Numero 5
Aprile 2008

ArsTEXnica

Rivista italiana di TEX e LATEX

GUIT

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

TEXnica Ars

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 una quota di iscrizione annuale diversificata: 30,00 € soci ordinari, 20,00 (12,00) € studenti (junior), 75,00 € Enti e Istituzioni.

Indirizzi

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

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

Codice ISSN 1828-2369

Stampata in Italia
Pisa: 15 Aprile 2008

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 5, Aprile 2008

Gruppo Utilizzatori Italiani di TEX

Editoriale

Gianluca Pignalberi

Come già annunciato dal mio predecessore Massimiliano Dominici, attualmente presidente del GUIT, questo numero di ArsTEXnica prevede un "cambio al vertice" dato che da questo numero, e fino alla scadenza naturale del mandato, sarò io il direttore.

La proposta per questo delicato e coinvolgente incarico mi è stata fatta per la mia precedente esperienza in campo editoriale, che va dalla collaborazione come giornalista con varie riviste di settore, alla collaborazione con Free Software Magazine in qualità di programmatore della classe LATEX con cui la rivista è stata composta finché ne è esistita una versione stampata, e di compositore *tout court*.

Purtroppo, tale esperienza non è indice che il mio impiego come direttore sarà immediatamente ben focalizzato. Dunque mi perdonerete il periodo di "rodaggio", indispensabile per prendere le misure del lavoro che dovrò svolgere. La mia eventuale crescita nel compito è merito di Massimiliano, che non mi ha negato il suo aiuto e la sua esperienza, e anzi si è prodigato in consigli e assistenza, e si è premurato di seguire con discrezione la sua "creatura" man mano che cresceva il numero delle sue pagine.

È con grande onore e altrettanta preoccupazione che ho accettato di candidarmi, ed è con gratitudine che ricevo l'investitura da parte dei membri della redazione: non sarà facile ripetere in qualità i numeri passati diretti da Massimiliano, sebbene questo numero è la prova che si può raggiungere, e anche superare, detta qualità. Se riusciremo costantemente in ciò, sarà esclusivamente merito di tutta la redazione e degli autori. Non solo essi mi hanno accordato la loro fiducia, ma si sono impegnati, e si impegnano costantemente, a non farmi mancare la loro professionalità, serietà e appoggio.

Il risultato lo avete sotto gli occhi: il quinto numero di ArsTEXnica è un concentrato di vari argomenti ottimamente presentati e spiegati. Per quanto mi consta, per la prima volta gli articoli vengono presentati in ordine di difficoltà. Questo vuole essere un segno, seppur minimo, non già di critica o differenziazione da quanto ottimamente svolto da Massimiliano, ma di contributo alla crescita di ArsTEXnica, che non può e non deve fermarsi mai allo *status quo*. Personalmente ho ulteriori idee al riguardo, ma non è qui e ora la sede per annunciarle; avremo modo di riparlarne nei prossimi numeri. Nell'intervallo tra un numero e l'altro, sarà mia cura lavorare, in collaborazione con i volontari della redazione, alla definizione e realizzazione di quegli strumenti che verranno giudicati utili alla crescita della rivista, e del Gruppo attraverso di essa. E dovrà necessariamente essere mia cura proseguire, o terminare, lo sviluppo degli strumenti che Massimiliano aveva già cominciato a sviluppare.

Basta chiacchiere e convenevoli; è ora di andare al cuore del nostro interesse comune: i contenuti del numero di ArsTEXnica che vi apprestate a leggere.

Si comincia con l'ottimo articolo di Massimo Guiggiani e Lapo Mori, che ci spiegano come (non) maltrattare le formule matematiche. L'articolo fornisce un gran numero di esempi e controesempi, per mostrarci come si può ottenere una serie di formule ben scritte e con un codice chiaro e coerente, a partire da esempi palesemente sbagliati o confusi. Il loro articolo non può che fare bene ad una gran quantità di persone che, per faciloneria o misconoscenza, si accontentano di avere formule che *sembrano* composte correttamente, ma che invece non lo *sono*.

Il secondo articolo di questo quinto numero si intitola Introduzione a XETEX, ed è a cura di Massimiliano Dominici. Nella sua introduzione, Massimiliano focalizza la sua e nostra attenzione sullo standard di codifica Unicode e sulle tecnologie dei cosiddetti *smart font*, e di come questi possono essere integrati in un sistema di composizione tipografica basato su TEX tramite XETEX. L'autore non lesina nella sua analisi, partendo da esempi di codice a basso livello (Plain TEX) ed arrivando ad esempi di alto livello, evidenziando l'uso di XETEX in LATEX e ConTEXt.

In un crescendo di difficoltà troviamo l'articolo di Claudio Beccari, incentrato sulle macroistruzioni con argomenti delimitati. Come spiegato dall'autore, l'argomento è adatto per chiunque abbia voglia di capire più profondamente i meccanismi di TEX, ma risulta essenziale solo a chi si debba cimentare con la scrittura di comandi. Gli autori puri dovrebbero astenersi dal mescolare dette istruzioni all'interno dei propri articoli.

Infine, la rivista che state leggendo si chiude con l'articolo di Enrico Gregorio, che ci spiega come abilitare la cesura a fine riga per documenti multilingua redatti in Plain TEX, fornendo anche un'interfaccia per la definizione delle lingue. L'articolo in questione ci introduce, con dovizia di spiegazioni e numerosi brani di codice, all'uso del pacchetto HyPlain.

Dopo aver fatto un ulteriore, doveroso ringraziamento a tutta la redazione e agli autori, non posso

non ringraziare tutti voi lettori di ArsTEXnica, sia privati che istituzioni. Il vostro appoggio è fondamentale per la rivista e il GUIT. Appoggio sempre maggiore con il passare del tempo. Grazie di cuore.

▷ Gianluca Pignalberi
g dot pignalberi at
freesoftwaremagazine dot com

Consigli su come *non* maltrattare le formule matematiche

Massimo Guiggiani, Lapo F. Mori

Sommario

Accade spesso di imbattersi in formule matematiche corrette, ma scritte in modo ambiguo o inutilmente complicato. In effetti, raramente si insegna come scrivere le formule e quali aspetti valutare al momento di scegliere una notazione. Questo articolo fornisce indicazioni per colmare queste lacune.

Abstract

Mathematical formulæ are often correct but ambiguous or uselessly complicated. Actually the rules to follow and the aspects to take into account when choosing a notation are very rarely taught. This article provides some suggestions to fill this gap.

1 Introduzione

Un ottimo modo per esprimere (certi) concetti matematici è mediante le *formule*. Sembra una cosa ovvia, ma si è arrivati a questa conclusione attraverso un processo storico abbastanza faticoso. A prima vista può sembrare che il linguaggio naturale sia più facile da usare, mentre le formule possono essere percepite come estranee. È tuttavia esperienza comune di tutti coloro che fanno scienza che le formule siano insostituibili per comunicare in modo chiaro, non ambiguo e sintetico.

Le formule matematiche fanno quindi parte di un linguaggio e, come tali, dovrebbero rispettare determinate regole. Purtroppo, accade frequentemente di imbattersi in formule "scritte male", cioè non chiare, ambigue, non sintetiche, con evidenti conseguenze sulla leggibilità.

Lo scopo di questo articolo è, appunto, quello di dare delle indicazioni di massima su come scrivere certe semplici formule, o forse su come *non* scriverle. In effetti, per imparare a scrivere nel linguaggio naturale occorre andare a scuola, fare errori ed essere corretti. Tipicamente, chi non è scolarizzato sa comunicare nella propria lingua, ma scrive in modo sgrammaticato, eppure comprensibile (chi non capisce "Io ho andato al mare"?). La stessa cosa è valida per le formule, però solo raramente viene insegnato quali regole seguire e quali errori evitare. Gli esempi che vengono esaminati sono tutti estratti da pubblicazioni, tesi di laurea e dispense universitarie.

L'articolo si rivolge prevalentemente ad autori di scritti di carattere tecnico-scientifico (ingegneri, fisici sperimentali, chimici e in genere coloro che trattano la matematica in quanto strumento) e non a matematici in quanto la tradizione tipografica per la matematica pura è in netto contrasto con le norme ISO non tanto per ragioni "sentimentali", quanto per motivi attinenti alla matematica stessa (Gregorio, 2007a).

Le considerazioni riportate nel seguito sono rivolte a chiunque scriva documenti tecnico-scientifici, indipendentemente dallo strumento di composizione tipografica utilizzato. Quando necessario, tuttavia, si riportano consigli su come applicare le regole suggerite con LaTeX. Per queste parti, il testo presume che il lettore conosca già i rudimenti di LaTeX, ovvero che abbia letto una delle numerose guide di base disponibili gratuitamente in rete (American Mathematical Society, 2002a; Baudoin, 1997; Flynn, 2005; Gregorio, 2007b; Oetiker *et al.*, 2007; The Tutorial Team, 2000; Voss, 2007) oppure un libro (Diller, 1999; Goossens *et al.*, 1995; Grätzer, 1999, 2000; Hahn, 1993; Higham e Griffiths, 1997; Knuth, 1992; Kopka e Daly, 1995; Lamport, 1994).

2 La regola generale

La regola da seguire è molto semplice: le formule matematiche devono essere non ambigue e sintetiche, ossia chiare e semplici.

In realtà questo vale (o dovrebbe valere) per molte forme di comunicazione, non solo scritte. Quando si fa scienza, però, si dovrebbe porre maggiore attenzione a come si espongono i concetti per facilitarne la comprensione.

Regole più dettagliate su come scrivere le formule matematiche sono esposte nella norma ISO 31 (ISO 31/11, 1982; ISO 31/12, 1982), discussa ampiamente su Wikipedia (a,b,c,d) e nell'articolo di Beccari (1997).

3 Operazioni

3.1 Moltiplicazione

L'errore più comune è l'uso del "pallino" per indicare la moltiplicazione fra scalari. Per esempio le seguenti formule

$$a \cdot x^2 + b \cdot x + c = 0, \qquad \sigma \cdot \varepsilon = 2 \cdot \alpha$$

andrebbero scritte così

$$ax^2 + bx + c = 0, \qquad \sigma\varepsilon = 2\alpha.$$

Infatti, in ossequio al principio di semplicità, la moltiplicazione fra lettere, o fra un numero e una lettera, non richiede nessun simbolo.

Se invece si moltiplicano due numeri occorre utilizzare il simbolo $\times$ oppure $\cdot$ per evitare ambiguità. Si scrive per esempio

$$2 \times 3 = 6 \quad \text{e non} \quad 2\,3 = 6.$$

Lo stesso simbolo andrebbe utilizzato quando si spezza una formula su più righe in corrispondenza di un prodotto, come mostrato nell'eq. (1).

Improponibile l'uso dell'asterisco:

$$a * x^2 + b * x + c = 0, \qquad \sigma * \varepsilon = 2 * \alpha.$$

Il pallino va utilizzato per indicare il prodotto scalare fra vettori. A tale scopo è possibile utilizzare sia il pallino sottile `\cdot` che un pallino più spesso `\bcdot`, definito con il seguente comando

```
\newcommand{\bcdot}{\boldsymbol{\cdot}}
```

I due comandi producono rispettivamente

$$\mathbf{a} \cdot \mathbf{c} = 0 \qquad \text{e} \qquad \mathbf{a} \bcdot \mathbf{c} = 0.$$

3.2 Operatori matematici

Gli operatori, incluso il simbolo del differenziale, vanno indicati con carattere dritto, come nei seguenti esempi

$$\int \sin x \,\mathrm{d}x, \qquad \frac{\mathrm{d}y}{\mathrm{d}x}, \qquad \lim_{x\to 0} \cos x = 1.$$

Si tratta infatti di abbreviazioni di parole e il font dritto contribuisce a ricordare questo fatto, oltre a evitare ogni ambiguità con possibili prodotti. Per esempio, $\tan x$ è indubbiamente la tangente di x, mentre $tanx$ potrebbe essere il prodotto di quattro termini. Si confronti anche la diversa leggibilità.

LaTeX e il pacchetto amsmath definiscono i più comuni operatori matematici tra cui `\lim`, `\sin`, `\min`, ecc. Se fosse necessario definirne di nuovi, il pacchetto amsmath mette a disposizione il comando `\DeclareMathOperator`, da utilizzarsi solo nel preambolo, e `\operatorname`, che permette di definirsi un nome di operatore 'al volo' senza bisogno di definire un comando apposito. Per esempio per scrivere

$$\operatorname{argmax} f(x)$$

è possibile definire l'operatore nei seguenti modi:

- dichiarando nel preambolo

  ```
  \DeclareMathOperator{\argmax}{argmax}
  ```

  e poi richiamandolo nel testo come

  ```
  \argmax f(x)
  ```

- dichiarando direttamente nel testo

  ```
  \operatorname{argmax} f(x)
  ```

Si noti che la definizione di un operatore non è equivalente all'uso di `\mathrm`[1] perché la prima si preoccupa anche di gestire lo spazio tra l'operatore e l'operando. Si noti per esempio la differenza tra

$$\sin x \qquad \text{e} \qquad \mathrm{sin}x$$

ottenuti rispettivamente con

```
\sin x
```

e

```
\mathrm{sin}x
```

Gli operatori, inoltre, permettono anche un'adeguata gestione di apici e pedici come nel caso dei limiti. Per esempio

$$\lim_{x\to\infty} \frac{1}{x}$$

si ottiene semplicemente con

```
\lim_{x\to\infty}\frac{1}{x}
```

Per scrivere correttamente il simbolo del differenziale è richiesta una definizione un po' particolare. Infatti è un operatore con spazio solo alla sua sinistra. In Beccari (2007b) viene proposta la seguente soluzione

```
\newcommand{\ud}{\mathop{}\!\mathrm{d}}
```

in cui si introduce un operatore vuoto, salvo poi eliminare lo spazio a destra con il comando `\!`.[2] Si noti la differenza tra

$$\int \sin x \mathrm{d}x \qquad \text{e} \qquad \int \sin x \,\mathrm{d}x,$$

dove il differenziale è ottenuto rispettivamente con `\mathrm{d}` e con `\ud`.

3.3 Equazioni su più righe

Quando si trattano equazioni lunghe può accadere di doverle spezzare su più righe. Vale come regola generale che i segni di operazione o relazione, cioè quelli che permettono di spezzare una formula, vanno indicati *una sola volta*, a termine riga in caso di formule in corpo, e ad inizio riga in caso di

1. Per ottenere caratteri dritti in ambiente matematico è possibile utilizzare il comando `\mathrm`. Gli altri metodi per scrivere caratteri dritti in ambiente matematico sono descritti nel par. 5.1.1.
2. `\!` inserisce uno spazio negativo uguale a quello che TeX inserisce fra un operatore e una variabile (Beccari, 2007a).

formule fuori corpo[3]

$$f = \frac{a}{b}\left[\sin\left(\frac{c}{d}\right) + \tan\left(\frac{c}{d}\right)\right] \times \left[1 + \sin\left(\frac{e}{f}\right) - \cos\left(\frac{g}{h}\right)\right]. \quad (1)$$

La doppia indicazione è sconsigliabile, oltre che per ragioni estetiche, anche per evitare ambiguità. Se per esempio si scrivesse

$$a = b + c + d - - e - f,$$

non sarebbe chiaro quale delle due si debba intendere

$$a = b + c + \boxed{d - e} - f$$
$$a = b + c + \boxed{d + e} - f.$$

Questa regola si estende anche ai casi in cui sono presenti più segni di uguaglianza come nel seguente esempio

$$\begin{aligned} a &= b + c + d \\ &= -e - f \\ &\simeq g. \end{aligned}$$

3.3.1 Comandi LaTeX

LaTeX si preoccupa automaticamente di spezzare formule lunghe in corpo, mentre spezzare le formule fuori corpo è compito dell'autore. Il pacchetto amsmath mette a disposizione un certo numero di ambienti per gestire le formule. Tra questi si ricordano split, multline, gather, align, aligned, alignat; per i dettagli si veda la documentazione di amsmath (American Mathematical Society, 2002b).

3.3.2 Gestione delle parentesi

Se si usano i comandi `\left` e `\right` per far scegliere in automatico a LaTeX la dimensione delle parentesi, possono nascere problemi nel caso che la formula sia spezzata tra `\left` e `\right`. Per esempio il codice

```
\begin{multline*}
   f(x,y,z)= (1+x+y-z)\left[\pi\\
      +\sin\left(\frac{a}{b}\right)
      +\cos\left( \frac{c}{d}\right)\right].
\end{multline*}
```

non è compilabile perché LaTeX si aspetta che tutti i `\left` vengano chiusi dai rispettivi `\right` all'interno di ogni riga.

Una soluzione consiste nel chiudere `\left` e `\right` rispettivamente con `\right.` e `\left.`

```
\begin{multline*}
   f(x,y,z)= (1+x+y-z)\left[\pi\right.\\
      \left.+\sin\left(\frac{a}{b}\right)
      +\cos\left( \frac{c}{d}\right)\right].
\end{multline*}
```

ma il problema è che adesso la dimensione delle parentesi di ogni coppia `\left` e `\right` viene calcolata indipendentemente e questo può produrre spiacevoli risultati come in questo caso

$$f(x,y,z) = (1 + x + y - z)\left[\pi + \sin\left(\frac{a}{b}\right) + \cos\left(\frac{c}{d}\right)\right].$$

La prima soluzione consiste nell'individuare il termine di altezza maggiore, in questo caso

$$\left(\frac{a}{b}\right),$$

e inserirlo all'interno del comando `\vphantom` nell'altra riga

```
\begin{multline*}
   f(x,y,z)= (1+x+y-z)\left[\pi\vphantom{
      \left(\frac{a}{b}\right)}\right.\\
      \left.+\sin\left(\frac{a}{b}\right)
      +\cos\left( \frac{c}{d}\right)\right].
\end{multline*}
```

ottenendo

$$f(x,y,z) = (1 + x + y - z)\left[\pi + \sin\left(\frac{a}{b}\right) + \cos\left(\frac{c}{d}\right)\right].$$

In questi casi è però consigliabile utilizzare le parentesi con dimensione fissa[4] per evitare di dover individuare l'elemento più alto. Per esempio il codice

```
\begin{multline*}
   f(x,y,z)= (1+x+y-z)\Bigl[\pi\\
      +\sin\Bigl(\frac{a}{b}\Bigr)
      +\cos\Bigl( \frac{c}{d}\Bigr)\Bigr].
\end{multline*}
```

produce

$$f(x,y,z) = (1 + x + y - z)\Big[\pi + \sin\Big(\frac{a}{b}\Big) + \cos\Big(\frac{c}{d}\Big)\Big].$$

Si noti che questa soluzione è valida solo nei casi in cui non si debbano usare parentesi di dimensione maggiore a `\Bigg` (necessario solo con matrici).

3. Con formula in corpo si intende una formula che compare all'interno di una riga di testo, come per esempio $c = a + b$, mentre con formula fuori corpo si intende una che da sola occupa una riga, come per esempio

$$c = a + b.$$

Le prime vengono ottenute in LaTeX con il comando `$...$` mentre le seconde vengono ottenute alternativamente con il comando `\[...\]` oppure con l'ambiente equation.

4. La dimensione fissa può essere ottenuta con i comandi `\bigx`, `\Bigx`, `\biggx` e `\Biggx` (dove *x* è 'l' per le parentesi sinistre, 'r' per le parentesi destre, 'm' oppure anche niente nel caso si voglia denotare un simbolo ordinario).

In aggiunta alle soluzioni presentate, si segnala che il pacchetto breqn, sviluppato da Michael Downes e preso in carico da Morten Høgholm, è proprio finalizzato alla cesura automatica delle formule e il dimensionamento automatico delle parentesi. Tale pacchetto fa funzionare i comandi `\left` e `\right` anche in presenza di interruzione di riga.

3.4 La punteggiatura

Esistono due scuole di pensiero sulla punteggiatura esterna, ovvero sulla punteggiatura in formule matematiche fuori corpo. Alcuni ritengono che questa non andrebbe mai usata (Beccari, 2007a), altri che ritengono tale punteggiatura necessaria ed essenziale.[5] La cosa fondamentale, qualunque dei due metodi si scelga, è essere coerenti e usare sempre un unico metodo.

Gli autori di questo articolo ritengono che le formule, sia nel testo che fuori corpo, facciano parte dell'argomentazione e, quindi, la punteggiatura vada usata per aiutare il lettore. Un esempio di buon uso della punteggiatura è il seguente:

Dal momento che

$$a = b$$

e che

$$b = c,$$

è provato che

$$a = c.$$

4 Variabili

4.1 Vettori e matrici

4.1.1 Notazione sintetica

I vettori sono spesso indicati con le lettere minuscole in neretto[6]

$$3\mathbf{a} + \alpha\mathbf{c} = \mathbf{v}.$$

Sono da evitare costruzioni come le seguenti

$$\vec{a}, \quad \underline{a}, \quad \{a\}, \quad [a].$$

Lo stesso vale per le matrici, che possono essere indicate con lettere maiuscole in neretto

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

evitando, perché meno leggibili, espressioni di questo tipo

$$[A]\{x\} = \{b\}.$$

5. La maggioranza dei libri italiani di matematica, nuovi e vecchi (compresi libri composti con il piombo a mano), usa, anche se in modo non sempre coerente, punto e virgola, virgola e punto nelle formule fuori corpo (Bellacchi, 1894; Bertini, 1907; Bianchi, 1899; Bonola, 1906; Burali-Forti, 1904; Caprilli, 1912; Cesaro, 1894; Dini, 1878; Fubini, 1908; Gazzaniga, 1903; Peano, 1887; Sacchi, 1854; Veronese, 1891; Vivanti, 1916).

6. Per ottenere in ambiente matematico lettere in neretto dritto si può utilizzare il comando `\mathbf`.

In un testo di algebra lineare, che tratta quindi interamente di matrici e vettori, è perfettamente legittimo abolire il neretto e scrivere semplicemente

$$Ax = b.$$

4.1.2 Notazione per componenti

In alcuni casi è necessario esprimere esplicitamente le componenti di vettori e matrici. Non è raro trovare in libri e tesi di laurea espressioni del tipo

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}.$$

Questa notazione è sconsigliabile perché utilizza parentesi differenti per matrici e vettori e perché utilizza le parentesi graffe per i vettori. Assai migliore la seguente:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{2}$$

Sebbene non esista una normativa o una consuetudine sul tipo di parentesi da utilizzare quando si esprimono vettori e matrici in componenti, è consigliabile:

- utilizzare parentesi tonde o quadre, evitando gli altri tipi (graffe, angolate, uncinate, ecc.);
- mantenere la stessa notazione sia per i vettori[7] che per le matrici lungo tutto il documento.

Per creare matrici e vettori in LaTeX è consigliabile l'impiego degli ambienti pmatrix (se racchiusi da parentesi tonde) e bmatrix (se racchiusi da parentesi quadre) forniti dal pacchetto amsmath. Per esempio l'eq. (2) può essere scritta

```
\begin{bmatrix}
    a_{11}&a_{12}&a_{13}\\
    a_{21}&a_{22}&a_{23}\\
    a_{31}&a_{32}&a_{33}\\
\end{bmatrix}
\begin{bmatrix}
    x_1\\
    x_2\\
    x_3
\end{bmatrix}=\begin{bmatrix}
    b_1\\
    b_2\\
    b_3
\end{bmatrix}
```

Talvolta è necessario riportare le componenti di un vettore all'interno del testo. In questi casi è consigliabile rappresentarlo come vettore riga trasposto in modo da non dilatare eccessivamente lo spazio tra le righe. Esistono almeno due convenzioni per scrivere vettori riga nel testo. La prima

7. Infatti, quando scritto in componenti, un vettore corrisponde a una matrice con una sola colonna (o riga, a seconda della convenzione usata).

consiste nell'utilizzare la convenzione dei vettori fuori corpo; per esempio $\mathbf{v} = [v_1\ v_2\ v_3]^T$. La seconda, adottata per esempio da STRANG (2005), è più compatta e consiste nel separare le componenti con virgole senza riportare il segno di trasposta, che è implicito; per esempio $\mathbf{v} = (v_1, v_2, v_3)$. In entrambi i casi, è consigliabile non utilizzare gli ambienti pmatrix e bmatrix, che produrrebbero parentesi troppo grandi.

La scelta tra parentesi tonde e quadre dipende dai gusti personali. Le parentesi quadre hanno come vantaggio una maggiore compattezza (occupano meno spazio in orizzontale) e una migliore resa grafica con matrici e vettori di molte righe.

