

Numero 8
Ottobre
2009

ArsT_EXnica

Rivista italiana di T_EX e L^AT_EX

Atti del G_UIT*meeting* 2009

G_UIT

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



G_{UIT} – Gruppo Utilizzatori Italiani di T_EX

ArsTeXnica è la pubblicazione ufficiale del G_{UIT}

Comitato di Redazione

Gianluca Pignalberi – *Direttore*

Claudio Beccari

Fabiano Busdraghi

Riccardo Campana

Massimo Caschili

Gustavo Cevolani

Massimiliano Dominici

Andrea Fedeli

Enrico Gregorio

Carlo Marmo

Lapo Mori

Ottavio Rizzo

Emiliano Vavassori

Emanuele Vicentini

Raffaele Vitolo

Emanuele Zannarini

ArsTeXnica è la prima rivista italiana dedicata a T_EX, a L^AT_EX 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 G_{UIT}, 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 T_EX e L^AT_EX 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 accettati, 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 nu-

mero in preparazione della rivista. Tuttavia gli articoli potranno essere inviati in qualsiasi momento e troveranno collocazione, eventualmente, nei numeri seguenti.

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

arstexnica@sssup.it.

Nota sul Copyright

Il presente documento e il suo contenuto è distribuito con licenza  Creative Commons 2.0 di tipo “Non commerciale, non opere derivate”. È possibile, riprodurre, distribuire, comunicare al pubblico, esporre al pubblico, rappresentare, eseguire o recitare il presente documento alle seguenti condizioni:

-  **Attribuzione:** devi riconoscere il contributo dell'autore originario.
-  **Non commerciale:** non puoi usare quest'opera per scopi commerciali.
-  **Non opere derivate:** non puoi alterare, trasformare o sviluppare quest'opera.

In occasione di ogni atto di riutilizzo o distribuzione, devi chiarire agli altri i termini della licenza di quest'opera; se ottieni il permesso dal titolare del diritto d'autore, è possibile rinunciare ad ognuna di queste condizioni.

Per maggiori informazioni:

<http://www.creativecommons.com>

Associarsi a G_{UIT}

Fornire il tuo contributo a quest'iniziativa come membro, e non solo come semplice utente, è un presupposto fondamentale per aiutare la diffusione di T_EX e L^AT_EX 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 T_EX:

