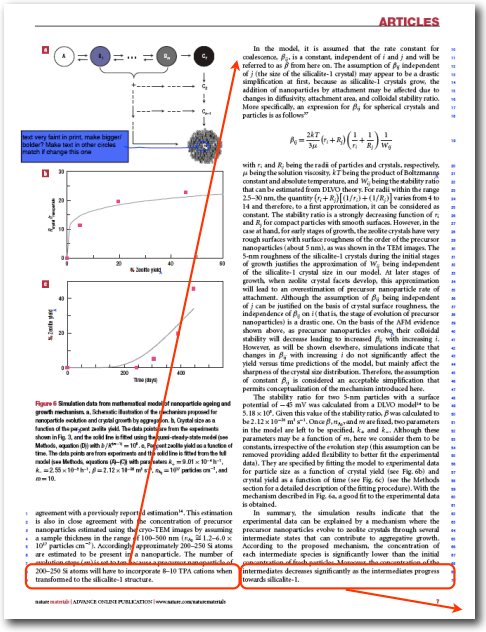


Figure 1: In TEX documents, taking two lines over from the first column does not necessarily result in two lines from the second column moving over.

1.3 Precise control over positioning of graphics

This is another related problem. Suppose we want to move a floating graphic slightly higher or lower, without affecting pagination. For example we might want the top of a graphic appearing at the beginning of a column to be moved up a fraction in order to align with the top of the text in the next column. With the usual LATEX commands this is not easy. I am mentioning this here as our macros for controlling glue allowed us to control the exact positioning of graphics too.

2 How TEX makes up pages

Figure 2 shows TEX's normal mechanism for setting a page. First of all the boxes are arranged in the vertical list, spaced out by the natural height of the glues assigned. Then, TEX stretches or shrinks the glues so as to fit the boxes and make the last baseline align with the bottom of the page. As is evident, it is difficult to control the positions of the baselines using this method.

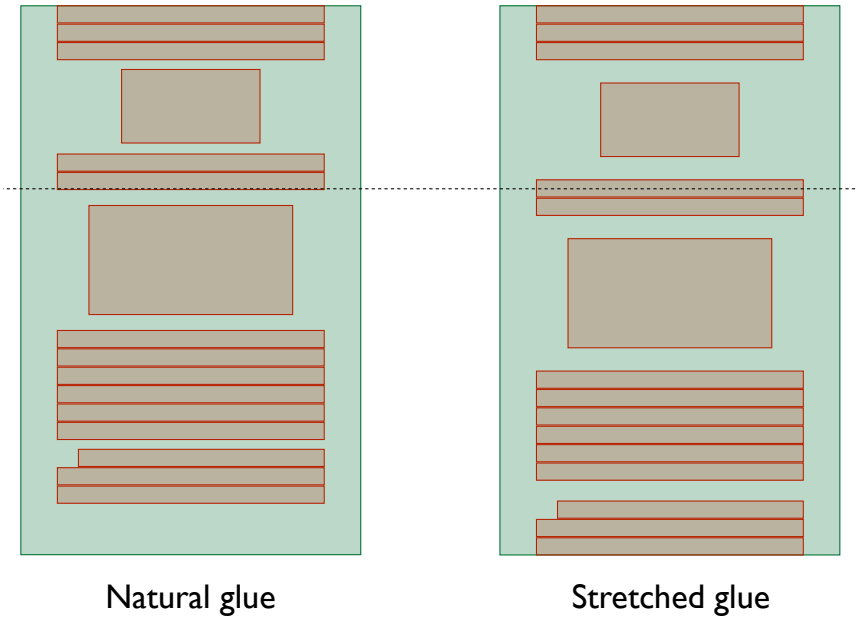


Figure 2: Stretching of glue to fit text to a page. The positions of the baselines are hard to predict when glue has stretchability.

3 Our approach to the solution

At River Valley, we have thought about and discussed extensively the methods we might use to effect grid setting. These include testing each box in the vertical list on the fly, and tweaking the vertical position. We tried to do this, both using TEX macros, and also doing it at compiler level, using pdfTEX. Unfortunately this approach did not work.

The more successful strategy was to try and make sure that all items in the vertical list had heights which were integral multiples of the value of `\baselineskip`. Examples of these items are:

- Floating elements (e.g. figures and tables)
- Displayed equations
- Section headers
- All skips between paragraphs, etc

Figure 3 shows some of the many glue parameters that are inserted in the vertical list and which must conform to this rule.

Let's look in more detail at how we dealt with specific issues.

3.1 Glue at the top of each column

When TEX starts typesetting a page, it inserts a glue called `\topskip` at the top of the column. The value of this glue is derived from a complex formula involving the elements in the first line. To simplify matters, we set `\topskip` to the value of `\baselineskip` which simplified matters considerably and did not produce any problems.

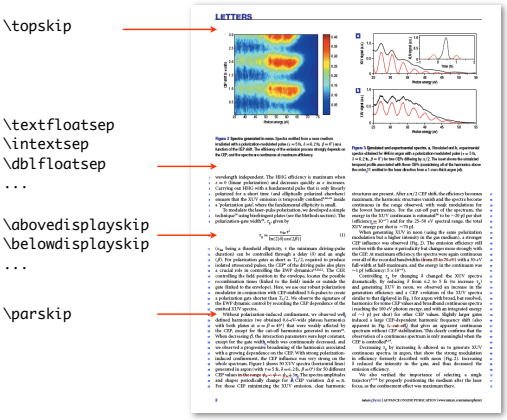


Figure 3: Some of the many vertical glues that can affect pagination. In general, baselines of the two columns are not aligned on a grid.

3.2 Stretching baseline glue

`\lineskip` and `\lineskiplimit` are two parameters that can cause big headaches in grid setting. Here is the logic: TEX sets paragraphs by putting one line above another, normally spacing out lines by the value of `\baselineskip`. The exception comes when TEX thinks two adjacent lines might clash. In particular, TEX examines the depth of each line of text and the height of the succeeding line. If, when placing the usual `\baselineskip`, TEX finds that the boundaries of the boxes containing the adjacent lines is closer than the value of `\lineskiplimit`, then the normal procedure is aborted, and a glue equal to the

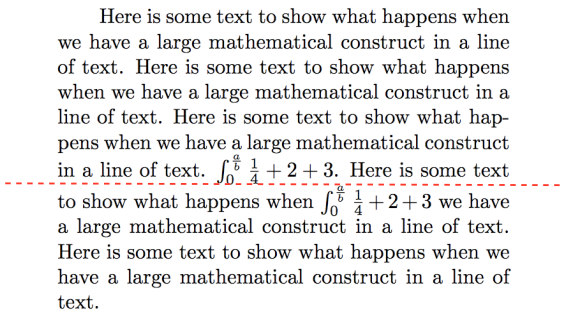


Figure 4: Variation of leading when large inline maths is present.

value of `\lineskip` is inserted. As we can see from figure 4, this can result in variable line spacing, making grid setting impossible.

We looked at the instances where `\lineskip` had been applied. In most cases, the normal leading would have sufficed, as the oversized elements were not aligned with each other. TEX does not know the horizontal position of the offending boxes, so in general there is no clash of text. Even when they were aligned, in most cases we could avoid the clash by reformatting the paragraph. The publications we were considering for grid setting contained only light mathematics, so we decided that we would do away with using `\lineskip` and just keep an eye out for clashing items, and deal with them on a case by case basis. So we chose the following values:

```
\lineskiplimit = -10pt
\lineskip = 0pt
```

The negative value of `\lineskiplimit` instructs TEX not to apply `\lineskip` unless there is an overlap of more than `10pt` between two adjacent lines. The value given to `\lineskip` is unimportant. Of course this might give very ugly overlapping lines, but we would pick these up while checking proofs, and we would deal with them manually. For seriously overlapping maths, we would change from inline to display math.

3.3 Dealing with floating elements.

Floating elements, such as figures or tables, generally consist of the main float, e.g. the graphic or the table, a caption, and three glue elements, one above the complete float, one below, and one separating the main element from the caption. In order to maintain grid setting, we need to control the vertical size of all these five elements. The three glue elements are set such that the total natural height is equal to an integral number of `\baselineskip`s. The main element and the caption are rounded up or down, so that their heights are also an integral number of `\baselineskip`s. This is done through a '`\roundoff`' macro that is executed at run time.

The general macro for floats is as follows:

```
\begin{figure}
 \centering
 \XFigure{figure1}
   {\label{fig1}Caption of figure.}
\end{figure}
```

which is similar to the normal LATEX float macro. We have made the control of spacing more useful by using `keyval.sty`, and adding the following options:

```
[beforegr  = ...pt]
[aftergr   = ...pt]
[beforecap = ...pt]
[aftercap  = ...pt]
[line      = ...]
```

These options allow the main element or the caption to be moved up or down. This is done before the `\roundoff` macro is executed, so the final height of each element will still be an integral number of `\baselineskip`s. The final option is a negative or positive integer, and adjusts the three glue parameters such that the complete height of the float is an integral number of lines larger or smaller. This is useful in solving pagination problems.

3.4 Displayed equations

For equations that do not break across pages, we again use `\roundoff` to make them fit the grid. The display glues such as `\abovedisplayskip` are set to fixed amounts, as before.

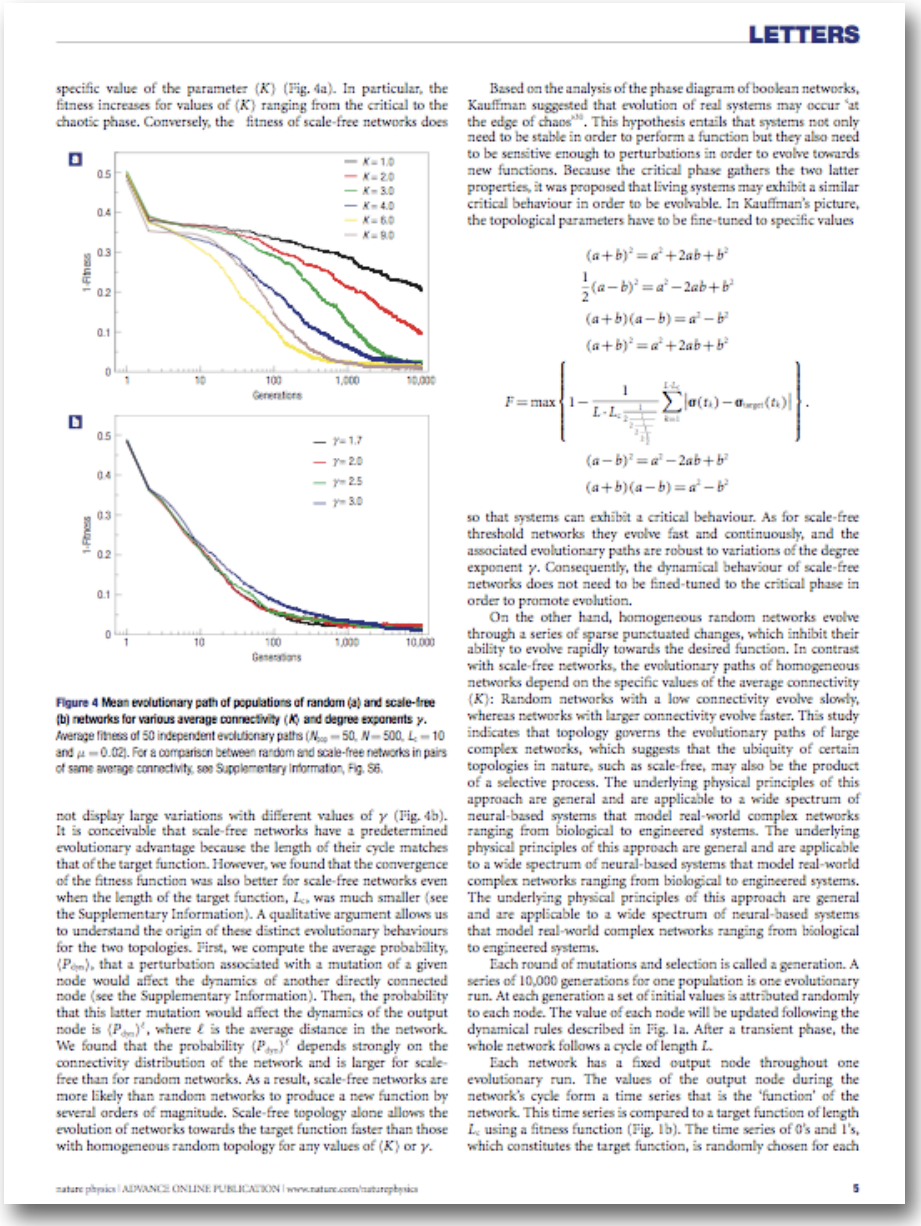
3.5 The one problem: Breaking displays

There is one problem we have not solved yet, namely that of automatically breaking displayed equations, and maintaining grid setting. The problem is that `\roundoff` produces one TEX box, so TEX's normal page breaking mechanism cannot be applied. For manuscripts with light mathematics, this is not a major problem, as the few such occurrences can be fixed manually, but it would be good to have an automated solution.

4 Results

Figures 5 and 6 shows a double column page set on a grid. The floating figure and the mathematics are all rounded off so that the text is set on a grid.

We have found that the time taken in pagination has reduced considerably after using the grid macros. When we need to move some lines from one column to the next, it results in exactly the same number of lines being taken over from the second column. The `keyval` options in floats allows us to deal easily with widows and orphans, by making a float one line longer or shorter.



LETTERS

specific value of the parameter ⟨K⟩ (Fig. 4a). In particular, the fitness increases for values of ⟨K⟩ ranging from the critical to the chaotic phase. Conversely, the fitness of scale-free networks does

Figure 4 Mean evolutionary path of populations of random (a) and scale-free (b) networks for various average connectivity ⟨K⟩ and degree exponents γ. Average fitness of 50 independent evolutionary paths ($N_{ini} = 50$, $N = 500$, $L_c = 10$ and $\mu = 0.02$). For a comparison between random and scale-free networks in pairs of same average connectivity, see Supplementary Information, Fig. S6.

not display large variations with different values of γ (Fig. 4b). It is conceivable that scale-free networks have a predetermined evolutionary advantage because the length of their cycle matches that of the target function. However, we found that the convergence of the fitness function was also better for scale-free networks even when the length of the target function, L_c, was much smaller (see the Supplementary Information). A qualitative argument allows us to understand the origin of these distinct evolutionary behaviours for the two topologies. First, we compute the average probability, $\langle P_{dyn} \rangle$, that a perturbation associated with a mutation of a given node would affect the dynamics of another directly connected node (see the Supplementary Information). Then, the probability that this latter mutation would affect the dynamics of the output node is $\langle P_{dyn} \rangle^{\ell}$, where ℓ is the average distance in the network. We found that the probability $\langle P_{dyn} \rangle^{\ell}$ depends strongly on the connectivity distribution of the network and is larger for scale-free than for random networks. As a result, scale-free networks are more likely than random networks to produce a new function by several orders of magnitude. Scale-free topology alone allows the evolution of networks towards the target function faster than those with homogeneous random topology for any values of ⟨K⟩ or γ.

Based on the analysis of the phase diagram of boolean networks, Kauffman suggested that evolution of real systems may occur 'at the edge of chaos'[10]. This hypothesis entails that systems not only need to be stable in order to perform a function but they also need to be sensitive enough to perturbations in order to evolve towards new functions. Because the critical phase gathers the two latter properties, it was proposed that living systems may exhibit a similar critical behaviour in order to be evolvable. In Kauffman's picture, the topological parameters have to be fine-tuned to specific values

$$(a+b)^2 = a^2 + 2ab + b^2$$
$$\frac{1}{2}(a-b)^2 = a^2 - 2ab + b^2$$
$$(a+b)(a-b) = a^2 - b^2$$
$$(a+b)^2 = a^2 + 2ab + b^2$$
$$F = \max\left\{1 - \frac{1}{L - L_c \frac{1}{2-\frac{1}{2^{L-1}}}} \sum_{k=1}^{L-L_c} |\sigma(t_k) - \sigma_{target}(t_k)|\right\}.$$
$$(a-b)^2 = a^2 - 2ab + b^2$$
$$(a+b)(a-b) = a^2 - b^2$$

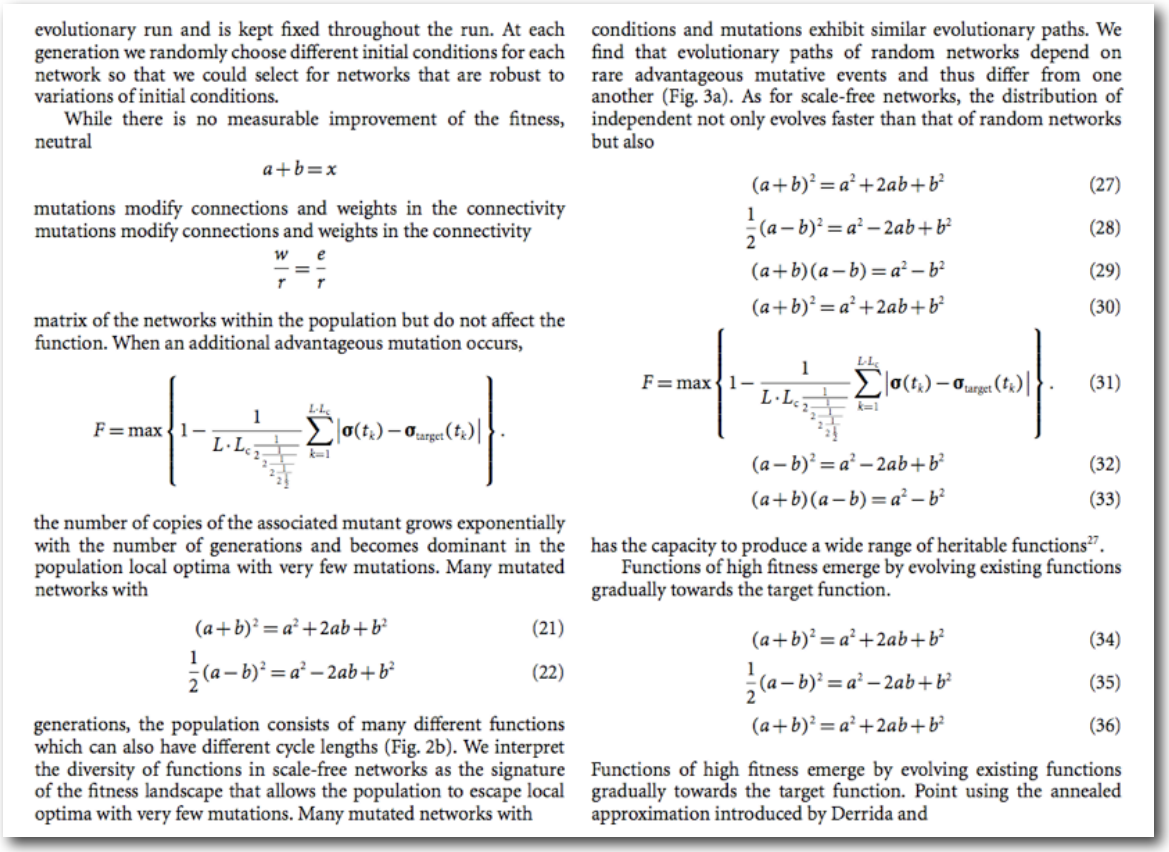
so that systems can exhibit a critical behaviour. As for scale-free threshold networks they evolve fast and continuously, and the associated evolutionary paths are robust to variations of the degree exponent γ. Consequently, the dynamical behaviour of scale-free networks does not need to be fined-tuned to the critical phase in order to promote evolution.

On the other hand, homogeneous random networks evolve through a series of sparse punctuated changes, which inhibit their ability to evolve rapidly towards the desired function. In contrast with scale-free networks, the evolutionary paths of homogeneous networks depend on the specific values of the average connectivity ⟨K⟩: Random networks with a low connectivity evolve slowly, whereas networks with larger connectivity evolve faster. This study indicates that topology governs the evolutionary paths of large complex networks, which suggests that the ubiquity of certain topologies in nature, such as scale-free, may also be the product of a selective process. The underlying physical principles of this approach are general and are applicable to a wide spectrum of neural-based systems that model real-world complex networks ranging from biological to engineered systems. The underlying physical principles of this approach are general and are applicable to a wide spectrum of neural-based systems that model real-world complex networks ranging from biological to engineered systems. The underlying physical principles of this approach are general and are applicable to a wide spectrum of neural-based systems that model real-world complex networks ranging from biological to engineered systems.

Each round of mutations and selection is called a generation. A series of 10,000 generations for one population is one evolutionary run. At each generation a set of initial values is attributed randomly to each node. The value of each node will be updated following the dynamical rules described in Fig. 1a. After a transient phase, the whole network follows a cycle of length L.

Each network has a fixed output node throughout one evolutionary run. The values of the output node during the network's cycle form a time series that is the 'function' of the network. This time series is compared to a target function of length L_c using a fitness function (Fig. 1b). The time series of 0's and 1's, which constitutes the target function, is randomly chosen for each

Figure 5: Example of a double column page set on a grid.



evolutionary run and is kept fixed throughout the run. At each generation we randomly choose different initial conditions for each network so that we could select for networks that are robust to variations of initial conditions.