4.2 Un simbolo, una lettera

Un simbolo matematico è normalmente indicato da *una* lettera, non da due o tre. Se, per esempio, si vuole suggerire che il *coefficiente di sicurezza* è pari a tre, non si deve scrivere

$$CS = 3$$

perché CS può sembrare il prodotto di due quantità. Molto meglio usare un pedice

$$C_\mathrm{s} = 3.$$

Abbastanza frequente è l'uso di CG per indicare il baricentro (*center of gravity*). Non sarebbe meglio usare semplicemente G?

Le norme ISO prevedono qualche eccezione, come il numero di Mach Ma e il numero di Reynolds Re, che sono rappresentati da due lettere. In questi rari casi in cui serve il carattere corsivo per un simbolo formato da più lettere è necessario utilizzare il comando `\mathit` per evitare problemi di spaziatura tra i caratteri. Si noti per esempio la differenza tra Ma e Ma, ottenuti rispettivamente con `$Ma$` e `$\mathit{Ma}$`. Nel primo la distanza tra M e a fa pensare che si stia considerando una moltiplicazione mentre nel secondo caso lo spazio è corretto.

Si dovrebbe anche evitare di inserire parole intere nelle formule

$$massa \times accelerazione = forza.$$

Intanto, questa dovrebbe essere una relazione vettoriale e ciò non viene comunicato dalla formula. Inoltre le parole andrebbero scritte in carattere dritto

$$\mathrm{massa} \times \mathrm{accelerazione} = \mathrm{forza},$$

in modo da avere le giuste spaziature fra le lettere e non rischiare di suggerire al lettore che la parola sia invece una moltiplicazione tra più grandezze scalari. In ogni caso il risultato non è comparabile in termini di chiarezza e sintesi con

$$m\mathbf{a} = \mathbf{F}.$$

Un altro errore comune è quello di scrivere le formule matematiche usando come simboli le variabili dei programmi di calcolo. Per esempio

$$A\_fl + B\_fl = d\_k\_fl$$

invece di

$$A + B = d.$$

4.3 Apici e pedici

L'uso dei pedici sarebbe da limitare ai soli casi in cui è necessario e, anche in questi casi, devono essere evitati pedici formati da abbreviazioni. Per esempio è sconsigliato scrivere σ_nominale o σ_nom, dove "nominale" ha il significato italiano; in questo caso si potrebbe semplicemente scrivere σ_n. Ovviamente è del tutto lecito usare pedici (o apici) per gli indici, come in un tensore del terzo ordine σ_{ijk}. In questi casi si deve usare il corsivo matematico con il classico comando `\sigma_{ijk}`.

I pedici e gli apici sottostanno alle stesse regole dei simboli (par. 5.1): devono essere in corsivo quando rappresentano grandezze fisiche o matematiche variabili, in dritto negli altri casi. Per esempio si deve scrivere A_T se T rappresenta la temperatura ma A_T se T rappresenta un nome come ad esempio "traiettoria"; si deve scrivere A_i se i rappresenta l'indice di una sommatoria.

L'uso delle parole come pedici può portare a deturpare formule altrimenti semplici, come nel seguente esempio

$$T_{frenante,ruota} = \frac{(T_{motore,erogata} - T_{motore,richiesta}) * C}{N}$$

che forse era meglio scrivere

$$T_\mathrm{f} = \frac{C(T_\mathrm{e} - T_\mathrm{r})}{N}. \tag{3}$$

Dato che i pedici "f", "e" ed "r" sono rispettivamente abbreviazioni di "frenante", "erogata" e "richiesta" vanno in carattere dritto per quanto detto sopra.

4.4 Notazione opportuna

Per facilitare la comprensione di un testo scientifico è fondamentale usare una notazione opportuna. Per rendersi conto di quanta importanza abbia la notazione basta provare a fare una moltiplicazione usando i numeri romani

$$\mathrm{MMCDXXVIII} \times \mathrm{XIX} \quad \text{invece di} \quad 2428 \times 19.$$

Un errore comune è usare *tipi* di lettere diversi per quantità dello stesso tipo: indicare, per esempio, alcune lunghezze con lettere minuscole (l, s, a) e altre con lettere greche (α, γ) non aiuta. Certamente possono esserci delle eccezioni, ma un po' di attenzione non guasta.

Anche i pedici devono essere scelti con cura. Si trovano figure in cui ω_1 è la velocità angolare del corpo 2 e ω_2 è quella del corpo 1.

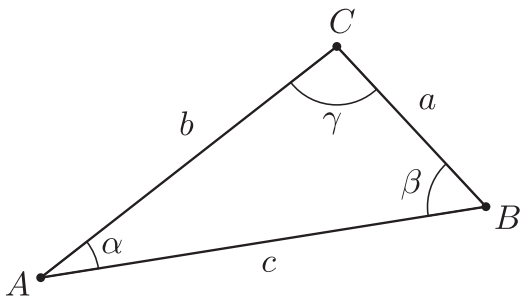


FIGURA 1: Esempio di variabili matematiche indicate con simboli in corsivo.

5 Grandezze fisiche e matematiche

5.1 Simboli

Le norme ISO prescrivono l'uso di simboli in corsivo per grandezze fisiche e matematiche che possono assumere valori differenti. Un esempio sono i nomi dei punti e degli angoli in geometria (Fig. 1).[8] Rientrano in questa categoria anche tutte le costanti fisiche il cui valore non è "costante" perché migliori misure possono modificarlo.[9]

Tutte le altre grandezze fisiche e matematiche devono essere scritte con carattere dritto. Tra queste rientrano l'unità immaginaria $2+4\mathrm{i}$ e la base dei logaritmi naturali $\mathrm{e} = 2{,}718\ldots$, nonché, ovviamente pi greco $\pi = 3{,}141\,592\ldots$ (BECCARI, 2007a).

Si ricorda che, come detto nell'introduzione, la tradizione matematica 'pura' non fa uso di queste convenzioni perché sono utili in campo tecnico ma sarebbero dannose in matematica (GREGORIO, 2007a).

I simboli di composti chimici devono essere scritti con carattere dritto. Per la scrittura sia di singoli composti chimici che di equazioni di reazioni chimiche si consiglia l'utilizzo di un pacchetto dedicato come mhchem.

5.1.1 Tipo di carattere in LaTeX

Di default le lettere scritte in ambiente matematico in LaTeX sono rappresentate in corsivo e quindi rappresentano grandezze matematiche e fisiche non costanti. Per esempio `$a=b$` produce $a = b$.

Per scrivere costanti, ovvero per utilizzare carattere dritto in ambiente matematico, è possibile utilizzare il comando `\mathrm`. Per esempio `$a=\mathrm{b}$` produce $a = \mathrm{b}$. `\mathrm`, pur usando caratteri dritti, è sempre in modalità matematica e quindi gli eventuali spazi non vengono considerati. Nel caso in cui sia necessario inserire spazi, si può usare il comando `\text`. Si noti per esempio la differenza tra

$$v_{\text{sezione ridotta}} \quad \text{e} \quad v_{\text{sezioneridotta}}$$

prodotti rispettivamente con

```
v_\text{sezione ridotta}
```

e

```
v_\mathrm{sezione ridotta}
```

`\mathrm`, rispetto a `\text`, fa sì che il pedice non diventi corsivo in contesto corsivo. Si consideri ad esempio

Vale la disuguaglianza $L_{\mathit{eff}} \neq L_{\mathrm{eff}}$

prodotto da

```
\textit{Vale la disuguaglianza%
$L_\text{eff}\ne L_\mathrm{eff}$}
```

Si ricorda, inoltre, che l'opzione italian del pacchetto babel attiva anche i comandi `\ap` e `\ped` per inserire rispettivamente un apice o un pedice sia in modo testo sia in modo matematico. Essi sono equivalenti a `\mathrm` se compaiono in ambiente matematico e a `\text` se compaiono nel testo. Per esempio il codice

```
A\ped{sezione 1} e $A\ped{sezione 1}$
```

produce

$\mathrm{A}_{\text{sezione 1}}$ e A_{sezione1}

Le lettere greche dritte sono fornite dal pacchetto upgreek. Si noti ad esempio la differenza tra $\upi$ e π ottenuti rispettivamente con `\uppi` e `\pi`.

Gli esempi precedenti servono solo per mostrare il comportamento dei vari comandi, ma resta fermo quanto detto nei paragrafi precedenti: apici e pedici non dovrebbero contenere abbreviazioni formate da più lettere.

5.2 Numeri

I numeri devono sempre essere scritti con carattere dritto (123) invece che in corsivo (*123*).[10] Questo comportamento è seguito di default da LaTeX in ambiente matematico, ma possono nascere problemi nel testo. Per questo motivo è sempre consigliabile utilizzare ambienti matematici per scrivere numeri anche nel testo. Se infatti il numero compare in una frase in corsivo, il fatto che sia in ambiente matematico impone che sia rappresentato da caratteri dritti. Per esempio

```
camminare \emph{al massimo 2 km} a piedi
```

produce

8. I punti e gli angoli, in quanto variabili matematiche, devono essere scritti in corsivo. A questa regola si aggiunge la convenzione adottata dalla maggioranza degli autori di indicare con le lettere latine maiuscole i punti (A, B, C, ecc.), con quelle latine minuscole i segmenti (a, b, c, ecc.) e con quelle greche, di solito minuscole, gli angoli (α, β, γ, ecc.).

9. Rientrano, per esempio, in questa categoria la carica dell'elettrone e, la costante di Planck h, la costante di Planck ridotta $\hbar$ e la costante di Boltzmann k.

10. Le date rappresentano un'eccezione a questa regola e possono essere scritte in corsivo se il contesto in cui compaiono è in corsivo.

camminare *al massimo 2 km* a piedi

anziché la forma corretta

camminare *al massimo* 2 *km* a piedi

che si ottiene con

```
camminare \emph{al massimo $2$ km} a piedi
```

Lesina (1986) fornisce alcune regole per scrivere i numeri nel testo. Queste regole possono essere riassunte in (Gregorio, 2005):

- I *numeri inferiori a venti*[11] vanno in lettere. Per esempio: il pignone ha undici denti.
- Se un numero rappresenta una *misura precisa*, va in cifre. Per esempio: il lato è lungo 1,5 m.
- Se è una *misura indicativa* va in lettere, purché il numero non sia troppo grande (quando è scritto). In questo caso è facile arrotondare, tanto non interessa se un posto è lontano un chilometro o 1123 m (tranne alcuni casi specifici). Quando si riportano numeri approssimati anche le unità di misura vanno scritte per esteso. Si scrive dunque "tra circa venti metri" e non "tra circa 20 m".
- Se il numero appare accoppiato a un altro che va scritto in cifre, anche questo va in cifre. Per esempio va scritto: ci sono 10 soggetti del primo tipo e 154 del secondo.
- Mai cominciare un periodo con un numero scritto in cifre.

I numeri romani, sebbene molto rari in testi scientifici, devono essere sempre scritti in carattere dritto, allo stesso modo dei numeri arabi. Si deve quindi scrivere XVI invece di *XVI*.

Numeri lunghi andrebbero scritti con un piccolo spazio ogni blocco di tre cifre:

125 362
0,398 276
12,345 4.

Lo spazio è ottenibile in LaTeX con il comando `\,`. Per fortuna questo può essere fatto in automatico con il comando `\np` del pacchetto numprint. Tra le funzioni di tale pacchetto si ricordano:

- L'inserimento di un *separatore ogni tre cifre per le migliaia*. Il separatore usato (`,` o `.` o `\,` o `~`) dipende dalla lingua in cui si sta scrivendo (definita dal pacchetto babel). Per l'italiano viene utilizzato di default lo spazio sottile `\,`. Per esempio in italiano `\np{15000000}` produce 15 000 000.
- La sostituzione del *segno decimale*. Indipendentemente da quale segno decimale sia usato nel codice LaTeX, numprint rappresenta i numeri con il separatore adatto per la lingua utilizzata: `.` per l'inglese e `,` per l'italiano e le altre lingue europee. Se, per esempio, si scrive in italiano `\np{3.15}` viene rappresentato come 3,15.
- L'*approssimazione dei numeri decimali* al numero di cifre significative impostato. Se per esempio si imposta l'uso di 3 cifre significative, `\np{2.742647826672}` produce 2,743.
- La conversione dei caratteri `E`, `e`, `D`, `d` nel *formato esponenziale*.[12] Per esempio `\np{1.234E-4}` viene rappresentato come $1{,}234 \cdot 10^{-4}$.
- L'*aggiunta automatica di zeri* qualora necessario. Ad esempio `\np{.12}` viene rappresentato come 0,12.

5.3 Cifre significative e notazione scientifica

Le scienze sperimentali sono solite esprimere valutazioni e risultati con quantità numeriche. Certe quantità numeriche sono esatte, come ad esempio π, e, $\sqrt{2}$, $2/3$. Le quantità derivanti, in modo diretto o indiretto, da misure o stime sono invece approssimate (o più propriamente incerte). La rappresentazione utilizzata per le quantità approssimate, sia in virgola mobile che in virgola fissa, è di fondamentale importanza perché fornisce informazioni sulla precisione con cui le quantità sono note. Per esempio

$$h = 1{,}23\,\mathrm{m} \qquad \text{e} \qquad h = 1{,}230\,00\,\mathrm{m}$$

seppure simili hanno due significati completamente diversi. Mentre nel primo caso h rappresenta una lunghezza nota al meglio di ± 5 mm e, quindi, deriva da misure piuttosto grossolane, nel secondo caso h rappresenta una lunghezza nota al meglio di ± 5 µm e, quindi, deriva da misure molto precise.

Prestare attenzione al numero di cifre significative con cui si rappresenta una quantità numerica è dunque molto importante, talvolta più importante delle cifre stesse.[13] Purtroppo spesso si rappresentano quantità numeriche con un numero eccessivo di cifre significative, in particolare quando si riportano i risultati di qualche calcolo.

La notazione scientifica, che è un modo di esprimere i numeri approssimati utilizzando le potenze intere di dieci, permette in primo luogo di utilizzare una notazione più concisa e in secondo luogo di identificare immediatamente il numero di cifre significative e quindi se ne consiglia l'impiego.

11. Altri autori dicono numeri al massimo di una cifra.

12. Questa funzione è particolarmente utile per riportare dati prodotti da software come FORTRAN, MATLAB, ecc.

13. Per esempio, a seconda del contesto, può esserci più differenza tra 2,1 e 2,150 2 che tra 2,1 e 2,9.

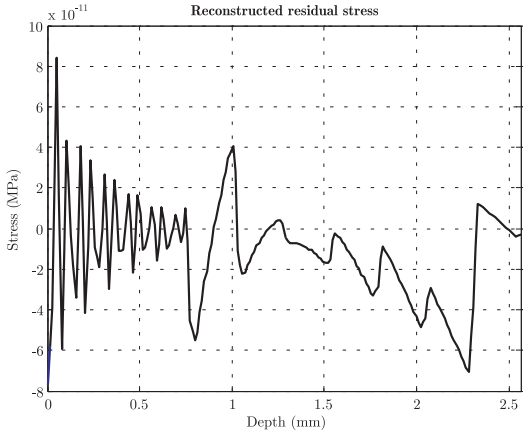


Figura 2: Unità di misura fra parentesi tonde.

5.4 Unità di misura

Le unità di misura (Bureau International des Poids et Mesures, 2006; ISO 1000, 1982) coinvolgono spesso più caratteri (kg, Pa, mm, MN, rad, ecc.) e devono seguire alcune regole:

- devono essere scritte in carattere dritto;
- non devono essere separate dalla quantità numerica a cui si riferiscono da un'interruzione di riga;
- non devono essere racchiuse fra parentesi quadre.

Le parentesi quadre non devono essere usate con le unità di misura perché in metrologia significano "unità di misura di" (Beccari, 2007a). Per esempio è lecito scrivere

$$a = 25 \text{ m/s}^2 \qquad \text{e} \qquad [a] = \text{m/s}^2$$

ma non

$$a = 25 \text{ [m/s}^2\text{]}.$$

Le parentesi tonde servono invece nelle tabelle e nei grafici quando l'unità di misura non compare vicino al valore numerico a cui si riferisce, ma al simbolo della corrispondente grandezza fisica. Si vedano per esempio la Tab. 1 e la Fig. 2.

A questi errori tipografici si aggiungono errori concettuali. Il più frequente consiste nell'indicare il secondo (unità di tempo) con sec (che peraltro significa secante in trigonometria) anziché con il simbolo s. Una buona rassegna sull'uso delle unità di misura è riportata nella guida dell'ottimo pacchetto Slunits (Heldoorn e Wright, 2007). Questo pacchetto permette di evitare di formattare a mano le unità di misura. Per esempio

```
\unit{32,1}{\micro\metre}
```

produce

χ	C_1 (N/rad)	C_2 (N/rad)	a_1 (m)
0	73 000,0	90 000,0	0,912
0,05	70 899,3	89 143,7	0,912
0,10	68 565,1	88 850,5	0,912

Tabella 1: Unità di misura fra parentesi tonde.

32,1 µm

Il pacchetto, tra l'altro, permette di scrivere le unità di misura in carattere dritto e di inserire uno spazio insecabile tra il numero e l'unità di misura. Se si utilizza l'opzione italian di babel, è necessario usare l'opzione italian anche su Slunits; se si usa il pacchetto amssymb, è necessario usare l'opzione squaren su Slunits perché il comando square viene già definito dal precedente.

Assolutamente sconsigliato è l'uso di abbreviazioni o acronimi come "mln" per "milioni" e "mld" per "miliardi" il cui uso è ormai una prassi in campo giornalistico ma si sta diffondendo anche in testi scientifici (Beccari, 2007a).

Un altro errore comune è utilizzare la k maiuscola per indicare il chilo. K, nell'ambito delle unità di misura, ha solamente il significato di kelvin,[14] mentre chilo si scrive con la k minuscola. Si scrive dunque kg, kB, ecc. per indicare chilogrammo, chilo byte ecc. Si ricorda che le unità si misura del sistema internazionale (SI) sono maiuscole quando costituiscono le abbreviazioni del nome di illustri scienziati (ad es. N, W, Pa, J) minuscole negli altri casi (ad es. m, lm, cd). Unica eccezione è il litro che in Europa può essere scritto maiuscolo o minuscolo, negli USA obbligatoriamente maiuscolo (Bureau International des Poids et Mesures, 2006; Ceraolo, 2007). Quando scritte per esteso, le unità di misura devono essere scritte con iniziale minuscola (Ceraolo, 2007; NIST, 2001).

6 Ambiguità

Indicare lo zero con il simbolo ∅ nella scrittura a mano è un'inutile pignoleria. Infatti, il contesto indica chiaramente che non si tratta di una "o" maiuscola. L'uso di ∅ è invece indispensabile nella stesura a mano di programmi di calcolo, proprio perché sarebbe difficile capire se si intende "O" oppure "0".

14. Sono ormai molti anni (dal 1968) che in ambito internazionale è stata abolita la parola "grado" per riferirsi ai kelvin. La ragione è che i "gradi" sono in qualche modo misure convenzionali di grandezze fisiche, ed, in particolare, possono anche essere espresse da numeri negativi; le unità prive della parola grado sono più legate alla fisica reale della cosa che si intende misurare e hanno valore zero in corrispondenza dell'azzeramento della grandezza (Bureau International des Poids et Mesures, 1969; Ceraolo, 2007; Terrien, 1968).

Nella composizione elettronica, invece, è fortemente consigliato indicare i litri con la lettera elle maiuscola (L) anziché minuscola (l). Sebbene le norme (Bureau International des Poids et Mesures, 1979; Giacomo, 1980; ISO 1000, 1982) prevedano l'uso di entrambe le forme,[15] quella minuscola rischia di confondersi con il numero uno e con la i maiuscola. Si noti la somiglianza tra i seguenti: l, 1, I.

7 Conclusioni

Queste note sono nate quasi per gioco, con l'intento di contribuire a ridurre, ci si passi il termine, l'analfabetismo con cui vengono talvolta scritte le formule, magari assai interessanti e profonde. Da una prima breve bozza si è poi passati gradatamente a considerare anche altri aspetti fino a ottenere un articolo serio e lungo ma, speriamo, non serioso e noioso. Tuttavia, mai sottovalutare l'importanza della chiarezza e dell'efficacia nella comunicazione, soprattutto scientifica.

8 Ringraziamenti

Desideriamo ringraziare Valeria Angeli, Luca Baldini, Riccardo Bartolozzi, Marco Beghini, Massimo Ceraolo, Andrea Domenici, Massimiliano Dominici, Beatrice Lazzerini, Caterina Mori, Pier Angelo Mori, Sebastiano Schillaci ed Antonio Sponziello per gli utili suggerimenti forniti nelle fasi di stesura e revisione di questo articolo. Si ringraziano inoltre Claudio Beccari ed Enrico Gregorio per le illuminanti discussioni svolte sul forum di GUIT (`http://www.guit.sssup.it/phpbb/index.php`) su alcuni temi trattati in questo articolo.

Riferimenti bibliografici

American Mathematical Society (2002a). *$\mathcal{AMS}$-LaTeX User's Guide*. URL `ftp://ftp.ams.org/pub/tex/doc/amsmath/amsldoc.pdf`.

— (2002b). «User's guide for the amsmath package». URL `ftp://ftp.ams.org/pub/tex/doc/amsmath/amsldoc.pdf`.

Baudoin, M. (1997). *Apprends LaTeX!* URL `http://tex.loria.fr/general/apprends-latex.pdf`.

Beccari, C. (1997). «Typesetting mathematics for science and technology according to ISO31XI». *TUGboat*, **18** (1), pp. 39–48.

— (2007a). Comunicazione privata.

— (2007b). *Introduzione all'arte della composizione tipografica*. URL `http://www.guit.sssup.it/downloads/GuidaGuIT.pdf`.

Bellacchi, G. (1894). *Introduzione storica alle funzioni ellittiche*. Barbera, Firenze.

Bertini, E. (1907). *Introduzione alla geometria proiettiva degli iperspazi*. E. Spoerri, Pisa.

Bianchi, L. (1899). *Lezioni sulla teoria dei gruppi di sostituzioni*. E. Spoerri, Pisa.

Bonola, R. (1906). *La geometria non-euclidea*. Zanichelli, Bologna.

Burali-Forti, C. (1904). *Lezioni di geometria metrico-proiettiva*. Fratelli Bocca, Torino.

Bureau International des Poids et Mesures (1969). «Resolution 4». In *Comptes Rendus de la 13^e Conférence Générale des Poids et Mesures (1967/68)*. p. 104. URL `http://www.bipm.org/en/CGPM/db/13/4/`.

— (1979). «Resolution 6». In *Comptes Rendus de la 16^e Conférence Générale des Poids et Mesures*. p. 101. URL `http://www.bipm.org/en/CGPM/db/16/6/`.

— (2006). *The International System of Units (SI)*. Pavillon de Breteuil, Sèvres, France, 8^a edizione. URL `http://www.bipm.org/en/si/si_brochure/`.

Caprilli, A. (1912). *Nuove formole d'integrazione*. Belforte, Livorno.

Ceraolo, M. (2007). Comunicazione privata.

Cesaro, E. (1894). *Corso di analisi algebrica con introduzione al calcolo infinitesimale*. Bocca, Torino.

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

Dini, U. (1878). *Fondamenti per la teorica delle funzioni di variabili reali*. Nistri, Pisa.

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

Fubini, G. (1908). *Introduzione alla teoria dei gruppi discontinui e delle funzioni automorfe*. E. Spoerri, Pisa.

Gazzaniga, P. (1903). *Gli elementi della teoria dei numeri*. Drucker, Verona-Padova.

Giacomo, P. (1980). *Metrologia*, **16** (1), pp. 55–61.

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

15. Le norme ISO prevedono l'uso dei litri anche se l'unità di misura del volume nel sistema internazionale (Bureau International des Poids et Mesures, 2006) è il m^3 con i suoi multipli e sottomultipli.

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

— (2000). *Math into LaTeX*. Birkhauser.

Gregorio, E. (2005). Discussione sul forum di GuIt. URL http://www.guit.sssup.it/phpbb/viewtopic.php?t=1220.

— (2007a). Comunicazione privata.

— (2007b). *LaTeX: Breve guida ai pacchetti di uso più comune*. URL 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.

Heldoorn, M. e Wright, J. (2007). *The SIunits package: support for the International System of Units*. URL http://tug.ctan.org/cgi-bin/getFile.py?fn=/macros/latex/contrib/SIunits/SIunits.pdf.