c/o Ufficio Statistica

Scuola Superiore Sant'Anna

Piazza Martiri della Libertà 33

56127 Pisa, Italia.

<http://www.guit.sssup.it>

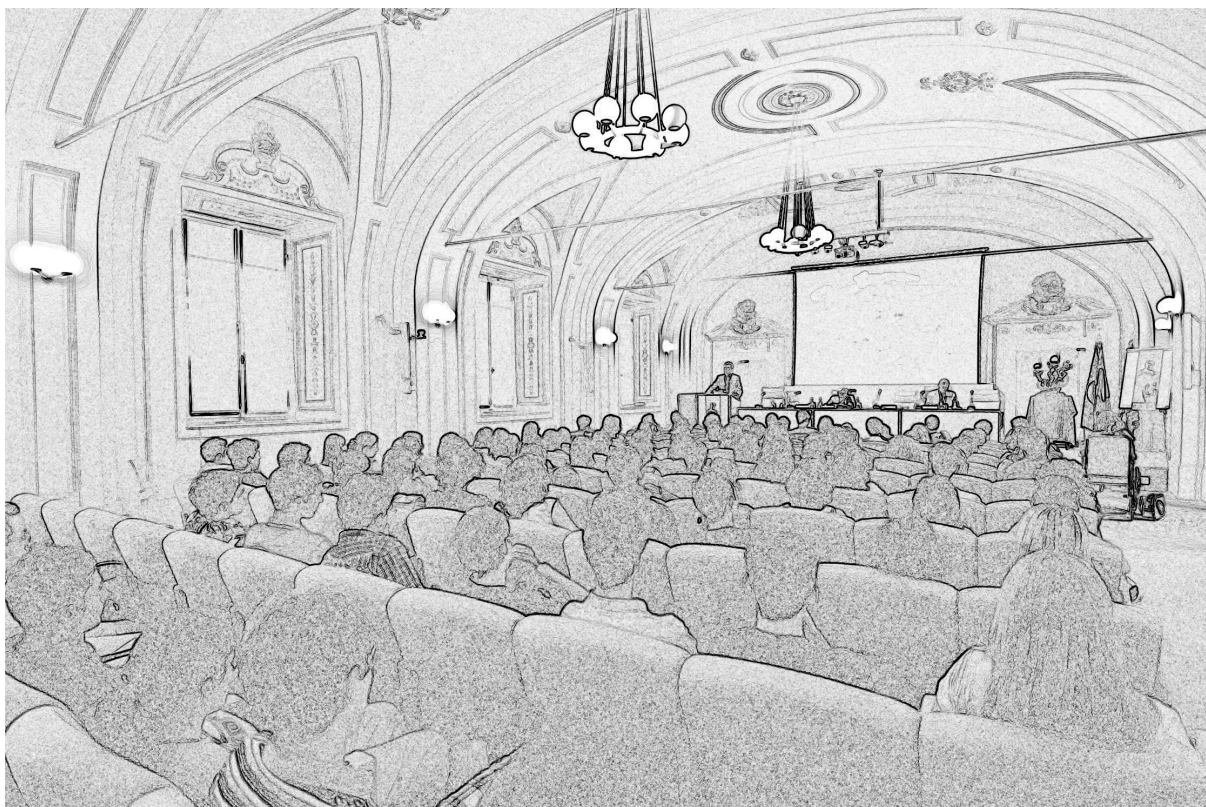
guit@sssup.it

Redazione ArsTeXnica:

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

arstexnica@sssup.it

Codice ISSN 1828-2369



Programma del Convegno

Sessione Mattutina

09:00 – Registrazione

09:30 – Simboli matematici in $\text{T}_\text{E}\text{X}$ e $\text{L}_\text{A}\text{T}_\text{E}\text{X}$

Enrico Gregorio, Università di Verona

10:00 – Produrre grafica vettoriale di alta qualità
programmando Asymptote

*Agostino De Marco, Università degli Studi di Napoli
“Federico II”*

10:30 – $\text{T}_\text{E}\text{X}$ as an ebook reader

Kabeh Bazargan, River Valley Technologies

11:00 – Coffee break

11:30 – La composizione di tabelle con larghezza
specificata

Claudio Beccari

12:00 – L'arte di gestire la bibliografia con biblatex

Lorenzo Pantieri

12:30 – $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ e CSV

Massimiliano Dominici

13:00 – Pausa Pranzo

Sessione Pomeridiana

14:30 – combelow: abbasso i segni diacritici
di serie B

Gianluca Pignalberi, Free Software Magazine

15:00 – Uso del comando `\write18` per comporre
l'indice analitico in modo sincrono

Claudio Beccari

15:30 – Coffee break

16:00 – Una estensione di luatex: luatex lunatic

Luigi Scarso, Logo S.R.L.

16:30 – Il respawn di Infomedia ($\text{L}_\text{A}\text{T}_\text{E}\text{X}$ -based)

Emmanuele Somma, Infomedia

17:00 – Processing “Computed” Texts

Jean Michel Hufflen, Université de Franche-Comté

18:00 – $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ Help Desk (esperti di $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ a vostra
disposizione)

18:30 – Chiusura dei lavori e cena sociale

ArsT_EXnica

Rivista italiana di T_EX e L^AT_EX

Numero 8, Ottobre 2009

Editoriale	
<i>Gianluca Pignalberi, Massimiliano Dominici</i>	5
Simboli matematici in T _E X e L ^A T _E X	
<i>Enrico Gregorio</i>	7
Produrre grafica vettoriale di alta qualità programmando <i>Asymptote</i>	
<i>Agostino De Marco</i>	25
T _E X as an ebook reader	
<i>Kaveh Bazargan</i>	40
La composizione di tabelle con larghezza specificata	
<i>Claudio Beccari</i>	42
L'arte di gestire la bibliografia con bibl _{at} ex	
<i>Lorenzo Pantieri</i>	48
L ^A T _E X e CSV	
<i>Massimiliano Dominici</i>	61
combelow: abbasso i segni diacritici di serie B	
<i>Gianluca Pignalberi</i>	70
Uso del comando <code>\write18</code> per comporre l'indice analitico in modo sincrono	
<i>Claudio Beccari</i>	76
Una estensione di luatex: luatex lunatic	
<i>Luigi Scarso</i>	79
Il respawn di Infomedia (L ^A T _E X-based)	
<i>Emmanuele Somma</i>	92
Processing "Computed" Texts	
<i>Jean-Michel Huppen</i>	102
Use of the <code>\write18</code> feature for composing indexes	
<i>Claudio Beccari</i>	110
Eventi e novità	112

Gruppo Utilizzatori Italiani di T_EX

Editoriale

Gianluca Pignalberi, Massimiliano Dominici

Siamo arrivati all'ultimo meeting prima di EuroTeX 2010. È con particolare orgoglio che, a nome di tutto il qJr, vi racconto quello che io vivo dal di fuori, non essendo residente a Pisa.