While there is no measurable improvement of the fitness, neutral

$$a+b=x$$

mutations modify connections and weights in the connectivity mutations modify connections and weights in the connectivity

$$\frac{w}{r}=\frac{e}{r}$$

matrix of the networks within the population but do not affect the function. When an additional advantageous mutation occurs,

$$F=\max\left\{1-\frac{1}{L\cdot L_c\frac{1}{2\frac{1}{2\frac{1}{2}}}}\sum_{k=1}^{L\cdot L_c}\left|\boldsymbol{\sigma}(t_k)-\boldsymbol{\sigma}_{\text{target}}(t_k)\right|\right\}.$$

the number of copies of the associated mutant grows exponentially with the number of generations and becomes dominant in the population local optima with very few mutations. Many mutated networks with

$$(a+b)^2=a^2+2ab+b^2 \tag{21}$$

$$\frac{1}{2}(a-b)^2=a^2-2ab+b^2 \tag{22}$$

generations, the population consists of many different functions which can also have different cycle lengths (Fig. 2b). We interpret the diversity of functions in scale-free networks as the signature of the fitness landscape that allows the population to escape local optima with very few mutations. Many mutated networks with conditions and mutations exhibit similar evolutionary paths. We find that evolutionary paths of random networks depend on rare advantageous mutative events and thus differ from one another (Fig. 3a). As for scale-free networks, the distribution of independent not only evolves faster than that of random networks but also

$$(a+b)^2=a^2+2ab+b^2 \tag{27}$$

$$\frac{1}{2}(a-b)^2=a^2-2ab+b^2 \tag{28}$$

$$(a+b)(a-b)=a^2-b^2 \tag{29}$$

$$(a+b)^2=a^2+2ab+b^2 \tag{30}$$

$$F=\max\left\{1-\frac{1}{L\cdot L_c\frac{1}{2\frac{1}{2\frac{1}{2}}}}\sum_{k=1}^{L\cdot L_c}\left|\boldsymbol{\sigma}(t_k)-\boldsymbol{\sigma}_{\text{target}}(t_k)\right|\right\}. \tag{31}$$

$$(a-b)^2=a^2-2ab+b^2 \tag{32}$$

$$(a+b)(a-b)=a^2-b^2 \tag{33}$$

has the capacity to produce a wide range of heritable functions[27].

Functions of high fitness emerge by evolving existing functions gradually towards the target function.

$$(a+b)^2=a^2+2ab+b^2 \tag{34}$$

$$\frac{1}{2}(a-b)^2=a^2-2ab+b^2 \tag{35}$$

$$(a+b)^2=a^2+2ab+b^2 \tag{36}$$

Functions of high fitness emerge by evolving existing functions gradually towards the target function. Point using the annealed approximation introduced by Derrida and

Figure 6: Example of a double column page with heavy maths, set on a grid.

5 Availability

We intend to release these macros in a free and open format. Our intention is to include them in a style file.

6 Acknowledgments

Han The Thanh did a lot of preliminary work in determining which route we should take. CV Rajagopal helped in writing the macros. Jagath and Rishi did the refinements and testing.

▷ Kaveh Bazargan
River Valley Technologies
www.river-valley.com

▷ CV Radhakrishnan
River Valley Technologies
www.river-valley.com

MlBibTeX's Architecture

Jean-Michel Hufflen

Abstract

This paper shows the architecture of MlBibTeX, our reimplementation of BibTeX focusing on multilingual features. Making precise the organisation and modules of this architecture allows us to show how MlBibTeX works and focus on the differences between MlBibTeX and BibTeX from a conceptual point of view. We also explain why this implementation using the Scheme programming language allows users to scrutinise the result of intermediate steps.

Keywords LaTeX, BibTeX, MlBibTeX, bibliography files, bibliography styles, bst, nbst, XML, XSLT.

Sommario

Questo paper illustra l'architettuta di MlBibTeX, una reimplementazione di BibTeX più adatta per lavorare in un contesto multilingua. Rendendo più precisa l'organizzazione ed i moduli è possibile mostrare come MlBibTeX lavora evidenziando le principali differenze esistenti tra MlBibTeX e BibTeX da un punto di vista concettuale. Verrà inoltre spiegato perché quest'implementazione fatta usando il linguaggio di programmazione Scheme permette di verificare i risultati degli step intermedi.

Parole chiave LaTeX, BibTeX, MlBibTeX, bibliografia, stile bibliografico, bst, nbst, XML, XSLT.

Introduction

Often the LaTeX (Lamport (1994)) word processor and BibTeX (Patashnik (1998)) bibliography processor are used in conjunction. Such an approach allows the 'References' section of a printed document to be given nice layout. Homogeneous typesetting for items being same type is provided by means of bibliography styles ruling the layout of articles, books, etc. As mentioned in (Mittelbach *et al.*, 2004, § 13.1), BibTeX has been stable for a long time. However, modern features, such as multilingual capabilities, have not been incorporated and are possible only by means of workarounds, most often by inserting LaTeX commands inside field values. As a consequence, using BibTeX to generate bibliographies for another output format than LaTeX may be difficult.

There are several projects aiming to develop BibTeX's successors (Mittelbach *et al.*, 2004, § 13.1.2). Among them, MlBibTeX—for '**M**ulti**L**ingual BibTeX—focuses on multilingual features. It uses XML[1] as a central formalism. Even if there is a compatibility mode for bibliography styles originating from BibTeX (Hufflen (2006a)), style designers are encouraged to develop new styles using a new language ((Hufflen, 2003, App. A & B)), nbst[2], close to XSLT[3] (W3C (1999)), the language of transformations used for XML texts.

The broad outlines of MlBibTeX have already been shown at the 2004 GUIT[4] conference and given in Hufflen (2005c). This present paper focuses on MlBibTeX's architecture, in the sense that we explain how MlBibTeX's modules are organised and how they interact. First, we recall how BibTeX proceeds. So we can point out the differences between BibTeX and MlBibTeX from a conceptual point of view. Reading this article only requires basic knowledge about LaTeX, BibTeX and XML. About its implementation, MlBibTeX is written using the Scheme programming language (Hufflen (2005b)). We will not go thoroughly into technical points in Scheme[5], but mention how intermediate results are built.

1 Situation

1.1 BibTeX's behaviour

As a recent reference, (Mittelbach *et al.*, 2004, § 12.1.3) explains how LaTeX and BibTeX can cooperate. *Citation keys*—e.g., '`eco1980`' in '`\cite{eco1980}`'—are stored by LaTeX into an auxiliary (.aux) file also used by BibTeX. Now let us describe BibTeX's behaviour roughly. First, it takes an .aux file[6] and searches it for citations. This .aux file also contains the information about a bibliography style. Then this bibliography style file is run. If we look at some standard bibliography files, some variables, functions and macros are defined, bibliographic databases are searched for citation keys, and the accurate functions of the style are applied. Figure 1 describes the organisation of a bibliography style file as it is sketched

1. e**X**tensible **M**arkup **L**anguage. Readers interested in an introductory book to this formalism can refer to Ray (2001).

2. **N**ew **B**ibliography **ST**yles.

3. e**X**tensible **S**tylesheet **L**anguage **T**ransformations.

4. ***G**ruppo **U**tilizzatori **I**taliani di TEX* (Italian TeX user group).

5. The current reference manual of this language is Kelsey *et al.* (1998).

6. Let us recall that when BibTeX is invoked by means of a command line such as '`bibtex` ⟨*job-name*⟩', ⟨*job-name*⟩ refers to an .aux file.

```
ENTRY {...}{...}{...}      % Declaration of variables and functions.
FUNCTION {...}{...}
...
FUNCTION {article}{...}
...
MACRO {oct}{"October"}
...
READ                       % Search bibliography data base .bib files for entries.
...                        % Some preprocessing, including the sort of selected entries.
SORT
...
EXECUTE {begin.bib}        % Preamble and \begin{thebibliography}.
EXECUTE {...}              % Some initialisations.
ITERATE {call.type$}       % Loop over entries producing outputs.
EXECUTE {end.bib}          % Write \end{thebibliography} command.
```

Figure 1: Bibliography style file used by BibTeX: framework and basic flow.

in (Mittelbach *et al.*, 2004, § 13.6.2). To sum up: once the citation keys and the bibliography style are known, the latter is run. The result is a .bbl file containing a `thebibliography` environment ((Mittelbach *et al.*, 2004, § 12.1.2)), each reference being prefixed by means of the `\bibitem` command.

1.2 Going further

At a first glance, a successor of BibTeX might conserve the same *modus operandi*. That is not always possible. BibTeX never reads the .tex file because the information it needs can be found in .aux files only. The same behaviour is unsuitable for MlBibTeX—as a program managing multilingual bibliographies—when it generates a 'References' section for a LaTeX source text. Let us consider the following title, which uses a syntactical extension provided by MlBibTeX:

```
TITLE = {[Danse macabre] : french}
```

This book of Stephen King is written in English—of course—but its title uses French words, what is specified by the '`[...] : french`' notation, where '`french`' is a *language identifier* (Hufflen (2003)). Let us assume that we are building a 'References' section for a document in English. In order for LaTeX to use the right hyphenation patterns if need be, the .bbl file generated should include such a title as:

```
\foreignlanguage{french}{Danse macabre}
```

provided that the babel package is loaded with the french option when LaTeX processes this document ((Mittelbach *et al.*, 2004, Ch. 9)). First, there are several ways to write in French with LaTeX with the babel package or with other *ad hoc* packages. Second, let us recall that all the languages used throughout a document must be specified as options when the babel package is loaded. This package is static and the languages it uses cannot be extended dynamically. As we explain in Hufflen (2005a), there are several solutions to this problem, but we think that the best consists of analysing the preamble of the .tex file, in order to get information about the multilingual capabiblity chosen by a user. Let us go back to our title: if the french option has not been selected, MlBibTeX will just produce '`Danse macabre`' and warn the user that some words may be incorrectly hyphenated.

Another drawback of bibliography styles of BibTeX is that they are monolithic: one file for one style. There is a tool, `custom-bib` ((Mittelbach *et al.*, 2004, § 13.5.2)), that allows a bibliography style to be build automatically from users' wishes. But there is no modular approach in the sense that a style designer could develop a new style from existing style fragments. In MlBibTeX, a style is assembled dynamically, according to what is needed. For example, let us consider that a plain style[7] is to be used for a document written in English and including some fragments in Italian. The files that can be used are:

plain.nbst general definitions for this style.

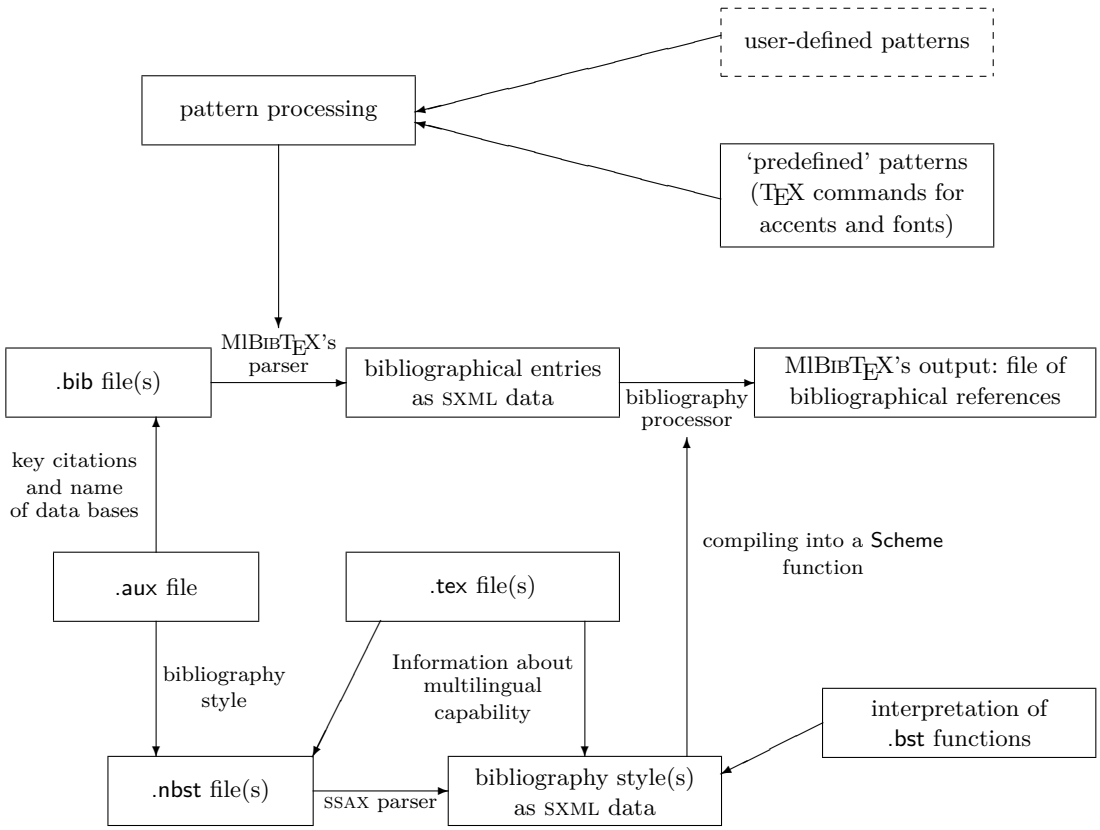
plain-{english,italian}.nbst definitions suitable for the English and Italian languages and belonging to this style.

{-english,-italian}.nbst general definitions for the English and Italian languages.

So we need this information about available languages before applying a style.

There is another difference, which can be noticed by an end-user, about abbreviations usable in a .bib file. Some may be defined using the `@STRING{...}`

7. That is, items are labelled by numbers.



'$A \leftarrow B$' means that A uses B. More precisely, functions or data put in A use functions of B or data from B. A dashed box means that the corresponding module is planned, but not fully implemented yet.

Figure 2: Data flow in MlBibTEX.

command of BibTEX ((Mittelbach *et al.*, 2004, § 13.2.3)), some may be defined as macros within the bibliography style ((Mittelbach *et al.*, 2004, Table 13.7)), for example, month names:

```
MONTH = oct
```

(cf. Figure 1). When BibTEX reads a .bib file, all the abbreviations must be expanded into strings. That is, all must be known at this time. On the contrary, in the styles of MlBibTEX, you can perform different operations whether or not a symbol is defined:

```
<nbst:choose>
  <nbst:when test="is-defined('roma')">
    ...
    <!-- The is-defined function does not exist
         in XSLT, but exists in nbst.
      -->
  </nbst:when>
  ...
</nbst:choose>
```

We think that all these differences between MlBibTEX and BibTEX are improvements but they lead to a redesign of the data flow in MlBibTEX.

2 Organisation of MlBibTEX

The organisation of MlBibTEX is pictured at Figure 2. Like BibTEX, MlBibTEX reads an .aux file and gets a list of citation keys. It also gets information about some bibliography data base files and a bibliography style.

Then it parses .bib files, searching them for the citation keys. This step results in an SXML[8] tree, that is, a representation of an XML tree using Scheme (Kiselyov (2005)). Let us consider that the entry given in Figure 3 is to be put within a bibliography. Conceptually, it can be viewed as the XML tree given in Figure 4, represented in SXML by the Scheme lists of Figure 5. Bibliography styles use XPath[9] expressions related to XML trees resulting from parsing a .bib file, that is, XML trees comparable with the example given in Figure 4.

Some symbols are not expanded because they have not been defined by a `@STRING` command in a .bib file—e.g., the `roma` symbol in Figure 3, see also Appendix A—and are retained for future pro-

8. Scheme implementation of XML.
9. XPath is the language used to address parts of an XML document (W3C (1993)).

```
@BOOK{eco1980,
        AUTHOR = {Umberto Eco},
        TITLE = {Il nome della rosa},
        PUBLISHER = {Bompiani},
        ADDRESS = roma,
        YEAR = 1980,
        LANGUAGE = italian}
```

Notice the `LANGUAGE` field and language identifier used to specify this entry for a book written in Italian.

Figure 3: MlBibTeX's bibliographical entry.

```
<book id="eco1980" language="italian">
  <author>
    <name>
      <personname>
        <first>Umberto</first>
        <last>Eco</last>
      </personname>
    </name>
  </author>
  <title>Il nome della rosa</title>
  <publisher>Bompiani</publisher>
  <year>1980</year>
  <address><symbol name="roma"/></address>
</book>
```

Figure 4: Entry as an XML tree (Fig. 3 cont'd).

cessing. In other words, that is a kind of 'benefit of the doubt'. The commands generating accented letters and other signs belonging to the Latin 1 encoding are expanded: for example, '`\'{e}`' is expanded to the accented letter 'é' of this encoding[10]. As a consequence, the accented and special letters included in the values of MlBibTeX's fields are viewed as single letters when entries are sorted. Besides, this sort operation can be performed w.r.t. a language's lexicographical order. The nbst language provides a `nbst:sort` element ((Hufflen, 2003, App. A)) that is analogous to the `xsl:sort` element of XSLT ((W3C, 1999, § 10)): if the data to be sorted are strings, a `language` attribute allows sorting w.r.t. a particular language (English, Italian, Spanish, ...)

Likewise, some predefined commands of LaTeX for fonts (`\emph`, `\textbf`, ...) are recognised and expanded to XML elements. This is done by means of patterns, expressed in Scheme. Now, only some predefined patterns are processed, e.g.:

```
(define-pattern "\'e" "é")
(define-pattern "\'{e}" "é")
```

we plan to extend this capability to user-defined patterns, so users will be able to mark up their .bib files, and tell MlBibTeX how it can understand this markup (Hufflen (2004a)).

During the next step, MlBibTeX deduces the source .tex file's name from the .aux file's[11]. Then it parses the preamble of this file and try to know:

- which languages are used throughout this document and how;
- which encoding may be used: pure ASCII[12] or Latin 1. So, users can type accented letters directly inside field values of .bib files processed by MlBibTeX, even if they do not use the inputenc package of LaTeX, for dealing with encodings other than ASCII ((Mittelbach *et al.*, 2004, § 7.5.2)). In this last case, the commands for generating accents will be used, for example, 'é' will be transformed into '`\'{e}`'.

This step results in:

- a list of possible file names[13] whose union is the multilingual bibliography style;
- updating a set of Scheme functions controlling MlBibTeX's output.

Then the files of the bibliography style are parsed by means of SSAX[14] (Kiselyov (2005)). This parser returns the SXML representation of an XML tree. So, the result is comparable to the expression given in Figure 5, but concerns the elements of a bibliography style. Finally, the fragments of a bibliography style are assembled and compiled into a Scheme function[15] (Hufflen (2006c)). Then this function is applied to the list of bibliographical entries selected.

Last, bibliography styles may use functions written using the bst language of BibTeX. That allows 'old' bibliography styles to be replaced progressively and explains the arrow, in Figure 2, from the box concerning bst functions to the box concerning bibliography styles expressed using the new language nbst.

3 Conclusion

When we started to develop MlBibTeX's first version (Hufflen (2001a)), we only aimed to provide a 'better BibTeX'. The only changes from end-users'

10. Now, most Scheme interpreters can only deal with the Latin 1 encoding. Future versions should be Unicode-compliant (Unicode Consortium (2003)) and able to deal with more encodings, such as Latin 2, UTF-8, ...

11. You can specify this .tex file's name by means of an option of MlBibTeX, if need be.

12. **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.

13. 'Possible files' because some files specific to a particular language may be inexistent, in which case they are ignored.

14. Scheme implementation of SAX (Simple API for XML).

15. Displaying a function's body is difficult and often not provided by most Scheme interpreters, that is why we do not give more details about this step. But we plan to develop tools for browsing bibliography styles.

```
(*top* ...
 (book (@ (id "eco1980") (language "italian"))
       (author
        (name (personname (first "Umberto")
                          (last "Eco"))))
       (title "Il nome della rosa")
       (publisher "Bompiani") (year "1980")
       (address (symbol (@ (name "roma"))))))
```

Figure 5: Entry in SXML (Fig. 4 cont'd).

point of view were additional syntactic features. This approach was not satisfactory enough and we decided to design a new language for bibliography styles (Hufflen (2001b)), taking advantage of XML features as far as possible. This design choice and the implementation of multilingual features led to an architecture quite far from BibTEX's. It may appear as more complicated, but we think that the role of each module is precise and can be understood by end-users, provided that they have got first experience with the Scheme programming language. But this architecture seems to us to be stable and robust. In particular, we succeeded in using MlBibTEX for deriving 'References' sections for ConTEXt (Hufflen (2006b)), another format built out of TEX[16] and very different from LaTEX. So, extending MlBibTEX to other output formats is possible.

A Abbreviations in MlBibTEX

This appendix aims to make precise how abbreviations—possibly multilingual—can be specified in MlBibTEX. As an example, we consider the `roma` symbol, that appears in the `eco1980` entry of Figure 3. Another example is given in Hufflen (2004b).

You can use a `@STRING` command, like in 'old' BibTEX ((Mittelbach *et al.*, 2004, § 13.2.3)):

```
@STRING{roma = {[Roma] * italian
                [Rome] * english
                [Rom] * german}}
```

the sequence of '`[...] * ...`' notations being a syntactical extension of MlBibTEX for information that *must* be put, possibly in another language if no translation is available (Hufflen (2003)). In other words, processing such a sequence never yields an empty string. But this `@STRING` command is expanded *verbatim* when a .bib file is processed—as in 'old' BibTEX—so it must be put in every .bib file where this `roma` symbol is used.

16. TEX provides a powerful framework to format texts nicely, but to be fit for use, the definitions of this framework need to be organised in a *format*. The first formats were *Plain TEX* and LaTEX, another format, more modern, is ConTEXt (Hagen (2001)).

A better solution, from our point of view, is to define this symbol like month names or predefined names of journals within the standard bst styles of BibTEX. If the `roma` symbol is not defined by a `@STRING` command, it is processed as we show in Figure 4 when a .bib file is parsed. It can defined within a bibliography style—that is, within a .nbst file—as follows:

```
<nbst:variable name="roma">
  <nonemptyinformation>
    <group language="italian">Roma</group>
    <group language="english">Rome</group>
    <group language="german">Rom</group>
  </nonemptyinformation>
</nbst:variable>
```

These two ways to define a symbol are close together, since a sequence of '`[...] * ...`' yields a `nonemptyinformation` element when .bib files are parsed. However, if an abbreviation has not been defined in 'old' BibTEX, a warning message is issued and the undefined symbol is just replaced by an empty string. As mentioned in § 1.2, the nbst language allows you to check that a symbol has been defined or not.

Acknowledgements

Many thanks to Maurizio W. Himmelmann for his patience when he was waiting for the first version of this article and for its Italian translation of the abstract. I also thank the referee of this article for the improvements suggested to me.

References

Hans Hagen: *ConTEXt, the Manual*. November 2001. `http://www.pragma-ade.com/general/manuals/cont-enp.pdf`.