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

ISO 1000 (1982). «SI units and recommendations for the use of their multiples and of certain other units». In *ISO Standards Handbook N. 2*, International Organization for Standardization, Geneva. 2ª edizione.

ISO 31/11 (1982). «Mathematical sign and symbols for use in physical sciences and technology». In *ISO Standards Handbook N. 2*, International Organization for Standardization, Geneva. 2ª edizione.

ISO 31/12 (1982). «Dimensionless parameters». In *ISO Standards Handbook N. 2*, International Organization for Standardization, Geneva. 2ª edizione.

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.

Lesina, R. (1986). *Il manuale di stile*. Zanichelli, Bologna, 2ª edizione.

NIST (2001). *NIST Special Publication 330: The International System of Units (SI)*. URL http://physics.nist.gov/Document/sp330.pdf.

Oetiker, T., Partl, H., Hyna, I. e Schlegl, E. (2007). *The (Not So) Short Introduction to LaTeX 2ε*. URL http://www.ctan.org/get/info/lshort/english/lshort.pdf.

Peano, G. (1887). *Applicazioni geometriche del calcolo infinitesimale*. Bocca, Torino.

Sacchi, G. (1854). *Sulla geometria analitica delle linee piane*. Bizzoni, Pavia.

Strang, G. (2005). *Linear Algebra and Its Applications*. Brooks Cole, 4ª edizione.

Terrien, J. (1968). «News from the international bureau of weights and measures». *Metrologia*, **4** (1), p. 43.

The Tutorial Team (2000). *On-line Tutorial on LaTeX*. Indian TeX Users Group. URL http://www.tug.org.in/tutorials.html.

Veronese, G. (1891). *Fondamenti di geometria a più dimensioni e a più specie di unità rettilinee esposti in forma elementare*. Tipografia del Seminario, Padova.

Vivanti, G. (1916). *Elementi della teoria delle equazioni integrali lineari*. Hoepli, Milano.

Voss, H. (2007). *Mathmode*. URL www.ctan.org/tex-archive/info/math/voss/mathmode/Mathmode.pdf.

Wikipedia (a). URL http://en.wikipedia.org/wiki/Typographical_conventions_in_mathematical_formulae.

— (b). URL http://en.wikipedia.org/wiki/ISO_31.

— (c). URL http://en.wikipedia.org/wiki/Mathematical_notation.

— (d). URL http://en.wikipedia.org/wiki/Mathematical_symbols.

▷ Massimo Guiggiani
Dipartimento di
Ingegneria Meccanica,
Nucleare e della Produzione
Università di Pisa
Pisa, Italia
guiggiani at ing dot unipi dot it

▷ Lapo F. Mori
Dipartimento di
Ingegneria Meccanica,
Nucleare e della Produzione
Università di Pisa
Pisa, Italia
lapo dot mori at ing dot unipi dot it

Introduzione a XeTeX

Massimiliano Dominici

Sommario

Unicode e *smart font technologies* rappresentano l'attuale standard *de facto* nella tipografia digitale. Questo articolo mostra come è possibile, con XeTeX, integrarli in un sistema di composizione tipografico basato su TeX.

Abstract

Unicode and *smart font technologies* are the current *de facto* standard in digital typography. This article should explain how they can be incorporated, with XeTeX, in a TeX-based typesetting system.

1 Introduzione

Per un sistema di composizione tipografica di alta qualità, quale è TeX, è necessario poter sfruttare appieno quelle tecnologie che si sono affermate come standard di fatto nel mondo della tipografia digitale. Nel corso della sua esistenza TeX ha dimostrato di sapersi adattare alle innovazioni che di volta in volta si sono imposte nell'uso comune. Nato in un contesto monolinguistico anglosassone e pensato, quindi, per risolvere esigenze legate a quel contesto; inizialmente in grado di ricevere solo testo codificato a 7-bit (ASCII); munito di un proprio formato particolare per il tracciamento dei caratteri (METAFONT); con il passare degli anni si è evoluto in modo da adattarsi ad un contesto multilinguistico, accettare testo codificato a 8-bit e usare quei formati di font (Type1 e TrueType) divenuti di uso comune.

Questo adattamento è avvenuto senza stravolgere l'implementazione originaria; così, ad esempio, l'uso di font Type1 e TrueType è possibile a patto che le metriche di tali font siano state preventivamente trasformate nel formato comprensibile da TeX (tfm, *TeX Font Metrics*), mentre la scelta della codifica del documento finale, o l'implementazione di funzioni tipografiche avanzate (quali le legature o i numeri "minuscoli") è effettuata per mezzo di "font virtuali" (vf, *Virtual Fonts*). A livello di immissione del testo, i vari tipi di codifica sono gestiti traducendo l'input nel formato interno di TeX.

Questa filosofia, se ha garantito un adattamento indolore e un alto grado di compatibilità all'indietro, ha però i suoi limiti e le sue controindicazioni. Limiti e controindicazioni che sono più evidenti oggi che gli standard di fatto nel mondo della tipografia digitale si chiamano Unicode e *smart font technologies*.

Unicode[1] è l'implementazione ufficiale dello standard ISO/IEC 10646 per la codifica dei caratteri appartenenti ai sistemi di scrittura di tutto il mondo. Ad ogni carattere "astratto"[2] Unicode attribuisce un codice numerico e un nome standard, evitando così il ricorso ad una pletora di codifiche parziali, spesso in conflitto l'una con l'altra (ad esempio "Windows cp1252" and "ISO/IEC 8859-1" che pretendono entrambe di rappresentare una codifica per "Latin 1"). Il suo uso facilita l'interscambio di dati fra diverse applicazioni fornendo uno standard condiviso per la codifica di ogni carattere.

Gli *smart font* ("font intelligenti"), sono quei formati di font in grado di istruire l'applicazione che vi accede (e che è in grado di interpretarne le informazioni) su come applicare al testo in entrata, in maniera automatica, una serie di funzioni tipografiche avanzate. Queste funzioni avanzate possono andare dalla semplice sostituzione di uno o più "glifi" con legature o forme storiche, in maniera sensibile al contesto oppure no, fino a complesse operazioni di posizionamento, tracciamento e riordino dei glifi rispetto a quelli che li precedono o seguono, operazioni necessarie per visualizzare correttamente alcuni sistemi di scrittura non occidentali.

I più comuni esempi di *smart font technologies* sono Apple Advanced Typograhy (AAT), sviluppato da Apple, OpenType, sviluppato congiuntamente da Adobe e Microsoft, e Graphite, sviluppato da SIL International.

Nonostante la tecnologia realizzata da Apple sia stata l'antesignana delle *smart font technologies* e rimanga tuttora superiore in alcuni settori, come quello dei *complex scripts*, il supporto di Adobe e Microsoft, e alcuni innegabili fattori di superiorità come quello di poter immagazzinare indifferentemente i contorni dei glifi sia nel formato tipico dei TrueType, che in quello, preferito dai tipografi professionali, dei Type1[3] hanno decretato il successo del formato OpenType. A partire dal 1999, con la decisione della stessa Adobe di convertire l'intera propria collezione di font nel nuovo formato OpenType, questo ha progressivamente soppian-

1. `http://www.unicode.org`
2. E non alla sua "presentazione": "A" e "ᴀ" sono due caratteri distinti per Unicode; non così "A" e "𝖠": entrambi rappresentano il carattere "U+0041 latin capital letter a".
3. In realtà, i dati relativi al contorno dei glifi, nel caso di OpenType di tipo PostScript, sono contenuti nel formato CFF (*Compact Font Format*), che è, in sostanza, una forma compressa del Type1.

tato sia il Type1 che il TrueType, affermandosi come nuovo standard di fatto.[4]

Sia l'uso di Unicode come codifica del testo in ingresso, sia l'uso e l'inclusione di font OpenType o AAT è possibile con TEX, ma non in maniera nativa. La codifica Unicode in ingresso deve essere "tradotta" da appositi moduli (ne esistono per i principali sistemi di composizione basati su TEX: LATEX e ConTEXt), mentre l'accesso ai font deve essere effettuato tramite TFM e VF, rendendo molto complicata la loro installazione e perdendo per strada alcune caratteristiche (ad esempio l'unitarietà del font: TEX è costretto, dalla limitazione interna dei TFM ad un massimo di 256 caratteri, a considerare ogni font virtuale come un font separato, anche se i singoli glifi sono immagazzinati nello stesso contenitore).

A questi inconvenienti si propone di porre rimedio XeTEX, una reimplementazione di TEX sviluppata da Johnatan Kew, della SIL International, con lo scopo di facilitare la composizione di documenti in sistemi di scrittura non occidentali, e originariamente disponibile solo per sistemi Macintosh.[5] Le innovazioni che XeTEX introduce si concentrano essenzialmente su tre punti: a) supporto nativo per Unicode; b) accesso diretto ai font installati nel sistema operativo, senza l'intermediazione dei TFM; c) capacità di interpretare direttamente le istruzioni contenute nelle tabelle interne dei font AAT, OpenType o Graphite. Un effetto collaterale, dovuto al modo in cui vengono passate al driver di stampa le informazioni relative ai font, è che non è più possibile produrre un output in DVI, ma solo in PDF (vedi sezione 2.3).

In questo modo l'uso, da parte dell'utente, di nuovi font residenti nel sistema operativo viene reso estremamente semplice, così come estremamente semplice diventa l'accesso ad eventuali funzioni tipografiche avanzate presenti nel font stesso.

2 Nozioni di base

In questa sezione verrà esposto, in linea di principio, il funzionamento di comandi di basso livello che l'utente di solito non ha bisogno di usare nei propri documenti, ma il cui esame può essere utile per comprendere il meccanismo con cui XeTEX opera.

L'utente, viceversa, ha a disposizione, per attivare le funzionalità avanzate di XeTEX, macro di alto livello, definite in appositi moduli, per quanto riguarda LATEX, o all'interno del sistema stesso, per quanto riguarda ConTEXt. Questi costrutti di alto livello saranno passati in rassegna nella sezione 3. Per maggiori dettagli su quanto verrà eaminato in questa sezione si può consultare Kew (2007).

2.1 Codifica del testo in ingresso

XeTEX è stato progettato in modo tale da poter interpretare un flusso di dati codificato indifferentemente in UTF-8 o UTF-16 (i due sistemi di codifica Unicode comunemente usati). Non è necessario, quindi ricorrere a pacchetti, moduli aggiuntivi o comandi particolari per compilare un testo espresso in questa codifica, a differenza di quanto avviene per LATEX, dove è necessario caricare un modulo di `inputenc` o per ConTEXt, dove si deve passare al programma l'istruzione `\enableregime[utf-8]`.

Al contrario, bisogna segnalare a XeTEX se è necessario tradurre in Unicode da una codifica diversa. Le primitive `\XeTeXinputencoding` e `\XeTeXdefaultencoding` indicano a XeTEX di operare questa traduzione, rispettivamente, per un singolo file e per tutti i file che verranno letti durante la compilazione, e garantiscono così compatibilità con file preparati con sistemi tradizionali.

È il caso di anticipare qui una caratteristica che logicamente dovrebbe apparire nella sezione 2.2, in quanto si tratta di un'opzione che può essere specificata al momento della selezione di un font. Tuttavia, trattandosi di un'opzione che ha effetto sul metodo di input, è bene esaminarla in questa sezione. L'opzione in questione specifica come tradurre in codifica Unicode alcuni costrutti particolari. In pratica viene usata generalmente (`map=tex-text`) per tradurre legature caratteristiche dell'input TEX: "`--`" → "–", "`!‘`" → "¡", ecc. La traduzione viene fatta tramite un file `.tec` che è il risultato di una compilazione, a partire da un file ASCII `.map`, con TECKit, un programma sviluppato da SIL International[6] per creare, appunto, tabelle di conversione da codifiche obsolete verso Unicode. Questo meccanismo può essere usato anche per "effetti speciali", come la traslitterazione in alfabeto latino di un testo in cirillico.

2.2 Accesso ai font

Per poter accedere ai font residenti nel sistema e poter sfruttare le tabelle interne degli *smart font*, la primitiva `\font` ha dovuto essere reimplementata e la sua sintassi estesa in modo da accettare argomenti opzionali.

Mentre la sintassi originale di `\font` è essenzialmente la seguente (si veda Knuth, 1984, p. 276):

`\font`⟨*comando*⟩`=`⟨*nome font*⟩⟨*dimensioni*⟩

dove ⟨*dimensioni*⟩ può indicare sia un corpo (come in `at 12pt`) o un fattore di scala (come in `scaled 500`), nella nuova sintassi estesa è possibile

4. Non tutti, però, condividono il giudizio entusiastico largamente diffuso sui muovi *smart font*. Per un autorevole parere contrario, si veda lo "sfogo" di Luc Devroye in `http://cg.scs.carleton.ca/~luc/opentyperant.html`.

5. Attualmente esistono versioni sia per Linux che per Windows, incluse nelle maggiori distribuzioni come TEX Live e MiKTEX.

6. Il programma è disponibile, sotto forma di sorgenti compilabili, dall'archivio delle versioni di XeTEX.

specificare una serie di opzioni relative alle caratteristiche generali di un font e a quali delle sue funzioni tipografiche avanzate si vuole eventualmente far ricorso:

```
\font⟨comando⟩=⟨nome font⟩/⟨opzioni⟩:
    ⟨caratteristiche avanzate⟩⟨dimensioni⟩
```

Innanzi tutto ⟨*nome font*⟩ accetta una sintassi leggermente diversa dall'omonimo argomento del comando tradizionale. Nelle sue prime implementazioni su sistemi Macintosh, XeTeX era in grado di accedere *solo* ai font installati nel sistema operativo. ⟨*nome font*⟩ doveva quindi specificare il nome del font, così come risulta al sistema operativo (ad esempio: "Times Roman"). Nelle implementazioni successive, e solo nel caso in cui si usi `xdvipdfmx`[7] per generare il PDF finale, la sintassi è stata estesa per dar modo a XeTeX di accedere ai font ovunque si trovino. Racchiudendo il nome del font tra parentesi quadrate è possibile adesso indicare a XeTeX di cercarlo all'interno della TDS,[8] oppure, sempre all'interno delle parentesi quadrate, è possibile specificare l'indirizzo preciso del font.

Tramite ⟨*opzioni*⟩ è possibile specificare alcune caratteristiche generali del font: variante (`/I`; corsivo, `/B`: neretto, ecc.), taglia ottica (`/S=`x, dove x indica il corpo),[9] specifica *smart font technology* da utilizzare (`/ATSUI` per AAT, `/ICU` per OpenType e `/GR` per Graphite).[10] Ogni ⟨*opzione*⟩, identificata da uno specifico 'tag', deve essere preceduta dal simbolo "`/`".

Per attivare (o disattivare) specifiche funzioni tipografiche avanzate, l'utente può passare a XeTeX una lista di elementi preceduti dal simbolo "`:`". La sintassi dei singoli elementi dipende dal tipo di *smart font technology* implementata nel font. Nel caso di font AAT, ad ogni funzione avanzata viene assegnato un nome dallo sviluppatore, cosicché l'utente dovrà specificare una coppia "chiave = valore" (per esempio: "Letter Case = Small Caps"). Nel caso di font OpenType, invece, le funzioni avanzate sono identificate da codici convenzionali di quattro lettere ('tag'): all'utente basta quindi specificare uno di questi codici preceduto dal simbolo "`+`", se vuole attivare la funzione, o dal simbolo "`-`", se vuole disattivarla. Gli OpenType prevedono anche due funzioni speciali: `script` e `language`, tramite le quali è possibile selezionare il supporto per un sistema di scrittura o per una lingua particolari. In questo caso è necessario far seguire al nome della funzione il 'tag' relativo al sistema di scrittura o alla lingua che si desiderano attivare, con il consueto sistema "chiave = valore".

Poiché sia il ⟨*nome font*⟩ che il nome delle funzioni tipografiche avanzate in font AAT possono contenere spazi bianchi al loro interno, è consigliabile racchiudere l'intera stringa rappresentante l'argomento di `\font` (tranne ⟨*dimensioni*⟩) tra apici doppi.

Per concludere, un paio di esempi serviranno a chiarire come è possibile usare questa sintassi estesa del comando `\font`.

```
\font\hoefler="Hoefler Text/B:Letter Case%
  =Small Caps" at 12pt
```

Al comando `\hoefler` viene associata la selezione del font "Hoefler Text", installato nel sistema operativo, nella variante "neretto", con sostituzione del maiuscoletto alle lettere minuscole, in corpo 12.[11]

```
\font\juni="[/percorso/Junicode_Regular]%
  :script=latn:language=ISL:+hlig:+hist"%
  at 11pt
```

Il comando `\juni` seleziona il font "Junicode Regular" (non installato nel sistema operativo, né nella TDS) in corpo 11, utilizzando il supporto specifico per l'alfabeto latino e la lingua islandese, e attivando forme e legature storiche.

2.3 Formato dell'output

Il formato ultimo dei documenti compilati con XeTeX è un PDF. Tuttavia XeTeX usa, nel corso del processo di compilazione, uno speciale formato intermedio (XDV, "eXtended DVI") allo scopo di trasmettere al driver di stampa[12] le informazioni necessarie ad interpretare tutto ciò che riguarda le caratteristiche dei font da includere nel documento finale. Questo formato, che non può essere interpretato dagli attuali visualizzatori di DVI, né correttamente disassemblato da programmi come `dvitype` o `dv2dt`, viene successivamente "distillato" nel PDF finale. Tutto ciò avviene in maniera del tutto automatica e trasparente per l'utente, che non deve compiere azioni aggiuntive; anzi, se vuole disattivare, nelle compilazioni intermedie, la generazione del PDF (che è la parte più dispendiosa

7. Si veda la sezione 2.3.

8. *TEX Directory Structure*, è l'insieme di cartelle in cui è distribuita un'installazione standard di TEX.

9. Alcuni font sono distribuiti in versioni differenti per specifiche gamme di corpi, in modo da evitare variazioni nel 'colore' della pagina (l'idea, ripresa dalla tipografia tradizionale, in cui un tipo di carattere ha un disegno leggermente diverso per ogni corpo, è stata sfruttata anche da Knuth nel disegnare il font *Computer Modern*). Ciascuna di queste versioni ha un suffisso proprio (ad esempio *Caption* o *Subhead*) ed è associata internamente ad una gamma di corpi. Le applicazioni in grado di avvalersi delle *smart font technologies*, tra cui XeTeX, scelgono automaticamente la versione più appropriata per il corpo selezionato, cosicché non è necessario specificare tale opzione se non per sovrascrivere il comportamento di default.

10. Di solito non è necessario specificare quest'ultima opzione, in quanto XeTeX è in grado di riconoscere da sé il formato del font. Tuttavia, alcuni font possono contenere tabelle in più di un formato e l'utente può volerne attivare uno in particolare, diverso da quello di default.

11. L'esempio è tratto da Robertson (2007a).

12. Il termine "driver di stampa" non è forse del tutto corretto, in quanto l'output finale è un file.

del processo, in termini di tempo), deve passare al compilatore `xetex` l'opzione `--no-pdf`.

XeTEX è in grado di usare due diversi driver di stampa, per ottenere il PDF finale. `xdv2pdf` è quello tradizionale, implementato nella versione originaria di XeTEX, ed è quello ancora abilitato di default su computer Macintosh. Ormai, però, le versioni per Linux e Windows utilizzano `xdvipdfmx`, una versione appositamente estesa di `dvipdfmx`, che, a sua volta, era la reimplementazione di `dvipdfm`. Tra i due driver ci sono alcune differenze, la più notevole è il supporto per i font non installati nel sistema operativo (vedi sezione 2.2), presente in `xdvipdfmx` e assente in `xdv2pdf`. Lo stesso vale per la possibilità di interpretare gli `\specials` inseriti da PSTricks (vedi sezione 4.2.3).

3 Uso con LaTeX e ConTeXt

Per quanto l'uso di TEX nella sua versione più "pura" (Plain TEX) sia ancora diffuso, tuttavia è certamente maggiore il numero di coloro che usano TEX attraverso "macro pacchetti" come LaTeX e ConTeXt. Questi utenti sono abituati ad usare un linguaggio di *markup* più astratto rispetto alle istruzioni di formattazione di basso livello tipiche di TEX. Per essi è poco conveniente dover ricorrere ad espressioni come quelle esaminate nella sezione 2.2, ogni volta che è necessario selezionare un determinato tipo di carattere o una sua specifica caratteristica. Si aspettano che ciò possa essere fatto per mezzo di dichiarazioni nel preambolo, per istruzioni generali, o per mezzo di specifici comandi di *markup* da applicare a singole porzioni di testo. Questa aspettativa è stata soddisfatta sotto forma di specifiche estensioni ("pacchetti") per quanto riguarda LaTeX, o di nuove funzionalità aggiunte al nocciolo, per quanto riguarda ConTeXt.[13]

3.1 LaTeX

3.1.1 Fontspec

La principale estensione che mette l'utente di LaTeX in grado di comporre i propri documenti con XeTEX è fontspec. Questo pacchetto, scritto da Will Robertson (Robertson, 2007b), mette a disposizione dell'utente una serie di istruzioni per una coerente selezione dei font e delle funzioni tipografiche avanzate ad essi associate. I comandi principali definiti nel pacchetto sono:

```
\setmainfont[⟨opzioni⟩]{⟨nome font⟩}
\setsansfont[⟨opzioni⟩]{⟨nome font⟩}
\setmonofont[⟨opzioni⟩]{⟨nome font⟩}
\fontspec[⟨opzioni⟩]{⟨nome font⟩}
```

Hanno tutti la medesima sintassi, per cui, nel seguito, gli esempi riguarderanno solo `\fontspec`,

13. Questo diverso tipo di approccio, modulare nel caso di LaTeX e monolitico nel caso di ConTeXt, rappresenta una delle differenze basilari tra i due programmi.

```
\fontspec[ExternalLocation,\
  LetterSpace=8,WordSpace=2,\
  Color=AA4444]{texgyreschola-regular}
\begin{center}
{\Large I PROMESSI SPOSI}\\[1ex]
ALESSANDRO MANZONI
\end{center}
```

I PROMESSI SPOSI

ALESSANDRO MANZONI

Figura 1: Maiuscolo spaziato

che è l'istruzione da utilizzare in mezzo al documento, mentre gli altri tre comandi riguardano la dichiarazione, nel preambolo, del font principale e del font senza grazie o a spaziatura fissa che eventualmente lo accompagnano.

L'argomento ⟨*nome font*⟩ può essere sia il nome interno del font ("Times Roman", "Linux Libertine O", ecc.), nel caso che questo sia installato nel sistema operativo, oppure il nome del file che contiene il font, se questo non è installato. In questo secondo caso, che come abbiamo visto nella sezione 2.3 funziona solo se si usa `xdvipdfmx` come driver di stampa PDF, è necessario specificare tra le ⟨*opzioni*⟩ `ExternalLocation`, eventualmente seguito dal percorso, se il font non si trova nella TDS.

Le ⟨*opzioni*⟩ si dividono sostanzialmente in due categorie: quelle che possono essere applicate a ciascun font indifferentemente (colore, spaziatura tra le lettere o tra le parole, ecc.) e quelle, invece, che dipendono da ciò che si trova (o non si trova) nelle tabelle interne al font, nel caso questo disponga di funzioni tipografiche avanzate. Naturalmente non verranno qui esaminate tutte (si rimanda il lettore alla documentazione del pacchetto per un analisi dettagliata), ma verranno mostrati solo alcuni esempi che si ritengono significativi.

Nella figura 1 è possibile osservare un esempio di maiuscolo spaziato. Questa pratica, di uso comune quando si voglia rendere esteticamente più gradevole l'aspetto di un titolo, non aveva, fino a non molto tempo fa[14] soluzioni realmente soddisfacenti. Come si vede dal codice che precede l'esempio, con fontspec è sufficiente passare ad uno dei comandi per la selezione dei font l'opzione `LetterSpace`, impostata al valore x, dove x rappresenta la dimensione dello spazio aggiunto, in frazioni percentuali del corpo del font. Per migliorare la resa visiva, è preferibile usare, contestualmente, l'opzione `WordSpace`, che moltiplica lo spazio normalmente inserito da TEX tra due parole per il valore fornito dall'utente (in questo caso '2', raddoppiando quindi uno spazio normale). In realtà `WordSpace` accetta come valore una tripla

14. Finché, cioè microtype non ha reso accessibili all'utente medio di LaTeX le funzioni di microtipografia implementate in `pdftex`. Il pacchetto soul forniva una soluzione limitata e poco robusta.