Già da tempo Massimiliano Dominici è impegnato a riportare a Duane Bibby i responsi sulla scelta del logo. Siamo arrivati alla versione definitiva, e abbiamo il disegno finale già in cassaforte. Il primo, piccolo passo è stato fatto. In realtà molto altro è stato fatto: bisognava fissare la data del meeting, prenotare l'aula e le strutture che lo ospiteranno. Fatto.

Il resto che rimane da fare (call for paper, organizzazione delle giornate e relativi servizi) verrà fatto con la precisione e la puntualità che contraddistinguono il qJr. Questo è il breve scorcio che intendo dare dei preparativi per questo importante appuntamento per la comunità TeX, per il cui successo il qJr si impegnerà al meglio delle proprie capacità. Il resto ve lo racconterò al prossimo appuntamento con ArsTeXnica.

Parliamo invece del qJrmeeting2009 a cui siamo arrivati. Gli interventi sono, al solito, numerosi e qualificati.

Enrico Gregorio ha applicato la sua conoscenza enciclopedica alla stesura di un articolo riguardante i simboli matematici. Non pensi il lettore «Ah, il solito elenco sui simboli che L^AT_EX mette a disposizione dell'utente.» La materia è molto più complessa, visto che Enrico fa un'introduzione ai comandi primitivi di TeX per comporre formule matematiche e ai corrispondenti comandi L^AT_EX; correda l'articolo di numerosi esempi d'uso e di suggerimenti per definire appropriatamente nuovi simboli, sfruttando le spaziature automatiche di TeX.

Agostino De Marco ha scritto uno dei suoi soliti articoli. Sia ben chiaro, la mia espressione è tutt'altro che spregiativa: voglio piuttosto dire che è un articolo orientato agli strumenti per fare grafica in/ con L^AT_EX; chiarissimo; contenente una quantità di esempi decisamente corposa e sempre significativa; qualitativamente alto. Questa volta ha concentrato i suoi sforzi su Asymptote, definito da molti il successore di METAPOST.

Kaveh Bazargan ci aggiorna, nei limiti del possibile, sull'ebook reader prodotto dalla sua River Valley Technologies, e basato su TeX. L'hardware su cui l'ebook gira non è il tanto chiacchierato Kindle di Amazon, ma il più diffuso e raggiungibile iPhone. Gli ho suggerito di prendere in considerazione il porting su uno dei suoi concorrenti nominali (il si-

stema Android e gli smartphone basati su di esso): vedremo se ne verrà fuori una *killer application*.

Claudio Beccari è presente con due lavori. Il primo lavoro riguarda la composizione di tabelle con larghezza specificata. Sebbene la cosa sembri banale, Claudio ci mostra tutte le difficoltà legate al conseguimento di questo obiettivo. Non poteva mancare la sua proposta migliorativa per il pacchetto `widetable`. Il secondo articolo ci mostra come sfruttare una capacità non banale di TeX: l'esecuzione di comandi esterni può far generare un indice analitico in modo sincrono alla composizione del documento. Questo meccanismo ci evita di dover compilare più volte un documento per ottenere lo stesso risultato. Questo articolo ha addirittura la versione inglese in appendice: questa versione è apparsa su TUGboat n° 95. A causa dei tempi di pubblicazione il "clone" è arrivato prima dell'originale.

Lorenzo Pantieri ci parla di BibL^AT_EX, un pacchetto che espande la potenza elaborativa delle bibliografie. Oltre al riconoscimento automatico della lingua del documento che ospiterà la bibliografia, si possono redigere bibliografie separate in base all'importanza dei documenti citati (ad es. introduttivi o specialistici, da leggere semplicemente o studiare attentamente). Si può anche scegliere il tipo di riferimento e la sua granularità.

Massimiliano Dominici ci introduce alle tecniche e ai pacchetti che servono a inserire dei file *comma separated value* (CSV) nei documenti L^AT_EX. Ci mostra i casi in cui vogliamo stampare il contenuto di semplici database le cui tabelle sono "banali" file CSV, e casi in cui vogliamo che un database di dati numerici diventi un grafico. I pacchetti sono anche in grado di modificare i database, ma l'argomento non è discusso nell'articolo.

Il sottoscritto ha tentato di espandere la fruibilità di L^AT_EX da parte di romeni e lettoni. Finora l'unica possibilità era scrivere il simbolo cediglia (segno diacritico) come tale anziché come una virgola sottoposta: accettabile per il romeno, meno per il lettone. L'autore ha scritto il pacchetto `combelow` proprio per rendere disponibile la composizione delle lettere con il segno diacritico corretto.

Luigi Scarso ha scritto un articolo molto dettagliato su un'estensione di LuaTeX: `luatex lunatic`. Quest'estensione serve a includere un interprete Python nell'interprete Lua di LuaTeX. Ci mostra come far funzionare tutto su un sistema Linux, arrivando a descrivere tutte le modifiche da fare ai vari file che lo compongono.