Jean-Michel Hufflen: "MlBibTEX: a New Implementation of BibTEX". In: Simon Pepping, ed., *EuroTEX 2001*, pp. 74–94. Kerkrade, The Netherlands. September 2001.

Jean-Michel Hufflen: "European Bibliography Styles and MlBibTEX". TUG*boat*, Vol. 24, no. 3, pp. 489–498. EuroTEX 2003, Brest, France. June 2003.

Jean-Michel Hufflen: "MlBibTEX's Version 1.3". TUG*boat*, Vol. 24, no. 2, pp. 249–262. July 2003.

Jean-Michel Hufflen: "MlBibTEX: beyond LaTEX". In: Apostolos Syropoulos, Karl Berry, Yannis Haralambous, Baden Hugues, Steven Peter and John Plaice, eds., *International Conference on TEX, XML, and Digital Typography*, Vol. 3130 of LNCS, pp. 203–215. Springer, Xanthi, Greece. August 2004.

Jean-Michel Hufflen: "Making MlBibTeX Fit for a Particular Language. Example of the Polish Language". *Biuletyn* GUST, Vol. 21, pp. 14–26. 2004.

Jean-Michel Hufflen: *Managing Languages within MlBibTeX*. To appear. June 2005.

Jean-Michel Hufflen: "Implementing a Bibliography Processor in Scheme". In: J. Michael Ashley and Michel Sperber, eds., *Proc. of the 6th Workshop on Scheme and Functional Programming*, Vol. 619 of *Indiana University Computer Science Department*, pp. 77–87. Tallinn. September 2005.

Jean-Michel Hufflen: "MlBibTeX: a Survey". In: *Proc. GUIT Meeting*, pp. 171–179. Pisa,Italy. October 2005.

Jean-Michel Hufflen: "BibTeX, MlBibTeX and Bibliography Styles". *Biuletyn* GUST, Vol. 23, pp. 76–80. In *BachoTeX 2006 conference*. April 2006.

Jean-Michel Hufflen: "MlBibTeX Meets ConTeXt". In: *EuroTeX 2006 conference, preprints*, pp. 71–76. Debrecen, Hungary. July 2006.

Jean-Michel Hufflen: "Implementing a variant of XSLT in Scheme". In: *Proc.* IFL *'06*, pp. 484–491. Budapest, Hungary. September 2006.

Richard Kelsey, William D. Clinger, Jonathan A. Rees, 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, Vol. 11, no. 1, pp. 7–105. August 1998.

Oleg E. Kiselyov: XML *and Scheme*. September 2005. `http://okmij.org/ftp/Scheme/xml.html`.

Leslie Lamport: *LaTeX: A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LaTeX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

Oren Patashnik: *BibTeXing*. February 1988. Part of the BibTeX distribution.

Erik T. Ray: *Learning* XML. O'Reilly & Associates, Inc. January 2001.

The Unicode Consortium: *The Unicode Standard Version 4.0*. Addison-Wesley. August 2003.

W3C: XML *Path Language (XPath). Version 1.0*. W3C Recommendation. Edited by James Clark and Steve DeRose. November 1999. `http://www.w3.org/TR/1999/REC-xpath-19991116`.

W3C: XSL *Transformations (*XSLT*). Version 1.0*. W3C Recommendation. Edited by James Clark. November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`.

▷ Jean-Michel Hufflen
LIFC (FRE CNRS 2661)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
`hufflen@lifc.univ-fcomte.fr`

GNU Emacs e AUCTEX per LATEX

Onofrio de Bari

Sommario

L'intento di questo articolo è raccogliere informazioni e risorse utili per l'impiego dell'editor GNU Emacs con l'estensione AUCTEX nella redazione di documenti LATEX, presentando istruzioni e consigli che fino ad ora non sono stati portati a conoscenza degli utenti con documenti in lingua italiana.

In una prima analisi si introduce l'editor GNU Emacs, soffermandosi poi sulla sua struttura logica e su quali istruzioni possono essere date a Emacs per risultare comodo da usare con TEX e LATEX; vengono infine analizzati nel dettaglio i moduli software AUCTEX e preview-latex, utili nella redazione del codice sorgente e nella visualizzazione preventiva del documento.

1 Emacs e GNU Emacs

La nascita di Emacs, almeno come concetto di editore di macro (Editor MACroS), si può collocare nella metà degli anni '70. All'epoca l'editor più diffuso nei laboratori del Massachusetts Institute of Technology (MIT) era TECO, il quale aveva la peculiarità di trattare come modalità separate la digitazione del testo, la modifica e la visualizzazione del documento, un po' come l'attuale vi. Alcuni sviluppatori di software al MIT, tra cui Guy Steele e Richard Stallman, contribuirono in diversi modi alla nascita di quello che poi sarebbe stato Emacs; il primo unificò le varie macro per TECO che si erano sviluppate in quel periodo, mentre il secondo – dopo una visita allo Stanford AI Lab – implementò una macro che permetteva di avviare un sottoprogramma TECO alla pressione di un tasto; se oggi questo può far quasi sorridere, va chiarito che per l'epoca era un grandissimo successo.

Furono diverse le implementazioni di Emacs per vari sistemi operativi dell'epoca, fino a quando nacque la prima per Unix realizzata in linguaggio C, chiamata Gosling Emacs, dal nome dello sviluppatore James Gosling. Nel 1984, in coincidenza con la nascita del progetto GNU's Not Unix (GNU), Stallman iniziò a sviluppare un'alternativa libera (nel senso definito nella GNU Public License) a Gosling Emacs, realizzando anche un nuovo interprete per il linguaggio LISP, che viene tuttora usato per le estensioni software e per le impostazioni; è così che nacque GNU Emacs, la cui prima versione pubblica è la 13, del 20 marzo 1985. La versione attuale è la 21.4, del 6 febbraio 2005.

Sono oggi disponibili versioni di GNU Emacs per i vari Unix, Linux, Microsoft Windows, Mac OS X e altri sistemi operativi.

Un'altra variante di Emacs è XEmacs, derivante da un *forking* del progetto Emacs; molte considerazioni contenute in questo articolo (se non tutte) sono valide impiegando XEmacs al posto di GNU Emacs, ma personalmente uso GNU Emacs, quindi non sono a conoscenza di eventuali discrepanze.

2 Fondamenti dell'editor GNU Emacs

Non è facile dire perché si dovrebbere scegliere GNU Emacs come editor; è questione di simpatia iniziale o di abitudine, anche considerando che nel mondo Unix vi è da decenni una rivalità, più o meno scherzosa, tra utenti di Emacs e utenti di vi.

Avviando GNU Emacs si nota subito che lo schermo (o la finestra) è diviso in due *buffer* (memorie di transito); vi è in particolare un'area grande per scrivere delle note e un'area molto più piccola, in basso, detta *minibuffer*, dove vengono visualizzati i comandi e in cui vi è il vero e proprio "dialogo" tra il programma e l'utente.

Un primo strumento d'apprendimento è il *tutorial* a GNU Emacs, disponibile anche in italiano, ottenibile dal menu Help selezionando la voce *Emacs Tutorial (choose language)*; il programma chiederà nel minibuffer quale lingua usare (con la stringa `Language:`), quindi basterà scrivere la parola `italian` e premere Invio perché venga visualizzato un buffer con la guida a GNU Emacs in italiano.

3 Adattare GNU Emacs a LATEX

Un utente di GNU Emacs che ha intenzione di compilare un documento scritto in TEX o LATEX vuole evidentemente trovarsi a proprio agio nell'ambiente di programmazione; una prima richiesta è di solito l'evidenziazione delle parole chiave del linguaggio, ottenibile con l'inserimento della riga

```
(add-hook 'tex-mode-hook
  'turn-on-font-lock)
```

nel file `.emacs`, presente nella propria *home directory* nel caso in cui si usi Linux. Il file in questione contiene tutte le personalizzazioni di GNU Emacs, che vengono implementate in Elisp, un "dialetto" del linguaggio Lisp.

Nel mio file `.emacs` ho personalmente ritenuto opportuno aggiungere le righe

```
(setq tex-dvi-view-command xdvi -s 4 *)
    (setq-default fill-column 72)
```

che servono rispettivamente a impostare xdvi come visualizzatore DVI predefinito e a mantenere le righe in 72 colonne per comodità di visualizzazione.

Una volta completate le operazioni di personalizzazione, non rimane molto altro che imparare dal tutorial e provare; dopo la fase iniziale di apprendimento di Emacs per l'uso con LATEX si inizierà a desiderare uno strumento di editing più potente, ed è allora che viene in aiuto AUCTEX.

4 L'estensione AUCTEX

Il modulo software AUCTEX per GNU Emacs è stato realizzato nel 1992 da Kresten Krab Thorup e da altri studenti dell'Aalborg University Center (Danimarca); attualmente è sviluppato e aggiornato da David Kastrup.

Il sottotitolo "Un sofisticato ambiente TEX per Emacs", con cui Kastrup descrive il software nella copertina del manuale, rende molto bene l'idea di quello che è AUCTEX. Non si tratta solo di una semplice raccolta di sequenze di tasti in GNU Emacs che facilitano la redazione del documento e l'immissione di comandi, ma di una collezione di funzionalità che coinvolge altresì l'interazione di Emacs con visualizzatori esterni per i formati DVI, Postscript e PDF, l'interpretazione della sintassi TEX e LATEX e il sostegno a particolarità linguistiche per documenti realizzati in lingue diverse dall'inglese.

AUCTEX è disponibile per sistemi alla Unix (Linux, FreeBSD e simili) così come per Microsoft Windows, Mac OS X e altri sistemi operativi.

Per quanto riguarda Linux, l'installazione di AUCTEX si esegue compilando il codice sorgente o utilizzando pacchetti precompilati per la propria distribuzione, mentre per Windows e MAC OS X è consigliabile procurarsi una versione precompilata di GNU Emacs contenente anche AUCTEX.

Dopo l'installazione, AUCTEX va attivato aggiungendo al proprio file `.emacs` le righe

```
(load auctex.el nil t t)
  (require 'tex-site)
```

allo scopo di caricare all'avvio il modulo software vero e proprio. La riga

```
(setq-default TeX-master nil)
```

fa sì che AUCTEX chieda quale sia il proprio file *master* per documenti costituiti da più file (un libro diviso in diversi capitoli, ad esempio), mentre le righe

```
(setq TeX-parse-self t)
(setq TeX-auto-save t)
```

abilitano l'interpretazione logica del codice sorgente TEX o LATEX con il salvataggio in una directory del risultato dell'interpretazione, in modo da non ripetere l'operazione nei successivi caricamenti del file.

4.1 Alcune funzionalità di AUCTEX

Il seguente elenco comprende alcune delle funzionalità aggiunte da AUCTEX alla modalità TEX e LATEX di GNU Emacs:

- inserimento di macro, ambienti e comandi mediante scorciatoie da tastiera, con un'impostazione "dialogica" nei confronti dell'utente, che viene guidato nelle scelte dei parametri opzionali e obbligatori del comando o dell'ambiente, anche tramite le funzioni di completamento della digitazione ottenibili con il meccanismo Unix della pressione del tasto Tab;
- scrittura di simboli matematici mediante scorciatoie da tastiera, attivabile con la combinazione `C-c ~`;
- evidenziazione e allineamento speciali per macro ed ambienti;
- esecuzione di TEX, LATEX, BibTEX da GNU Emacs;
- visualizzazione degli errori di compilazione;
- esecuzione di visualizzatori di file DVI, Postscript e PDF da GNU Emacs.

4.2 Impiego basilare di AUCTEX

L'inserimento di macro, ambienti e comandi con l'uso di scorciatoie è, all'inizio, l'aspetto più comodo di AUCTEX; in altri termini è possibile, a partire da un buffer vuoto di Emacs, non dover digitare nulla dei vari comandi

```
\begin{document}
...
\end{document}
```

per creare un documento LATEX. Si può realizzare un documento dal titolo `prova.tex` aprendo un nuovo buffer in GNU Emacs con il comando

```
C-x C-f
```

salvandolo con il nome `prova.tex`; AUCTEX presenta quindi nel minibuffer la richiesta del file *master*

```
Master file: (default /home/guit)  /guit/
```

intendendo con questo nome il file principale che contiene il preambolo.

Dopo aver risposto alla richiesta compare sullo schermo un buffer vuoto e digitando la sequenza per creare un nuovo ambiente LATEX

```
C-c C-e
```

nella finestra minibuffer di GNU Emacs apparirà la scritta

```
Environment type: (default document)
```

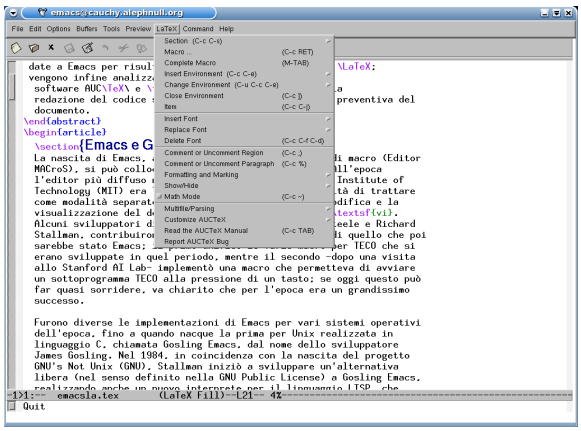


Figura 1: Il menu LaTeX di AUCTeX.

che chiederà quale ambiente si intende usare (in questo caso è `document` perché si vuole iniziare un documento *ex–novo*).

Dopo la pressione del tasto Invio si vedrà comparire la scritta

```
Document class: (default article)
```

e quindi si sceglierà la classe di documento; a questo punto nella finestra del minibuffer apparirà la scritta

```
Options:
```

che invita l'utente a digitare le opzioni per la classe di documento.

Il risultato ottenuto sarà qualcosa di simile a quanto segue

```
\documentclass[a4paper,11pt]{article}

\begin{document}

\end{document}
```

e ottenuto senza scrivere codice LaTeX.

Quanto è stato ora descritto è solo un banale e primitivo aspetto delle potenzialità di AUCTeX, ma può servire a dare l'idea di quanto ci si può aspettare da questo modulo aggiuntivo per GNU Emacs.

4.3 Il menu LaTeX in AUCTeX

La prima voce, *Section*, del menu LaTeX di AUCTeX permette di inserire i comandi di sezionamento nel documento.

La voce *Macro* permette l'inserimento di una macro, mentre con *Complete Macro* è possibile scrivere parti di comando LaTeX per far completare il nome a AUCTeX; a titolo di esempio, per scrivere un comando `\renewcommand` basta scrivere "`\renewc`", quindi premere la sequenza `M-Tab` (nei personal computer il tasto `Alt` seguito dal tasto `Tab`) per ottenere il comando completo `\renewcommand`.

Selezionando invece *Insert Environment* si può inserire un ambiente senza dover digitare tutti i caratteri che compongono la sintassi; in questo senso AUCTeX è un antesignano dei sistemi "visuali" come LyX, con la differenza che in AUCTeX si possono utilizzare tutte le potenti funzionalità di GNU Emacs e si può facilmente lavorare "a mano" con il codice, possibilità che –personalmente– trovo abbastanza preclusa in ambienti di scrittura LaTeX con approccio visuale.

Change Environment si rivela anch'essa molto utile, dato che molto spesso accade di cambiare ambiente in LaTeX perché la resa tipografica non soddisfa. Si può pensare al caso in cui, nella redazione di un documento $\mathcal{AMS}$-LaTeX, si sceglie prima un ambiente `align` per scrivere una formula, mentre poi alcune modifiche nel contenuto della stessa fanno optare per un ambiente `gather`; a quel punto basta selezionare *Change Environment* dal menu quando si è con il cursore dentro l'ambiente da sostituire (o servirsi della scorciatoia `C-u C-c C-e`) per ottenere il nuovo ambiente desiderato.

Per concludere la descrizione del sottomenu riguardante gli ambienti, *Close Environment* permette di aggiungere semplicemente un

```
\end{environment}
```

dove `environment` indica l'ambiente utilizzato.

Il sottomenu successivo riguarda i comandi relativi alle fonti; molto semplicemente, con *Insert Font* si potrà scegliere se inserire il carattere in corsivo, maiuscoletto e così via, mentre con *Replace Font* si possono sostituire tali attributi in modo analogo a quello descritto per gli ambienti; con *Delete Font*, infine, si cancella il punto più interno contenente una specifica di fonte.

In un altro sottomenu troviamo funzionalità inerenti alla gestione del codice sorgente nel suo aspetto; la voce *Comment or Uncomment Region*, in particolare, permette di far precedere ogni riga della regione da un carattere % di commento. In GNU Emacs una regione è un'area selezionata con il mouse o con un *mark* (comando `C-<SPC>`, documentato a pag. 71 in Stallman (2002)). L'opzione *Comment or Uncomment Paragraph* svolge lo stesso compito per un paragrafo.

Con *Formatting and Marking* si possono ottenere degli spazi bianchi all'inizio delle righe che riflettono la struttura sintattica del documento, al fine di permettere una maggiore leggibilità e una più agevole ricerca degli errori (il mancato bilanciamento delle parentesi, ad esempio).

Da *Show/Hide* si può accedere al *Fold Mode*, un'interessante caratteristica che permette di nascondere oggetti che non sono parte del testo che si sta scrivendo, come note o citazioni, oppure di evitare che compaiano nel testo le sequenze di cambiamento degli attributi tipografici delle fonti. Se ad esempio in una parte di testo si è scritto

```
\texttt{pippo}
```

quando si selezionerà la voce *Show/Hide → Fold Mode* per attivare la funzione e quindi *Show/Hide → Hide Current macro* si leggerà nel buffer di Emacs solo la stringa `pippo`, opportunamente evidenziata e/o colorata in base alle impostazioni predefinite o personalizzate della modalità *Font Lock*, che permette di evidenziare con un particolare colore o fonte tipografica il codice di *markup* costituito da ambienti o comandi. Tutto questo si può evidentemente estendere al nascondere e al far riapparire tali stringhe "accessorie" in una regione o in un intero buffer.

La scrittura di simboli matematici e testo "matematico" mediante scorciatoie, attivabile selezionando la voce *Math Mode* è uno dei punti di forza di AUCTEX; servendosi di tale modalità si possono, ad esempio, inserire un comando come `\Delta` semplicemente con la sequenza `‘D`. È superfluo dire che tutto questo vale anche per inserire in breve nomi di operatori, simboli relazionali, caratteri in grassetto e quanto è utile per indicare oggetti diversi nella tipografia matematica. Per conoscere in dettaglio le sequenze da usare si consulti la *AUCTEX Reference Card*, presente nella distribuzione di AUCTEX nel file `tex-ref.tex` e comunque facilmente reperibile su Internet.

Con *Multifile/Parsing* si può controllare in maniera alquanto agevole un documento con struttura multipla costituito da un file principale (*master file*) e da altri file subordinati.

La voce *Customize AUCTEX* permette di accedere al buffer di personalizzazione di AUCTEX, nello stile grafico delle ultime versioni di GNU Emacs; dalla versione 20 infatti le modifiche al file `.emacs` possono essere effettuate non solo digitando codice Elisp ma anche impostando le variabili di ambiente di Emacs dal buffer di personalizzazione.

È molto comoda la possibilità di leggere come buffer di Emacs il manuale di AUCTEX selezionando la voce *Read the AUCTEX Manual*; verrà visualizzato un buffer con il manuale in formato TeXinfo.

Selezionando infine la voce *Report AUCTEX Bug* è possibile informare gli autori del software di bug presenti (naturalmente avendo cura di controllare che non siano già elencati tra i bug noti).

5 L'estensione preview-latex

Il modulo preview-latex permette di visualizzare nel buffer di GNU Emacs contenente il sorgente del documento un'anteprima di quello che sarà il risultato finale della compilazione, potendosi così vedere le formule matematiche, le figure, i titoli di capitoli e paragrafi. preview-latex è stato realizzato da David Kastrup, Alan Shutko, Jan-Åke Larsson e Nick Alcock; in precedenza veniva distribuito separatamente, ma dalla versione 11.81 è integrato in AUCTEX almeno nel senso che le procedure di installazione dei due moduli software sono state unificate.

Dopo aver installato preview-latex occorre attivarlo con la riga

```
(load preview-latex.el nil t t)
```

inserita nel file `.emacs`. All'apertura di un file con estensione `.tex` in GNU Emacs si vedrà quindi un menu *Preview* con dei sottomenu autoesplicativi che seguono la logica del menu *LATEX* di AUCTEX. Il primo è dedicato alla generazione di un'anteprima per un particolare punto del testo, per un ambiente, per un paragrafo, per una regione o per tutto il documento, mentre il secondo riguarda la rimozione delle anteprime.

Se si usa Emacs in modalità grafica si può, con l'aiuto del mouse, agire sulle singole anteprime. Portando, ad esempio, il puntatore del mouse su un titolo di paragrafo del quale è visualizzata un'anteprima, compare un piccolo menu; se si clicca con il tasto centrale del mouse si disabilita l'anteprima rendendo esplicito il codice "sottostante", mentre se si clicca con il tasto destro si apre un menu con più opzioni, tra cui quella di disabilitare l'anteprima, rimuoverla per tornare alla normale visualizzazione del codice o rigenerarla.

Il sottomenu *Turn Preamble Cache* attiva o disattiva la funzionalità con la quale, alla prima richiesta di anteprima, il preambolo del documento viene messo in cache per realizzare un "file di formato" che servirà per velocizzare la resa delle successive anteprime, evitando che ogni volta vengano di nuovo esaminati i comandi che compongono la parte dichiarativa del documento.

Le voci *Customize*, *Read Documentation* e *Report Bug* richiamano le medesime funzioni di quelle presenti nel menu LATEX di AUCTEX.

6 Conclusioni

Non sarebbe stato possibile, non solamente per ovvi motivi di spazio, descrivere tutte le potenzialità dell'uso di GNU Emacs e AUCTEX; quello che però voglio sperare sia avvenuto è almeno un semplice interessamento nei confronti di un approccio "vecchio stile" alla redazione di documenti LATEX che, a mio personale parere, ha non solo "fascino" per un utente un po' *geek* ma contiene in sé una grande funzionalità.