PSALM LXVIII. 20, *in fine.*

And unto God the Lord belong the issues of death (i.e. from death).

Buildings stand by the benefit of their foundations that sustain and support them, and of their buttresses that comprehend and embrace them, and of their contignations that knit and unite them. The foundations suffer them not to sink, the buttresses suffer them not to swerve, and the contignation and knitting suffers them not to cleave. The body of our building is in the former part of this verse. It is this: *He that is our God is the God of salvation*; *ad salutes*, of salvations in the plural, so it is in the original; the God that gives us spiritual and temporal salvation too.

(a) Numeri 'oldstyle'

PSALM LXVIII. 20, *in fine.*

And unto God the Lord belong the issues of death (i.e. from death).

Buildings stand by the benefit of their foundations that sustain and support them, and of their buttresses that comprehend and embrace them, and of their contignations that knit and unite them. The foundations suffer them not to sink, the buttresses suffer them not to swerve, and the contignation and knitting suffers them not to cleave. The body of our building is in the former part of this verse. It is this: *He that is our God is the God of salvation*; *ad salutes*, of salvations in the plural, so it is in the original; the God that gives us spiritual and temporal salvation too.

(b) Legature non comuni

PSALM LXVIII. 20, *in fine.*

And unto God the Lord belong the iſſueſ of death (i.e. from death).

Buildingſ ſtand by the benefit of their foundationſ that ſuſtain and ſupport them, and of their buttreſſeſ that comprehend and embrace them, and of their contignationſ that knit and unite them. The foundationſ ſuffer them not to ſink, the buttreſſeſ ſuffer them not to ſwerve, and the contignation and knitting ſufferſ them not to cleave. The body of our building iſ in the former part of thiſ verſe. It iſ thiſ: *He that iſ our God iſ the God of ſalvation*; *ad ſaluteſ*, of ſalvationſ in the plural, ſo it iſ in the original; the God that giveſ uſ ſpiritual and temporal ſalvation too.

(c) Forme storiche

PSALM LXVIII. 20, *in fine.*

And unto God the Lord belong the issues of death (i.e. from death).

Buildings stand by the benefit of their foundations that sustain and support them, and of their buttresses that comprehend and embrace them, and of their contignations that knit and unite them. The foundations suffer them not to sink, the buttresses suffer them not to swerve, and the contignation and knitting suffers them not to cleave. The body of our building is in the former part of this verse. It is this: *He that is our God is the God of salvation*; *ad salutes*, of salvations in the plural, so it is in the original; the God that gives us spiritual and temporal salvation too.

(d) Maiuscoletto

Figura 2: Funzioni tipografiche avanzate per il font *Junicode*

di numeri, separati da virgola: i tre valori verranno applicati, come fattore di scala, rispettivamente, al valore nominale dello spazio, al suo 'allungamento' massimo (*stretch*) e al suo 'accorciamento' massimo (*shrink*). Specificare un solo valore significa assegnare lo stesso a ciascuno dei tre elementi della tripla. Abbiamo infine aggiunto un tocco di colore (che nella versione stampata apparirà in una diversa tonalità di grigio) con l'opzione `Color`, seguita da una tripla di valori RGB (ogni valore è formato da due cifre espresse nel sistema esadecimale). È possibile aggiungere anche un quarto valore per indicare il grado di trasparenza.

Negli esempi seguenti (riassunti nella figura 2[15]) verrà utilizzato il font Junicode per esaminare le opzioni relative alle funzioni tipografiche avanzate contenute nelle tabelle OpenType.

Nell'esempio della figura 2a i numeri normali sono stati sostituiti da numeri 'oldstyle', tramite l'opzione `Numbers=Oldstyle`. `Numbers` accetta come valori possibili qualunque combinazione tra `Lining`, `Oldstyle`, `Monospaced` e `Proportional` ('normali', 'oldstyle', a spaziatura fissa, a spaziatura variabile). Solo per font OpenType è possibile specificare anche il valore `SlashedZero`, che sostituisce lo zero normale con uno zero barrato.

Nell'esempio seguente (figura 2b) sono state aggiunte alcune legature opzionali,[16] aggiungendo all'argomento opzionale di fontspec la stringa `Ligatures=Discretionary`. Si può notare, infatti, che la forma di `st`, nella parola 'stand' di riga 4 è costituita da un'unico carattere. Se si esaminano attentamente i quattro esempi, poi, ci si accorge che il numero romano 'LXVIII' ha una forma diversa, più compatta, negli esempi delle figure 2b e 2c: anche in questo caso si tratta, per l'appunto, di una legatura. Non è qui il caso di riportare tutti i possibili valori dell'opzione `Ligatures`; oltre ad essere molti, infatti, variano anche a seconda che si abbia a che fare con un font AAT oppure OpenType. Si rimanda, quindi, per ulteriori dettagli, al manuale di fontspec già citato (Robertson,

15. Dove sono riprodotte le prime righe di *Death's Duel*, di John Donne, come appaiono nella versione E-Text di Project Gutenberg dell'originale Donne (1959).

16. Legature comuni come `fi`, ecc. sono attivate automaticamente, ma possono eventualmente essere disattivate tramite l'opzione `Ligatures=NoCommon`.

2007b).

Nella figura 2c è stata aggiunta un'ulteriore funzione tipografica avanzata: alcuni caratteri sono stati sostituiti con forme arcaiche degli stessi (in particolare è stata sostituita `s` con `slong` e tutte le sue legature). L'opzione da aggiungere, in questo caso, è: `Style=Historic`. Anche in questo caso si rimanda al manuale di fontspec per informazioni più dettagliate, dal momento che l'opzione accetta un lungo elenco di valori predefiniti, diversi a seconda che il font incorpori la tecnologia AAT o OpenType.

Infine, nell'esempio riportato nella figura 2d, sono state sostituite le lettere dell'alfabeto minuscolo con le rispettive forme del maiuscoletto. È da notare che il testo in corsivo è rimasto immutato: infatti la versione corsiva di Junicode non possiede un maiuscoletto e di conseguenza l'opzione è stata in questo caso ignorata.[17] `Letters=Smallcaps` è quanto serve per attivare la suddetta funzione. Altri possibili valori per `Letters` sono `Uppercase`, che trasforma tutto il testo in maiuscolo, `Lowercase`, che trasforma tutto in minuscolo, e `UppercaseSmallCaps` che opera nella direzione opposta a `SmallCaps`, trasformando in maiuscoletto le maiuscole. I font AAT possono anche avere una funzione `InitialCaps` per trasformare in maiuscolo tutte le lettere a inizio parola.

È possibile anche aggiungere o togliere determinate caratteristiche al font in uso al momento, usando il comando `\addfontfeature`, il cui unico argomento accetta la stessa sintassi dell'argomento opzionale di `\fontspec`. Inoltre, con `\newfontfamily`⟨*comando*⟩`[`⟨*opzioni*⟩`]{`⟨*font*⟩`}`, o con `\newfontface`, che accetta la stessa sintassi, è possibile creare degli appositi comandi per la selezione rapida e coerente di caratteri per contesti specifici.

Infine fontspec fornisce una serie di comandi per definire nuove funzioni tipografiche avanzate che non sono coperte dal pacchetto stesso, pur essendo presenti nel font. `\newAATfeature`, `\newICUfeature` e `\newfontfeature` servono, appunto, a questo scopo.

3.1.2 Polyglossia

XeTeX è stato ideato per lavorare con font OpenType, AAT o Graphite, in codifica Unicode. Questi font dispongono di solito di un insieme di caratteri che copre numerosi linguaggi e sistemi di scrittura. È quindi naturale considerarlo un programma particolarmente adatto ad operare in un contesto multilinguistico. Peccato, quindi, che esistano profondi conflitti tra XeTeX e il pacchetto babel, che è il pilastro portante del supporto multilinguistico per LaTeX. Un conflitto profondo a tal punto da rendere inutilizzabile l'accoppiata XeTeX+babel per ambiti linguistici quali il greco e l'ebraico. Ciò avviene perché, nel momento in cui fu ideato e sviluppato, babel aveva bisogno di fare supposizioni a proposito della codifica e dei font utilizzati, supposizioni che intralciano sostanzialmente il funzionamento di XeTeX.

Esistono dei modi per aggirare queste limitazioni, ma tutti portano ad un uso subottimale di babel. Ad esempio, se si vuole utilizzare il modulo di babel per il greco con XeTeX, è possibile caricarlo come non predefinito (basta che non appaia come ultimo nella lista dei moduli caricati) e usarne solo le funzionalità relative alla corretta sillabazione, racchiudendo il testo greco tra `\begin{hyphenrules}{greek}` e `\end{hyphenrules}`. In questo modo, però, si perdono, per esempio, tutti i nomi predefiniti per i vari elementi del testo (Capitolo, Figura, Bibliografia, ecc.).

Per ovviare a questo inconveniente è in via di sviluppo il pacchetto polyglossia, che dovrebbe costituire l'alternativa futura a babel per XeTeX e, eventualmente, per gli altri sistemi di composizione derivati da TeX e basati su Unicode (ad esempio LuaTeX).

Il pacchetto è ancora in una fase sperimentale e non si trova su CTAN, ma è disponibile dall'archivio dei sorgenti di XeTeX.[18] L'attuale versione sembra comunque funzionare già abbastanza bene. Non esistendo una documentazione del pacchetto, bisogna dedurre la sua interfaccia utente dagli esempi allegati.

Dopo aver caricato il pacchetto, i vari moduli vengono caricati nel preambolo con il comando `\setdefaultlanguage`, nel caso della lingua principale, o `\setotherlanguage`. Entrambi i comandi, oltre all'argomento obbligatorio con il quale va indicata la lingua da attivare, specificandone il modulo, prevedono anche un argomento opzionale, sotto forma di una lista di elementi "chiave = valore". Ad esempio tramite l'opzione `variant=ancient`, passata al modulo `greek`, è possibile attivare il supporto per il greco antico politonico. Ogni volta che si vorrà attivare, nel testo, una lingua differente da quella predefinita, basterà racchiudere il testo in questione tra `\begin{`⟨*lingua*⟩`}` e `\end{`⟨*lingua*⟩`}`, oppure inserire l'istruzione `\selectlanguage{`⟨*lingua*⟩`}`

Nelle figure 3[19] e 4[20] sono riportati due esempi corredati del relativo codice. Nel preambolo sono state inserite le seguenti dichiarazioni:

```
\usepackage{polyglossia}
\setdefaultlanguage{italian}
```

17. Ma di questa mancata applicazione rimane traccia nel file di log, sotto forma di avviso da parte del pacchetto fontspec.

18. `http://scripts.sil.org/svn-view/xetex/TRUNK/texmf/tex/xelatex/polyglossia/`.

19. Il testo riproduce il primo capoverso delle *Storie* di Erodoto, come appare in Erodoto (2006).

20. Si tratta della prima strofa del poemetto di Vladimir Majakovskij *La nuvola in calzoni* (Majakovskij, 1989).

```
\fontspec{GFS Didot}
\begin{greek}
῾Ηροδότου ῾Αλικαρνησσέος ἱστορίης ἀπόδεξις
ἥδε, ὡς μήτε τά γενόμενα ἐξ ἀνθρώπον τῷ χρόνῳ
ἐξίτηλα ψένται μήτε ἔργα μεγάλα τε καὶ
θωυμαστὰ τὰ μὲν ῾Έλλησι, τὰ δὲ βαρβάροισι
ἀποδεχθέντα ἀκλεᾶ γένται, τά τε ἄλλα καί δι'
ἣν αἰτίην ἐπολέμησαν ἀλλήλοισι.
\end{greek}
```

῾Ηροδότου ῾Αλικαρνησσέος ἱστορίης ἀπόδεξις ἥδε, ὡς μήτε τά γενόμενα ἐξ ἀνθρώπον τῷ χρόνῳ ἐξίτηλα ψένται μήτε ἔργα μεγάλα τε καὶ θωυμαστὰ τὰ μὲν ῾Έλλησι, τὰ δὲ βαρβάροισι ἀποδεχθέντα ἀκλεᾶ γένται, τά τε ἄλλα καί δι᾽ ἣν αἰτίην ἐπολέμησαν ἀλλήλοισι.

Figura 3: Greco antico

```
\fontspec{Linux Libertine O}
\begin{russian}
\begin{verse}
Вашу мыслб\\
мечтающую на размягченном мозгу,\\
как выжиревший лакей на засалеииой кушетке,\\
буду драэнить об окровавленный сердца лоскут;\\
досыта изъиздеваюсьб нахальный и едкийю.
\end{verse}
\end{russian}
```

Вашу мыслб
мечтающую на размягченном мозгу,
как выжиревший лакей на засалеииой кушетке,
буду драэнить об окровавленный сердца лоскут;
досыта изъиздеваюсьб нахальный и едкийю.

Figura 4: Cirillico (russo)

```
\setotherlanguage{russian}
\setotherlanguage[variant=ancient]{greek}
```

I font utilizzati sono, nel primo esempio, GFS Didot e, nel secondo, LinuxLibertine. Il codice, invece, è composto in UM Typewriter.

3.1.3 Altri pacchetti

Oltre a fontspec e polyglossia esistono altri pacchetti 'minori' dedicati a XeTeX. In genere si tratta o di pacchetti che l'utente non ha bisogno di caricare direttamente ma che svolgono, ad esempio, compiti di supporto per quanto riguarda la codifica (xunicode e euenc), oppure di pacchetti che offrono alcune funzionalità relative ad ambiti linguistici specifici (arabxetex, xgreek, xecyr). Gli unici due pacchetti che meritano qualche riga di descrizione, in questa sede, sono xltxtra e philokalia.

Il primo (Robertson, 2007c) comprende una serie di correzioni e di aggiunte al formato base di XeLaTeX: incorpora, ad esempio, il pacchetto fixltx2e; ridefisce il logo di TeX e di alcuni programmi derivati (LaTeX, XeTeX, XeLaTeX); ridefisce `\textsuperscript` e `\textsubscript` in modo da usare automaticamente le cifre in apice e in pedice fornite dal font; fornisce il comando `\vfrac` per comporre frazioni all'interno del testo usando le funzioni tipografiche avanzate del font; fornisce, tramite `\namedglyph{`⟨*nomeglifo*⟩`}`, un'interfaccia di alto livello alla primitiva di XeTeX che consente di recuperare un carattere tramite il suo nome; infine ridefinisce il comando `\showhyphens` in modo che funzioni con XeTeX.

philokalia (Syropoulos, 2007), invece, accompagna l'omonimo font, riproduzione di un carattere usato nella composizione di una serie di libri di devozione, assemblati nel monastero del Monte Athos in una collezione detta, appunto, 'Philokalia'.[21] Il pacchetto, che si appoggia a fontspec, xunicode e xltxtra, fornisce all'utente i comandi necessari ad usare il font Philokalia come carattere principale del documento o relativamente a singole porzioni di esso.

21. Si veda, per maggiori dettagli, `http://orthodoxwiki.org/Philokalia`.

3.2 ConTeXt

Gli utenti di ConTeXt non hanno bisogno di ricorrere a pacchetti aggiuntivi. Il supporto per le funzionalità di XeTeX è incluso nel nocciolo del programma. L'uso è semplice e ricalca la sintassi già vista per TeX puro, ovviamente adattata alle convenzioni in vigore in ConTeXt.

Per usare le funzioni tipografiche avanzate di uno *smart font* una volta tanto, è sufficiente la seguente istruzione:

```
\definedfont["'Linux Libertine O':%
   mapping=tex-text;+onum;+smcp"]
```

con la quale si comunica a ConTeXt di usare il font Linux Libertine, installato nel sistema, nella forma "maiuscoletto", corredato di numeri "oldstyle". Si noti che è stata attivata anche l'opzione per tradurre nei rispettivi codici Unicode i costrutti tipici di TeX (per esempio la legatura `--`, equivalente ad un trattino medio '–') e che le varie opzioni possono essere separate da punto e virgola invece che da due punti. Nel caso che contenga degli spazi, il nome del font deve essere racchiuso tra ulteriori apici. Questo non è necessario, come vedremo qui sotto, nel caso di dichiarazioni globali.

Il comando `\definedfont` è utile nel caso si debba ricorrere una sola volta ad una particolare istanza di un font. Se invece il font in questione deve essere impostato come carattere principale del documento, in tutte le sue forme (normale, corsivo, maiuscoletto, ecc.) è necessario che le relative dichiarazioni vengano organizzate in maniera coerente. ConTeXt adotta un approccio particolare che può essere considerato l'equivalente di un *Font Definition file* per LaTeX. Un possibile esempio di tale organizzazione è il seguente:

```
\starttypescript[serif][libertineos][uc]
```

```
\definefontsynonym[LibertineOS]
   ["Linux Libertine O%
    :mapping=tex-text:onum"]
   [encoding=uc]
\definefontsynonym[LibertineItalicOS]
   ["Linux Libertine O/I%
    :mapping=tex-text:+onum"]
   [encoding=uc]
\definefontsynonym[LibertineBoldOS]
   ["Linux Libertine O/B%
    :mapping=tex-text:+onum"]
   [encoding=uc]
\definefontsynonym[LibertineBoldItalicOS]
   ["Linux Libertine O/BI%
    :mapping=tex-text:+onum"]
   [encoding=uc]
\definefontsynonym[LibertineSCOS]
   ["Linux Libertine O%
    :mapping=tex-text:+smcp:+onum"]
   [encoding=uc]

\stoptypescript

\starttypescript[serif][libertineos][name]

\definefontsynonym[Serif]
                  [LibertineOS]
\definefontsynonym[SerifBold]
                  [LibertineBoldOS]
\definefontsynonym[SerifItalic]
                  [LibertineItalicOS]
\definefontsynonym[SerifBoldItalic]
                  [LibertineBoldItalicOS]
\definefontsynonym[SerifSlanted]
                  [LibertineItalicOS]
\definefontsynonym[SerifBoldSlanted]
                  [LibertineBoldItalicOS]
\definefontsynonym[SerifCaps]
                  [LibertineSCOS]

\stoptypescript

\starttypescript [LinuxLibertineO][uc]

\definetypeface[LinuxLibertine][rm][serif]
       [libertineos][default][encoding=uc]

\stoptypescript

\usetypescript[LinuxLibertineO][uc]
\setupbodyfont[LinuxLibertine,10pt]
```

Nella prima parte si trova la mappatura tra il livello "fisico" (il font "reale") e un primo livello astratto. Le mappature seguenti aumentano il livello di astrazione, fino a giungere al nome simbolico (`LinuxLibertine`) che verrà usato nel documento per identificare la collezione di font usata per il testo principale. Si può notare, dal codice riportato sopra, che la codifica usata è Unicode (`uc`) e che non c'è bisogno di un secondo livello di apici attorno al nome del font, anche se questo contiene degli spazi.

Questa interfaccia non è del tutto soddisfacente, perché ancora troppo legata ad una sintassi di basso livello, per quanto riguarda la specificazione delle funzioni tipografiche avanzate da attivare. Non a caso, nelle ultime versioni di ConTEXt[22] (successive alla pubblicazione di TEX Live 2007), è stata sviluppata una nuova interfaccia, ancora sperimentale, in grado di trattare ad un livello più astratto anche le funzioni tipografiche avanzate. Il seguente codice

```
\definefontfeature[OSC]
   [onum=yes,smcp=yes]
\definefontsynonim[Libertine]
   [Linux Libertine O]
   [feature=OSC]
```

definisce il font `Libertine`, che potrà essere richiamato in qualsiasi punto del testo tramite il comando `\definedfont`, e gli applica l'insieme di funzioni tipografiche avanzate definite dal nome simbolico `OSC`. A questo nome sono state in precedenza, tramite il comando `\definefontfeature`, associate le funzioni "maiuscoletto" (`smcp`) e "numeri oldstyle" (`onum`). Un discorso simile vale anche per le definizioni globali.

Per concludere facciamo presente che per usare XeTEX insieme a ConTEXt il file sorgente deve essere compilato passando all'eseguibile `texexec` l'opzione `--xtx`.

4 Formule e grafica

4.1 Formule

Dal momento che XeTEX è in grado di usare anche i tradizionali font in versione TFM, le formule possono essere composte senza problemi usando pacchetti e costrutti tradizionali. Tuttavia cominciano ad essere disponibili font matematici in formato OpenType: Microsoft distribuisce Cambria Math insieme al proprio sistema operativo, su CTAN si trova Asana Math, e i font STIX[23], che dovrebbero diventare lo standard per le pubblicazioni scientifiche, sono appena usciti dalla fase beta dello sviluppo e, a breve, ne dovrebbe essere disponibile la prima versione stabile.

Per poter sfruttare questi nuovi font, e per poter usare un input in formato Unicode anche per le parti matematiche del testo, è necessario il pacchetto unicode-math, per LaTEX. Questo pacchetto, ancora sperimentale e perciò reperibile solo dall'archivio delle versioni di XeTEX e non su CTAN, è composto da un file di stile, due file che mappano i simboli in codifica Unicode nella rappresentazione

22. Le informazioni che seguono sono tratte da Hagen (2007).
23. `http://www.stixfonts.org`.

$$u(t,x) = \frac{1}{2}\left[f(x-t) + f(x+t)\right] + \frac{1}{2}\int_{x-t}^{x+t} g(s)\,ds.$$

$$(\mathcal{P})\begin{cases} \partial_t^m u = f(t,x,u,\partial_t u,\partial_x u,\ldots,\partial_t^j\partial_x^k u,\ldots) \\ \qquad\qquad 0 \le j \le m-1, \quad j+k \le m, \\ \partial_t^j u(0,x) = u_j(x). \end{cases}$$

(a) Computer Modern

$$\boldsymbol{u(t,x) = \frac{1}{2}\left[f(x-t) + f(x+t)\right] + \frac{1}{2}\int_{x-t}^{x+t} g(s)\,ds.}$$

$$(\mathscr{P})\begin{cases} \boldsymbol{\partial_t^m u = f(t,x,u,\partial_t u,\partial_x u,\ldots,\partial_t^j\partial_x^k u,\ldots)} \\ \qquad\qquad \boldsymbol{0 \le j \le m-1, \quad j+k \le m,} \\ \boldsymbol{\partial_t^j u(0,x) = u_j(x).} \end{cases}$$

(b) Garamond Math Design

$$u(t,x) = \frac{1}{2}\left[f(x-t) + f(x+t)\right] + \frac{1}{2}\int_{x-t}^{x+t} g(s)\,ds.$$

$$(\mathscr{P})\begin{cases} \partial_t^m u = f(t,x,u,\partial_t u,\partial_x u,\ldots,\partial_t^j\partial_x^k u,\ldots) \\ \qquad\qquad 0 \le j \le m-1, \quad j+k \le m, \\ \partial_t^j u(0,x) = u_j(x). \end{cases}$$

(c) Asana Math

$$u(t,x) = \frac{1}{2}\left[f(x-t) + f(x+t)\right] + \frac{1}{2}\int_{x-t}^{x+t} g(s)\,ds.$$

$$(\mathscr{P})\begin{cases} \partial_t^m u = f(t,x,u,\partial_t u,\partial_x u,\ldots,\partial_t^j\partial_x^k u,\ldots) \\ \qquad\qquad 0 \le j \le m-1, \quad j+k \le m, \\ \partial_t^j u(0,x) = u_j(x). \end{cases}$$

(d) Asana Math colorato

FIGURA 5: Formule composte con XeTeX.

interna di LaTeX, e una serie di mappe compilate con TECKit.

All'utente viene fornito il comando `\setmathfont` con il quale è possibile scegliere il font per le formule e applicargli una serie di opzioni. Queste opzioni possono essere applicate anche solo ad una determinata serie di caratteri, individuata per mezzo di codici Unicode, nomi LaTeX, o nomi collettivi predefiniti (ad esempio `\mathopen` indica tutti i delimitatori di sinistra).