Pur sfuggendo alla banalità del "semplice è bello", ritengo che in un'epoca in cui si cerca eccessivamente nei software il fascino "visivo" sia una buona cosa servirsi di software che, pur se solo testuale, garantisce un valido compromesso tra l'essere a proprio agio con la tastiera e il mouse e la comoda visualizzazione su schermo delle informazioni, in special modo per l'utente che opera nel settore scientifico e non cerca fronzoli, bensì rigore ed efficienza. GNU Emacs e AUCTEX usati per redigere

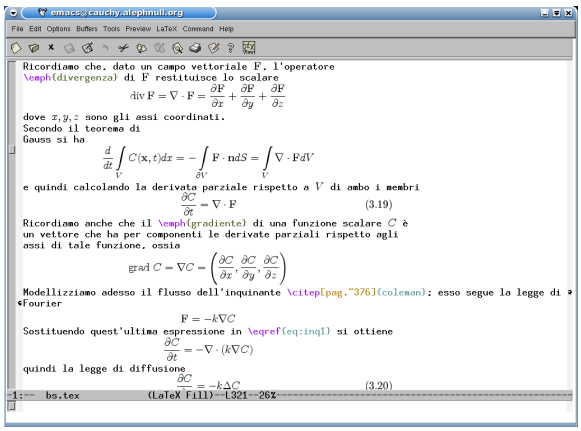


Figura 2: L'estensione preview-latex.

documenti TEX e LATEX sono sicuramente dei buoni candidati a raggiungere questo scopo – magari dopo una fase di apprendimento iniziale un po' ripida; non è però una sensazione che ognuno di noi ha provato iniziando a imparare TEX o LATEX?

Lascio valutare a chi legge se GNU Emacs e AUCTEX possono essere un buon mezzo per non limitarsi a "lasciar fare tutto al computer" e per avere comodità di uso unita al facile controllo su quel codice sorgente che da anni permette di ottenere i meravigliosi documenti realizzati con il sistema tipografico del professor Donald Knuth.

Riferimenti bibliografici

Stallman, R. (2002). *GNU Emacs Manual*. Free Software Foundation.

▷ Onofrio de Bari
onodebari@gmail.com

Test interattivi di Matematica e Fisica on-line: il LaTeX come strumento di sviluppo

Salvatore Palma

Sommario

Nell'anno scolastico 2005/06 appena trascorso, ho incominciato a realizzare sul sito della mia scuola un progetto di recupero e approfondimento di argomenti di Matematica e di Fisica per i miei studenti.

Nei miei lavori faccio uso soprattutto di AcroTeX del Prof. D.P. Story dell'università di Akron e di PdfScreen del Prof. C.V. Radhakrishnan.

1 Obiettivi e difficoltà

L'idea da cui sono partito era quella di dare agli allievi diversi tipi di "appunti" personali su alcune parti di programma, che trattavo in modo diverso dal libro di testo in uso, e chiarimenti su argomenti importanti per i quali occorrevano frequenti richiami. Volevo, inoltre, creare una serie di test per dare loro l'opportunità di verificare la propria preparazione con la possibilità di comprendere gli errori commessi.

Avevo già l'abitudine di scrivere in LaTeX le verifiche in classe, quindi, produrre appunti con formule non mi creava alcun problema.

La ricerca sul Web per creare test di Matematica, mi fornisce una quantità notevole di esempi, alcuni prodotti da case editrici per facilitare il lavoro dell'insegnante. Quasi tutti questi test venivano scritti in HTML e avevano la caratteristica di essere di un solo tipo: occorreva "spuntare" la risposta corretta scelta tra più risposte proposte.

I più belli, dal punto di vista didattico, davano la possibilità di correggersi, di valutarsi e di stampare la correzione.

Se volevo seguire questa strada dovevo convertire il LaTeX in HTML o incollare le formule necessarie come immagini, tutte le volte che la formulazione delle domande lo richiedeva.

Tra gli esempi di quiz on-line esaminati sul Web, segnalo quelli dell'Università di Vienna. Sono scritti in Java e sono di notevole impatto per lo studente.

Oltre alle tipologie classiche Vero/Falso o a risposta multipla, comparivano i puzzle, esercizi in cui si dovevano associare le risposte trascinandole accanto alle relative domande. A richiesta, l'Università di Vienna dava la possibilità di scriverne di personalizzati.

Io però volevo progettarli e scriverli senza disturbare nessuno e, soprattutto, non cercavo solo test a risposta multipla, del tipo V/F o puzzle, ma, ad una domanda del tipo: calcola la derivata della seguente funzione. . . , l'allievo doveva poter inserire la sua risposta e il test doveva interpretare la correttezza o meno della risposta inserita.

Lancio l'ennesima ricerca con Google, digitando distrattamente "Test interattivi in Latex" e trovo un articolo in PDF del Prof. PierLuigi Zezza:

Introduzione a MetM@t versione per lo schermo.

Nell'articolo, il Professore presentava il suo progetto di Matematica e quiz on-line e ringraziava i Proff. **C.V. Radhakrisnan** per il package PdfScreen, **Sebastian Rahtz** per il package hyperref, **Christian Schenk** per il progetto MikTeX e **Donald P. Story** *per il suo progetto AcroTeX ed in particolare per il package* exerquiz *che è stato utilizzato per tutti i test e quiz interattivi.*

Nei miei lavori, non avevo mai utilizzato i packages PdfScreen, hyperref e exerquiz.

Dopo un periodo di studio e documentazione mi convinco, definitivamente, che ciò che volevo fare era già stato fatto dal Prof. D.P. Story per i suoi allievi. Dovevo solo imparare ad usare i packages sopra citati e adattare, alle mie personali esigenze, i numerosi esempi presenti nei lavori del professore.

Segnalo le uniche due mie conoscenze di "lavori italiani" con questi pacchetti:

1. A.M.B.O. Argomenti di Matematica di Base per l'Orientamento. Un grosso lavoro per l'orientamento coordinato dal Prof. Giuseppe Accascina dell'università di Roma, cui hanno collaborato anche alcuni docenti delle superiori.

2. MetM@t - Modelli e Metodi Matematici del Prof. PierLuigi Zezza della Università di Firenze.

2 Cos'è AcroTeX

AcroTeX, acronimo di Acrobat e LaTeX, è un progetto che si propone di pubblicare su Web quiz interattivi, test da spedire al docente, lavori per casa. Una interazione docente-studente via web!

In "AcroTeX eDucation Bundle", il Prof. D.P. Story definisce così il suo lavoro. . . *È una collezione di macro files in LaTeX, corredata di vari files di supporto ed esempi, progettata per la ePublication e concepita per il settore dell'educazione.*

Usa LaTeX come sistema autore per le applicazioni e Adobe Acrobat come programma per gestire e leggere i documenti prodotti. Il lavoro nasce nel 1999, ma è stato continuamente aggiornato e ampliato. Oggi si presenta così strutturato:

1. **web package** per presentare in modo piacevole i propri lavori;
2. **exerquiz package** per creare con facilità esercizi e test interattivi;
3. **eforms package** per gestire i form in Acrobat.

Altri packages: **insdljs** (per inserire javascript nei documenti PDF), **dljslib** (funzioni di libreria in Javascript), **eqExam** (per creare esami, test, lavori per casa da inviare al docente), **eq2db** (per salvare i risultati dei test in un database), completano l'opera veramente ammirevole del Prof. D.P. Story.

Exerquiz è il package che permette la realizzazione di tutti i lavori interattivi e mette a disposizione i seguenti ambienti:

- Exercise Environment: macro per creare esercizi on-line;
- Shortquiz Environment: macro per creare quiz interattivi con risposta immediata, ma senza soluzioni;
- Shortquiz con soluzioni: macro per creare quiz interattivi con risposta immediata con link alle soluzioni;
- Quiz Environment: macro per creare quiz, in cui i link, alle risposte, alle soluzioni, alla valutazione sono governati da JavaScripts.

Quest'ultimo ambiente è arricchito, nelle sue funzionalità, dal package **insdljs**. I test prodotti in questo ambiente sono molto belli.

Se si sanno creare istruzioni in JavaScript, insdljs permette di inserirle nei documenti PDF, digitandole nel sorgente LaTeX, anche senza usare exerquiz.

Exerquiz dà, inoltre, la possibilità di inserire risposte distinguendo tra due tipi di domande:

1. una domanda che richiede come risposta un'espressione matematica;
2. una domanda che richiede come risposta un testo;

e mette a disposizione le macro `\RespBoxMath` e `\RespBoxTxt` per "leggere" la risposta inserita. Un esempio verrà dato più avanti.

Come pacchetto per presentare, **web package** non è indispensabile, e, non essendoci incompatibilità tra exerquiz e PdfScreen, molti autori, me compreso, preferiscono quest'ultimo.

Voglio ancora sottolineare una caratteristica di questo pacchetto: la libertà di modificare dimensioni e colori dei bottoni, cambiare la scala di valutazione, attribuire punteggi diversi alle singole domande, punteggi parziali diversi ai singoli punti di una domanda strutturata in più quesiti ...

3 "RemeDialMaths"

Quando mi sono sentito sufficientemente sicuro nell'adattare alle mie esigenze, i numerosi esempi tovati in AcroTeX, ho presentato al Collegio dei Docenti del mio Istituto, un progetto di recupero e approfondimento di Matematica (e Fisica) on-line. I materiali prodotti dovevano essere inseriti sul sito della scuola e potevano essere usati da tutti gli studenti del Liceo. Ai colleghi chiedevo collaborazione nella progettazione dei test e di segnalarmi eventuali errori. Il progetto mi viene approvato all'unanimità il 9/9/05.

Il nome "RemeDialMaths", mi è stato suggerito successivamente da una collega di Inglese, che avendo capito cosa volevo realizzare, intendeva sottolineare l'azione a distanza della mia didattica.

Il progetto si proponeva di produrre:

- Test di autovalutazione con correzione;
- Test di verifica con punteggio, ma senza correzione;
- Test da compilare e inviare al docente (...ancora non realizzati);
- Esercizi e problemi (con o senza soluzione);
- Approfondimenti monotematici sotto forma di libro elettronico.

Quello che fin qui ho realizzato è:

1. *Test di ottica geometrica* (424 kb)
2. *Test di ottica geometrica con correzione* (520 kb) – Test eseguito durante un tirocinio attivo SIS. Si utilizza anche il file esterno di Cabri: specchio.fig
3. *Test sui logaritmi* (516 kb)
4. *Test sui logaritmi con correzione* (643 kb) – Test di valutazione
5. *Rotazioni di centro l'origine* (377 kb) – Approfondimento con test
6. *Funzioni trigonometriche* (348 kb) – Test di valutazione con punteggio e correzione
7. *Logaritmi* (791 kb) – Richiami di teoria e test di valutazione con punteggio e correzione
8. *Derivate 1* (461 kb) – Esercizi sulle applicazioni

9. *Derivate 2* (435 kb) – Test di valutazione con punteggio e correzione

A parte, *Relatività ristretta*. Un e-book sulla relatività ristretta con un test finale di verifica con versione a schermo e versione stampabile

Particolare importanza ho dato alla spiegazione degli errori.

Alcuni test possono fare riferimento a file esterni di Cabri o Derive, programmi molto noti nelle nostre scuole, da esplorare prima di rispondere ad una specifica domanda.

Le verifiche in classe rispecchiano il modello proposto on-line.

4 Esempi

Un tipico preambolo presente nei test è il seguente:

```
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage[absolute]{textpos}
\usepackage{multicol}
\usepackage{amsmath}
\usepackage{color,xcolor}
\usepackage[screen,sectionbreak]{pdfscreen3}
%pdfscreen3=versione modificata
%con men in italiano
\usepackage[pdftex,italian,execJS,%
  insdljs]{exerquiz}
\usepackage[colorlinks=true]{hyperref}
\usepackage{chemarrow}
\usepackage[italian]{babel}
```

Per il dimensionamento dello schermo ho fatto la scelta `\bottombuttons` con le istruzioni

```
\margins{.65in}{.65in}{.65in}{.65in}
\screensize{6.25in}{8.5in}
\overlay{Immagine5.pdf}
%overlay1.pdf righeg.pdf quartafront.pdf
%4c3.pdf panel.pdf gbhole1bis.pdf
%(vari sfondi personali per la
%prima pagina)
\bottombuttons
```

Bottoni personalizzati in dimensione, colori, messaggi...

```
\everyqRadioButton{\textColor{0 0 1 rg}%
  \BC{1 0 0}\BG{1 .973
  .863}\F{\FPrint}}%.690 .769 .871
\useBeginQuizButton[\BC{255 0 0}%
  \textColor{1 0 0}%
  \CA{Inizio }\RC{Test}\AC{}] % use buttons
\useEndQuizButton[\BC{255 0 0%
  }\textColor{1 0 0}%
  \CA{Fine}\RC{Test}\AC{}]
\everyCheckBox{\BC{1 0 0}%
  \BG{0.98 0.92 0.73}%
  \F{\FPrint}}%0.98 0.92 0.73
\everyeqTextField{\BG{1 .973 .863}}
\everyRespBoxMath{\rectW{1.8in}\textSize{0}}
\everyRespBoxTxt{\rectW{1.8in}\textSize{0}}
```

La valutazione finale, in relazione al punteggio, si può ridefinire anche con messaggi del tipo: correggiti, bravo, discreto... Io ho preferito la valutazione classica.

```
% \begin{verbatim}
\renewcommand\eqGradeScale{%
  "10",[95, 100],"9 ",[90,95],"9",[85,90],
  "8 ",[80,85],"8",[75,80],"7 ",[70,75],
  "7",[65,70],"6 ",[60,65],"6",[55,60],
  "5 ",[50,55],"5",[45,50],"4 ",[40,45],
  "4",[35,40],"3 ",[30,35],"3",[25,30],
  "2 ",[20,25],"2",[15,20],"1",[10,15],
  "0",[0,10]}
```

Un'osservazione importante. Quando si stampano i test, può accadere che alcuni form (nel mio caso i **RadioButton**), ed il loro contenuto, non vengano stampati.

Al termine del tirocinio di ottica geometrica, i miei allievi dovevano compilare il test progettato dal tirocinante, stamparlo e consegnarlo al docente. Oltre al punteggio in automatico, si doveva aggiungere la valutazione di una risposta libera che poteva essere di parecchie righe.

Provando il test, mi sono accorto di questo problema e ho controllato che tra gli attributi di campo dei vari form Story (a) fosse inserito `\F{\FPrint}`.

Ho quindi rimediato nel modo seguente:

```
% \begin{verbatim}
 \everyqRadioButton{\textColor{0 0 1 rg}
  \BC{1 0 0}\BG{1 .973 .863}\F{\FPrint}}
% \end{verbatim}
```

Lo schema di un test molto semplice, con punteggi e soluzioni si presenta così:

```
\begin{quiz}*{tiroc1}\label{tir1}
 \begin{questions}
   \item\PTs{3} domanda 1 con punti 3
     \begin{answers}
     .....
     \end{answers}
   \item\PTs{2} domanda 2 con punti 2
     \begin{answers}
     .....
     \end{answers}
   .......
   .......
 \end{questions}
\end{quiz}
```

mentre un parziale listato di *testlog2*, riferito alla prima domanda (figura 1), è dato in appendice.

```
% \begin{verbatim}
 \begin{quiz}*{log1}
 \hypertarget{log1}{}\label{log1}

 \begin{questions}
```

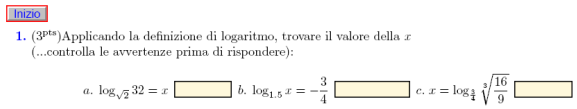


Figura 1: testlog2

```
%%%%%%%%%%%%%%%%%%%%1    %%%%%%%%%%%%%%%
\item\PTs{3}\label{def1}\hypertarget{1}{}
  Applicando la definizione di logaritmo,
  trovare il valore della $x$ (...controlla
  le avvertenze prima di rispondere):
  \vspace{-.1in}
 $$
 \def\vi#1{\setlength{\fboxrule}{0pt}%
   \setlength{\fboxsep}{1bp}
   \fbox{\RespBoxMath[\rectW{11bp}\Q{1}%
   \textSize{0}]{#1}*{1}{.0001}{[0,1]}}}
%
  \begin{mathGrp}[mygrpEval]\PTs*{1}
       a.\;\;\(\log_{\sqrt{2}}32=x\)\;\;
  \RespBoxMath[\rectW{.60in}\Q{1}%
    \textSize{0}]{10}{1}{.0001}{[0,1]}\quad
    b.\;\;\(\log_{1.5}x=%
      -\displayfrac{3}{4}\)\;\;
  \RespBoxTxt[\rectW{.80in}
    \Q{1}]{1}{0}{1}{sqrt[4](8/27)}\quad
  c.\;\;\(x=\log_{\frac{3}{4}}\sqrt[3]{%
    \displayfrac{16}{9}}\)\;\;
  \RespBoxTxt[\rectW{.60in}\Q{1}]%
    {1}{0}*{1}{-2/3}
  \end{mathGrp}\ \quad%
  \CorrAnsButtonGrp{10,sqrt[4](8/27),-2/3}
 $$
          %spiegazione degli errori
\begin{solution}\hyperlink{1}{..1 }\\
 Risolviamo i tre esercizi:\\
 \medskip
 \myblue{1a. }\triangolo
  Scriviamo, in base alla definizione di
 logaritmo, l'espressione
 data in forma esponenziale\\
 \[\log_{\sqrt{2}}32=x\rightarrow
   \big(\sqrt{2}\big)^x=32\rightarrow
   \Big(2^\frac{1}{2}\Big)^x=
   2^5\rightarrow 2^{\frac{x}{2}}=2^5
\]
 quindi\\
 \[\displayfrac{x}{2}=5\rightarrow x=10\]\\
 \medskip
 \myblue{1b. }\triangolo Idem
 \[\log_{1.5}x=-\displayfrac{3}{4}
   \rightarrow (1,5)^{-\frac{3}{4}}=x
   \rightarrow
   \Big(\displayfrac{15}{10}\Big)%
     ^{-\frac{3}{4}}=x
   \rightarrow
   \Big(\displayfrac{3}{2}\Big)%
     ^{-\frac{3}{4}}=x
\]\\
 quindi \\
 \[x=\Big(\displayfrac{2}{3}\Big)%
     ^{\frac{3}{4}}=\sqrt[4]
   {\Big(\displayfrac{2}{3}\Big)^3}=
   \sqrt[4]{\displayfrac{8}{27}}
 \]\\
 \medskip
 \myblue{1c. }\triangolo Idem
 \[x=\log_{\frac{3}{4}}\sqrt[3]{%
   \displayfrac{16}{9}}
   \rightarrow
   \Big(\displayfrac{3}{4}\Big)^x=%
     \sqrt[3]{\displayfrac{16}{9}}
   \rightarrow
   \Big(\displayfrac{3}{4}\Big)^x=%
     \Big(\displayfrac{16}{9}\Big)%
       ^{\frac{1}{3}}
   \rightarrow
   \Big(\displayfrac{3}{4}\Big)^x=%
     \Big(\displayfrac{4}{3}\Big)%
       ^{\frac{2}{3}}
 \]
 quindi
 \[\Big(\displayfrac{3}{4}\Big)^x=%
   \Big(\displayfrac{3}{4}\Big)%
     ^{-\frac{2}{3}}
   \rightarrow
   x=-\displayfrac{2}{3}\]\\
 \hrulefill\end{solution} \vspace{-.2in}
%%%%%%%% Fine domanda 1
```

5 Alcuni limiti

L'entusiasmo per AcroTeX non deve però farmi tralasciare alcuni problemi di notazione che ho incontrato.

Quando si deve inserire una risposta di tipo matematico, tutto funziona se l'inserimento rispetta le regole matematiche standard e si usano le funzioni classiche di libreria $\ln x, \arctan x \ldots$(fig.2)

Figura 2: derivata

ma se devo inserire $\log_2 x$, non posso farlo da tastiera.

Lo stesso accade per i simboli di operazioni $\cup, \cap$, tra insiemi, $\sqrt[n]{x}\ldots$

Io sono ricorso a notazioni non standard, che passo da tastiera, facendole accettare come testo. Dunque una risposta matematica passata come testo. Ad es., in figura 3, l'allievo deve inserire la risposta corretta $\sqrt[4]{\frac{8}{27}}$, leggendo le mie istruzioni (prima di fare il test), deve digitare da tastiera sqrt[4](8/27).

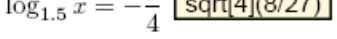


Figura 3: radice quarta

Discorso analogo per $\log_3 \frac{1}{2}$, nella domanda relativa al cambiamento di base (figura 4)

$$\log_{\frac{1}{2}} \frac{5}{8} = \frac{\log_3 \frac{5}{8}}{\dots}$$ log(1/2,3)

Figura 4: cambiamento di base

Le notazioni che suggerisco sono arbitrarie, anche se plausibili. In Derive, per es., $\log_2 x$ deve essere digitato come log(2,x).

Credo sia questo il principale motivo per cui, autori più preparati del sottoscritto preferiscano non usare i campi a riempimento.

6 Considerazioni finali e ricaduta didattica

Il lavoro di progettare e costruire test con il LaTeX è abbastanza oneroso, sia nella stesura che nella progettazione. È infatti improbabile che le due abilità siano concentrate, in maniera equilibrata, in un'unica persona.

Ho comunque voluto provare, anche se questa esperienza giunge verso la fine della mia carriera e non è capita dai colleghi; è opinione comune che ad una certa età si debbano "tirare i remi in barca", e attrezzarsi per una vita più tranquilla.

Alcuni, però, mi danno suggerimenti e promettono di provare questi materiali nelle loro classi.

Un collega più giovane, che sa usare LaTeX, è positivamente impressionato e mi ha promesso collaborazione.