Nella figura 5 sono riportati quattro esempi riguardanti la composizione di formule con XeTeX, usando il pacchetto amsmath. Nei primi due sono stati usati font tradizionali, in particolare per l'esempio 5b è stato usato il pacchetto mathdesign con l'opzione garamond. Negli ultimi esempi il font usato è Asana Math, ed è stato caricato nel preambolo il pacchetto unicode-math. Per ottenere la colorazione dell'esempio 5d (in tonalità di grigio, nella versione a stampa), è stato impiegato il seguente codice:

```
\setmathfont[Range=ALL]{Asana Math}
\setmathfont[Range={\mathopen,\mathclose},%
             Color=FF0000]{Asana Math}
\setmathfont[Range={\mathop},%
              Color=0000FF]{Asana Math}
\setmathfont[Range={\mathbin,\equal},%
             Color=00FF00]{Asana Math}
```

Va notato che la selezione di opzioni per diverse gamme definite di simboli rallenta notevolmente il processo, fino al punto, nel caso si cerchi di applicarle a `\mathalpha`, di dover abortire il processo stesso (quest'ultima notazione, ovviamente, è vera relativamente al computer su cui sono state effettuate le prove, equipaggiato con un processore Athlon XP 1600+ e 512Mb di memoria RAM).

4.2 Grafica

4.2.1 Inclusione di illustrazioni esterne

XeTeX supporta l'inclusione di illustrazioni prodotte con programmi esterni, nei formati riconosciuti dal driver di stampa in uso. Vi sarà, quindi, differenza, a seconda che venga usato `xdv2pdf` o `xdvipdfmx`. Il primo, che funziona, come si è detto, solo su sistemi Macintosh, si appoggia alle librerie di QuickTime ed è, dunque, in grado di operare l'inclusione di una gamma piuttosto estesa di formati, che vanno dal Bitmap al TIFF, dall'EPS al PDF, passando per PICT, TGA, ecc.[24] `xdvipdfmx`, invece, è in grado di includere nel documento gli stessi formati supportati da `dvipdfm`: PNG, JPEG, PDF, EPS.

Per effettuare l'inclusione, l'utente può avvalersi, in maniera del tutto trasparente, dei comandi usuali: quelli definiti da graphicx se usa LaTeX, o quelli forniti di base da ConTeXt, nel caso usi quest'ultimo programma.

4.2.2 pgf

Dalla fine del 2007 la libreria di utilità grafiche pgf, e quindi i programmi costruiti a partire da essa, come Ti*k*z, includono il file `pgfsys-xetex.def` che fornisce il supporto per il driver `xdvipdfmx`. A patto, quindi, di compilare con questo driver di stampa, è possibile inserire nel proprio documento grafici espressi nel linguaggio proprio di pgf. Tuttavia, non tutte le funzionalità avanzate sono al momento disponibili. Non è possibile, ad esempio, riempire un'area dell'illustrazione con motivi

24. È bene ricordare, però, che per l'inclusione di immagini in formato PDF, è preferibile usare la primitiva `\XeTeXpdffile`, piuttosto che `\XeTeXpictfile`, se si vuole che venga preservato il carattere vettoriale dell'immagine stessa, nel caso, ovviamente, che si stia usando `xdv2pdf` come driver di stampa.

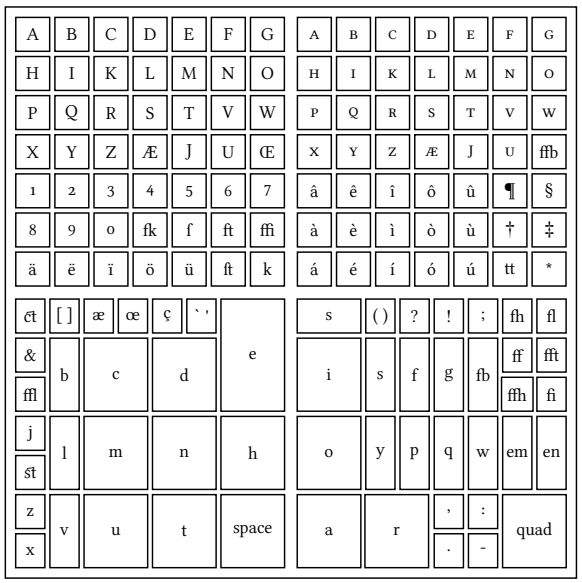


Figura 6: Una cassetta di caratteri disegnata con Ti*k*z.

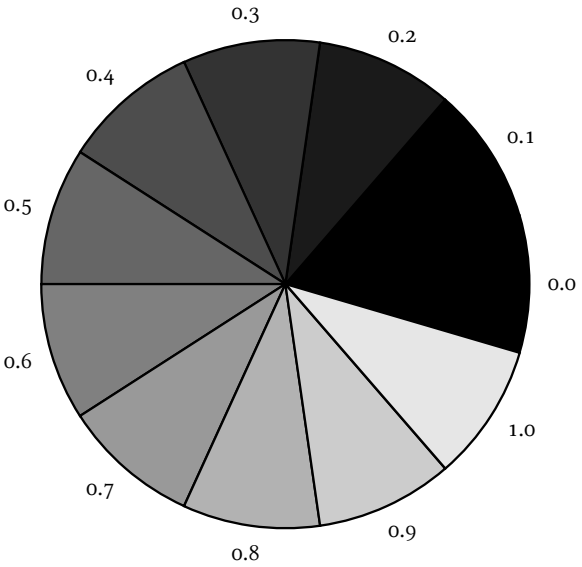


Figura 7: Esempio di grafico con pstricks

ripetuti (tassellatura), né applicare ad un'immagine esterna una maschera di trasparenza selettiva. pgf, infatti, pur mirando ad essere perfettamente portabile da un compilatore all'altro, dipende, per l'effettivo grado di supporto delle singole operazioni, dai diversi driver di stampa, ed è ottimizzato per funzionare con `pdftex`. Tutti gli altri driver, in misura maggiore o minore, mancano invece di qualche funzionalità.

Nella figura 6 è visibile il risultato di un grafico prodotto con tikzpicture e compilato con XeLaTeX.

4.2.3 PSTricks

PSTricks è, probabilmente, il sistema grafico più utilizzato in combinazione con LaTeX. La possibilità di inserire nel file DVI frammenti di codice PostScript che dovranno poi essere interpretati dai vari driver di stampa, lo rende uno dei più potenti e flessibili strumenti per la grafica nel mondo di TeX. Non è sorprendente, perciò, che si stia cercando di rendere PSTricks compatibile con XeLaTeX. Anche in questo caso il supporto viene fornito per `xdvipdfmx`. Dall'archivio delle versioni di XeLaTeX è possibile scaricare il file `xetex-pstricks.con` che, una volta rinominato `pstricks.con` e copiato in una cartella dove pstricks può trovarlo, configura quest'ultimo in modo da usare `xdvipdfmx` come driver di stampa.[25]

Nel concreto, l'interpretazione del codice PostScript inserito nell'XDV viene fatta per mezzo di Ghostscript, di cui si consiglia di installare una versione recente, se si vuole che vengano supportate operazioni di carattere avanzato, come, ad esempio, effetti di trasparenza.

25. Alternativamente, è possibile usare `xdvipdfmx.con`, facente parte della distribuzione base di PSTricks, che però appare leggermente indietro nello sviluppo, rispetto al suo omologo distribuito da XeLaTeX.

Per quanto questo progetto sia ancora in fase di sviluppo, i risultati sono già buoni, come si può vedere nella figura 7.

5 Conclusioni

Questo articolo mostra come XeLaTeX sia ormai un prodotto maturo, disponibile su tutte le piattaforme e utilizzabile insieme ai principali "dialetti" di TeX (Plain TeX, LaTeX, ConTeXt). Non è ancora esente da imperfezioni: deve migliorare il supporto per alcune funzioni tipografiche avanzate implementate dalle *smart font technologies*, ha bisogno di un sostituto per babel (polyglossia è il promettente candidato), deve migliorare la gestione dei font matematici in codifica Unicode e il supporto per sistemi di applicazioni grafiche come PSTricks e pgf.

Malgrado queste piccole imperfezioni, la facilità d'uso che introduce in un settore, quello della gestione dei caratteri da stampa, tradizionalmente tra i più complicati e ostici, nel mondo TeX, lo rende uno strumento utile e, in alcuni casi, come la composizione di documenti contenenti sistemi di scrittura non occidentali, addirittura quasi indispensabile. Essendo un programma in fase di attivo sviluppo e con una base di utilizzatori numericamente non indifferente, possiamo tranquillamente aspettarci ulteriori miglioramenti nel futuro prossimo.

Per concludere, vorrei aggiungere una nota finale sui font. Per quanto l'utilità di XeLaTeX si dimostri anche soltanto nel caso in cui si voglia usare un font senza bisogno di doverlo installare nel sistema TeX, il programma dà il meglio di sé quando si trova a gestire degli *smart font*. Questi font, che siano AAT, OpenType o Graphite, si trovano oggi con facilità. Senza voler citare font commerciali, distribuiti da Adobe, Monotype, Linotype, e da

quasi tutte le altre case, grandi, medie o piccole che siano, la cui qualità è certamente garantita, è possibile ormai reperire *smart font* di buona qualità anche nel circuito del software libero. Molti dei font prodotti da *e-foundry* per TEX (Latin Modern, la collezione TEX Gyre, Antykwa Toruńska, Iwona) sono disponibili anche in formato OpenType ed è possibile scaricarli sia da CTAN che dal sito dalla *e-foundry*.[26] Ancora disponibili su CTAN sono UM Typewriter, un font a spaziatura fissa codificato Unicode, e Asana-Math, font matematico anch'esso in codifica Unicode, entrambi di Apostolos Syropoulos.

Per uscire dal mondo TEX, Junicode,[27] font per medievalisti e non solo, e Linux Libertine,[28] sono tra i più completi e ben fatti, rilasciati sotto i termini della licenza GNU GPL. Accanto a questi, vanno ricordati i font prodotti da SIL International con licenza OFL, in particolare Gentium, Doulos e Charis, corredati di tabelle sia OpenType che Graphite; e quelli della Greek Font Society,[29] anch'essi con licenza OFL, soprattutto GFS Bodoni e GFS Didot, che contengono, oltre ai caratteri dell'alfabeto greco, anche l'alfabeto latino.

Né, infine, sono da dimenticare Old Standard, di Aleksej Kryukov,[30] e i font dedicati a sistemi di scrittura antichi prodotti da George Douros[31] tra cui il bellissimo Alexander, il cui alfabeto greco è ricalcato su quello disegnato da Alexander Wilson nel XVIII secolo, purtroppo disponibile solo in versione corsiva.

6 Ringraziamenti

Questo articolo non sarebbe mai stato scritto, se non mi fosse stato, praticamente, "commissionato" da Lapo F. Mori, a cui vanno, quindi, per primo i miei ringraziamenti.

Desidero inoltre ringraziare l'anonimo recensore dell'articolo per avermi indicato i punti in cui poteva essere migliorato, e per avermi aiutato a rettificare alcune inesattezze riguardo alla relazione tra Unicode e lo standard ISO/IEC 10646.

Riferimenti bibliografici

Apple Computer (2002). *TrueType Reference Manual*, Apple Computer, capitolo 6: Font files. `http://developer.apple.com/fonts/TTRefMan/RM06/Chap6AATIntro.html`.

Beeton, B., Freytag, A. e Sargent III, M. (2007). «Unicode support for mathematics». `http://www.unicode.org/reports/tr25/`.

Donne, J. (1959). *Devotions Upon Emergent Occasions Together with Death's Duel*. Ann Arbor Paperbacks, University of Michigan Press. La versione utilizzata è quella fornita da *Project Gutenberg*: `http://www.gutenberg.org/2/3/7/7/23772/`.

Erodoto (2006). *Le storie*. UTET. A cura di Aristide Colonna e Fiorenza Bevilacqua. Con testo a fronte.

Hagen, H. (2007). «ConTEXt MKII & MKIV.» Disponibile all'indirizzo `http://www.pragma-ade.com/general/manuals/mk.pdf`.

Hosken, M., Hallissy, B., Cleveland, W., Correll, S. e Ward, A. (2007). *Graphite Language Description*. SIL Non-Roman Scripts Initiative (NRSI). La versione elettronica in formato PDF è disponibile all'indirizzo `http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&cat_id=RenderingGraphite`.

Kew, J. (2002). *TECkit version 2.0. A Text Encoding Conversion toolkit*. SIL Non-Roman Scripts Initiative (NRSI). La versione elettronica in formato PDF è disponibile all'indirizzo `http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=TECkitDownloads`.

— (2005). «XeTeX the Multilingual Lion: TEX meets Unicode and smart font technologies». *TUGboat*, **26** (2), pp. 115–124.

— (2007). «About XeTeX». Disponibile all'indirizzo `http://scripts.sil.org/xetex/XeTeX-notes.pdf`.

— (2008). «XeTeX Live». *TUGboat*, **29** (1), pp. 146–150.

Knuth, D. E. (1984). *The TEXbook*. Addison-Wesley, Reading, MA, USA.

Majakovskij, V. (1989). *La nuvola in calzoni*. Marsilio. A cura di Remo Faccani. Con testo a fronte.

Microsoft Corporation (2004). «OpenType specification». `http://www.microsoft.com/typography/otspec/default.htm`.

Robertson, W. (2005). «Advanced font features with XeTeX – the fontspec package». *TUGboat*, **26** (3), pp. 215–223.

— (2007a). «The XeTeX reference guide». Disponibile su CTAN, all'indirizzo `http://www.ctan.org/tex-archive/info/xetexref/XeTeX-reference.pdf`.

— (2007b). «The fontspec package.» Disponibile su CTAN, all'indirizzo `http://www.ctan.org/tex-archive/macros/xetex/latex/fontspec.pdf`.

26. `http://www.gust.org.pl/projects/e-foundry/`.
27. `http://junicode.sourceforge.net`.
28. `http://linuxlibertine.sourceforge.net`.
29. `http://www.greekfontsociety.gr/`.
30. `http://www.thessalonica.org.ru/en/fonts.html`.
31. `http://users.telar.gr/~g1951d/downloads`.

— (2007c). «The xltxtra package.» Disponibile su CTAN, all'indirizzo http://www.ctan.org/tex-archive/macros/xetex/latex/xltxtra.pdf.

— (2008). «Experimental unicode mathematical typesetting: The unicode-math package». http://scripts.sil.org/svn-public/xetex/TRUNK/texmf/source/xelatex/unicode-math/unicode-math.pdf.

Syropoulos, A. (2007). «The philokalia package.» Disponibile su CTAN, all'indirizzo http://www.ctan.org/tex-archive/macros/xetex/latex/philokalia.pdf.

Unicode Consortium (2007). *The Unicode standard, Version 5.0.* Addison-Wesley, Boston, MA.

▷ Massimiliano Dominici
mlgdominici@interfree.it

Macroistruzioni con argomenti delimitati

Claudio Beccari

Sommario

Il pacchetto di macroistruzioni comunemente definito LaTeX non tratta mai delle macro con argomenti delimitati; esso stesso ne mette a disposizione del compositore un numero minimo. Con i comandi primitivi dell'interprete TeX è possibile definire macro con argomenti delimitati che si possono usare anche con LaTeX. Queste macro possono risultare estremamente comode in certe circostanze, specialmente quando si creano file di classe o pacchetti di estensione, dove permettono di identificare con maggiore facilità il significato di ogni argomento. L'esposizione viene eseguita servendosi di un problema concreto: il Registro delle Lezioni.

Abstract

The macro package commonly known as LaTeX does not describe any means for defining macros with delimited arguments; furthermore it offers a very small number of them to the user. By means of the primitive commands of the TeX interpreter it is easy to define macros with delimited arguments that may be used also while using the other LaTeX macros. This kind of macros may be very useful in some instances, particularly when writing class or package files, where they make it easy to identify the function that any argument plays in the macro expansion. The subject is described with the aid of a practical problem: the Lecture Log.

1 Introduzione

Le macroistruzioni, comunemente chiamate 'macro', servono per far eseguire all'interprete tex una serie complessa di operazioni, che il compositore non è costretto a riscrivere ogni volta per disteso.

L'interprete, che sia tex stesso o pdfetex o altro, è un programma eseguibile che interpreta quanto è scritto nel file sorgente `.tex`, per eseguire la composizione del testo che vi è contenuto insieme ai comandi di *mark-up* che dichiarano certe parti del testo come appartenenti a certe categorie, che vanno composte in questa o quella maniera.

L'interprete nella totalità dei casi non costringe il compositore a scrivere il testo inframmezzato ai comandi di composizione, siano essi cambiamenti di font, variazioni della giustezza o della giustificazione, passaggio alla composizione della matematica, eccetera; anche la semplicissima collezione di macro contenuta nel file `plain.tex` permette al compositore di usare un set essenziale di macro che consentono di arrivare ad un modesto risultato senza troppa fatica[1]. La collezione di macro contenuta nel file `latex.ltx` mette a disposizione del compositore un numero sterminato di macro che agevolano ulteriormente la composizione, lasciando l'autore e il compositore liberi di concentrarsi sul contenuto, piuttosto che sulla sua estetica. Le centinaia di pacchetti di estensione disponibili negli archivi internazionali estendono ulteriormente le funzionalità delle macro di LaTeX.

Lo stesso succede quando si usano altri programmi di composizione quali Omega (Ω) e Lambda, oppure Aleph e Lamed, oppure ConTeXt o XeTeX e XeLaTeX.

Per rendere più spedito l'avvio della compilazione, fin dall'inizio TeX è stato progettato con la capacità di leggere una volta per tutte il file `plain.tex`, o il file `latex.ltx`, o altri simili file, trasformandone il contenuto in linguaggio macchina, per poi scaricare la parte di memoria occupata da questa trasformazione in un file, detto di 'formato', nella fattispecie `plain.fmt` e `latex.fmt`, così che in seguito è più veloce caricare il file binario di formato e si risparmia tutto il lavoro di interpretazione e di trasformazione in linguaggio macchina già fatto una volta per tutte. Quando si lancia LaTeX direttamente dalla linea di comando o cliccando su di un opportuno bottone dello *shell editor*, in realtà si dà il comando di lanciare l'interprete dicendogli anche quale file di formato usare, in questo caso il formato `latex.fmt`.

Quando è in azione, l'interprete legge il flusso di dati che gli arriva dal file sorgente, e assegna una categoria ad ogni oggetto (*token*) che legge; poi, a seconda della categoria, esegue quanto ogni token dice di fare; se si tratta di una semplice lettera, la accoda al testo precedentemente raccolto, se si tratta di un segno di interpunzione lo inserisce nel testo con le opportune spaziature, eccetera.

Ma se si tratta di un comando, esamina se si tratta di un vero comando primitivo, nel qual caso lo esegue, altrimenti si tratta di una macro: in questo caso ne va a cercare la definizione (nel formato caricato in memoria oppure nella parte di memoria dove ha raccolto le macro definite dall'utente o

1. Qui non si discutono i pregi e i difetti di questa o quella collezione di macro; gli estimatori di `plain` TeX hanno dei buoni motivi per preferire questa collezione minimale di macro, perché essa consente loro di specificare quello che vogliono per le parti della composizione che già non hanno una macro definita. Il fatto è che per comporre bene `plain` TeX richiede una enorme esperienza e non poca attenzione; in parole povere un gran lavoro e quindi una notevole fatica.

lette dai pacchetti di estensione), cerca anche se gli argomenti sono delimitati o non lo sono, ed in ogni caso 'tokenizza' gli argomenti tratti dal file di ingresso, esegue le formali sostituzioni nei segnaposto di ogni argomento, esamina se la definizione del comando fa ricorso ad altre macro, ripete il processo per ogni altra macro che incontra, finché ha completamente sviluppato la macro iniziale e il suo contenuto ed ha solo una lista di token non ulteriormente sviluppabili; a questo punto esegue in sequenza tutti questi token.

Sembra complicato e lo è; il vantaggio per l'utente del sistema TEX è che questi non deve preoccuparsene, perché l'interprete fa tutto da solo. Al massimo l'utente può trarre giovamento dalla comprensione di questo meccanismo quando incontra degli errori e deve provvedere in merito.

Come si è visto nella sommaria descrizione fatta sopra, l'interprete vede se la definizione di ogni macro prevede argomenti delimitati, oppure se questi argomenti sono semplici token non ulteriormente sviluppabili ovvero sono stringhe di lettere e, talvolta, ulteriori comandi, racchiusi dal compositore dentro una coppia di parentesi graffe, che costituiscono i delimitatori espliciti di default di ciascun argomento. Per esempio, se il compositore vuol comporre una tabella e vi vuole collocare una cella che occupi due colonne con l'argomento costituito da una parola centrata ed evidenziata con il comando `\emph`, inserisce nel file sorgente la dichiarazione

```
\multicolumn2c{\emph{Pippo}}
```

dove i token `2` e `c` non necessitano di parentesi di gruppo perché sono due token non ulteriormente scomponibili, mentre il terzo argomento di `\multicolumn` richiede le parentesi graffe che lo delimitano, così come il terzo argomento stesso contiene un comando con un argomento formato da più lettere, che va quindi racchiuso a sua volta fra parentesi graffe.

Si noti l'ambiguità delle parole che si stanno usando per descrivere la funzione delle parentesi graffe; esse definiscono il loro contenuto come un gruppo, e in un certo senso ne costituiscono i delimitatori; tuttavia esse non delimitano l'argomento, nonostante le apparenze, ma definiscono il gruppo.

La mancanza di delimitatori è ancora più evidente se si va a cercare nel file `latex.ltx` la definizione di `\multicolumn` che, semplificando molto, si riduce a

```
\def\multicolumn#1#2#3{⟨definizione⟩}
```

e, come si vede i tre argomenti `#1`, `#2`, e `#3` non sono separati da nulla, nemmeno da spazi.

2 Macro con argomenti delimitati

Dopo questa lunga introduzione vale la pena di descrivere come, tecnicamente, si possono definire macro con argomenti delimitati.

Innanzi tutto non si possono creare simili definizioni usando i comandi di alto livello messi a disposizione da LATEX: `\newcommand` o `\renewcommand` oppure il loro parente stretto `\providecommand` o, infine, `\DeclareRobustCommand` e le loro versioni asteriscate. Questi comandi, infatti, evitano al compositore di doversi preoccupare di molti dettagli e di molte verifiche, e perciò isolano il compositore dalla effettiva definizione eseguita durante l'esecuzione di quei comandi, definizione che alla fine si riduce all'esecuzione di uno dei quatto comandi primitivi dell'interprete TEX: `\def`, `\edef`, `\gdef` e `\xdef`.

Il problema di questi quattro comandi è che sono 'primitivi' non solo nel senso che sono all'origine di qualunque altro comando (macro) che possa essere definito tramite loro, ma anche nel senso traslato del termine: nel senso, cioè, che 'senza ragionare' eseguono incondizionatamente quello che devono fare, con il rischio che essi vengano usati malamente dal compositore quando involontariamente ridefinisce un comando del nucleo di LATEX; a causa di questo errore, LATEX smette (o può smettere) di funzionare. Se dovesse succedere una cosa del genere, la caccia all'errore deve essere ricercata per prima cosa nelle definizioni imprudenti fatte dal compositore stesso.

Per noi non anglofoni c'è un metodo molto semplice ed economico per definire comandi diversi da quelli del nucleo di LATEX: basta usare nomi italiani, invece che inglesi, come sono invece tutti i comandi interni di LATEX.

Però il metodo corretto, e valido indipendentemente dalla lingua del compositore, è quello di eseguire la verifica che il nome del comando sia effettivamente disponibile per una definizione: basta scrivere le proprie definizioni all'interno di un file di stile con estensione `.sty`, da leggere nel preambolo del file di ingresso mediante il comando `\usepackage`, oppure far precedere il comando `\makeatletter` alle nuove definizioni, che a loro volta devono essere seguite da `\makeatother`. Volendo per esempio definire la macro `\pipe` mediante il comando `\def`, basta scrivere nel preambolo:

```
\maketatletter
\@ifdefinable{\pipe}{⟨definizione⟩}
\makeatother
```

Il comando `\@ifdefinable`, che contiene nel suo nome la 'non lettera' `@`, è il motivo per il quale bisogna premettere `\makeatletter` prima di farne uso, e poi ripristinare `@` con la categoria di 'non lettera' dopo averne fatto uso. Per altro, il suddetto comando `\@ifdefinable` controlla che il comando

proposto sia effettivamente non ancora definito, che non cominci per `\end`[2] e sia diverso da `\relax`, visto che questi due ultimi comandi giocano un ruolo importantissimo nel nucleo di LaTeX.

Riferendoci alla predisposizione di una classe per comporre i registri delle lezioni, è possibile che sia necessario separare il nome dal cognome del docente, per associare al cognome un significato particolare; per non costringere il compositore a ricordarsi troppe regole, decidiamo che la stringa di caratteri contenenti il nome e il cognome, che verrà usata più volte nella composizione del registro, sia introdotta dal compositore (verosimilmente il docente stesso) come una stringa contenente un solo spazio di separazione fra nome e cognome[3].

Questo unico spazio ci servirà da delimitatore del nome; potremo usare un altro carattere per delimitare il cognome, per esempio un punto esclamativo, sicuramente non presente nel cognome. La nostra definizione sarà composta di due macro; la prima serve al docente per impostare il suo nominativo, che verrà memorizzato in una macro di servizio; il secondo servirà ad isolare il cognome e definirà un'altra variabile temporanea da usare in altre macro, in relazione al significato che vorremo assegnare al cognome. Ecco dunque che avremo:

```
\let\@docente\empty% Default iniziale
\def\docente#1{\def\@docente{#1}}
\def\getcognome#1 #2!{\def\Cognome{#2}}
```

Si noti lo spazio fra il primo e il secondo argomento di `\getcognome` e il punto esclamativo che delimita la fine del cognome.

Ecco, questa è la definizione di una macro ad argomenti delimitati; se si preparasse il seguente file `.tex` e lo si compilasse[4]:

```
\documentclass{minimal}
\makeatletter
\let\@docente\empty
\def\docente#1{\def\@docente{#1}}
\def\getcognome#1␣#2!{\def\Cognome{#2}}
\begin{document}
\docente{Mario␣Rossi}
\makeatletter
\expandafter\getcognome\@docente!
\show\Cognome
\end{document}
```

`\show` mostrerebbe sullo schermo che `\Cognome` è una macro la cui definizione vale proprio `Rossi`.

Il comando `\expandafter` serve per alterare la sequenza delle operazioni: prima viene sviluppata la macro temporanea `\@docente`, che ridiventa la stringa introdotta nel processo di composizione e memorizzata mediante il comando `\docente`, poi questa stringa viene elaborata dal comando `\getcognome` che assegna il solo cognome alla macro `\Cognome`.

3 Il Registro delle Lezioni

Presso il Politecnico di Torino ogni docente deve compilare un Registro delle Lezioni formato da una camicia e tanti intercalari quanti bastano[5].

La camicia sulla prima pagina, figura 1, riporta le solite intestazioni presenti nei documenti di qualunque università, cioè il nome e il logo; contiene poi il tipo di corsi di studi e la sede di svolgimento dell'insegnamento, il nome dell'insegnamento con il suo codice; se questo insegnamento aggrega anche studenti di altri corsi di studi, vanno elencati tutti i codici e in chiaro la sigla di ciascun corso di laurea; ci vuole evidentemente il nome del docente e la sua facoltà di appartenenza (non necessariamente coincidente con quella della facoltà a cui afferisce l'insegnamento); ci vuole l'anno accademico e il periodo didattico[6].

La quarta pagina della camicia, figura 2, contiene due tabelle: una serve per indicare quanta didattica riceve ogni studente secondo l'organizzazione del corso, e la seconda indica quanta didattica hanno fornito il docente e i suoi collaboratori: il corso potrebbe essere diviso in squadre e, a seconda della tipologia di attività, potrebbero esserci diversi docenti compresenti; le attività sono divise per tipologie identificate da sigle: L per le lezioni, EA per le esercitazioni in aula, EL per le esercitazioni in laboratorio sperimentale, eccetera.

Gli intercalari, figura 3, contengono una tabella per ogni docente (principale o collaboratore) che forma una 'riga' di una grande tabella che prende tutta la pagina.

Ogni riga deve contenere oltre la data, l'orario, la tipologia di didattica a cui si riferisce quella registrazione, le ore complessive dedicate a quella tipologia di attività, la lingua in cui si svolge

2. Tutti i comandi che terminano un ambiente sono indicati dal compositore con `\end{`⟨*ambiente*⟩`}`, ma, dopo un poco di amministrazione e di pulizia eseguito dal comando `\end`, il comando che effettivamente chiude l'ambiente è costituito dal nome stesso dell'ambiente preceduto da `\end` senza l'uso delle graffe. In altre parole, l'ambiente `quote` viene effettivamente chiuso da `\endquote`.

3. Per i nomi e/o i cognomi formati da più parole, bisognerà unirle, lasciando le maiuscole, in modo che la stringa contenga effettivamente un solo spazio: PierPaolo, invece di Pier Paolo; DellaCasa invece di Della Casa. Non è una regola troppo complicata da ricordare.

4. Si ricorda che la classe `minimal` serve appunto per collaudare le definizioni di nuove macro; che `\begin{document}` esegue anche `\makeatother` e che il segno ␣ serve per evidenziare gli spazi fra i caratteri.

5. La camicia è una cartellina costituita, volendo, da un foglio A3 piegato a metà; ogni intercalare è un foglio A4 inserito ordinatamente dentro la cartellina. In pratica, vista la rarità delle stampanti A3 negli uffici dei docenti, si può stampare tutto su fogli A4, mettere la quarta pagina di copertina come ultima pagina, con la parte scritta all'esterno del fascicolo, e infine pinzare il tutto con punti metallici.

6. In altre università i corsi, forse, sono solo annuali e/o semestrali; presso il Politecnico ci sono corsi annuali, semestrali, trimestrali e bimestrali; si svolgono in periodi successivi, detti periodi didattici.

l'attività, la squadra per la quale l'attività è stata svolta, il numero di docenti compresenti; infine il nome del docente specifico e la sua firma, che ovviamente dovrà essere autografa, quindi la classe del documento deve provvedere solo alla composizione del riquadro destinato alla firma. Ovviamente il riquadro più grande di questa 'riga' contiene il testo che descrive la materia trattata in quella lezione.

Come si vede, il registro richiede una composizione piuttosto complessa; la necessità di ripetere alcune informazioni in righe successive dove cambia solo il nome del docente richiede la formazione di liste; la necessità di eseguire i semplici calcoli richiesti implica altre liste, docente per docente; la necessità di eseguire un minimo di controlli richiede ulteriori comandi.

4 La classe per la composizione dei registri

L'idea di creare una classe per questo tipo di composizione, non è solo quella di comporre bene il registro mediante l'uso dello strumento compositivo migliore, ma anche di sfruttare LaTeX per fargli eseguire le scritture ripetitive, nonché per fargli fare i calcoli necessari; le capacità di calcolo di LaTeX sono piuttosto limitate, ma, senza nemmeno ricorrere al pacchetto di estensione calc, si riesce a fare tutte le somme necessarie per la quarta pagina della copertina sia per righe sia per colonne.

Le liste di docenti e di tipologie di attività didattiche richiedono altre liste. I controlli potrebbero anch'essi avvantaggiarsi dalle macro con argomenti delimitati.

Qui non si descriverà tutta la classe; la descrizione completa può essere ottenuta compilando con LaTeX il codice documentato contenuto nel file `registro07.dtx`, Beccari (2007). La documentazione serve per la corretta manutenzione del codice e non è certamente una lettura divertente. Qui ci si limita solamente a descrivere sommariamente quello che interessa per la spiegazione delle macro con argomenti delimitati.

Si prevede che il docente componga due file; il primo sarebbe il master-file che contiene solo le informazioni per la camicia e per eseguire la lettura del secondo; questo secondo file dovrebbe contenere solo le informazioni per gli intercalari; questa divisione dei ruoli non è essenziale, ma è comoda per molti motivi, specialmente quando sono coinvolti corsi omonimi paralleli[7].

Per non dover riscrivere le righe 2, 3 e 4 dell'intercalare della figura 3, bisogna che il codice da inserire nel file sorgente sia ridotto al minimo indispensabile; nello stesso tempo il codice non deve essere molto diverso da quello usato per le righe 1 e 5, che si riferiscono ad un solo docente: il titolare per la riga 1 e un collaboratore per la riga 5; solo il contenuto della lezione deve variare, evidentemente, ancorché esso debba essere lo stesso per le tre righe 2, 3 e 4.

Il codice della riga 1, relativa al solo titolare il cui nome è già stato memorizzato con i comandi specificati per creare la prima pagina della camicia, dovrebbe essere semplicemente:

```
\begin{lezione}{2004-09-22}%
   {12:30--14:30}{L}{2}
Introduzione al corso.

Ripasso ... più comuni.
\end{lezione}
```

Il codice per la riga 5, relativo ad un solo collaboratore dovrebbe essere semplicemente:

```
\begin{lezione}{2004/09/24}%
{10:30--12:30}{EA}{2}(1,2/2,EN)%
  [GianPaolo Neri]
Esercizi 1.4, ..., 1.7
\end{lezione}
```

Infine per le tre righe 2, 3 e 4 occorre la lista dei docenti:

```
\begin{lezione}{2004/09/23}%
   {10:30--12:30}{EA}{2}(3,1/2)%
      [Mario Rossi,Fabio Bianchi,%
         GianPaolo Neri]
Nozioni elementari sui trasformatori.

Esercizi 1.1, 1.2, 1.3
\end{lezione}
```

Questa descrizione permette di capire che ci vogliono due diversi tipi di argomenti facoltativi: il primo è racchiuso fra parentesi tonde e contiene nell'ordine: (*a*) il numero di docenti compresenti; (*b*) il numero della squadra riferito al numero totale di squadre; (*c*) la lingua in cui si svolge l'attività.

Il secondo tipo di argomento facoltativo, racchiuso fra parentesi quadre, indica una lista di docenti i cui nomi e cognomi sono separati da virgole senza spazi superflui[8], eventualmente ridotta ad un solo nominativo se il docente corrispondente, diverso dal titolare, è l'unico presente in aula. Non c'è mai bisogno di indicare espressamente il titolare, tranne nei casi in cui egli sia compresente con altri docenti.

Si noti che non è necessario specificare la lingua italiana, perché è quella di default; così non è necessario specificare le squadre se ce ne è una sola; non è nemmeno necessario specificare il numero di docenti compresenti se ce ne è uno solo; quindi l'argomento facoltativo fra parentesi tonde ha, per

7. Questi sono i corsi che dovrebbero procedere in modo sincrono, visto che contengono allievi divisi fra i corsi in base alla lettera iniziale del cognome.

8. In generale non è necessario spezzare le righe come fatto sopra, ma se lo si fa, è necessario inserire i segni di commento senza lasciare spazi superflui.

POLITECNICO DI TORINO
FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Corsi di Laurea – Sede di Torino

Registro del corso di **ELETTROTECNICA II**
07AUQBT, 07AUQBW,
07AUQCT / IFM+FIS+IFF

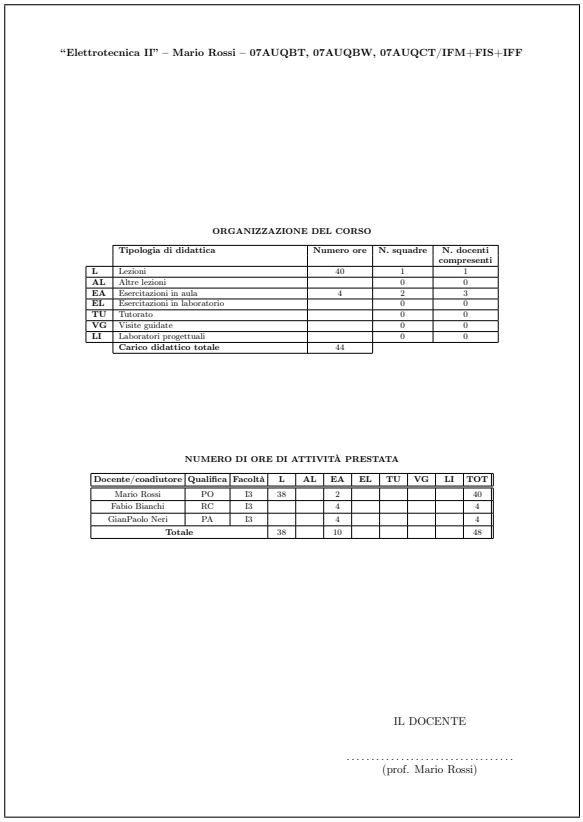
Prof. **MARIO ROSSI**
Dipartimento di Elettronica

Anno Accademico **2004/2005**
1º PD

Visto
IL PRESIDE DELLA FACOLTÀ

.....

FIGURA 1: Prima pagina della camicia



"Elettrotecnica II" – Mario Rossi – 07AUQBT, 07AUQBW, 07AUQCT/IFM+FIS+IFF

ORGANIZZAZIONE DEL CORSO

	Tipologia di didattica	Numero ore	N. squadre	N. docenti compresenti
L	Lezioni	40	1	1
AL	Altre lezioni		0	0
EA	Esercitazioni in aula	4	2	3
EL	Esercitazioni in laboratorio		0	0
TU	Tutorato		0	0
VG	Visite guidate		0	0
LI	Laboratori progettuali		0	0
	Carico didattico totale	44		

NUMERO DI ORE DI ATTIVITÀ PRESTATA

Docente/coadiutore	Qualifica	Facoltà	L	AL	EA	EL	TU	VG	LI	TOT
Mario Rossi	PO	I3	38		2					40
Fabio Bianchi	RC	I3			4					4
GianPaolo Neri	PA	I3			4					4
Totale			38		10					48

IL DOCENTE

.....
(prof. Mario Rossi)

FIGURA 2: Quarta pagina della camicia

Insegnamento: Elettrotecnica II ..... Codici: 07AUQBT, 07AUQBW, 07AUQCT

Data 2004-09-22	Tipo L	N° Docenti compresenti 1	Argomento Introduzione al corso. Ripasso di elettrotecnica; le leggi delle tensioni e delle correnti, i nodi indipendenti, i tagli indipendenti, le maglie indipendenti; le equazioni del connettore. Le leggi costitutive dei componenti più comuni		Docente Mario Rossi
Orario 12:30–14:30	N.ore 2	Squadra di squadre 1/1		Lingua IT	Firma
Data 2004/09/23	Tipo EA	N° Docenti compresenti 3	Argomento Nozioni elementari sui trasformatori. Esercizi 1.1, 1.2, 1.3		Docente Mario Rossi
Orario 10:30–12:30	N.ore 2	Squadra di squadre 1/2		Lingua IT	Firma
Data 2004/09/23	Tipo EA	N° Docenti compresenti 3	Argomento Nozioni elementari sui trasformatori. Esercizi 1.1, 1.2, 1.3		Docente Fabio Bianchi
Orario 10:30–12:30	N.ore 2	Squadra di squadre 1/2		Lingua IT	Firma
Data 2004/09/23	Tipo EA	N° Docenti compresenti 3	Argomento Nozioni elementari sui trasformatori. Esercizi 1.1, 1.2, 1.3		Docente GianPaolo Neri
Orario 10:30–12:30	N.ore 2	Squadra di squadre 1/2		Lingua IT	Firma
Data 2004/09/24	Tipo EA	N° Docenti compresenti 1	Argomento Esercizi 1.4, 1.5, 1.6, 1.7		Docente GianPaolo Neri
Orario 10:30–12:30	N.ore 2	Squadra di squadre 2/2		Lingua EN	Firma

FIGURA 3: Prime righe di un intercalare

così dire, una importanza minore rispetto a quello racchiuso fra parentesi quadre e all'occorrenza quello può mancare anche se questo è presente.

Similmente i tre parametri dell'argomento facoltativo fra parentesi tonde sono elencati gerarchicamente, nel senso che indicare il primo non richiede di indicare il secondo e il terzo, ma indicare il secondo richiede che si indichi il primo; all'occorrenza, se va bene il valore di default, gli argomenti non indicati possono essere omessi, purché non si omettano le virgole; in altre parole va bene scrivere `(,,EN)` ma non va bene indicare solamente `(EN)`: al contrario si può indicare sia `(2)` sia `(2,,)`, perché producono lo stesso risultato.

A parte queste specificazioni, ci si rende subito conto che sia il primo, sia il secondo argomento facoltativo sono liste di stringhe separate da virgole; la tipica situazione nella quale per esaminarle bisogna fare uso di macro con argomenti delimitati.

5 Alcune macro con argomenti delimitati

In teoria una macro può avere simultaneamente sia argomenti delimitati sia argomenti non delimitati; per chiarezza di programmazione sarebbe meglio non ridursi a queste situazioni ibride, ma non è vietato. Una macro avente solo argomenti delimitati segue la sintassi seguente[9]:

`\def\`⟨*macro*⟩⟨*delim*$_0$⟩`#1`⟨*delim*$_1$⟩`#2`⟨*delim*$_2$⟩`...%`
`#n`⟨*delim*$_n$⟩`{`⟨*definizione*⟩`}`

con $n \leq 9$. Il nome della ⟨*macro*⟩ può essere qualsiasi, ma per evitare inconvenienti è meglio fare uso del test `\@ifdefinable`. Gli argomenti, come è noto, non possono essere più di nove, e corrispondentemente non ci possono essere più di dieci delimitatori. Questi a loro volta possono essere costituiti da qualunque stringa, anche dei semplici spazi, o, eventualmente, delle stringhe che hanno la struttura di comandi veri o non definiti; solo ⟨*delim*$_0$⟩ non può essere una lettera, perché potrebbe essere confusa con l'ultima lettera della ⟨*macro*⟩.

Esaminando il codice del pacchetto babel, si possono per esempio trovare le definizioni di due comandi usati molto frequentemente in quel codice, introdotti apposta per rendere 'robusti' i comandi condizionali; semplificando un poco, si tratta di:

```
\def\bbl@afterelse#1\else#2\fi{\fi#1}
\def\bbl@afterfi#1\fi{\fi#1}
```

La lettura di questi due comandi permette di capire che servono per 'gettare' la prima clausola di un comando condizionale, oppure la seconda clausola oltre la fine dell'espressione condizionale; per esempio, se si volesse controllare se questa pagina sia dispari si potrebbe scrivere in termini di comandi primitivi, estesi con i due comandi di babel citati sopra:

```
\expandafter\ifodd\value{page}%
\bbl@afterelse dispari\else
\bbl@afterfi pari\fi
```

I separatori `\else` e `\fi`, benché siano anche dei comandi primitivi, non vengono eseguiti, quando sono letti come separatori, ma servono solo per delimitare la prima clausola dalla seconda; la chiusura dell'istruzione condizionale `\ifodd` viene eseguita dal comando `\fi` contenuto dentro la definizione.

Tornando alle liste formate dai comandi facoltativi della classe dei registri, vediamo dunque che la separazione dei parametri fra parentesi tonde si può eseguire facilmente con i comandi seguenti[10]:

```
\def\set@numeri(#1){\lista@numeri#1,,,!}
%
\def\lista@numeri #1,#2,#3,#4!{%
 \def\@tempA{#1}%
 \ifx\@tempA\empty\NumDoc=\@ne\relax
 \else\NumDoc=#1\relax\fi
%
 \def\@tempA{#2}%
 \ifx\@tempA\empty\def\Squadre{1/1}%
 \else\def\Squadre{#2}\fi
%
 \def\@tempA{#3}%
 \ifx\@tempA\empty\def\Lingua@{\@Lingua}%
 \else\def\Lingua@{#3}\fi
}%
```

Il trucco è semplice: `\set@numeri` estrae l'argomento facoltativo delimitato dalla coppia di parentesi tonde (i delimitatori), poi lo passa a `\lista@numeri`, allungandolo con altri tre elementi della lista 'vuoti', poi completa la lista delimitandola con un punto esclamativo.

`\lista@numeri` a sua volta prende la lista, eventualmente allungata, ma la separa in quattro elementi: il primo dovrebbe rappresentare il numero di docenti compresenti, il secondo il numero relativo di squadre e il terzo la lingua; se il compositore avesse specificato solo il primo, i tre elementi vuoti aggiunti alla lista assicurano che l'insieme di elementi che formano l'argomento facoltativo siano tutti presenti, eventualmente vuoti; se invece il compositore li ha specificati tutti e tre, il quarto argomento raccoglie tutto quel che resta della lista, eventualmente una sequenza di virgole. Le istruzioni successive copiano ciascun valore della lista

9. Qui si fa un esempio che usa `\def` come comando di definizione; si potrebbero usare anche `\gdef`, `\edef` e `\xdef`; `\def` e `\edef` eseguono definizioni locali al gruppo entro il quale vengono eseguite, mentre `\gdef` e `\xdef` eseguono definizioni globali; `\def` e `\gdef` eseguono le definizioni senza sviluppare le macro eventualmente presenti nella ⟨*definizione*⟩, mentre `\edef` e `\xdef` le sviluppano.

10. `\NumDoc` è un contatore interno definito 'alla `plain` TEX', con il quale è possibile fare le assegnazioni in modo più agevole, ma non li si può usare 'alla LATEX', per esempio, per i riferimenti incrociati.

in una macro temporanea `\@tempA`; questa viene confrontata con la macro vuota `\empty`; se entrambe sono vuote, allora assegna il valore di default, altrimenti assegna il parametro specificato.

La lista dei docenti è più complicata da separare nei suoi elementi, perché i nominativi dei docenti sono in numero imprecisato e, memoria dell'elaboratore permettendo, in numero arbitrariamente grande. Quindi bisogna estrarne uno alla volta e ciclicamente verificare che cosa resta nella lista. Ma per ciascun nominativo estratto bisogna anche comporre la riga nel registro e raccogliere i dati numerici di impegno per ogni tipo di attività eseguito dal docente che si sta elaborando.

Un primo comando carica la lista dei docenti estraendola dall'argomento fra parentesi quadre:

```
\def\set@docente[#1]{%
                \edef\lista@docenti{#1}%
% seguono altri comandi inessenziali per
% questa esposizione; infine si esegue:
\componi@scatole}
```

Successivamente la `\lista@docenti` viene usata dal comando ciclico che ripete la composizione della riga dell'intercalare per ogni docente elencato:

```
\def\componi@scatole{%
\@ripeti\@@@docente:=\lista@docenti\do{%
\expandafter\getcognome\@@@docente!%
\expandafter\ifx
  \csname\Cognome\endcsname\relax%
  \expandafter\gdef
  \csname\Cognome\endcsname{EA=0}%
\fi
\start@@boxes\lez@i\lez@ii\lez@iii%
             \lez@iv\lez@v\lez@vi\lez@vii
\emettiscatola
}}

\def\@ripeti#1:=#2\do#3{%
  \expandafter\def\expandafter%
     \@tempA\expandafter{#2}%
  \ifx\@tempA\@empty \else
  \expandafter\@riciclo#2,,\@@#1{#3}\fi
}%

\def\@riciclo#1,#2,#3\@@#4#5{\def#4{#1}%
#5%
\def#4{#2}%
\ifx #4\@empty
\expandafter\@finisci%
\else
      \expandafter\@riciclo\fi
      #2,#3,\@nil\@@#4{#5}%
}

\def\@finisci#1\@@#2#3{}%
```

Il comando `\componi@scatole` esamina la lista e per ogni docente compreso nella lista definisce `\@@@docente`; si noti il costrutto:

`\@ripeti`⟨*macro*⟩`:=`⟨*lista*⟩`\do{`⟨*comandi*⟩`}`

dove ⟨*macro*⟩ e ⟨*lista*⟩ sono argomenti delimitati.

Questo è un tipico costrutto informatico per ripetere una serie di ⟨*comandi*⟩ finché una certa variabile segue una certa enumerazione rappresentata da numeri, da lettere, o da elementi ordinati in una lista. Il nucleo di LaTeX contiene una simile macro `\@for` la cui sintassi è identica a quella di `\@ripeti`, ma il cui svolgimento è diverso, perché è prevista per un uso più generale. Nello stesso tempo i delimitatori `:=` e `\do`, al di là della loro funzione strettamente meccanica di delimitatori, esprimono concetti che al compositore sono più chiari di qualunque altro segno scelto a caso, anche se potrebbe svolgere la stessa funzione delimitatrice.

`\@ripeti` inizialmente esamina la lista dopo averla copiata in `\@tempA`; se questa è vuota, non fa nulla, ma se contiene qualcosa allora procede e passa i suoi argomenti al comando ad argomenti delimitati `\@riciclo`; si vede che la lista su cui operare è prolungata con altri due elementi vuoti e il delimitatore finale è costituito dal delimitatore `\@@` prima di indicare gli ultimi due argomenti non delimitati.

`\@riciclo` prende cinque argomenti, i primi tre dei quali sono delimitati da virgole e terminati con `\@@`; ma si noti: i primi tre argomenti contengono rispettivamente il primo nominativo, il secondo nominativo (eventualmente vuoto) e poi la lista residua dei nominativi dopo i primi due (la lista residua potrebbe essere costituita da una lista vuota, contenente solo le virgole).

Viene assegnato il primo nominativo alla ⟨*macro*⟩ che nel nostro caso è `\@@@docente`, poi si esegue la lista di comandi contenuta nel quinto argomento. Fatto questo la macro deve decidere se proseguire o terminare.