La scelta di usare colori e sfondi può sembrare fastidiosa o pesante, ma si deve tenere conto che questi test sono concepiti principalmente per essere visualizzati e non stampati, tuttavia è possibile mettere sfondi bianchi quando si vogliono dare le due opportunità o prevedere la versione stampabile accanto a quella a schermo. E poi c'è un altro problema. Come si può *catturare* l'attenzione dello studente superficiale e, contemporaneamente, dare risposte che soddisfino lo studente seriamente motivato se l'aspetto estetico del test non è accattivante? Noi prof. delle superiori lavoriamo con studenti che vanno dai 14 ai 19 anni e che non sempre hanno scelto il liceo consapevolmente; talvolta l'indirizzo P.N.I.(Piano Nazionale dell'Informatica), con maggior carico di lavoro, è scelto dal quattordicenne perchè...lì si lavora col computer!

Lo studente reagisce bene a proposte didattiche messe sul web esplicitamente per lui dal proprio docente e consulta i lavori soprattutto se si accorge che le verifiche in classe sono dello stesso tipo o trattano gli stessi argomenti. Il test sui logaritmi, in quarta, era stato preceduto da due test di autovalutazione e la verifica in classe non è andata male. Lo stesso era accaduto per le trasformazioni del piano applicate alle funzioni trigonometriche e l'approfondimento sulle rotazioni, oggetto di verifica orale.

Nella classe quinta, il lavoro sulla Relatività ristretta, è stato fatto tutto sulle dispense e, il test finale di autoverifica con correzione, è stato anche un ottimo ripasso dell'argomento. Alcune applicazioni delle derivate, gli studenti le hanno studiate sugli esercizi on-line.

Molti miei studenti hanno propri siti e sanno fare con l'HTML cose eccezionali. Trovano strano che un vecchio docente come me produca questi materiali e li valutano anche sotto il profilo estetico, esprimendo pareri e suggerimenti... con un misto di ammirazione.

Proprio basandomi su questa ammirazione ho proposto un mini corso per apprendere il LaTeX. L'iniziativa è stata accolta con entusiasmo, come tutte le iniziative che implicano l'utilizzo del laboratorio informatico, ma il tempo è tiranno e non se ne è fatto nulla. Ho preferito proporlo solo ad alcuni veramente motivati a proseguire negli studi universitari in facoltà scientifiche, avvertendoli che il grosso del lavoro dovevano farlo nel loro tempo libero.

Un grande risultato l'ho ottenuto con uno studente particolarmente bravo[1]. La classe aveva fatto in laboratorio uno studio su alcuni luoghi geometrici: la cissoide di Diocle, la strofoide retta, la versiera di Agnesi, la concoide di Nicomede utilizzando il programma Cabri per tradurre le proprietà geometriche di questi luoghi e, successivamente, in un riferimento cartesiano si doveva dedurre l'equazione del luogo, fare lo studio completo e verificare con Derive l'esattezza delle conclusioni. Mentre i suoi compagni hanno utilizzato PowerPoint, Word o FrontPage per creare la loro relazione, lo studente mi chiede istruzioni per scriverla in LaTeX. Incredulo gli fornisco tutte le informazioni necessarie e il CD con MikTeX e WinEdt. Solo alcune domande al termine delle ore di lezione per chiedermi come si inseriscono i grafici o tabelle... mi fanno capire che Luca non aveva desistito. Un mese dopo mi consegna la relazione in un ottimo LaTeX (migliore del mio), i file sorgente e i file pdf finali, chiedendomi se, gentilmente, potevo metterli sotto forma di un e-book.

Riferimenti bibliografici

Dodson, C. «Including colour .pdf graphics and hyperlinks».

Pakin, S. «The comprehensive LaTeX symbol list».

Radhakrishnan, C. «pdfscreen.sty - manual».

Rahtz, S. e Oberdiek, H. «Hypertext marks in LaTeX: a manual for hyperref».

1. Luca Ferreri, attualmente iscritto al II anno di Matematica nell'ateneo Torinese

Story, D. (a). «Acrotex bundle-eform supporte».

Story, D. (b). «The acrotex education bundle».

Zezza, P. «Introduzione a metm@t – versione per lo schermo».

▷ Salvatore Palma
Liceo Scientifico A. Volta - Torino
palmasal@hotmail.com

Esperienze didattiche con LaTeX: un corso su edizioni critiche

Jerónimo Leal

Sommario

L'articolo offre alcune esperienze dirette sull'organizzazione di un corso di LaTeX per la stampa di edizioni critiche. Vengono qui trattate le due fasi organizzative del corso. In primo luogo la preparazione: scelta della distribuzione, scelta dell'editor, preparazione delle lezioni, degli esempi ed esercizi, pubblicizzazione e iscrizione; e poi la realizzazione: invio del materiale, installazione, verifica dell'apprendimento ed analisi dei risultati.

1 Introduzione

Le esperienze maturate nell'apprendimento dell'uso di un sistema adatto alla stampa di edizioni critiche (1999–2003) e nell'organizzazione di un corso finalizzato alla diffusione di questo sistema (2003–2007), possono essere utili a chi si accinge a preparare un corso su LaTeX o di fatto lo sta già facendo. Il corso svolto si rivolge a professori, ricercatori e dottorandi che devono affrontare l'editing di documenti molto particolari (le edizioni critiche) che difficilmente possono essere realizzati con un comune processore di testi. L'edizione critica, nei suoi tratti basilari, richiede righe numerate e diversi livelli di apparato, equiparabili alle note a piè di pagina ma con riferimento al numero della riga e non a un numero progressivo di nota. Essa richiede anche la possibilità di inserire simboli speciali, scrivere in greco classico ed altro ancora. Si ritiene che questa esperienza possa essere utile per la diffusione di LaTeX in ambito umanistico e per la preparazione di corsi di introduzione a TeX/LaTeX.

2 Impostazione del corso

Il prototipo dello studente, da quanto si desume dalle iscrizioni dei tre corsi già svolti, è un ricercatore di Università, perlopiù utente di Windows (ma senza escludere altri sistemi), solitamente possessore di un portatile, non troppo esperto nell'installazione di programmi nel suo computer (senza escludere che ne abbia installato qualcuno con l'aiuto di qualche amico più esperto) e sopratutto con problemi di stampa e presentazione della sua edizione critica. Non si esclude che lo studente, avendo già provato qualche programma del tipo WYSIWYG non sia riuscito ad approfondire la logica del sistema in modo da impararne autonomamente l'impiego e quindi necessiti di un corso di questo tipo. Infine, lo studente non possiede particolari padronanze di linguaggi di programmazione e neanche specifiche necessità di elaborazione di scritti con formule matematiche o altre caratteristiche appartenenti all'ambito prettamente scientifico (scienze matematiche, fisiche, chimiche, ecc.).

Da questa breve descrizione si può desumere quanto importante sia prevedere prima dell'inizio del corso ogni possibile difficoltà che lo studente si troverà ad affrontare. Solo in una fase successiva, si potrà preparare il materiale da impiegare: verrà trattata quindi la preparazione (scelta della distribuzione e dell'editor, preparazione delle lezioni, degli esempi ed esercizi, promozione e iscrizione) e la realizzazione del corso (invio del materiale del corso, installazione, controllo dell'apprendimento, risultati). Fin dall'inizio si è deciso di non focalizzarsi mai su un singolo sistema operativo, ma di implementare un corso in grado di soddisfare le esigenze anche di utenti Mac OS X e Linux[1].

3 Distribuzione LaTeX

Prima dell'organizzazione del corso si sono dovute testare diverse distribuzioni (MiKTeX[2], TeXLive[3], teTeX[4], proTeXt[5], fptex[6]) per essere sicuri della compatibilità del materiale. La cosa più importante è che la distribuzione sia la stessa per tutti gli studenti e sia aggiornata alle versioni più recenti dei pacchetti impiegati, ma non meno importante è accertarsi dell'effettivo funzionamento dei pacchetti impiegati in quella distribuzione: era importante poter lavorare con `utf8x`, pdflatex e naturalmente `ledmac` (possibilmente la versione più recente ma sempre testata sia sugli esercizi che sugli esempi preparati). All'inizio sono stati riscontrati dei problemi con il pacchetto `utf8x` in qualche distribuzione ed anche col pacchetto `ledmac` dopo l'ultimo aggiornamento. La scelta più conveniente è quindi ricaduta sulla distribuzione TeXLive che è identica sia per i sistemi Linux che Windows in modo che sia più facile affrontare i problemi indipendentemente dal sistema operativo utilizzato.

1. Fino ad ora abbiamo avuto diversi utenti Mac ma non ancora di Linux.
2. `http://www.miktex.org`
3. `http://www.tug.org/texlive`
4. `http://www.tug.org/tetex`, non più aggiornata da maggio 2006.
5. `http://www.tug.org/protext`
6. `http://www.ctan.org/tex-archive/systems/win32/fptex`

Uno degli obiettivi era quello di rendere particolarmente semplice l'ingresso in LaTeX per gli utenti Windows, senza trascurare gli altri sistemi (Mac OS X e Linux, principalmente), come già detto.

4 Editor

La scelta dell'editor è stata particolarmente lunga ed approfondita. Le nostre richieste in fatto di shell erano molto esigenti: capacità di scrivere con unicode, poiché più di uno studente doveva scrivere in greco classico, e possibilità di tasti programmabili, poiché i comandi di `ledmac` sono lunghi e nessun editor li ha integrati. L'ideale sarebbe stato *Kile*[7], ma la possibilità di installare Linux – oppure cygwin – è stata ritenuta troppo complicata. Perciò si è cominciato, nei primi corsi, con la coppia *jEdit*[8]-*Crimson*[9], per poi arrivare a *Texmaker*[10]. La scelta andava ben oltre le pure preferenze personali: facilità e universalità insieme alla possibilità di impiegare unicode e programmazione di comandi erano dei requisiti fondamentali. *jEdit* richiedeva l'installazione di java e diverse altre estensioni e non disponeva di nessun comando già pronto per l'uso, benché molto accattivante dal punto di vista della universalità e della possibilità di lavorare con unicode. Caratteristiche diametralmente opposte sono possedute da *Crimson* a cui è però preclusa la possibilità di lavorare con unicode.

Nel confronto tra *Winshell*[11] e *Texmaker*, i due più confacenti alle nostre necessità, è stata decisiva la caratteristica di universalità del secondo (Texmaker è disponibile per tutte le piattaforme, incluso Mac OS X). Unica pecca è costituita dal dovere associare manualmente il file `.tex` la prima volta che si apre il programma.

5 Lezioni

Il passaggio compiuto dalle lezioni in aula alle lezioni on-line ha richiesto solo un maggior dettaglio nelle spiegazioni, perciò il lavoro è stato distribuito nelle diverse fasi percorse negli ultimi anni. Ogni lezione è stata impostata con delle slides implementate con il pacchetto `prosper` e successivamente `beamer`, in quanto dovendo far riferimento al codice era necessario impiegare comandi di tipo `verbatim`. La scelta di usare slides come materiale didattico è stata pedagogicamente necessaria poiché in questo modo l'informazione è più diretta e trasmessa in forma essenziale, in questo modo il rischio di smarrimento dello studente è ridotto al minimo. Inoltre, le lezioni sono sempre corredate da una serie di documenti di aiuto in formato `.pdf`, come, per esempio, *Una (mica tanto) breve introduzione a LATEX*[12]; le slides di *Introduzione al LATEX* di Gianluca Gorni[13]; Introduzione a LaTeX per Windows (PGW) di Luigi Carusillo[14]; *Impara LATEX*, di Marc Baudoin[15]. Si allegano anche i files di documentazione dei pacchetti impiegati: `ledmac` (inglese), *psgreek* (inglese), *ucs* (inglese). Molto utile per trovare alcuni simboli necessari è il file *The Comprehensive LaTeX Symbol List*[16]. Come complemento alla fase di approfondimenti si aggiungono i files *TEX made easy*[17], *A Gentle Introduction to TEX*[18] e il codice fonte e istruzioni per l'uso di `EDMAC` con TeX[19].

Il cambiamento da `prosper` a `beamer` come pacchetto per la preparazione di slides è stata una scelta di ordine prettamente pratico, perché lavorare con `pdflatex` era molto più comodo e veloce e `beamer` permette questo modo di compilazione.

Piattaforme di *e-learning* sono state prese in considerazione come possibili alternative per la preparazione del corso, ma trattandosi di un numero abbastanza ridotto di studenti (il corso più numeroso ha avuto una ventina di iscritti) non sarebbe stato giustificato lo sforzo di implementare una piattaforma non facilmente configurabile, e che in ultima analisi avrebbe più confuso che aiutato lo studente alle prime armi. L'elenco delle lezioni è stato inserito in una normale pagina `html` ed ognuna di esse è stata protetta da una password, in modo che lo studente potesse leggere solo una lezione per volta, per regolare bene il ritmo di apprendimento. Per la protezione con la password dei files pdf (creati con `pdflatex`) è stato impiegato *iText*[20], una *Free Java-PDF library*, che permette l'introduzione di una password in un documento pdf esistente.

Il numero di lezioni è stato fissato a dieci di modo che il corso possa essere portato a compimento in meno di tre mesi con una lezione a settimana. Il livello di difficoltà è crescente: si parte da tre lezioni generiche di introduzione a LaTeX e all'utilizzo di Texmaker, per poi passare – nella quarta lezione – all'introduzione a `ledmac`. Le lezioni dalla quinta alla settima servono ad approfondire le caratteristiche di `ledmac` ed insieme per spiegare altri comandi necessari per lavorare con LaTeX. L'ottava lezione è dedicata completamente al greco classico attraverso l'uso dei suoi font e dei comandi:

```
\usepackage{ucs}
\usepackage[utf8x]{inputenc}
```

7. http://kile.sourceforge.net
8. http://www.jedit.org
9. http://www.crimsoneditor.com
10. http://www.xm1math.net/texmaker
11. http://www.winshell.de
12. http://www.tug.org/tex-archive/info/italian/lshort/itlshort.pdf
13. http://www.dimi.uniud.it/gorni
14. http://www.webalice.it/lgcrsll/pgw/latex.html
15. http://www.mat.uniroma1.it/centro-calcolo/manuali/impara_latex.pdf
16. http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf
17. http://tex.loria.fr/general/all.dvi
18. http://tex.loria.fr/general/gentle.dvi
19. ftp://ftp.ucl.ac.uk/pub/users/ucgadkw/edmac
20. http://www.lowagie.com/iText

Complemento necessario per questa fase, ma non limitato ai soli iscritti al corso, è l'installazione di una tastiera greca e dei font unicode, che vengono introdotti nella parte pubblica della pagina web del corso[21]. Nella nona lezione si fa una prospettiva delle alternative a `ledmac`: `ledpar`, `ednotes` e `EDMAC`, quest'ultimo descritto con maggiore dettaglio rispetto ai precedenti. Infine la decime lezione è riservata alla *continuità* del corso: dove reperire le informazioni, aiuto, pubblicazioni ecc. Come conclusione delle lezioni si allega anche un file con un riassunto di tutti i comandi ordinati per lezione, in modo che la revisione dei contenuti sia più semplice.

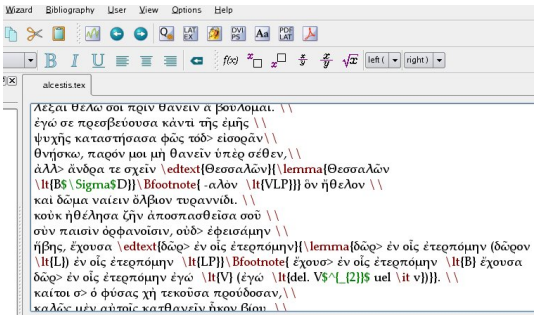


Figura 1: Lavorare con unicode semplifica molto le cose

6 Esempi

L'elaborazione di un numero adatto di file di esempio è stata la chiave (e parallelamente un grandissimo lavoro) per ricombinare insieme i tasselli di un corso in cui tutti gli argomenti sono fortemente interconnessi tra loro. Infatti, il pacchetto `ledmac` richiedeva una distribuzione e un editor adeguato allo scopo, ma l'unico modo per testare appieno le funzionalità era quello di preparare gli esempi e verificare se la sequenza di comandi:

```
\edtext{\abb{}}{\killnumber%
   \Bfootnote{\textbf{VI.} Incipit VI %
   \textit{praem. A}}}
```

funzionava correttamente, valutando quindi se mantenere la distribuzione o sostituirla con un'altra (cosa che avrebbe comportato rivedere le decisioni precedenti). I nomi molto espliciti che abbiamo dato ai nostri files di esempio sono prova della gradualità e delle difficoltà contenute: `modello.tex`, `simplex.tex`, `complex.tex` e `recomplex.tex`.

7 Esercizi

Poiché il corso era incentrato su una forte caratteristica di interattività, ci si è trovati a preparare anche una nutrita serie di esercizi (approssimativamente uno per ogni lezione) in cui lo studente doveva dimostrare di aver imparato le principali caratteristiche di ogni comando. L'arte del commentare e decommentare le righe s'impara di pari passo con i comandi LaTeX negli esercizi più semplici; nei più sofisticati, invece, bisogna scrivere comandi nuovi, ma sempre sulla base dei files di esempio a questo scopo preparati. Il secondo principio pedagogico è stato quello di "non scrivere mai un documento LaTeX da zero". Infatti, gli utenti che iniziano a lavorare vengono dal mondo del *New–Text–Document* in cui il rischio di partire da zero è minimo. Nel nostro caso, invece, l'imitazione di quello che già funziona è il metodo più saggio per risparmiare tempo ed essere più efficienti. Agli umanisti questa regola d'oro sembra invece molto vicina al plagio e pertanto bisogna lottare fortemente contro quest'errato pregiudizio.

8 Promozione

Abbiamo applicato diversi metodi lungo il nostro percorso, ma alla fine è prevalso il criterio secondo cui nel fare troppa pubblicità si può attirare gente senza nemmeno le conoscenze per capire di cosa si tratta. Si è quindi deciso di limitare la propaganda alla sola pagina web del corso che a partire da dicembre 2004 ha registrato più di duemila visite. L'uso di internet come unico mezzo di promozione del corso dovrebbe inoltre costituire un filtro per ricevere dei partecipanti dotati di un minimo di dimestichezza con l'uso di strumenti informatici.

9 Iscrizione

L'iscrizione, come ormai avviene per quasi tutte le attività, è anche in questo caso *online*: attraverso una normale pagina tipo *form* vengono richiesti alcuni dati per poter conoscere il profilo del futuro studente. Successivamente, a titolo di rimborso spese per l'invio del materiale attinente al corso e per contribuire allo sviluppo del corso e alla diffusione del software *open source* è stata prevista una quota di partecipazione dall'importo abbastanza ridotto. Quest'elemento si è dimostrato pedagogicamente molto utile, in quanto a nostro avviso ha contribuito ad evitare iscrizioni dovute al capriccio del momento e addirittura ad assicurare la continuità degli studi partendo dal principio che chi ha investito dei propri soldi è in genere più restio ad abbandonare alle prime difficoltà.

10 Invio del materiale

La fase successiva è l'invio del materiale. Questo consiste in un CD-Rom con la distribuzione di LaTeX e l'installatore dell'editor di testo. In un primo momento si era pensato di inserire anche le lezioni, gli altri documenti di compendio e gli esempi, ma poi si è deciso di evitare poiché sarebbe

21. `http://antiqua.pusc.it/CeTeX`

stata un'ulteriore fonte di problemi per alcuni degli utenti: in questo caso è infatti più rapido scaricare i file da internet piuttosto che rintracciarli nel proprio computer. Così nelle lezioni sono inclusi dei link per il download dei diversi documenti, compressi quelli in formatto zip, man mano c'è bisogno di essi. Per la preparazione del CD-Rom abbiamo impiegato *Kiso*[22], una *gui* scritta per Linux che permette facilmente di aggiungere files ad un'immagine ISO 9660. Con *wxbasic*[23] è stata creata anche una semplicissima *gui* dell'installazione che si auto-avvia quando si inserisce il CD.

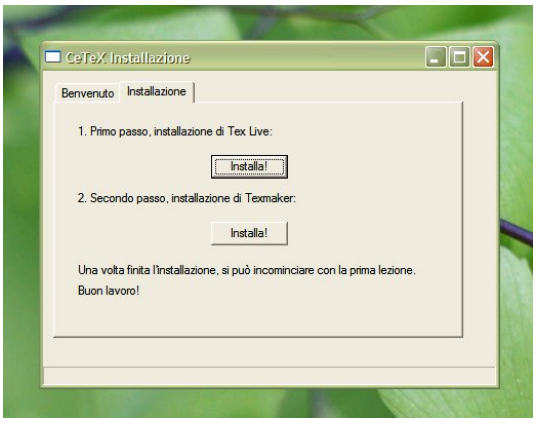


Figura 2: Gui per l'installazione

11 Installazione

Ci siamo inspirati al principio "meglio installare tutto", in modo che l'utente alle prime armi non si trovi costretto ad installare altri pacchetti individualmente, il che per alcuni sarebbe stato probabilmente difficile. Abbiamo quindi preparato un tutorial dell'installazione di TEXLive nella nostra pagina web (oltre ad altri materiali supplementari e informazioni utili per gli studenti del corso e per non iscritti). Tutta la procedura viene comunque guidata dalla *gui* del CD-ROM.

12 Controllo dell'apprendimento

Nelle prime versioni (presenziali) del corso abbiamo seguito tutto il gruppo di studenti contemporaneamente il che ha rallentato moltissimo le lezioni, poiché ognuno ha manifestato un problema diverso. Nella versione on-line il controllo è stato invece effettuato tramite una semplice e-mail, in modo che una volta letta la lezione e risolti gli esercizi lo studente potesse inviare il risultato (generalmente un file pdf) e le eventuali domande al professore. Dopo una verifica del materiale ricevuto e dello stato di apprendimento dello studente il professore avrebbe rispedito la password della lezione successiva. Come anticipato nella pagina web, esiste anche la possibilità di un incontro personale per risolvere gli eventuali dubbi, ma quasi sempre si è riusciti a risolvere tutte le incertezze semplicemente attraverso la posta elettronica. Qualche volta, per la risoluzione di particolari problemi, è anche successo di essere stati obbligati a prendere il controllo del computer dello studente in modalità di amministrazione remota. Per il futuro pensiamo che qualche sistema di *voip* possa essere molto d'aiuto. Comunque, mantenere il contatto con gli antichi studenti è stato un problema di non facile risoluzione.

13 Risultati

Le maggiori difficoltà a cui il neofita va incontro sono costituite dall'installazione, poiché prevede diverse fasi non evidenti per chi è alle prime armi e, in una fase successiva, l'interpretazione dei messaggi di errore del compilatore. Per questo abbiamo facilitato molto (nella misura delle nostre possibilità) l'installazione e abbiamo dedicato un'ampio spazio alla risoluzione degli errori: in ogni lezione c'è sempre stata una sezione *provochiamo il compilatore* nella quale sono stati volutamente provocati degli errori per poter studiare i messaggi restituiti.

Per quanto riguarda gli studenti iscritti, si rileva che alcuni non avevano un vero interesse in LATEX (in quanto già utenti di un altro sistema) e hanno confessato di essersi iscritti un po' per curiosità ed un po' per imparare più cose; altri hanno incontrato immediatamente delle difficoltà e si sono scoraggiati forse troppo presto; altri ancora sono riusciti ad imparare bene ma poi non abbiamo avuto più notizie; infine rimane un gruppo ridotto di utenti che ogni tanto conferiscono con il professore per risolvere eventuali dubbi; e, in più, ci è arrivato qualche dubbio di altri utenti di `ledmac` che non avevano seguito il corso precedentemente. Questi ultimi gruppi stanno forse chiedendo una lista di discussione su `ledmac` in italiano. Mi auguro che il GUIT possa venire incontro a questa necessità.

▷ Jerónimo Leal
Pontificia Università
della Santa Croce
jleal@pusc.it

22. http://kiso.sourceforge.net
23. http://wxbasic.sourceforge.net

Edizione con LaTeX delle opere di Francesco Maurolico

Massimiliano Dominici, Pier Daniele Napolitani

Sommario

Il Progetto Maurolico è nato alcuni anni fa con lo scopo di pubblicare, a stampa e in versione elettronica, l'edizione critica delle opere di Francesco Maurolico (1494–1575). Nell'ambito di questo progetto è stato realizzato un sistema, il *MAURO*-TeX, che a partire da un linguaggio di *mark-up* di tipo LaTeX, consente, attraverso l'uso di appositi strumenti informatici, di ottenere un *output* HTML per la pubblicazione su internet e un *output* in LaTeX standard come formato intermedio verso PDF e POSTSCRIPT.

MAURO-TeX è in fase di profonda trasformazione. Da una parte si sta spostando verso l'XML la parte riguardante la codifica dei testi; dall'altra l'avvio della pubblicazione a stampa delle opere offre l'occasione per una profonda revisione delle macro usate dal formato intermedio LaTeX per generare l'*output* PDF e POSTSCRIPT.

1 Introduzione

Circa otto anni fa cominciava un progetto di edizione delle opere scientifiche di Francesco Maurolico, un importante matematico del Rinascimento (1494–1575). L'edizione, una volta conclusa, consisterà di 10 volumi, per un totale, circa, di 5000 pagine. Essa riunirà circa 250 testi, che vanno da semplici frammenti di appena mezza pagina a un'opera intera divisa in più libri. La produzione dell'autore copre un arco di 60 anni, e tutto il campo della matematica dell'epoca: matematica propriamente detta, ma anche ottica, astronomia, meccanica, gnomonica, ecc. Essa rimaneggia, a volte a partire da alcuni frammenti, nuove edizioni di autori dell'antichità: Euclide, Archimede, Apollonio, Teodosio, Menelao, ecc., che completa con lavori personali di particolare originalità. Scarsamente diffusi durante la vita dell'autore, questi lavori ebbero una sicura influenza sui matematici della fine del XVI secolo e di quello successivo, in Italia, per il tramite del *milieu* matematico gesuita, ma anche nel resto d'Europa.

Il progetto riunisce le competenze di un gruppo di persone, principalmente storici della matematica, che abitano in paesi diversi, lavorano con macchine, strumenti e *software* differenti. Le esigenze del lavoro d'*équipe* hanno spinto all'esplorazione e alla realizzazione di soluzioni informatiche che permettessero a tutti i ricercatori coinvolti di lavorare in maniera uniforme. Malgrado fossero presenti, all'epoca in cui è partito il progetto, alcuni strumenti alternativi per la gestione di un'edizione critica (EDMAC, *tustep*, *cte*), queste soluzioni, o erano sconosciute ai membri del progetto o non rispondevano alle caratteristiche volute.

Poiché il progetto era partito da un Dipartimento di Matematica (quello dell'Università di Pisa), era naturale che si pensasse a LaTeX per la realizzazione di un tale sistema di edizione. Il risultato è *MAURO*-TeX, un linguaggio di *mark-up* in grado di rispondere a tutte le esigenze di un'edizione critica.

In un primo tempo *MAURO*-TeX è stato concepito come un linguaggio capace di padroneggiare le esigenze di codifica di un'edizione critica e, allo stesso tempo, di fornirne una versione *stampabile*, compilando il sorgente con il solo ausilio del file `mauro.sty`. Man mano che il progetto cresceva e si sviluppava, è risultato evidente che ciò non era possibile. Parallelamente a ciò, uno degli obbiettivi del Progetto era quello di costruire un sito internet dove pubblicare (in formato HTML) l'intera edizione: a questo scopo è stato sviluppato un *parser bison*[1] in grado di interpretare la grammatica *MAURO*-TeX e un programma in C++, `m2hv`, in grado di trasformarla in HTML. A questo punto, le funzioni che venivano aggiunte al linguaggio erano sempre più difficili da implementare in termini di macro LaTeX, mentre era relativamente semplice inserirle in un programma in C++ che utilizzasse il suddetto *parser*. Così è stato sviluppato `m2lv`, l'equivalente per LaTeX di `m2hv`. Questo programma trasforma il sorgente codificato in *MAURO*-TeX in un nuovo file codificato in LaTeX standard da cui ottenere l'output finale in PDF o POSTSCRIPT.

Attualmente è stata progettata, ed è in corso di sviluppo, la trasformazione del linguaggio *MAURO*-TeX da un *mark-up* di tipo LaTeX ad uno di tipo XML. L'XML infatti si presta meglio di LaTeX ad esprimere la codifica di un testo e ad essere analizzato da opportuni strumenti. Tuttavia LaTeX rimarrebbe come formato intermedio finalizzato alla produzione di file PDF e POSTSCRIPT.

In quest'ottica, la preparazione, da poco iniziata, dell'edizione a stampa delle opere di Maurolico offre quindi l'occasione, soprattutto in questa pri-

1. Un *parser* è un programma in grado di analizzare un flusso di dati (letto ad esempio da un file o da tastiera) e di interpretarlo in base ad una grammatica formale, per costruirne un albero degli elementi.

ma fase, di individuare quale forma dovrà avere l'*output* LaTeX del sistema *MAURO*-TeX così trasformato (o del suo eventuale successore). Una particolare cura verrà posta nell'assicurarsi che questa forma si mantenga comunque flessibile ed in grado di collaborare con i più importanti pacchetti dedicati alla gestione di edizioni critiche.

2 Il *MAURO*-TeX

2.1 Testo critico e apparati

L'edizione critica deve prendere in considerazione, innanzitutto, le diverse modalità con cui ci pervengono i testi degli autori: sotto forma, a volte, del solo manoscritto, altre volte copiati in alcune centinaia di esemplari, permettendo così la creazione di una *tradizione* testuale. A partire dal XV secolo i testi possono essere stati stampati. L'insieme dei testi in mano all'editore costituiscono i *testimoni* dell'opera.

I testimoni sono generalmente lontani dall'essere identici. L'autore stesso può aver modificato il testo nel tentativo di migliorarlo. Se il testo è stato copiato, il copista può aver introdotto degli errori. L'esperienza mostra che questo tipo di errori è piuttosto frequente e può introdurre (a causa della dimenticanza di parole, del salto di intere righe di testo) delle modifiche che stravolgono il senso del testo. Se sono presenti note marginali, spesso di mano diversa da quella dell'autore, il copista può decidere di integrarle senza darne alcun avviso. Inoltre bisogna tener conto di difficoltà più materiali: le ingiurie del tempo, lacerazioni della carta, fogli andati perduti, l'inchiostro che si cancella, macchie di umidità che rendono illeggibile il testo. Infine, il numero degli errori aumenta in maniera esponenziale con l'aumentare dei *testimoni* disponibili.

L'editore ha il dovere di presentare, a partire da tutti i testimoni, un *testo critico*, ovvero un testo che rappresenti i diversi testimoni e che egli ritiene dia conto della volontà dell'autore. Il testo critico è accompagnato da un *apparato critico* che deve dare al lettore la possibilità di ricostruire a) il percorso attraverso il quale l'editore è giunto al testo critico, a partire dai testimoni; b) il contenuto di ogni singolo testimone. L'editore deve segnalare in apparato tutte le *varianti* tra i testimoni: parole e frasi differenti, diverse disposizioni del testo, note marginali, aggiunte in interlinea, in margine, lacune, ecc. Inoltre deve segnalare i propri interventi sul testo: le parole che ha dovuto sopprimere o aggiungere o correggere per arrivare a dare al testo critico un senso compiuto. Infine, deve segnalare i cambiamenti di scrittura e d'inchiostro che indicano l'intervento di diverse mani (o anche della stessa in tempi diversi).

Nel quadro di un'edizione che presenti una tradizione molto ricca, l'editore deve anche provvedere a individuare i legami di parentela e filiazione dei vari testimoni, il cosiddetto *stemma codicum*. Questo lavoro può essere parte integrante dell'edizione stessa, in quanto, una volta individuata, per un testimone, la sicura derivazione da un altro, l'editore può decidere di non tenerne più conto ai fini dell'edizione, con una evidente semplificazione del lavoro.

Le edizioni scientifiche presentano delle particolarità proprie rispetto alle edizioni classiche. I testi scientifici presentano una forte strutturazione di cui l'editore deve rendere conto. I testi antichi rimandano a numerosi riferimenti interni ed esterni. Le centinaia di riferimenti agli *Elementi* di Euclide (la Bibbia dei matematici e il loro riferimento principale nel corso di molti secoli), sono costantemente citati, esclusivamente per mezzo di un numero, in tutti i lavori matematici, fino all'avvento dei matematici moderni. Queste citazioni sono estremamente importanti per lo storico della matematica e devono essere inventariate, classificate e rese per esteso. Inoltre, la quantità di figure e di numeri riveste particolare importanza nei testi scientifici, ed è fonte di numerosi errori dei copisti anch'essa. Infine una difficoltà supplementare è data dalla notazione matematica stessa, così differente da quella dei matematici moderni.

2.2 Il linguaggio

Il *MAURO*-TeX è un linguaggio di *mark-up* che permette di costruire, passo per passo, un'edizione critica. L'utente trascrive il testo dei vari testimoni, riunendoli in un unico file LaTeX. Il linguaggio permette di trattare tutti i casi che si presentano nel corso di un'edizione che comprenda uno o più testimoni. Per ogni occasione che dia luogo ad una o più varianti nei testimoni, egli segnala consecutivamente il loro contenuto nei diversi campi di una apposita macro (`\VV{}`). Qualche esempio basterà a dare un'idea del meccanismo. Si consideri il caso di tre testimoni che presentino le rispettive lezioni:

- A: Sit data gratia, sit datus cubus...
- B: Sit data latio, sit datus cubus...
- C: Sit data ratio, sit datus cubus...

La trascrizione si effettua nella seguente maniera:

```
Sit data \VV{{A :gratia}{B :latio}
{C :ratio}}, sit datus cubus ...
```

E l'output è il seguente:

Sit data gratia[a], sit datus cubus

a. gratia *A* latio *B* ratio *C*

Per definizione, il primo campo della macro `\VV{}` contiene il testo critico; gli altri campi sono destinati a riempire l'apparato. Una semplice

permutazione dei campi basta a modificare il testo critico e, di conseguenza, l'apparato. Per trattare le differenti situazioni che si incontrano nel corso di un'edizione critica, l'utente ha a sua disposizione un certo numero di scorciatoie finalizzate a riempire l'apparato critico con abbreviazioni e argomenti standard della filologia. Supponiamo che nel testimone *C* la parola *gratia*, cancellata con un tratto di penna, segua la parola *ratio*. Il trascrittore utilizza la macro predefinita `\POSTDEL` e scrive:

```
Sit data \VV{{C:\POSTDEL{gratia}:ratio}
{A:gratia}{B:latio}}, sit datus cubus ...
```

E ottiene:

> Sit data ratio[a], sit datus cubus

a. post ratio *del.* gratia *C* gratia *A* latio *B*

Se la parola *ratio* appare come aggiunta in interlinea:

```
Sit data \VV{{C:\INTERL:ratio}
{A:gratia}{B:latio}}, sit datus cubus ...
```

ottenendo

> Sit data ratio[a], sit datus cubus

a. ratio *in interl. C* gratia *A* latio *B*

Naturalmente è possibile riempire l'apparato critico con tutto ciò che sembra più opportuno all'editore, soprattutto nei casi non espressamente previsti dal linguaggio. Ad esempio, se si desidera segnalare che la parola *ratio* è scritta con inchiostro rosso, si può ricorrere alla macro `\DES{}`, il cui argomento è libero.:

```
Sit data
\VV{{C:\DES{rubro atramento}:ratio}
{A:gratia}{B:latio}}, sit datus cubus ...
```

E si avrà

> Sit data ratio[a], sit datus cubus

a. ratio *rubro atramento C* gratia *A* latio *B*

Per tutti i casi ricorrenti è stata prevista una scorciatoia del tipo di quelle viste sopra: omissioni, lacune, parole illeggibili, integrazioni in margine e in interlinea, correzioni del copista, congetture e correzioni dell'editore, ecc. È stata prevista una codifica dei nomi dei testimoni in modo da poter descrivere le diverse mani che hanno lavorato sul manoscritto. Le varianti cosiddette "banali" (errori di ortografia e refusi) sono codifcate tramite la macro `\VB{}`, che ha la stessa sintassi di `\VV{}`. L'uso vuole che queste varianti vengano segnalate nella codifica del testo ma non riportate nell'apparato, in modo da non appesantirlo inutilmente.

Il *mark-up* sembra sufficientemente completo per poter realizzare una vera e propria classificazione delle varianti utili all'editore e allo storico, e altri elementi di codifica permettono di descrivere formalmente i testimoni. Il linguaggio fornisce inoltre una serie di macro più elementari per il trattamento dei cambi di foglio, delle date, dei titoli, di citazioni e riferimenti, degli enunciati delle proposizioni, ecc.

3 Il lavoro sull'edizione a stampa

3.1 Specifiche e linee guida

Abbiamo già fatto notare che il lavoro di revisione delle macro utilizzate dall'*output* LaTeX di *MAURO*-TeX, ha un duplice obbiettivo: quello immediato di avere una edizione a stampa confacente ai criteri espressi dall'editore, e quella, sperimentale, di determinare la forma futura del suddetto *output*, una volta che *MAURO*-TeX (o ciò che prenderà il suo posto) verrà trasformato in linguaggio di *mark-up* puro e semplice.

Finora il lavoro si è limitato alla parte concernente l'apparato critico vero e proprio, che rappresenta comunque il nucleo centrale del *MAURO*-TeX, e, anche limitatamente a ciò, non è ancora concluso.

Piuttosto che implementare da zero un ennesimo pacchetto per edizioni critiche, cosa che non è tra gli scopi di questo lavoro, si è pensato di utilizzare un pacchetto già esistente e la scelta è caduta su ledmac (Wilson (2005)), che può vantare, rispetto ai concorrenti, ednotes (Lück (2006)) e bigfoot (Kastrup (2006)), un livello di confidenza maggiore (ledmac è l'erede di EDMAC, che viene usato ormai da una decina di anni nella produzione di edizioni critiche) e una documentazione più estesa e organica.

L'impostazione di fondo, però, che si è tenuta presente nell'elaborare le macro, è stata quella di mantenere un livello di astrazione maggiore possibile. L'idea sarebbe quella di avere, a conclusione del lavoro, un *front-end* in grado di lavorare all'occorrenza con diversi *back-end*. In parole povere l'eventuale classe maurolico, risultato finale di questo esperimento, dovrebbe essere in grado di lavorare, internamente, con diversi pacchetti dedicati alle edizioni critiche, quali ledmac, ednotes o bigfoot, lasciando all'utente la possibilità di selezionare, tramite una semplice opzione della classe, quale di questi pacchetti usare.

Per tutti i riferimenti al funzionamento specifico di ledmac si rimanda ai testi citati in bibliografia.

3.2 Installazione dei font

La prima specifica indicata dall'editore è stata la scelta dei caratteri da utilizzare per la stampa. Questi caratteri, in maggioranza assenti, poiché commerciali, dalle distribuzioni TeX, hanno richiesto un processo di installazione non banale, in

quanto doveva rispondere a particolari esigenze di armonizzazione e di uso.

L'insieme dei caratteri da utilizzare per l'edizione a stampa delle opere di Franceso Maurolico, è costituito dal *Monotype Dante*, per il testo in tondo e in corsivo; dal *Gill Sans*, per le parti evidenziate in obliquo; dal *Math Garamond*, unico carattere presente in alcune distribuzioni TEX, per i simboli matematici; e dall'*Odyssea* per il testo in greco antico (politonico).

Considerata la complessità dell'installazione, si è ritenuto utile fare uso di un programma come fontinst (Jeffrey *et al.* (2004)), che garantisce la possibilità di controllare il processo nei dettagli e di effettuare in piena sicurezza operazioni di trasformazione dei caratteri stessi – quale ad esempio la scalatura del *Gill Sans*, necessaria per armonizzarne le proporzioni rispetto a quelle del *Dante*.

fontinst permette di definire all'interno di un file (nel nostro caso `maurofonts-inst.tex`) una serie di istruzioni che regolano la costruzione dei file ausiliari (`.vf` e `.tfm`) necessari a TEX per poter utilizzare il font in questione[2].

Nel seguito daremo solo qualche cenno sui dettagli dell'installazione, in quei casi in cui si sono riscontrate particolari difficoltà, rimandando, per tutto ciò che è ordinaria amministrazione, alla bibliografia sull'argomento (Jeffrey *et al.* (2004); Lehman (2004)).

Corpo del testo

Per il corpo del testo, l'editore ha scelto il *Monotype Dante*, un carattere ideato e disegnato dall'italiano Giovanni Maiderstag sulla base dei caratteri del primo umanesimo incisi da Francesco Griffo per Aldo Manuzio.

Il carattere – disponibile in formato Postscript Type1, nei pesi *regular*, *medium* e *bold*, e corredato di *expert font set*, cioè di un insieme separato di font contenente legature, numeri minuscoli[3], numeri e lettere esponenti, ecc. – non ha comportato particolari problemi. fontinst è stato creato proprio avendo in mente l'installazione di un font Type1 con codifica Adobe Standard, e questo è il caso, per l'appunto, del *Dante*. Così, è sufficiente l'attenta lettura di un ottimo *tutorial* come *The Font Installation Guide* (Lehman (2004)) per ottenere una installazione perfettamente funzionante, al termine della quale l'utente ha a disposizione alcune famiglie di font che può sfruttare per determinare l'aspetto tipografico del testo.

Ad esempio, le seguenti linee di codice, inserite nel file `maurofonts.sty`, permettono di usare il *Dante* con i numeri "normali" (caricando il pacchetto con l'opzione lining: `\usepackage[lining]{maurofonts}`), o con i numeri "saltellanti" (opzione oldstyle: `\usepackage[oldstyle]{maurofonts}`).

```
\RequirePackage{nfssext}
\DeclareOption{lining}{%
  \renewcommand*{\rmdefault}{mdtx}}
\DeclareOption{oldstyle}{%
  \renewcommand*{\rmdefault}{mdtj}}
```

L'uso del *Gill Sans* è limitato all'apparato delle note, dove viene utilizzato per far risaltare i testimoni (indicati di norma con una lettera maiuscola) o alcune notazioni particolari (ad esempio l'indicazione che una particolare parola o frase è stata scritta con un inchiostro diverso da quello utilizzato nel resto del manoscritto: *diverso atramento*).

Anche in questo caso l'installazione è stata lineare e non ha comportato particolari problemi. Il carattere è stato scalato al 90% delle sue dimensioni effettive, per renderlo più armonico al disegno del *Dante*.

Simboli matematici

Per quanto le opere di Francesco Maurolico trattino di argomenti matematici, la notazione simbolica che vi si trova utilizzata è abbastanza semplice e non ha niente a che vedere, in quanto a complessità, con quella che si sarebbe sviluppata nei secoli successivi.

Tuttavia, si è ritenuto comunque opportuno fornire il sistema di caratteri da utilizzare per l'edizione, di un insieme di simboli matematici che coprisse non solo le esigenze più elementari ma fosse adatto anche ad esprimere una notazione più avanzata.

È stato quindi necessario individuare un font matematico che si armonizzasse abbastanza bene con il carattere utilizzato per il corpo del testo, e cioè il *Dante*. Considerando che le caratteristiche morfologiche del *Dante* non sono così dissimili da quelle di un altro carattere umanistico come il *Garamond*, la decisione di utilizzare il font *Math Design URW Garamond*, contenuto nel pacchetto mathdesign (Pichaureau (2004)) di Paul Pichaureau e progettato per essere usato insieme all'*URW Garamond*, si è imposta come la più semplice.

A differenza di quanto avviene per il testo, dove LATEX utilizza la codifica T1 a 8 bit, per quanto riguarda la notazione matematica vengono ancora utilizzate delle codifiche a 7 bit (si veda a questo proposito Vieth e Hoekwater (1999)). In pratica LATEX utilizza quattro famiglie di font, con diversa codifica:

2. Questo non è esattamente vero. fontinst genera file con estensione .pl e .vpl, i cosidetti *Property List* e *Virtual Property List*, versioni ascii dei *TeX Font Metrics* e *Virtual Fonts* (con estensioni .tfm e .vf) che TEX richiede per poter utilizzare un font. Questi ultimi si ottengono compilando i primi con `pltotf` e `vptovf`.