Per questo scopo assegna il secondo elemento della lista al quarto argomento, cioè a `\@@@docente`; se questo è vuoto non bisogna continuare, altrimenti bisogna cominciare un nuovo ciclo; nello stesso tempo bisogna anche usare gli argomenti residui.

Se dunque il prossimo nominativo è vuoto, viene chiamato il comando `\@finisci` che si mangia tutta la lista rimanente e tutti gli argomenti rimanenti senza fare assolutamente nulla d'altro; altrimenti viene chiamato di nuovo il comando `\@riciclo` accodando alla lista un oggetto qualsiasi, che non verrà mai esaminato, perché nella lista è preceduto da almeno un nominativo vuoto.

Un altro uso particolare delle macro con argomenti delimitati è quello con il quale viene costruita una lista di informazioni nella macro identica al cognome del docente. Si riprendono alcuni comandi riportati sopra, ma sui quali si è evitato di fare commenti:

```
\expandafter\ifx
  \csname\Cognome\endcsname\relax%
```

```
    \expandafter\gdef
    \csname\Cognome\endcsname{EA=0}%
  \fi
```

Questa serie di brevi comandi serve per definire una macro col nome identico al cognome del docente: se il nominativo del docente era `Mario Rossi`, il comando `\Cognome`, estratto con `\get@cognome`, contiene `Rossi` come sua definizione.

Ricordiamo che i comandi `\csname` e `\endcsname` racchiudono una stringa con la quale costruiscono il nome di una macro; l'`\expandafter` messo prima del test `\ifx` serve per sviluppare `\csname` prima di eseguire il test; `\csname` esamina i token successivi fino al comando `\endcsname` e, se ce ne è qualcuno sviluppabile, lo sviluppa; nel nostro esempio sviluppa `\Cognome` e quindi la coppia `\csname` e `\endcsname` crea il nome della macro `\Rossi` con la particolarità che le assegna di default l'equivalenza con `\relax` nel caso che questo nome di macro non abbia già una definizione: la stessa cosa succede con il comando successivo. Tutto il costrutto nel nostro esempio diventa quindi equivalente a:

```
\ifx\Rossi\relax \gdef\Rossi{EA=0}\fi
```

Tutta quella lunga sequela di comandi diventa un semplice trucco per procedere nella stessa maniera qualunque sia il cognome del docente.

In relazione alle righe 2, 3 e 4 della figura 3, dopo l'esecuzione dei comandi ciclici eseguiti nello sviluppo della lista di nominativi, esisteranno le macro `\Bianchi` che vale `EA=0,EA=2`; `\Rossi` che vale `LL=2,EA=2` grazie al fatto che prima di quella esercitazione in aula il docente aveva già fatto una lezione di due ore, e `\Neri` che vale `EA=0,EA=2`.

Tutte queste liste verranno poi esaminate come argomenti delimitati da altre macro; i delimitatori ora sono alternativamente un segno di `=` e una virgola; quest'ultima serve per isolare un elemento alla volta, mentre il primo serve per separare il tipo di attività didattica dalle ore svolte. Nuovamente due macro ad argomenti delimitati serviranno per estrarre gli elementi della lista e, all'interno di queste, il codice dell'attività e il tempo corrispondente. Esse serviranno per comporre l'ultima tabella della figura 2 per inserire i numeri giusti nelle caselle corrette e per fare le somme per colonne e per righe.

6 Conclusione

Nel TEXbook, (Knuth, 1990, capitolo 20 e appendice D), si descrivono le macro con le definizioni eseguite mediante i comandi primitivi; vi si trovano anche molti esempi di uso di simili macro. Il file `latex.ltx` (che si trova nella distribuzione del sistema TEX all'indirizzo `.../texmf-dist/tex/latex/base/`) ne contiene moltissime altre; ovviamente si possono ristudiare le macro appena descritte nel file `registro07.dtx`, Beccari (2007) o meglio, nella sua composizione mediante LATEX.

Quello che si è esposto qui non serve per spaventare i possibili utenti, ma per indirizzare gli scrittori di file di classe o di estensione verso comandi che rendono il codice più leggibile e quindi più facile da correggere o da modificare.

Si consiglia di usare delimitatori intelligibili; non sempre è necessario, ma più complicato è un comando, tanto più espliciti dovrebbero essere i nomi dei comandi e quelli dei delimitatori; così facendo, il codice sarà immediatamente comprensibile senza bisogno di complicati commenti o frasi circonvolute.

Certamente si sconsiglia vivamente di usare questi comandi nel corpo di un file sorgente di un documento da comporre; se li si usa, è bene farlo in file di classe o di estensione. D'altra parte Leslie Lamport, il padre di LATEX, è stato piuttosto esplicito: egli ha definito e usato moltissimi comandi con argomenti delimitati nel nucleo di LATEX, ma non ne ha lasciato praticamente nessuno a disposizione dei compositori; questi li possono usare (senza saperlo: per esempio quando usano il comando `\cline` per comporre tabelle; quale delimitatore viene usato?) e non ha creato nessun comando di alto livello per definirli. Questo è un segno evidente che si tratta di cose delicate da non lasciare in mano al compositore, il quale è invece invitato a concentrarsi sul contenuto, piuttosto che su come comporlo.

Riferimenti bibliografici

Beccari, C. (2007). «Il pacchetto registro07». `registro07.dtx`. URL `ftp://ftp.polito.it/pub/tex/polito/registro/registro07.zip`.

Knuth, D. E. (1990). *The TEXbook*, volume A di *Computers and Typesetting*. Addison Wesley, 18ª edizione.

▷ Claudio Beccari
`claudio dot beccari at alice.it`

HyPlain, più lingue insieme anche in Plain

Enrico Gregorio

Sommario

Si descrive un sistema per abilitare la cesura a fine riga in più lingue usando Plain TEX, insieme a un'interfaccia per definire lingue e le loro convenzioni.

Abstract

We describe a system for enabling hyphenation in several languages under Plain TEX, along with an interface to define the used languages and their conventions.

1 Introduzione

Nella giovinezza di TEX, cioè prima della versione 3, ogni formato poteva contenere l'insieme delle regole di cesura a fine riga per una sola lingua. Con un trucco piuttosto oscuro era in realtà possibile inserire le regole per due lingue, ma la faccenda era davvero complicata.

A quell'epoca LATEX era poco usato per lavori di matematica, dato che gli ambienti per il trattamento delle formule si riducevano a equation e eqnarray. Di buono aveva la numerazione automatica e i riferimenti incrociati; troppo poco per farlo preferire a $\mathcal{AMS}$-TEX, il formato basato su Plain creato da Michael Spivak per l'American Mathematical Society.

Come si faceva per avere le cesure corrette in italiano, visto che la versione standard di Plain forniva le regole per l'inglese americano? C'erano due modi: la prima soluzione era di ignorare il problema, con cesure inaccettabili come 'fas-tidio'. La seconda era di modificare il formato Plain in modo che caricasse le regole per la cesura dell'italiano che Claudio Beccari aveva prodotto.

Il formato Plain viene creato con iniTEX dal documento `plain.tex` il quale a un certo punto contiene la riga

```
\input hyphen
```

e bastava sostituirla con

```
\input ithyph
```

rinominando il documento come `plainit.tex`.

Ai tempi la riga di comando non faceva molta paura: non c'erano altri modi per lavorare, su certe piattaforme! Per chi usava il Macintosh c'erano OzTEX o *Textures* con i quali creare un nuovo formato era molto semplice.

È bene ricordare una delle particolarità di TEX. All'avvio del programma, con un metodo che dipende dal sistema operativo e dalla distribuzione, l'interprete TEX (che può anche essere pdfTEX) carica un *formato*, cioè uno stato iniziale. La maggior parte degli utenti usa LATEX, ma non si è certo ristretti a questo. La creazione del formato coinvolge l'uso di iniTEX, una versione speciale del programma, ed è eseguita automaticamente, di solito in fase di installazione; tuttavia a volte è necessario rifarla, per esempio proprio per abilitare nuove lingue con babel in LATEX. In questo articolo si farà riferimento a TEX, ma tutto vale anche per pdfTEX o XETEX.

Nel 1989 arrivò TEX versione 3 (Knuth, 1989), con la possibilità di inserire in un singolo formato le regole di cesura anche per 256 lingue diverse e subito si cercò di sfruttare l'opportunità per creare formati compatibili con più lingue. Il documento più semplice su cui far girare iniTEX per ottenere un formato in cui sia possibile scrivere correttamente in inglese e in italiano è

```
\input plain
\language1
\input ithyph
\language0
\def\english{\language0 }
\def\italian{\language1 }
\dump
```

Con il formato creato da questo si possono avere le cesure corrette sia in inglese americano che in italiano, basta dichiarare all'inizio del proprio documento la lingua preferita. Si può anche avere un documento in cui le lingue coesistono, basta dichiarare la lingua in un gruppo.

Non è qui il luogo per un'analisi di come funziona la scelta delle regole di cesura; basti sapere che TEX tiene conto del valore assegnato alla variabile `\language` e che

```
\language=0
\language0
```

sono modi del tutto equivalenti di assegnare a questa variabile il valore zero. TEX associa un insieme di regole al valore della variabile; perciò, con l'esempio precedente, quando `\language` vale zero si useranno le regole inglesi (lette tramite `\input plain`), mentre quando vale uno si useranno quelle italiane. Conseguenza di questo è che per disabilitare le cesure è sufficiente assegnare il valore 255 a `\language`, visto che è alquanto improbabile che si siano definite le regole per 256 lingue,

questo vale anche in LaTeX. Tuttavia, se il valore è un intero non compreso fra 0 e 255, verranno usate le regole come per il valore 0. Si veda l'Appendice H del TeXbook.

È evidente che l'interfaccia proposta è troppo primitiva, in realtà occorre fare qualcos'altro; per esempio, per avere le cesure corrette con parole precedute dall'apostrofo in italiano è necessario dare il comando

```
\lccode‘\’=‘\’
```

perché TeX abbandona la sillabazione se in una sequenza di caratteri ne compare uno il cui `\lccode` sia zero.[1] Inoltre l'italiano permette di spezzare anche le ultime due lettere di una parola, a differenza dell'inglese, che ne richiede almeno tre sulla nuova riga. Dunque occorre che la definizione di `\italian` sia modificata con

```
\def\italian{\language1
  \lccode‘\’=‘\’
  \righthyphenmin2 }
```

Analogamente occorre ridefinire `\english` per disfare ciò che `\italian` può avere fatto:

```
\def\english{\language1
  \lccode‘\’=0
  \righthyphenmin3 }
```

Ovviamente la situazione si complica di molto se invece di due sole lingue se ne hanno di più, ciascuna con le sue convenzioni e necessità. Si pensi al francese che richiede spaziature prima di certi segni di interpunzione e molte altre cose.

Daniel Flipo e Laurent Siebenmann crearono proprio per le esigenze del francese il pacchetto HyMaster (Siebenmann, 1993), che risolveva in modo decente certi problemi. A partire da quello mi creai un sistema meno complesso che poteva servire le mie esigenze e quelle dei colleghi, allora a Catania. Poi me ne dimenticai, perché nel frattempo LaTeX aveva incorporato, tramite il pacchetto amslatex, le funzionalità di $\mathcal{AMS}$-TeX. Inoltre babel (Braams, 1991) permetteva non solo di definire le regole di cesura per più lingue, ma anche di modificare 'al volo' le parole fisse.

Non c'era quindi più bisogno di rimanere ancorati a $\mathcal{AMS}$-TeX e la maggior parte dei matematici si trasferì a LaTeX, soprattutto con l'uscita di LaTeX 2_ε, nel 1994.

Fu con grande stupore che lessi sul forum del GuIt un intervento[2] in cui si chiedeva aiuto per installare un formato basato su Plain che riconobbi come le mie antiche macro! Da quella discussione si sviluppò l'idea di migliorare quel pacchetto, in modo da rendere più facile il passaggio da una lingua all'altra e offrire un'interfaccia per aggiungere codice da eseguire quando si passa a una lingua diversa.

Lo scopo di questo articolo non è tanto di descrivere un pacchetto di uso probabilmente molto limitato, quanto di scoprire qualcuno dei trucchi che ho usato e che possono essere utili anche in altre situazioni.

2 Il pacchetto HyPlain

Il pacchetto consiste di tre documenti: `hyplain`, `hyrules` e `hylang`, tutti con estensione `.tex`. Il primo è quello su cui si fa girare iniTeX, il secondo quello in cui si definisce l'interfaccia utente. Il terzo è modificabile dall'utente, che può scegliere quali lingue abilitare e definire le convenzioni da usare nelle varie lingue.

Non discuterò qui l'installazione del formato prodotto nelle varie distribuzioni, visto che chi usa Plain o i suoi derivati è abituato a mettere le mani a basso livello. Su un sistema Unix, per esempio con TeX Live, è sufficiente la seguente sequenza di comandi:

```
$ tex -ini hyplain
...
Beginning to dump on file hyplain.fmt
...
$ ln -s ‘which tex‘ hyplain
```

per avere nella directory sia il formato che un eseguibile richiamabile dalla linea di comando con `./hyplain`. Si può naturalmente trasferire il file `hyplain.fmt` e l'eseguibile in luoghi opportuni, quali siano dipende dalla distribuzione.

Su questo formato è possibile poi sovrapporre $\mathcal{AMS}$-TeX al modo usuale:

```
$ tex -ini -jobname hyamstex \&hyplain
...
* \input amstex
...
* \dump
...
Beginning to dump on file hyamstex.fmt
...
$
```

(con `$` indico l'invito della shell, con `*` quello di TeX e con `...` riassumo i messaggi inviati sullo schermo).

Il contenuto di `hyplain.tex` è molto semplice:

```
1 \catcode‘\{=1
2 \catcode‘\}=2
3 \catcode‘\@=11
4
5 \let\@@input\input
6
7 \def\input hyphen {%
8   \let\input\@@input
```

1. La notazione '\' permette di riferirsi al codice ASCII del carattere che compare dopo la barra rovescia.

2. http://www.guit.sssup.it/phpBB2/viewtopic.php?t=1451

```
9   \input hyrules }
10
11 \@@input plain
12
13 \def\fmtname{plain %
14   (multiple language support)}
15 \def\hyplainversion{1.0}
16
17 \dump
```

Le righe 1 e 2 servono per definire le graffe come caratteri di inizio e fine dei gruppi, perché questo non è stabilito da iniTEX. La riga 3 serve per dichiarare `@` come lettera, al modo solito, per definire comandi interni.

Nella riga 5 memorizzo il significato di `\input`. Infatti voglio evitare che sia caricato il file `hyphen.tex`. Una delle consegne di Knuth è che il file `plain.tex` non sia modificato che da lui stesso; in questo file perciò ci sarà sempre (a meno che Knuth non cambi idea) la riga

```
\input hyphen
```

Con le righe 7–9 ridefinisco `\input` come una macro delimitata dalla stringa `hyphen␣`, il cui effetto è di ridare a `\input` il suo significato originale e di caricare `hyrules`.

A seguire carichiamo `plain.tex` e durante l'elaborazione di questo verrà eseguita la macro (ridefinita) `\input`, con il risultato detto prima.

Il resto del file è chiaro: si definiscono due macro che contengono un'identificazione del pacchetto e poi si esegue `\dump` che scrive il file `hyplain.fmt`.

3 hyrules.tex

Esaminiamo ora il secondo file del pacchetto, quello dove vengono introdotti i comandi dell'interfaccia per usare più lingue.

```
1 \chardef\myh@us@catcode=\the\catcode‘\_
2 \catcode‘\_=11
3 \let\@xp\expandafter
4 \newlinechar‘^^J
```

Fin qui nulla di complicato: si salva il codice di categoria del carattere `_` che poi diventa una lettera, che userò per alcuni comandi interni. Si veda Gregorio (2006) per informazioni sui codici di categoria. Introduco anche un'abbreviazione per rendere meno pesante il codice che segue e poi definisco il carattere che, nei messaggi sullo schermo, provoca un 'a capo'.

Ogni lingua che sarà caricata con `hylang.tex` è identificata da una macro della forma `\⟨stringa1⟩_⟨stringa2⟩`; queste stringhe possono essere arbitrarie, ma consiglio di usare i codici ISO di lingua e nazione; per esempio, l'inglese americano sarà identificato da `\en_US`, l'italiano in Italia sarà `\it_IT` e l'italiano in Svizzera `\it_CH`.

Naturalmente l'utente non potrà scrivere direttamente questi comandi, a cui si accede indirettamente; infatti il codice di categoria di `_` ritornerà quello usuale.

Uno di questi comandi è riservato: `\zz_ZZ` indica una lingua *universale*, nella quale non sono definite regole di cesura e che verrà usata in mancanza di una definizione in `hylang`. Trovo che questa sia una strategia migliore di quella di babel, che invece usa le cesure dell'inglese americano per quelle lingue di cui TEX non possieda le regole.

Compariranno spesso costrutti del tipo

```
\@xp\let\@xp\foo\csname bar\endcsname

\@xp\edef\csname bar\endcsname{xyz}
```

in cui `bar` contiene i parametri `#1` e `#2`. Non c'è da prenderne paura: il primo equivale a

```
\let\foo\bar
```

perché il primo provoca l'espansione del secondo che a sua volta espande `\csname`. Il secondo equivale a

```
\edef\bar{xyz}
```

per motivi analoghi. Ricordo che `\edef` espande ciò che è contenuto nelle graffe, ma nel caso di `\the` seguito da un registro token si limita ad estrarne il contenuto. Inoltre

```
\csname ...\endcsname
```

produce una sequenza di controllo il cui nome consiste dei caratteri che risultano dall'espansione dei token compresi fra i due comandi, senza distinzione sui codici di categoria. È importante ricordare che dall'espansione devono risultare solo caratteri. Perciò, se l'argomento `#1` è `ab` e `#2` è `CD`, l'espansione di

```
\csname #1_#2\endcsname
```

produrrà un token equivalente alla sequenza (in genere illegale) `\ab_CD`.

```
5 \def\selectlanguage#1#2{%
6   \@xp\ifx\csname #1_#2\endcsname\relax
7     \global\@xp\let
8       \csname #1_#2\endcsname\zz_ZZ
9     \myh@nel{#1}{#2}%
10  \fi
11  \csname #1_#2\endcsname
12  \ignorespaces}
```

La macro `\selectlanguage`, come in babel, è quella che sceglie una nuova lingua. A differenza del comando di babel ha due argomenti, le due stringhe di cui si compone l'identificativo. Se si scrive

```
\selectlanguage{ab}{CD}
```

la macro esamina per prima cosa se il comando `\ab_CD` è definito. Le regole di TEX dicono che se

```
\csname ab_CD\endcsname
```

non corrisponde a un comando definito, il suo valore equivale a `\relax`; dunque, se `\ab_CD` non è definito, il condizionale è vero e si rende `\ab_CD` equivalente a `\zz_ZZ`, emettendo un messaggio di avviso. Poi il comando `\ab_CD` viene eseguito e ogni spazio che segua il secondo argomento di `\selectlanguage` è ignorato.

```
\def\myh@nel#1#2{%
  \message{^^Jhyplain warning:
    Language #1 for #2 undefined,
    using fallback instead^^J}}
```

La macro `\mhy@nel` contiene il messaggio di avviso che la lingua scelta non è definita, starà all'utente correggere l'errore.

```
\def\addalias#1#2#3{%
  \@xp\let\@xp#1\csname #2_#3\endcsname}
```

Questa macro è l'interfaccia per definire comandi abbreviati o più espliciti per la scelta di una lingua; per esempio si potrebbe dare, nel proprio documento o in `hylang`,

```
\addalias\IT{it}{IT}
\addalias\usenglish{en}{US}
```

Così `\IT` e `\usenglish` diventano equivalenti, rispettivamente, a

```
\selectlanguage{it}{IT}
\selectlanguage{en}{US}
```

Ora viene la parte più complessa del pacchetto: quando si sceglie una lingua si deve anche eseguire un certo codice, come abbiamo visto prima, e questo codice va anche 'neutralizzato' quando si passa a una lingua diversa.

Un modo potrebbe essere quello di scrivere il comando `\selectlanguage` o le abbreviazioni sempre dentro un gruppo; ma non si può contare su questo, che potrebbe anche diventare pesante per l'utente. Si pensi a un testo scritto prima in inglese e poi in italiano, per un lungo tratto, e poi ancora in inglese: occorrerebbe mettere un inizio di gruppo quando si comincia a scrivere in italiano e poi ricordarsi di chiuderlo e questo potrebbe portare a errori a volte incomprensibili.

Molto meglio definire un insieme di comandi da eseguire cominciando a scrivere in una lingua e il codice che annulla queste modifiche, da eseguire quando si comincia a scrivere in una lingua diversa. Se poi `\selectlanguage` è dato in un gruppo, le modifiche saranno annullate automaticamente. Li chiameremo, rispettivamente, codice *IN* e *OUT*; saranno ovviamente diversi per ciascuna delle lingue abilitate.

Una strategia analoga, e più complessa, è usata da babel, che però ha molte cose di cui tenere conto, come i testi fissi, i registri, i caratteri attivi e così via (Gregorio, 2007). Molti di questi aspetti vengono lasciati da HyPlain all'utente, che si suppone più addentro alla programmazione TEX di quanto non sia quello normale di LATEX.

In tutto quanto segue supporremo sempre che la lingua da considerare abbia l'identificativo `\ab_CD`. Oltre a questa macro ne vengono definite altre due, `\extras@ab_CD` e `\noextras@ab_CD`; la prima contiene il codice *IN* di quella lingua, la seconda il codice *OUT*.

```
\newtoks\myh@toks
\def\addto#1#2#3#4{%
  \@xp\let\@xp\myh@temp
    \csname extras@#1_#2\endcsname
  \myh@toks=\@xp{\myh@temp}%
  \myh@toks=\@xp{\the\myh@toks #3}
  \@xp\edef\csname extras@#1_#2\endcsname
    {\the\myh@toks}%
  \@xp\let\@xp\myh@temp
    \csname noextras@#1_#2\endcsname
  \myh@toks=\@xp{\myh@temp}%
  \myh@toks=\@xp{\the\myh@toks #4}
  \@xp\edef\csname noextras@#1_#2\endcsname
    {\the\myh@toks}%
}
```

Si comincia allocando un registro token. La macro `\addto` ha quattro argomenti:

1. la prima parte dell'identificativo,
2. la seconda parte dell'identificativo,
3. il codice 'in ingresso', che denoteremo con *IN*,
4. il codice 'in uscita', che denoteremo con *OUT*.

Si definisce una macro temporanea equivalente a `\extras@ab_CD`, poi si carica il registro token con la sua espansione di primo livello; l'istruzione successiva aggiunge al contenuto del registro il terzo argomento, cioè il codice *IN*, e, finalmente, si ridefinisce `\extras@ab_CD` con il contenuto di questo registro.

Per non essere troppo astratti, l'effetto sarà questo, se `\extras@ab_CD` vale `\foo\bar=2␣` e il codice *IN* è `\xyz=9␣`: alla fine del processo, sarà come avere dato

```
\def\extras@ab_CD{\foo\bar=2 \xyz=9 }
```

Lo stesso si fa al comando `\noextras@ab_CD` con il codice *OUT*. Il comando ha quindi effetto cumulativo.

```
\def\definebaselanguage#1#2#3{%
  \message{***^^JDefining language
    #1 for #2 using^^J}%
  \@xp\chardef\csname l@#1_#2\endcsname=0
  \language\csname l@#1_#2\endcsname
```

```
  \input #3%
  \message{This is \string\language
    \@xp\the\csname l@#1_#2\endcsname
    ^^J***^^J}%
  \myh@initialize{#1}{#2}}
\def\definelanguage#1#2#3{%
  \message{***^^JDefining language
    #1 for #2 using^^J}%
  \csname newlanguage\@xp\endcsname
    \csname l@#1_#2\endcsname
  \language\csname l@#1_#2\endcsname
  \input #3%
  \message{This is \string\language
    \@xp\the\csname l@#1_#2\endcsname
    ^^J***^^J}%
  \myh@initialize{#1}{#2}}
```

I due comandi definiti in questa parte di codice sono molto simili. In effetti il primo si usa una volta sola per definire la lingua di base che è opportuno sia l'inglese americano.