3. Numeri "minuscoli", o "saltellanti", o, all'inglese "*old style*", sono quei numeri che, come le lettere minuscole, hanno dimensioni variabili in altezza, caratteristica che li distingue dai numeri "maiuscoli" o "normali", più comunemente usati, che hanno invece tutti la stessa altezza.

- OT1 (la vecchia codifica per il testo) per estrarre le lettere (generalmente in tondo) usate per comporre gli operatori matematici (come in "sin" o in "log");
- OML per estrarre le lettere (generalmente in corsivo) usate per le variabili (come in "f" o in "ω");
- OMS per estrarre i simboli;
- OMX per estrarre simboli aggiuntivi.

In tutti questi casi abbiamo fatto in modo che venissero utilizzati singoli caratteri presi dal *Dante*, ogni volta che fossero presenti, poi da *Odyssea* e infine dal *Math Design URW Garamond*. Questo è possibile richiamando delle apposite procedure nel file `maurofonts-inst.sty` (vedi la sezione 3.2). Per comodità riportiamo qui sotto un esempio che si riferisce unicamente alla codifica OML:

```
\installfont{mdtri7m}{%
  mdtri8r,odysi,mf-gmri7m,mathit}{%
   oml}{OML}{mdt}{m}{it}{}
```

Le istruzioni riportate indicano in sostanza a TEX di costruire il font virtuale[4] `mdtri7m` nella codifica OML prendendo dal *Dante* (`mdtri8r`) tutti i caratteri che può, utilizzando poi il font *Odyssea* (`odysi`) per le lettere greche e riempiendo infine quanto manca con *Math Design URW Garamond* (che qui è stato rinominato come `mf-gmri7m`).

Greco

L'installazione del font greco *Odyssea*, si è rivelata la parte più ardua di questa prima fase. In questo caso si sommavano due singolarità: una al livello della codifica interna al font, e l'altra al livello del metodo di *input* usato da MAURO-TEX.

Per quanto riguarda il primo di questi aspetti, il font *Odyssea*, un prodotto commerciale della *Linguist's Software*, usa una codifica interna poco diffusa, e utilizzata quasi esclusivamente dalla suddetta casa. Questo implica che è difficile trovare un supporto "prefabbricato" a questa codifica per fontinst. Anche la documentazione in materia è scarsa, e di pochissimo aiuto è il file AFM che si può ottenere tramite `ttf2afm`. L'unico aiuto reperibile in rete si trova sul sito della *SIL International* (`www.sil.org`), da cui è possibile scaricare il file `LingSoftmaps.zip` che contiene la codifica del font *GraecaII*, anch'esso distribuito da *Linguist's Software* e che presenta sostanzialmente la stessa codifica di *Odyssea*.

4. Un font virtuale (o *virtual font*) è un file binario che viene usato dal driver DVI (o da `pdfetex`, se l'*output* generato è in PDF) per tracciare effettivamente i caratteri. Tramite un font virtuale è possibile compiere operazioni complesse come la rimappatura di un font, la combinazione di caratteri provenienti da font diversi, o l'applicazione di effetti speciali (scalatura, inclinazione, spaziatura, ecc.).

Allo stesso modo, il metodo di *input* dei caratteri greci utilizzato da MAURO-TEX è l'*Ibycus*, una variante di *Beta Code*. Quest'ultimo è effettivamente lo standard adottato da una serie di organizzazioni e siti internet (*Thesaurus Linguae Grecae*, *Perseus*, ecc.), ma non lo è affatto per TEX, dove viene comunemente adottata la codifica LGR. Da ciò segue che la sequenza di caratteri da inserire per ottenere la medesima lettera accentata è diversa nei due casi; e quindi, in generale, nel predisporre l'occorrente per generare i *virtual fonts* che dovranno essere utilizzati da LATEX, bisognerà costruire in maniera diversa le legature. Ad esempio, mentre, utilizzando il sistema standard, per ottenere "ὦ" bisognerà digitare la sequenza di caratteri: "`~ > w`", lo stesso risultato potrà essere ottenuto in *Ibycus* digitando la sequenza "`w ( =`". Anche in questo caso non esistono soluzioni "prefabbricate", ed è stato quindi necessario scrivere da zero i file che contengono le istruzioni da passare a fontinst per costruire le opportune legature.

3.3 Definizione delle caratteristiche della pagina

Raccolte le indicazioni dell'editore sulle specifiche del layout si è provveduto a trasformarle in parametri per LATEX. Le specifiche erano le seguenti:

- dimensioni della pagina: 17cm × 24cm;
- gabbia del testo: 13cm × 20cm, incluso lo spazio per i titoli correnti;
- corpi dei caratteri: normale 11/13pt, infratesto 10/11pt, note 9/10pt, titoli 12/24pt
- titoli correnti in maiuscoletto spaziato;
- titoli in maiuscolo spaziato.

Per quanto riguarda i parametri della pagina, è stato utilizzato il pacchetto geometry (UMEKI (2002)):

```
\RequirePackage{geometry}
\geometry{
    includehead,
    includemp,
    marginparwidth=1.1cm,
    paperwidth=17cm,
    paperheight=24cm,
    width=13cm,
    height=20cm,
    headsep=.2cm
}
```

Poi sono stati ridefiniti i corpi dei caratteri (riportiamo, per brevità, solo la ridefinizione di `\normalsize`):

```
\renewcommand\normalsize{%
   \@setfontsize\normalsize\@xipt{13}%
```

```
\abovedisplayskip 11\p@ \@plus3\p@%
   \@minus6\p@
\abovedisplayshortskip \z@ \@plus3\p@
\belowdisplayshortskip 6.5\p@%
   \@plus3.5\p@ \@minus3\p@
\belowdisplayskip \abovedisplayskip
\let\@listi\@listI}
```

E, per l'impostazione delle testatine e piè di pagina, è stato utilizzato il pacchetto fancyhdr (VAN OOSTRUM (2004)):

```
\RequirePackage{fancyhdr}
\fancyhead{}
\fancyfoot{}
\fancyhead[CE]{
    \capsselect{maiuscoletto}
    \caps{\leftmark}}
\fancyhead[CO]{
    \capsselect{maiuscoletto}
    \caps{\rightmark}}
\fancyhead[RO,LE]{\thepage}
```

Infine, la spaziatura del maiuscolo e del maiuscoletto è stata ottenuta facendo ricorso ai comandi forniti dal pacchetto soul (FRANZ (2003)):

```
\RequirePackage{soul}
\capsdef{/mdtx///}{%
  \scshape}{.08em}{.35em}{.5em}
\capssave{maiuscolo}
\capsdef{/mdtx///}{%
  \scshape}{.062em}{.35em}{.5em}
\capssave{maiuscoletto}
```

Tuttavia è in corso di studio la possibiltà di abbandonare questa soluzione e sostituirla con la creazione di un font virtuale con le opportune spaziature.

3.4 Apparato

L'apparato critico dell'edizione delle opere di Francesco Maurolico prevede tre livelli di note. Il primo si riferisce alle varianti del testo, il secondo ricostruisce l'apparato delle citazioni, il terzo è riservato agli interventi dell'editore. Quest'ultimo livello non comporta particolari problemi di conversione, essendo l'equivalente di una normale nota a piè di pagina. Il preprocessore, quindi, si limiterà ad esportare letteralmente la macro `\Adnotatio{<arg>}` e sarà poi il particolare *back-end* scelto dall'utente a definire questa macro in maniera significativa. Ad esempio, nel nostro caso, usando ledmac come *back-end*, avremo la seguente definizione:

```
\let\Adnotatio\footnoteA
```

L'apparato testuale e quello delle fonti presentano invece una maggiore complessità. Per il momento solo il primo di questi apparati è stato trattato in maniera sufficientemente approfondita. Esaminiamolo nei dettagli.

Nel file sorgente MAURO-TEX una variante testuale è marcata in questa maniera:

```
sit rectus: \VV{
           {*:\ED{conieci}:cum}
           {A:\NL}
           }
```

Come descritto nel paragrafo 2.2, la macro `\VV` può contenere un numero indefinito di campi delimitati da parentesi graffe: ognuno di questi rappresenta una variante testuale. I sottocampi (delimitati da `:`) in cui ogni variante è organizzata sono tre, di cui uno, il secondo, opzionale, e rappresentano rispettivamente: la fonte manoscritta (`*` sta a significare la lezione del testo critico), eventuali aggiunte dell'editore, e la variante vera e propria. La lezione da inserire nel testo è quella indicata nella prima variante (nel nostro esempio *cum*). La macro `\ED`, contenuta nel secondo campo indica un commento dell'editore e ha come risultato quello di stampare il proprio contenuto opportunamente formattato (nel nostro caso `conieci` verrà stampato in *Gill Sans* obliquo.

Possiamo quindi immaginare di trasformare il codice precedente nel seguente, che ha un aspetto più "LATEX friendly":

```
sit rectus: \testocritico{cum}{%
  \variante[\ED{conieci}]{*}{cum}
  \variante{A}{\NL}
   }
```

Resta da dare un significato pratico alle macro `\testocritico`[5] e `\variante`, in riferimento al nostro *back-end* basato su ledmac.

Nel caso di `\testocritico`, l'operazione è lineare:

```
\newcommand\testocritico[3][]{%
  \edtext{#2}{#1\Afootnote{#3\unskip}}}
```

Cioè, la nostra macro è l'equivalente della macro `\edtext` di ledmac, utilizzata per generare l'apparato critico, dove il secondo argomento viene direttamente passato ad uno degli apparati in nota, per l'esattezza `\Afootnote`[6]. Rispetto al comportamento di default di `\Afootnote`, questo verrà modificato in modo che non venga riportato in nota il lemma (*cum*), ma solo il numero di riga:

```
\renewcommand{\Afootfmt}[3]{%
  \ledsetnormalparstuff
  {\notenumfont\printlines#1|}%
   \strut\hspace{3pt}%
%   {\select@lemmafont#1|#2\rbracket}%
```

5. La scelta di utilizzare nomi italiani per le istruzioni è dovuta al fatto che dovendo la futura classe maurolico cooperare con i pacchetti ledmac e ednotes, era oggettivamente difficile trovare nomi inglesi che rispecchiassero il significato dell'istruzione e non fossero già usati da uno dei due suddetti pacchetti. Tuttavia, essendo ancora il progetto nella sua fase iniziale e di sperimentazione, questa scelta potrebbe essere rivista.

6. L'argomento opzionale serve ad inserire, manualmente, eventuali informazioni da passare a ledmac relative al lemma e ai numeri di riga

```
%     \enskip#3\strut\par}
  #3\strut\par}
```

`\variante`, invece comporta maggiori complicazioni. Ci si aspetterebbe che fosse sufficiente far stampare alla macro i propri argomenti nella sequenza appropriata: lemma, eventuali commenti provenienti dall'argomento opzionale, indicazione della fonte. Questo non è possibile, data la possibilità di avere, come argomento opzionale, macro del tipo `\POSTDEL{<arg>}` che devono stampare il seguente testo: "*post* `<lemma>` *del* `<arg>`". Dove `<lemma>` è, ovviamente, il secondo argomento obbligatorio della macro `\variante`.

La soluzione sta nel salvare il lemma in una macro interna e interpretare l'argomento opzionale come una "funzione" da applicare a quest'ultima:

```
\newcommand\variante[3][\pm@lemma]{%
  \def\pm@lemma{#3}#1\pm@lemma%
  \ifx*#2\pm@lemmasep\else\ \pm@fonte{#2}\fi%
  \variantsep}
\newcommand\MARG{%
  \pm@lemma\ \pm@varfont{in marg.}%
  \let\pm@lemma\relax}
\newcommand\POSTDEL[1]{%
  \def\pm@lemmasep{}\pm@varfont{post}%
  \ \pm@lemma\ \pm@varfont{del} #1%
  \let\pm@lemma\relax}
\newcommand\ANTEDEL[1]{%
  \def\pm@lemmasep{}\pm@varfont{ante}%
  \ \pm@lemma\ \pm@varfont{del} #1%
  \let\pm@lemma\relax}
\newcommand\INTERL{%
  \pm@lemma\ \pm@varfont{supra lineam}%
  \let\pm@lemma\relax}
```

Il codice riportato qui sopra, mostra appunto, oltre alla definizione di `\variante`, alcune di queste "funzioni".

Un'idea visiva, molto sommaria, dei risultati finora ottenuti può essere fornita dalla figura 1.

4 Conclusioni

Riteniamo che quanto abbiamo qui esposto possa avere un interesse che va al di là dell'edizione dell'opera di Maurolico. L'esperienza del *MAURO*-TEX è sicuramente, per molti motivi, limitata alle peculiarità di questa edizione. Tuttavia da essa sono nati almeno due filoni di sviluppo:

1. la creazione di un linguaggio XML capace di supportare un'interfaccia "amichevole" e, soprattutto, capace di adattarsi alle disparate esigenze dell'editore critico in modo flessibile;
2. la preparazione di una serie di strumenti per l'*output* di un'edizione capaci di adattarsi alle disparatissime esigenze non solo dell'editore critico, ma anche della tipografia e della creazione di siti web.

Le esperienze che stiamo attualmente conducendo su entrambi questi fronti ci sembrano confermare le scelte fondamentali del *MAURO*-TEX: individuare correttamente l'informazione rilevante e distinguerla, attraverso un opportuno sistema di *mark-up* dal risultato grafico che si vuole ottenere.

Riferimenti bibliografici

Franz, M. (2003). «The soul package». Disponibile su CTAN, `macros/latex/contrib/soul`.

Jeffrey, A., McDonnell, R. e Hellström, L. (2004). «fontinst: Font installation software for TEX». Disponibile su CTAN, `fonts/utilities/fontinst/doc/manual`.

Kastrup, D. (2006). «The bigfoot». Disponibile su CTAN, `macros/latex/contrib/bigfoot`.

Lehman, P. (2004). «The font installation guide». Disponibile su CTAN, `info/Type1fonts/fontinstallationguide`.

Lück, U. (2006). «ednotes: for critical editions with LaTeX». Disponibile su CTAN, `macros/latex/contrib/ednotes`.

Pichaureau, P. (2004). «The mathdesign package». Disponibile su CTAN, `fonts/mathdesign`.

Umeki, H. (2002). «The geometry package». Disponibile su CTAN, `macros/latex/contrib/geometry`.

van Oostrum, P. (2004). «Page layout in LaTeX». Disponibile su CTAN, `macros/latex/contrib/fancyhdr`.

Vieth, U. e Hoekwater, T. (1999). «Surviving the tex font encoding mess». Disponibile su CTAN, `fonts/utilities/fontinst/doc/talks`.

Wilson, P. (2005). «ledmac: A presumptuous attempt to port edmac, tabmac, edstanza, to LaTeX». Disponibile su CTAN, `macros/latex/contrib/ledmac`.

▷ Massimiliano Dominici
`mlgdominici@interfree.it`

▷ Pier Daniele Napolitani
Dipartimento di Matematica
Università di Pisa
`napolita@mail.dm.unipi.it`

| [A:20r]

DIALECTICA MAUROLYCI

De terminis

28 Sonus est aër ictus et auditus obiectum. Sonorum alius est strepitus seu fragor ex collisione corporum factus, alius vero vox animalis. Vocum alia non significativa, ut temere prolata, bu, baf, alia significativa. | **29** Adhuc terminorum mentalium non est terminus, significat, dato quod esset: ut lapis, ens; alius 2ae intentionis, qui terminum significat: ut nomen, propositio. [A:20v]

30 Alius secundae impositionis, qui terminum significat, ut nomen, oratio, in voce vel scriptura. **31** Unde manifestum est, quod in ordine primus terminus est mentalis, secundus vocalis, qui mentalem, tertius scriptus, qui vocalem declarat. Imago tamen lineata vel sculpta omnibus est una quae repraesentat imaginem mentalem.

De trascendentibus

32 Ex qua distinctione patet numerus et vocabula transcendentium terminorum. Est igitur transcen|dens terminus maxime universalis quidquid existit aut concipitur significans. **33** Malum falsum hinc excluduntur, quandoquidem non essentiam secundum defectum dicunt. [A:21r]

De insolubilibus

34 Sub factiva ordinantur omnes illae artes, quas Graeci, βαναυσικάς. Foramen illud σπάρτον a Graecis dicitur. Vectem esse instrumentum, sive ligneum, sive ferreum, quo pondera moventur et elevantur. **35** Superponitur enim vectis sustentaculo, sive fulcro solido, quod vocatur ὑπομόχλιον, ita ut cuspis vectis subiecta sit saxo, sive oneri movendo, longius autem manubrium sit ad manum moventis, vel e contrario cuspis alicui stramento valido innitatur, ut per longius manubrium moles aliqua moveri et elevari possit σφαίρωμα esse appendiculum staterae. Citatus autem a magistro coram judice, respondit: si vincam, non solvam ex sententia, si vinces non solvam ex pacto.

36 Contra vero Corax sic discipulum redarguebat: immo, si vincam, solves ex sententia, si vinces, solves ex pacto. Quorum argutiam iudex admiratus dixisse fertur κακος κορακος κακον Ον.

In Arce Catanensi, 17 maii 1554

7 mentalium *signo posito in marg. A* 7 ut lapis, ens; *in marg. A* 9 *ante* impositionis *del* Intentionis *A* 10–13 Unde ∼ mentalem *in marg. A* 13 imaginem *supplevi* 17–18 Malum ∼ dicunt *in marg. A* 20 βαναυσικάς S^1 *spatio relicto om. S* 21 σπάρτον *coniecimus* σπάριον S^1 *spatio relicto om. S* 23 ὑπομόχλιον S^1 *spatio relicto om. S* 26 σφαίρωμα S^1 *spatio relicto om. S* 31 κακος κορακος κακον Ον *correxi* κακος κορακος κακον Ον *A*

1

Figura 1: Una pagina dell'edizione a stampa delle opere di Francesco Maurolico

L'edizione critica di un'opera matematica: *MAURO*-TEX e MetaPost

Roberta Tucci

Sommario

Questo articolo è costituito da tre parti: una prima parte in cui vengono esposti alcuni problemi filologici che si presentano quando ci si appresta a fare un'edizione critica di un'opera matematica; una seconda parte di tipo descrittivo, in cui vengono presentati gli strumenti informatici (*MAURO*-TEX e MetaPost) che sono stati scelti per fare l'edizione; infine una terza parte in cui vengono esposti sinteticamente i risultati ottenuti dall'utilizzo di tali strumenti.

1 Introduzione

In questo articolo presenteremo un'esperienza dell'utilizzo di LATEX per la cura della trascrizione, della collazione e dell'edizione critica delle opere matematiche di Francesco Maurolico[1]. In particolare, presenteremo il pacchetto *MAURO*-TEX, un sistema LATEX ideato e sviluppato all'interno del "Progetto Maurolico"[2] per la trascrizione e l'edizione critica di testi scientifici. Suggeriamo che l'uso di *MAURO*-TEX e del programma MetaPost per quanto riguarda la gestione critica delle figure dell'opera originale costituisce un'ottima soluzione al problema della cura di un'edizione critica di un testo scientifico.

1. Francesco Maurolico (Messina 1494 - Messina 1575) rappresenta esemplarmente la figura dell'umanista cinquecentesco: impegnato nella ricostruzione del sapere antico, si occupò praticamente di tutto lo scibile, spaziando dalla matematica alla letteratura, dalla teologia alla musica, dalla meccanica alla filosofia. Ideò e descrisse strumenti ottici e astronomici, disegnò carte geografiche e diresse numerose opere edilizie nella sua città natale Messina. Nel campo delle scienze matematiche, Maurolico si interessò di geometria solida e piana, di aritmetica, di centrobarica, di ottica e di astronomia, ottenendo in ciascun settore pregevoli risultati. Studioso dotato di una notevole creatività matematica, interpretò in maniera originale ed innovativa argomenti noti e formalizzò nuove teorie, fra cui ricordiamo, per non fare che qualche esempio, la teoria del momento meccanico sviluppata all'interno del *De momentis aequalibus*, la legge di rifrazione nei *Diaphana* e la teoria delle coniche. La varietà degli argomenti trattati e l'importanza dei risultati conseguiti lo pongono di diritto tra le menti matematiche più produttive e brillanti del suo secolo.

2. A metà degli anni Novanta a seguito di un workshop tenutosi presso il Dipartimento di Matematica dell'Università di Pisa intitolato "All'alba della matematica moderna. Francesco Maurolico e il ritorno dei classici", la comunità scientifica si è resa conto della necessità di un'edizione completa degli scritti di Maurolico. Si è costituito così un gruppo di ricerca, formato da studiosi di vari paesi con lo scopo di produrre una tale edizione. Per maggiori informazioni si consulti in sito internet: `www.maurolico.unipi.it/`.

L'articolo non vuole essere una guida a *MAURO*-TEX, né tanto meno a MetaPost, bensì è la presentazione di un'esperienza concreta di uso di questi programmi per l'edizione critica di un testo storico-scientifico. L'articolo prende vita dal nostro lavoro di editori, iniziato con l'edizione critica del *De momentis aequalibus* — opera di Francesco Maurolico che tratta questioni di meccanica — e proseguito con la cura dell'edizione di altre opere mauroliciane e di Giorgio Valla[3].

2 Aspetti filologici di un'edizione critica

Un testo antico di matematica è pur sempre un "testo" e dunque va trattato con regole filologiche che sono imprescindibili nel lavoro di collazione e/o edizione dell'opera stessa.

Un'opera ci può essere giunta in due differenti tradizioni: tradizione diretta e tradizione indiretta. Nel primo caso possediamo testimoni, manoscritti o a stampa, che riportano il testo "originale", ovvero l'opera così come fu pensata dall'autore; mentre nel secondo si hanno citazioni o estratti dell'opera presenti all'interno di altre opere. Poiché in nessun caso i testi giunti fino ai giorni nostri sono privi di errori — di copiatura, di stampa, lacune, aggiunte, trasposizioni, cambiamento di lezioni — l'editore[4] è tenuto a rendere conto di tutte le varianti presenti nella tradizione, scegliendo di volta in volta quelle importanti, e dunque da riportare in apparato, e quelle banali da non riportare, il tutto tenendo ben presente lo scopo finale dell'editore stesso, vale a dire restituire un testo che sia il più vicino possibile all'originale.

3. Giorgio Valla (Vigoleno 1447 - Venezia 1500) rappresenta la figura dell'umanista del maturo e tardo Quattrocento. In vita godette di grande fama e stima tra i suoi contemporanei; nato vicino a Piacenza trascorse l'intera esistenza, ad eccezione di qualche breve viaggio, nel nord della penisola italica svolgendo la sua attività di insegnante in alcuni dei centri più importanti e vitali del settentrione — ovvero nella zona in cui vi fu un maggiore interesse all'orientamento della cultura in senso sia tecnico che scientifico — fu infatti maestro a Pavia, Milano, Genova e in ultimo a Venezia. Giorgio Valla fu uno studioso completo, egli rappresenta il tipico esempio di umanista impegnato attivamente nelle scienze; fu in possesso di numerosi codici scientifici — alcuni dei quali all'epoca semi sconosciuti — e fu il primo che tradusse dal greco al latino molti di quei testi scientifici.

4. Col termine "editore" si intende la persona che sulla base delle proprie specifiche conoscenze dopo aver studiato tutti i testimoni e aver valutato ogni variante, produce il testo critico dell'opera affidata alle sue cure.

Prima della rivoluzione della stampa a caratteri mobili le opere erano tramandate solo in forma di manoscritti e, per ovvi motivi, ad eccezione dei pochi manoscritti olografi, tutti gli altri erano tramandati da una schiera di copisti. Quando si parla di copisti si deve tenere ben presente il periodo che si sta trattando: quello latino, in cui "l'industria editoriale" era monopolio escusivamente laico; quello medievale, che rappresenta un arco di tempo molto ampio e dunque si va dall'Alto Medioevo, in cui erano solo i monaci a fare il lavoro di copiatura dei codici, al Basso Medioevo in cui, grazie anche allo sviluppo delle università, si passa ad un mercato del manoscritto, in cui i copisti non sono più solo esclusivamente monaci, ma sono anche mercanti, studenti, insegnanti o semplicemente persone che ricopiano i codici per poterne avere una loro copia personale. L'attività del copista travalica spesso il suo compito specifico: egli diventa filologo o addirittura coautore, aggiungendo a proprio arbitrio postille, variando la lezione dell'antigrafo (modello) ed emendando lacune con parole ed idee del tutto personali, con il risultato di mettere in crisi il povero editore che a volte può scambiare una congettura o una riscrittura del copista per una lezione originale.

Da quanto abbiamo detto, sebbene in modo molto sintetico, si comprende la necessità che ha un'editore moderno di presentare l'opera con tutte le varianti presenti nei suoi testimoni. Il fruitore dell'edizione critica deve essere in grado di ricostruire per intero uno qualsiasi dei testimoni leggendo l'apparato, ed inoltre deve poter trovare tutte le informazioni che gli necessitano al fine di ricostruire lo *stemma codicum* dell'opera — l'albero genealogico dei codici, cioè le parentele tra i codici che permettono di risalire all'archetipo. Dunque è essenziale avere uno strumento informatico che permetta di gestire in modo adeguato un numero variabile n di testimoni e che consenta di farne l'apparato critico.

3 Aspetti informatici di un'edizione critica

Fare un'edizione critica significa non solo trascrivere un testo, ma anche collazionare tutti i testimoni esistenti dell'opera in questione[5], fare delle scelte di tipo editoriale che non precludano la possibilità di ricostruire interamente uno qualsiasi dei testimoni usati e tutte le operazioni devono rispettare regole filologiche ben precise. MAURO-TEX è lo strumento informatico che consente di fare tutto ciò tenendo presenti tutti i testimoni; con esso è possibile fornire in apparato tutte le varianti presenti nella *traditio*[6] dell'opera, permettendo in tal modo di fare una collazione che rispetti gli standard filologici. Questo pacchetto aggiuntivo a LATEX consente, con dei comandi base, tra le altre funzionalità di tenere conto di tutte le correzioni e le aggiunte, sia in margine che in interlinea, presenti in ciascuno dei testimoni presi in esame.

Un editore è posto nella condizione di dover risolvere anche problemi concreti di gestione della tradizione di un'opera; egli deve dare conto delle note a margine, delle annotazioni interlineari, delle lacune, delle cancellazioni, delle aggiunte, delle varianti tra i testimoni e delle figure presenti nel testo ma tenendo presente sempre il suo obiettivo finale, vale a dire quello di ottenere un'edizione, anche cartacea, che rispetti dei criteri editoriali e d'eleganza predefiniti[7].

3.1 Il pacchetto MAURO-TEX

MAURO-TEX è un pacchetto LATEX sviluppato nell'ambito del "Progetto Maurolico" per la trascrizione e l'edizione critica di testi matematici traditi in più testimoni. Questo pacchetto nasce dalla necessità di trascrivere in modo scientifico conoscenze e informazioni filologiche; fare un'edizione critica di un testo significa molto spesso dover trascrivere un numero anche elevato di testimoni (sia a stampa che manoscritti) e per fare ciò è necessario integrare un linguaggio di descrizione di pagina per pubblicazioni scientifiche quale è LATEX con degli opportuni comandi di tipo "filologico". I comandi di MAURO-TEX, permettono di avere uno standard editoriale per l'impaginazione, l'inserimento delle figure, sia nel testo sia a margine, l'utilizzo di simboli particolari, la scrittura con l'alfabeto greco, l'evidenziazione grafica delle diverse parti di una proposizione matematica e della sua dimostrazione, consentono la ricostruzione integrale di uno qualsiasi dei testimoni usati per l'edizione poiché vengono gestite con dei comandi particolari sia le note a margine, sia gli interlinea e anche una qualsiasi *adnotatio* distinguendo se esse siano dell'autore del manoscritto o aggiunte in un secondo tempo.

Alcuni comandi tipici di MAURO-TEX sono `\begin{Enunciatio}` e `\end{Enunciatio}` i quali cambiano sia il corpo che il formato del testo che si è posto fra essi racchiudendolo in evidenza; una funzione analoga, ma con caratteristiche stilistiche differenti, viene svolta anche dai comandi `\begin{Protasis}` e `\end{Protasis}`. Il comando `\Folium{A:88r}` indica il passaggio a un nuovo

5. La *collatio codicum* è il confronto tra i codici. L'editore prende un testo base come punto di riferimento, che può essere rappresentato dall'edizione precedente o da un codice, poi prendendo un codice (o un altro codice), direttamente o in riproduzione, lo confronta, passo per passo; si segnano tutte le varianti del codice rispetto al testo base. Lo stesso si fa con ogni codice esistente, sempre rispetto al testo base.

6. La *traditio* di un'testo è l'insieme dei manoscritti e degli stampati in cui è trasmesso tale testo.

7. Riguardo al problema delle edizioni critiche elettroniche è rilevante il lavoro svolto da Jerónimo Leal, al riguardo si consulti il sito: `http://antiqua.pusc.it/CeTeX/index.html`; interessante anche la comunicazione orale di Leal durante il GuIT Meeting 2004 dal titolo: *CeTeX: Edizioni Critiche con LATEX*.

foglio nel testimone A — precisamente la fine del foglio 87 v e l'inizio del foglio 88 r; inoltre se nel preambolo è presente `\FoliumInMargine`, genera nell'edizione critica il simbolo "|" e, a margine, l'indicazione A:88r. Se invece nel preambolo viene aggiunto lo stesso comando con l'aggiunta di `\FoliumInTesto`, il comando `\Folium` genera nel testo dell'edizione, nel punto preciso di cambio di foglio, l'indicazione [Fol. A:88r].

Poiché molte delle opere di matematica antica che ci sono pervenute sono in lingua greca — ed anche nei tantissimi casi che la *traditio* dell'opera è solo in latino molto spesso i testimoni riportano frasi in greco — vi è la necessità di codificare i caratteri dell'alfabeto greco ed una funzione importante di MAURO-TEX è perciò la possibilità di scrivere in lingua greca. Tale comando é `\GG`, il quale converte le lettere latine in quelle dell'alfabeto greco seguendo una tabella di conversione, genera le maiuscole semplicemente ponendo maiuscola la lettera latina corrispettiva e permette anche di inserire gli accenti, gli spiriti e le dieresi. Il comando più importante di MAURO-TEX è quello che gestisce le varianti registrate fra i vari testimoni: `\VV`. La sintassi di base è molto semplice: ogni testimone è associato ad una lettera maiuscola racchiusa fra due parentesi graffe {}.

3.2 Il programma MetaPost

Un posto rilevante nell'edizione di testi matematici è occupato dalle figure. Attualmente, non esiste uno standard condiviso relativo all'edizione e alla collazione delle immagini. Nella nostra figura di editori, abbiamo quindi liberamente scelto di utilizzare lo strumento informatico che, a nostro parere, era il più adatto ai fini dell'edizione critica non solo del testo ma anche delle figure incluse nell'opera; la nostra scelta è caduta su un programma già presente nel mondo informatico, il MetaPost.

Il MetaPost è un linguaggio di programmazione finalizzato alla creazione e alla gestione di figure all'interno di documenti scientifici. Il suo inventore è John D. Hobby, il quale ha basato il suo linguaggio sul MetaFont, un sistema per la creazione di font sviluppato da Donald Knuth. Il MetaPost, con la sua sintassi concisa ma chiara si e' dimostrato particolarmente adatto alla rappresentazione dell'iconografia matematica dei testi da noi trattati.

Quando si cura l'edizione critica di un'opera matematica, occorre trattare le figure presenti con la stessa attenzione dedicata al testo scritto. Varianti nelle figure relative alle dimostrazioni possono avere lo stesso peso di varianti all'interno del testo della dimostrazione stessa. Può succedere che si riesca a ricostruire lo stemma di un codice proprio usando le varianti presenti nelle figure; risulta quindi essenziale poter presentare anche le figure dell'opera in forma critica, compito per il quale il MetaPost si rivela essere uno strumento molto adatto.

La struttura di un file MetaPost è la seguente:

```
beginfig(1);
comandi MetaPost
endfig;
beginfig(2);
comandi MetaPost
endfig;
...
end
```

Nella sintassi MetaPost le uguaglianze denotate per mezzo di = sono equazioni soddisfatte dalle coordinate dei punti, e non assegnazioni; sono invece assegnazioni quelle denotate dal simbolo :=, ed, implicitamente, quelle dei cicli `for`. Vi sono anche differenti tipi di dato: `numeric` e `pair`.

4 Esempi di MAURO-TEX e MetaPost

4.1 MAURO-TEX

Come abbiamo detto all'inizio di questo articolo, collazionare e fare un'edizione critica di un'opera richiede la conoscenza e l'utilizzo di regole filologiche e la codifica imformatica di queste regole viene fatta con MAURO-TEX.

Un primo comando usato frequentemente è `\Cit` col quale vengono codificate le citazioni, la cui sintassi è formata da tre sottocampi: nel primo va inserito il testo originale della citazione, nel secondo l'etichetta che identifica univocamente la citazione in un'edizione di riferimento, nel terzo eventuali note. Ad esempio

> *A* suspendatur `\Cit{{per primum postulatum}{MAU/DMA/1/pos1}{senza dubbio si riferisce al primo postulato del De momentis}}` a puncto *B*

genera:

> *A* suspendatur *per primum postulatum* a puncto *B*

Un'altra macro con sintassi simile a quella di `\Cit` è `\Date` che codifica le date: Castellobono `\Date{{6. Decembris die Martis M.D.XLVII}{06.12.1547}}` che genera "Castellobono *6. Decembris die Martis M.D.XLVII*". Ma, come abbiamo già detto, la macro fondamentale di MAURO-TEX è la `\VV`; consideriamo per esempio il caso in cui il testimone A riporta la lezione "Centrum gravitatis circuli est" mentre il testimone B riporta "Centrum gravitatis trianguli est". Si userà allora il comando `\VV{{A:circuli}{B:trianguli}}`, che genera:

> Centrum gravitatis circuli[1] est

[1] circuli *A* trianguli *B*.

Il comando `\VV` mette a disposizione molte altre caratteristiche utili. Un esempio è il caso in cui in un testimone si trovi una lacuna; supponiamo per esempio che il testimone A riporti la lezione "Centrum gravitatis circuli est" mentre il testimone B abbia al posto di "circuli" una macchia illeggibile. In questo caso si scriverà `\VV{{A:circuli}{B:\NL}}`, che genera:

> Centrum gravitatis circuli[1] est

[1] circuli *A non legitur B*.

Questa macro gestisce dunque tutte le casistiche che si possono presentare ad un'editore e/o trascrittore, comprese le annotazioni marginali, interlineari, le correzioni, le cancellature etc., utilizzando, come abbiamo visto, una sintassi molto semplice. Vediamo la sintassi nel caso di aggiunte interlineari e di cancellazioni:

> opposite `\VV{{M+:\INTERL:minorum}}` sunt aliae quandoque partes nomine, `\VV{{M+:\ANTEDEL{nisi}:sola}}` sub

genera:

> opposite minorum[1] sunt aliae quandoque partes nomine sola[2], sub

[1] minorum *in interl. M*
[2] *ante* sola *del.* nisi *M*

Senza continuare nella descrizione dettagliata dei comandi MAURO-TEX, possiamo comprendere come questo pacchetto sia risolutivo nel lavoro dell'editore di testi matematici, fornendogli uno strumento informatico essenziale e permettendo di produrre un documento filologicamente impeccabile.

4.2 MetaPost

Di seguito riportiamo tre esempi di figure riprodotte all'interno dell'edizione critica[8] dell'opera di Francesco Maurolico il *De mometis aequalibus*, per la cui edizione sono state fatte oltre 200 figure:

Codice sorgente:

```
beginfig(1)
u:=0.5cm;
pair A, B, C, a, b, c, d, e, f;
a:=(0,0); c:=(4*u,0); b:=(6*u,0);
A:=(0,1*u);
C:=(4*u,1*u); B:=(6*u,1*u);
d:=(0,2*u); e:=(4*u,2*u); f:=(6*u, 2*u);
draw a-c-b; draw A-C-B;
draw d-e-f; draw a-A-d;
draw c-C-e; draw b-B-f;
label.llft(btex $A$ etex, A);
label.llft(btex $C$ etex, C);
label.lrt(btex $B$ etex, B);
endfig;
```

8. Cfr. Tucci (2004).

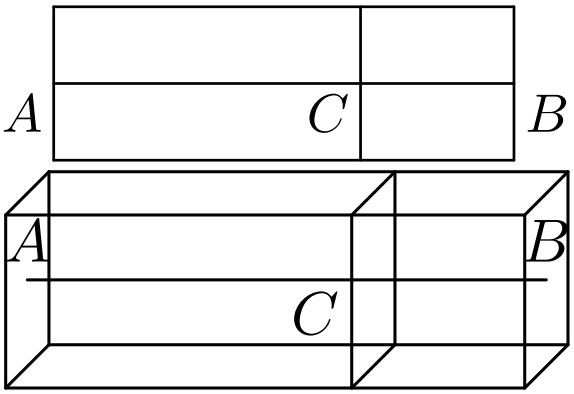


Figura 1: Figura della Proposizione I.26 del *De momentis aequalibus*

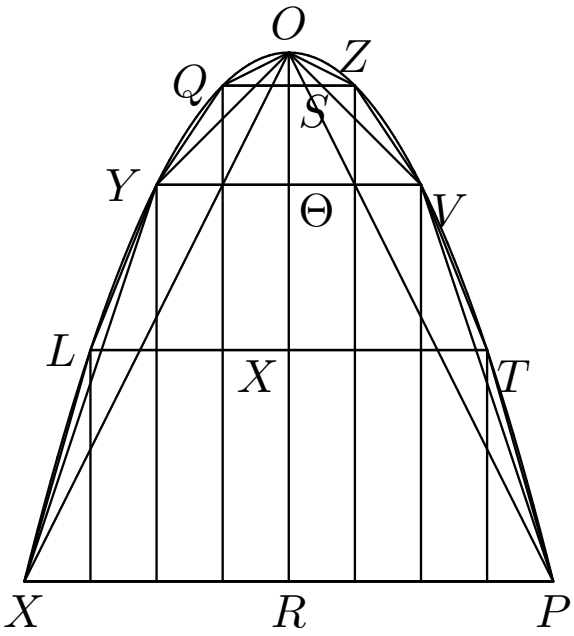


Figura 2: Figura della Proposizione IV.12 del *De momentis aequalibus*

```
beginfig(2)
u:=0.5cm;
pair A, B, C, a, b, c, d, e, f, g, h, i,
l, m, n;
a:=(0,0); c:=(4*u,0); b:=(6*u,0);
A:=(1/4*u,5/4*u);
C:=(4*u,5/4*u); B:=(25/4*u,5/4*u);
d:=(0,2*u);
e:=(4*u,2*u); f:=(6*u, 2*u);
g:=(1/2*u,1/2*u); h:=(9/2*u,1/2*u);
i:=(13/2*u,1/2*u);
l:=(1/2*u,5/2*u); m:=(9/2*u,5/2*u);
n:=(13/2*u,5/2*u);
draw a-c-b; draw A-C-B;
draw d-e-f; draw g-h-i;
draw l-m-n; draw a-g;
draw c-h; draw b-i;
draw e-m; draw d-l;
```

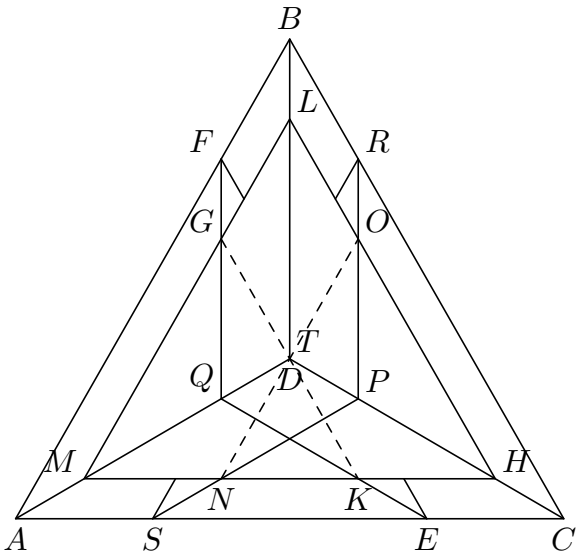


Figura 3: Figura della Proposizione III.16 del *De momentis aequalibus*

```
draw f-n; draw g-l;
draw h-m; draw i-n;
draw a-d; draw c-e;
draw b-f; label.top(btex A etex, A);
label.llft(btex C etex, C);
label.top(btex B etex, B);
endfig;
beginfig(4)
u:=2cm;
pair A,S,E,C,M,N,K,H,Q,D,P,T,G,O,F,R,L,B;
A:=(0,0); C:=(4*u,0); B:=(2*u,3.5*u);
pair b; b:=(2*u,0);
D:=1/3[b,B]; M:=1/4[A,D];
Q:=3/4[A,D]; H:=1/4[C,D];
P:=3/4[C,D]; S:=1/4[A,C];
F:=3/4[A,B]; E:=3/4[A,C];
N:=whatever[M,H]=whatever[S,P];
K:=whatever[M,H]=whatever[Q,E];
L:=1/4[B,D];
G:=whatever[F,Q]=whatever[M,L];
R:=1/4[B,C];
O:=whatever[L,H]=whatever[R,P];
pair s,e,f,r;
f:=whatever[F,E]=whatever[L,M];
e:=whatever[F,E]=whatever[M,H];
s:=whatever[M,H]=whatever[S,R];
r:=whatever[L,H]=whatever[S,R];
T:=whatever[G,K]=whatever[N,O];
draw A-B-C-cycle;
draw M-H-L-cycle;
draw A-D; draw C-D;
draw B-D; draw s-S;
draw f-F; draw R-r;
draw E-e; draw F-Q;
draw Q-E; draw R-P;
draw P-S; draw N-O dashed evenly;
draw G-K dashed evenly;
label.bot(btex A etex, A);
label.bot(btex S etex, S);
label.bot(btex E etex, E);
label.bot(btex C etex, C);
label.bot(btex N etex, N);
label.bot(btex K etex, K);
label.ulft(btex M etex, M);
label.urt(btex H etex, H);
label.ulft(btex Q etex, Q);
label.urt(btex P etex, P);
label.bot(btex D etex, D);
label.urt(btex T etex, T);
label.ulft(btex G etex, G);
label.urt(btex O etex, O);
label.urt(btex R etex, R);
label.ulft(btex F etex, F);
label.urt(btex L etex, L);
label.top(btex B etex, B);
endfig;
```

Riferimenti bibliografici

Hobby, J. D. (1997). «The MetaPost system».

Hobby, J. D. (1998). *A User's Manual for MetaPost*. AT&T Bell Laboratories, Murray Hill, NJ 07974.

Sutto, J.-P. (a cura di) (2006). *Manuale del MAURO-TEX*. `http://www.maurolico.unipi.it/mtex/mtex.ht`.

Tucci, R. (2003–2004). *Il "De momentis aequalibus" di Francesco Maurolico: una proposta di ricostruzione della sua stratificazione testuale*. Tesi di Laurea, Università di Pisa. (Rel. P.D. Napolitani).

▷ Roberta Tucci
Università di Pisa
`tucci@dm.unipi.it`

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 16 Marzo 2007. 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 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 in questa pagina web, un termine di scadenza per la presentazione degli articoli da pubblicare nel numero in preparazione della rivista. Tuttavia gli articoli potranno essere inviati in qualsiasi momento e troveranno collocazione, eventualmente, nei numeri seguenti.

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

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 2, Ottobre 2006