A ogni lingua è associato un intero fra 0 e 255, il cui valore è contenuto nel comando `\l@ab_CD`; la differenza fra il primo e il secondo comando è che il primo associa alla lingua che si definisce l'intero 0, mentre il secondo impiega il meccanismo di allocazione definito in Plain TEX tramite `\newlanguage`. Ci sono tre argomenti:

1. la prima parte dell'identificativo,
2. la seconda parte dell'identificativo,
3. il nome del file che contiene le regole per la cesura.

Si comincia associando alla lingua l'intero; se la macro è chiamata tramite

```
\definelanguage{ab}{CD}{abhyphen}
```

l'effetto complessivo (oltre all'emissione di un paio di messaggi) è di eseguire

```
\newlanguage\l@ab_CD
\language\l@ab_CD
\input abhyphen
\myh@initialize{ab}{CD}
```

Nel caso di `\definebaselanguage` il primo comando eseguito sarebbe

```
\chardef\l@ab_CD=0
```

Infatti un comando definito tramite `\chardef` si comporta come un numero quando TEX se ne aspetta uno.

È importante emettere `\language\l@ab_CD` prima di caricare il file delle regole di cesura, perché iniTEX memorizza queste regole nel formato proprio associandole al valore della variabile `\language` in vigore. Discuteremo fra poco l'ultimo comando, che serve a eseguire gli opportuni codici *OUT* e *IN*.

C'è una curiosità alle righe 15–16; il comando `\newlanguage` è definito in Plain come `\outer`, quindi non può comparire così com'è nel testo di sostituzione in `\def`. Con il trucco di immergerlo in `\csname...\endcsname`, TEX non *vede* il comando proibito. Inoltre possiamo sfruttare il fatto che ciò che compare fra `\csname` e `\endcsname` viene espanso; perciò l'`\expandafter` provoca l'espansione del comando che segue (`\csname`), risparmiandoci un po' di lavoro; l'effetto sarebbe equivalente, nel solito esempio, a

```
\newlanguage\l@ab_CD
```

```
\def\definedialect#1#2#3#4{%
  \@xp\let\csname l@#1_#2\@xp\endcsname
    \csname l@#3_#4\endcsname
  \myh@initialize{#1}{#2}}
```

Alcune lingue possono condividere le regole di cesura; l'esempio classico è il tedesco parlato in Germania e in Austria: sono la stessa lingua, ma con convenzioni leggermente diverse. Per esempio, in Germania il mese di gennaio si chiama *Januar*, in Austria è *Jänner*. Con

```
\definedialect{ab}{CD}{ab}{EF}
```

si definisce effettivamente una nuova macro `\ab_EF` con le due associate a cui assegnare diversi codici *IN* e *OUT*, ma questo dialetto si comporta come la lingua di base per quanto riguarda le regole di cesura.

```
\def\refinelanguage#1#2#3#4{%
  \@xp\def\csname extras@#1_#2\endcsname
    {#3}%
  \@xp\def\csname noextras@#1_#2\endcsname
    {#4}%
}
\let\refinedialect=\refinelanguage
```

Nel file `hylang` l'utente dovrebbe provvedere i codici *IN* e *OUT* iniziali per le lingue che definisce. Questo viene fatto tramite `\refinelanguage`. L'utente potrà aggiungere codice tramite `\addto` nel proprio documento. Gli argomenti sono come già descritto per `\addto` e non c'è bisogno di altre spiegazioni.

```
\def\myh@initialize#1#2{%
  \@xp\def\csname #1_#2\endcsname{%
    \language\csname l@#1_#2\endcsname
    \myh@undo
    \def\myh@lan{#1_#2}%
    \csname extras@#1_#2\endcsname}%
  \@xp\def\csname extras@#1_#2\endcsname{}%
  \@xp\def\csname noextras@#1_#2\endcsname{}}
\def\myh@undo{%
  \csname noextras@\myh@lan\endcsname}
```

La macro `\myh@initialize` è la principale del pacchetto. I suoi argomenti sono le componenti dell'identificativo della lingua. Con `\myh@initialize{ab}{CD}` vengono eseguiti i seguenti comandi:

```
\def\ab_CD{%
  \language\l@ab_CD
  \myh@undo
  \def\myh@lan{ab_CD}%
  \extras@ab_CD}
\def\extras@ab_CD{}
\def\noextras@ab_CD{}
```

Si definiscono cioè le macro `\ab_CD` e le due associate. Quando poi `\selectlanguage{ab}{CD}` esegue la macro `\ab_CD`, succedono le seguenti cose:

1. si passa alla lingua prescelta;
2. si esegue la macro `\myh@undo`;
3. si definisce la macro `\myh@lan` che contiene l'identificativo della lingua prescelta;
4. si esegue il codice *IN* relativo.

Un'altra curiosità è la mancanza di `\expandafter` prima di `\language`; questo comando è primitivo ed è uno di quelli che si aspettano di essere seguiti da un numero, quindi provocano l'espansione del token successivo. Analogamente, alla riga 7, `\global` è una primitiva che deve essere seguita da un comando di assegnazione e quindi anch'essa espande il token successivo.

La macro `\myh@undo` è definita come

```
\csname noextras@\myh@lan\endcsname
```

e, quando è eseguita, causa l'espansione di `\myh@lan`, quindi produce l'identificativo della lingua precedentemente in uso, come abbiamo visto poco fa e quindi l'esecuzione del codice *OUT* della lingua che si sta abbandonando.

Facciamo un paio di esempi. Supponiamo che l'utente scriva

```
\selectlanguage{ab}{CD}
testo
\selectlanguage{ef}{GH}
altro testo
\selectlanguage{ab}{CD}
```

Il primo comando esegue `\ab_CD` e quindi il significato di `\myh@lan` è, a questo punto, `ab_CD`. Quando viene eseguita la scelta della seconda lingua, prima di `\extras@ef_GH` viene espansa `\myh@undo`, cioè precisamente la macro `\noextras@ab_CD`. In seguito il comando `\myh@undo` si espanderà, come si desidera, in `\noextras@ef_GH`, per annullare, al terzo `\selectlanguage`, gli effetti del codice *IN* relativo alla lingua `\ef_GH`.

Se invece l'utente scrive

```
\selectlanguage{ab}{CD}
testo
{\selectlanguage{ef}{GH}
altro testo}
```

il testo nel gruppo sarà composto secondo le regole valide per la lingua `\ef_GH`. La macro `\myh@undo` verrà ridefinita all'interno del gruppo, ma alla sua chiusura riotterrà il valore precedente.

Dunque, in ogni caso, gli effetti del codice *IN* relativo alla seconda lingua scelta, verranno annullati, per effetto di comandi espliciti oppure per via dell'uscita da un gruppo.

```
76 \def\hyphenmins#1#2{%
77   \lefthyphenmin=#1\relax
78   \righthyphenmin=#2\relax}
79
80 \input hylang
81
82 \let\definebaselanguage\undefined
83 \let\definelanguage\undefined
84
85 \def\myh@lan{} % initialization
86 \def\noextras@{} % initialization
87 \en_US
88
89 \catcode`\_=\myh@us@catcode
```

La fine del file `hyrules` contiene cose semplici; il primo comando è l'interfaccia per definire la minima lunghezza dei tronconi di parola dopo la cesura; andrà usato nel codice *IN* di ogni lingua e non sarà quindi necessario annullarne gli effetti nel codice *OUT*.

A questo punto viene caricato `hylang.tex` di cui ci occuperemo in seguito. Poi i comandi per le definizioni di nuove lingue sono disabilitati, perché non ha senso farlo se non quando si compila con iniTEX, e si definiscono due comandi per non far arrabbiare TEX all'esecuzione di `\en_US`. Infatti a questo punto non c'è alcuna lingua precedente e, se si guarda il codice di cui abbiamo discusso finora, ci si rende conto della necessità di queste definizioni.

Per ultimo, il codice di categoria di `_` viene riportato a quello in vigore quando si è cominciato.

4 hylang.tex

Questo è il documento modificabile dall'utente a seconda delle sue esigenze. Qui vengono caricate le lingue che si vogliono usare; se si desidera cambiare la scelta, occorrerà modificare questo file e ricreare il formato `hyplain.tex`.

```
1 %%% US English must always come first
2 \definebaselanguage{en}{US}{hyphen}
3 \definelanguage{zz}{ZZ}{zerohyph}
4 %%% don't modify the lines above
5 \refinelanguage{en}{US}{\hyphenmins{2}{3}}{}
6 \refinelanguage{zz}{ZZ}{}{}
7
8 %%% Italian
9 \definelanguage{it}{IT}{ithyph}
10 \refinelanguage{it}{IT}
11   {\hyphenmins{2}{2}\lccode`\'=`\'}
12   {\lccode`\'=0 }
```

```
13
14 %%% Add other languages if needed
15
16 %%% Aliases
17 \addalias\US{en}{US}
18 \addalias\IT{it}{IT}
19 \addalias\ZZ{zz}{ZZ}
20 \addalias\nohyphens{zz}{ZZ}
```

Le prime due righe non commentate sono quelle per definire la lingua di base che è, come già detto, l'inglese americano. Poi vediamo un esempio per l'italiano, con l'impostazione dei minimi di cesura e il cambiamento del codice `\lccode` per l'apostrofo.

Nelle moderne distribuzioni è presente il file `zerohyph.tex` che non definisce alcuna regola di cesura. Scegliendo questa 'lingua' si inibisce del tutto la sillabazione a fine riga. Come detto, questa è la lingua 'universale' che viene scelta se si dà un comando per una lingua sconosciuta.

Per finire, alcuni sinonimi: per esempio, invece di `\selectlanguage{it}{IT}` è possibile scrivere semplicemente `\IT`.

5 Commenti

L'interfaccia prevista è molto primitiva, ma ampliabile con una certa facilità. Supponiamo, per esempio, che si voglia, come nel modulo italiano di babel modificare i valori di `\clubpenalty` e `\widowpenalty`. Occorrerà modificare anche il codice *IN* per le altre lingue, in modo che i valori desiderati per esse siano ripristinati al cambio di lingua. Vogliamo anche che in italiano si usi la spaziatura 'francese', cioè che gli spazi dopo i segni di interpunzione siano come quelli fra le parole.[3]

Il contenuto di `hylang` dovrà diventare (togliendo le cose superflue)

```
\definebaselanguage{en}{US}{hyphen}
\definelanguage{zz}{ZZ}{zerohyph}

\refinelanguage{en}{US}
  {\hyphenmins{2}{3}%
   \clubpenalty150
   \widowpenalty150 }
  {}

\refinelanguage{zz}{ZZ}
  {\clubpenalty150
   \widowpenalty150 }
  {}

\definelanguage{it}{IT}{ithyph}
\refinelanguage{it}{IT}
  {\hyphenmins{2}{2}\lccode`\'=`\'
   \clubpenalty3000
   \widowpenalty3000
   \frenchspacing}
  {\lccode`\'=0
   \nonfrenchspacing}
```

3. Qualcuno obietta che non è questa la spaziatura francese; in effetti Knuth, con questo nome, intendeva riferirsi solo agli spazi *dopo* i segni di interpunzione.

L'utente che usa Plain è in genere abbastanza smaliziato per accorgersi di queste sottigliezze. Va in *OUT* solo il codice di cui si sa che annulla ciò che si produce con il codice *IN*. I valori dei parametri interni vanno sempre specificati nel codice *IN* e, se li si cambia in una lingua, occorre modificarli in tutte.

6 Sviluppi futuri

In babel c'è un meccanismo per conservare i valori delle variabili e dei comandi, probabilmente sarà incluso nella prossima versione di HyPlain. L'idea di babel è di mantenere in un comando il codice da eseguire a ogni cambio di lingua per ritornare a uno stato 'iniziale'.

Dovrebbe essere sufficiente aggiungere l'espansione di questo comando, che chiamerò `\myh@saved`, a ogni `\selectlanguage`, in modo da rimettere a posto valori delle variabili e significati delle macro modificate.

```
1 \def\myh@saved{}
2 \newcount\myh@savecnt
3 \def\hyplainsave#1{%
4   \@xp\let\csname myh@%
5     \number\myh@savecnt\endcsname #1\relax
6   \begingroup
7     \toks@\@xp{\myh@saved \let#1=}%
8     \edef\x{\endgroup
9       \def\noexpand\myh@saved{%
10        \the\toks@ \@xp\noexpand
11          \csname myh@\number\myh@savecnt
12            \endcsname\relax}}%
13   \x
14   \advance\myh@savecnt\@ne}
15 \def\hyplainsavevariable#1{\begingroup
16     \toks@\expandafter{\myh@saved #1=}%
17     \edef\x{\endgroup
18       \def\noexpand\myh@saved{%
19         \the\toks@ \the#1\relax}}%
20   \x}
```

È interessante vedere come funziona questo codice. Supponiamo di scrivere `\hyplainsave\pippo` e che il valore del contatore `\myh@savecnt` sia 25. La prima cosa che viene eseguita è

```
\let\myh@25\pippo
```

La sequenza `\myh@25` è illegale, se scritta così, ma in realtà accediamo a essa tramite `\csname...\endcsname` e, una volta che è entrata nel meccanismo di lettura e assorbita da TEX non è diversa dalle altre.

Nel seguito indicheremo con `<myh@saved>` il testo di sostituzione della macro `\myh@saved`.

Viene poi aperto un gruppo e in esso eseguito

```
\toks@\@xp{\myh@saved \let\pippo=}
```

cosicché il registro token temporaneo `\toks@` contiene

```
{<myh@saved> \let\pippo=}
```

Ora viene il bello. Si esegue un `\edef` che espande tutto ciò che c'è tra le graffe. Quindi questo diventa equivalente a

```
\def\x{\endgroup
  \def\myh@saved{<myh@saved>
    \let\pippo=\myh@25\relax}
```

La successiva esecuzione di `\x` ridefinisce `\myh@saved` che quindi conterrà il codice necessario a ridare a `\pippo` il suo significato originale. A questo punto il contatore viene fatto scattare di uno, in modo che il prossimo uso di `\hyplainsave` usi un comando diverso dal precedente, in questo caso `\myh@27`. Il gruppo viene chiuso durante l'espansione di `\x`, così TEX dimentica la definizione di questo e i valori assegnati nel gruppo.

Con `\hyplainsavevariable` la faccenda è analoga; se `\pluto` è un registro dimensionale il cui valore è 42 pt, il registro `\toks@` conterrà

```
{<myh@saved> \pluto=}
```

e la successiva definizione (con `\edef`) di `\x` equivale a

```
\def\x{\endgroup
  \def\myh@saved{<myh@saved>
    \let\pluto=42pt\relax}
```

In questo modo dovrebbe essere possibile far diventare il file `hyrules` dell'esempio precedente

```
\definebaselanguage{en}{US}{hyphen}
\definelanguage{zz}{ZZ}{zerohyph}

\refinelanguage{en}{US}
  {\hyphenmins{2}{3}}{}

\refinelanguage{zz}{ZZ}{}{}
\definelanguage{it}{IT}{ithyph}

\refinelanguage{it}{IT}
  {\hyphenmins{2}{2}\lccode`\'=`\'
   \clubpenalty3000
   \widowpenalty3000
   \frenchspacing}
  {\lccode`\'=0
   \nonfrenchspacing}

\hyplainsavevariable\clubpenalty
\hyplainsavevariable\widowpenalty
```

con evidente economia di pensiero: basta dichiarare tutte le variabili o comandi che si vogliono preservare. Questi comandi potranno anche essere dati nel documento dell'utente, cambiando i valori per mezzo di `\addto`.

Ogni suggerimento o richiesta di nuove funzioni è bene accetto: ogni tanto è bene giocare un po' con Plain TEX.

Riferimenti bibliografici

Braams, J. (1991). «Babel, a multilingual style-option system for use with LATEX's standard document styles». *TUGboat*, **12** (2), pp. 291–301.

Gregorio, E. (2006). «Codici di categoria». *ArsTEXnica*, **1**, pp. 5–14.

— (2007). «Enjoying babel». *TUGboat*, **28** (2), pp. 247–255.

Knuth, D. E. (1989). «The new versions of TEX and METAFONT». *TUGboat*, **10** (3), pp. 325–328.

Siebenmann, L. (1993). «A format compilation framework for European languages». *TUGboat*, **14** (3), pp. 212–221.

▷ Enrico Gregorio
Dipartimento di Informatica
Università di Verona
Enrico dot Gregorio at univr dot it

Eventi e novità

BachoTEX 2008

BachoTEX 2008, the XVI BachoTEX conference, will take place from April 30 until May 4, 2008. It is organized yearly since 1993 by the Polish TEX User Group – GUST, in Bachotek, near Brodnica, in the north-east of Poland.

BachoTEX 2008 will celebrate its XVI$^{\text{th}}$, or 10000_2 or 100_4 or 10_{16} jubilee!

To whet your appetite here is a list of some speakers (for the current list of papers and schedule see the link provided below):

- Hans Hagen – author of ConTEXt, co-author of LuaTEX
- Piotr Fuglewicz – Polish spellchecking expert;
- Jean-Michel Hufflen – XML expert, author of MlBibTEX;
- Bogusław Jackowski – Polish TEX and font guru, co-author of the Latin Modern and TEX Gyre families of fonts;
- David Kastrup – maintainer of AUCTEX, author of bigfoot and other LATEXpackages;
- Jonathan Kew – author of XETEX;
- Robert Oleś book designer, lecturer in book design, lead editor of the Polish edition of Bringhurst's "The Elements of Typographic Style";
- Marek Ryčko – Polish TEX guru and practitioner;
- Martin Schröder – PDFTEX project admin;
- Andrzej Tomaszewski – typographer, book designer;
- Adam Twardoch – font expert;
- Marcin Woliński – Polish LATEX guru, author of the Polish document classes mwcls.

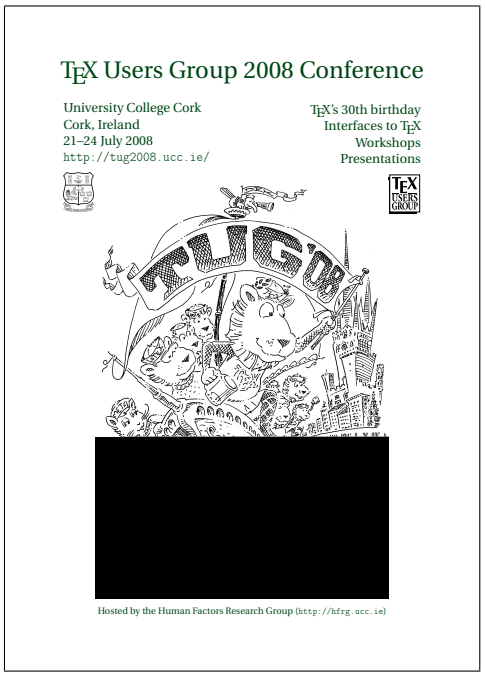
`http://www.gust.org.pl/BachoTeX/2008/index_html`

TUG Conference 2008

Il convegno annuale 2008 del TEX Users Group si terrà dal 21 al 24 Luglio 2008 presso lo University College Cork, in Irlanda.

Nel 2008 cade il trentesimo anniversario della nascita di TEX. Temi del convegno saranno: Interfacce, LATEX, ConTEXt, METAFONT, METAPOST e XETEX.

Domenica 20 Giugno (prima della conferenza) si terranno alcuni seminari. Attualmente sono allo studio sessioni dedicate all'uso di base e avanzato di LATEX e a PSTricks.



Il convegno è ospitato dallo *Human Factors Research Group* presso il Dipartimento di Psicologia Applicata.

Per ulteriori informazioni: `http://www.ucc.ie/en/tug2008/`

2$^{\text{nd}}$ International ConTEXt User Meeting

The 2$^{\text{nd}}$ ConTEXt user meeting will take place in Slovenia from 20$^{\text{th}}$-25$^{\text{th}}$ August 2008 in Bohinj, in the heart of Triglav National Park, around a lake in Alps.

Organizers

The ones responsible for your enjoyable stay:

- prof. dr. Rudolf Podgornik;
- Mojca Miklavec `<mojca dot miklavec dot lists at gmail dot com>`;
- assist. dr. Primož Peterlin.

The two responsible for quality and professional feeling of the meeting:

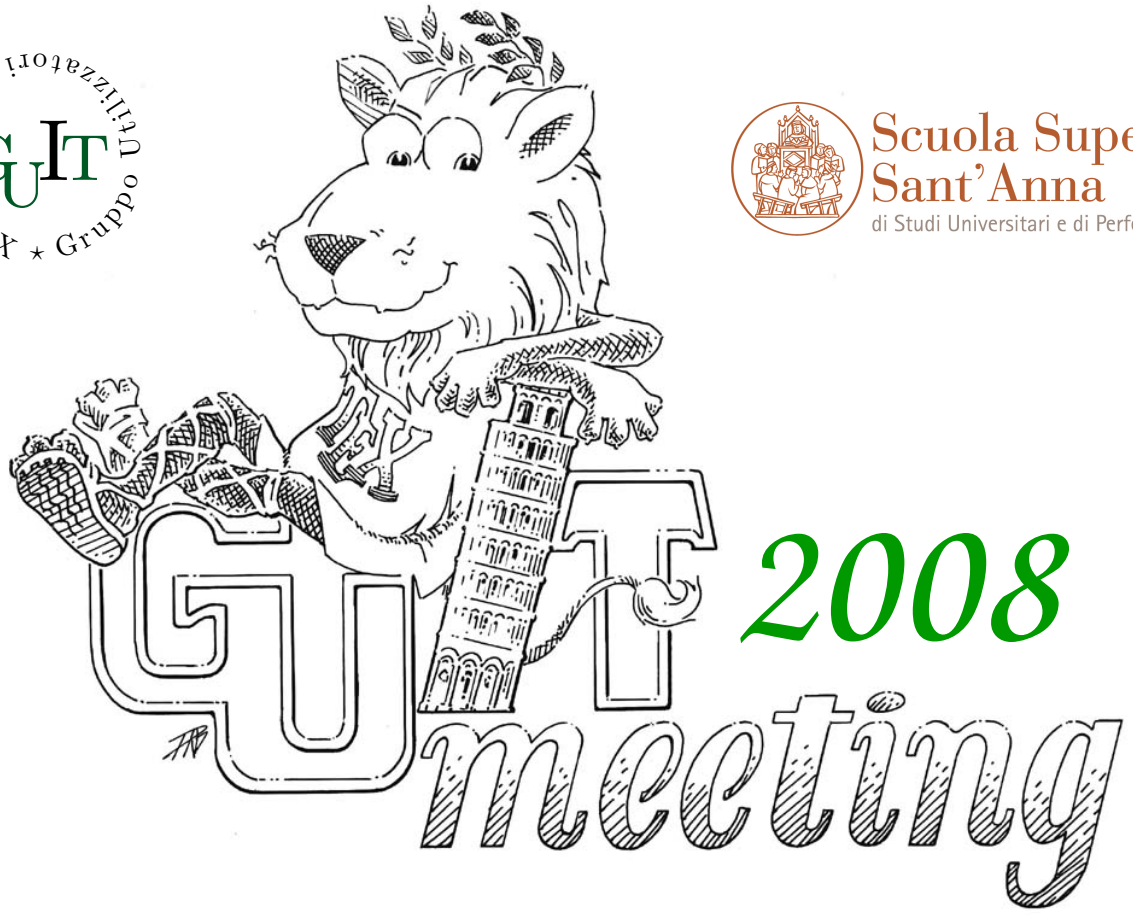
- Hans Hagen;
- Taco Hoekwater.

With special thanks to our Faculty of Mathematics and Physics, University of Ljubljana and:

- prof. dr. Franc Forstnerič;
- prof. dr. Vladimir Batagelj;
- prof. dr. Tomaž Pisanski.

`http://meeting.contextgarden.net/`

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



Quinto convegno nazionale su TEX, LATEX e tipografia digitale

Pisa, 18 ottobre 2008

Aula Magna, Scuola Superiore Sant'Anna

Call for Paper

Sabato 18 ottobre 2008 presso l'Aula Magna della Scuola Superiore Sant'Anna di Pisa si terrà il quinto Convegno annuale su TEX, LATEX e tipografia digitale organizzato dal Gruppo Utilizzatori Italiani di TEX. Il Convegno sarà un momento di ritrovo e di confronto per la comunità LATEX italiana, tramite una serie di interventi atti sia a contribuire all'arricchimento sia a supportarne lo sviluppo.
Maggiori informazioni sul Convegno e sulle modalità di presentazione degli interventi sono disponibili all'indirizzo:

`http://www.guit.sssup.it/guitmeeting/2008/`

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 5, Aprile 2008