Emmanuele Somma ci racconta, posizionando gli eventi cronologicamente, come si è arrivati a far rinascere lo storico editore Infomedia, e del ruolo che L^AT_EX riveste nella composizione delle sue riviste. Emmanuele ci parla di quali strumenti abbia usato per riformare una redazione ispirata a elementi non comuni in Italia: peer-review, licenze libere, decentramento. È con particolare piacere che posso includere questo articolo in *ArsTeXnica*: nel periodo 2003–2005 ho pubblicato diversi articoli su tre delle riviste Infomedia. In particolare Dev ha ospitato una decina di miei articoli proprio su L^AT_EX e affini. Infomedia è media partner di *ArsTeXnica* per questo numero.

Jean-Michel Hufflen fa una comparazione tra metodi e strumenti che possono essere usati per *derivare* testi da strutture più grandi: in questo caso file con struttura simil-XML. L'articolo ci mostra diversi modi di derivare automaticamente un file T_EX da questi file con diversa struttura. L'utilità della cosa sta nella possibilità di comporre tipograficamente testi che normalmente non lo sarebbero (si pensi, ad esempio, a report automatici).

Non crediate che abbia dimenticato il mio impegno di migliorare il framework di *ArsTeXnica*. Con i suggerimenti di Claudio Beccari ho ridefinito la struttura dei template per gli autori di *ArsTeXnica*: dovrebbe ora essere più facile raccogliere automaticamente i lavori per organizzare la rivista, così come dovrebbe essere più logica la struttura per una persona. Ulteriori informazioni in merito le troverete sul sito del G_JT_R. Da questo numero, poi, cambia la generazione dell'indice della rivista. Gli autori e i lettori più attenti avranno visto che il titolo di un articolo era spesso diverso da quello dell'indice e della testatina. L'autore è tenuto a dare un titolo breve, insieme a quello dell'articolo, proprio per comporre correttamente la testatina. Ora, pensate a chi voglia citare un articolo contenuto in *ArsTeXnica*, o cercarlo in rete. Prende l'indice, lo legge, e riporta *quel* titolo. Errore: l'articolo si intitola spesso in maniera diversa. Ora, finalmente, il titolo dell'indice è uguale a quello dell'articolo. Proprio come le riviste scientifiche internazionali. È altresì possibile inserire articoli senza autore: a differenza delle vecchie versioni della classe l'indice viene composto senza problemi. Prima, invece, la voce dell'indice conteneva nella prima riga il titolo imbandierato a sinistra, e nella seconda riga il solo numero di pagina imbandierato a sinistra. Era evidente la differenza con le altre voci dell'indice, che avevano nella seconda riga il nome dell'autore e il numero di pagina separati da un `\dotfill`.

Verso EuroT_EX 2010

Di solito questo è lo spazio che rubo a Gianluca per fare il punto sull'anno trascorso, e non mancherebbero certo cose da raccontare. Dovrò invece

parlare soprattutto del futuro che, come ha già preannunciato Gianluca all'inizio dell'editoriale, si chiama EuroT_EX 2010. Dato che sarà nostro compito organizzarlo, e vogliamo farlo bene per fare buona figura con i nostri ospiti, stiamo cominciando, proprio in questi giorni, ad entrare nella fase operativa. Abbiamo già il logo del convegno, che non poteva che essere opera di Duane Bibby, e potete ammirarlo in tutto il suo splendore proprio su questa rivista, nella sezione "Eventi e novità" (non proprio in tutto il suo splendore, perché questa che trovate sulla rivista è la versione in bianco e nero).

Un logo è importante, ma nei prossimi giorni, settimane e mesi dovremo darci da fare per cose più "di sostanza": decidere il tema del convegno, pianificare la ripartizione delle giornate, sistemare nel migliore dei modi tutto ciò che riguarda l'accoglienza, pensare alla raccolta degli interventi e alla stesura degli atti.

Anticipo qui alcune cose che proporrò al consiglio direttivo (o, più probabilmente, che avrò già proposto, quando leggerete queste righe). Innanzitutto, dato che il convegno nazionale del 2010 si terrà all'interno di EuroT_EX, ci terrei che la giornata ad esso dedicata rimanesse aperta a tutti, come è sempre stato per il nostro meeting, e non solo agli iscritti a EuroT_EX. Stiamo peraltro cercando, per rimanere in tema, di mantenere basse le quote di iscrizione a EuroT_EX per favorire una partecipazione la più alta possibile.

Anche per quanto riguarda la stesura degli atti, valuteremo se sia possibile mantenere l'attuale sistema, per cui i partecipanti ricevono già una copia del volume in versione finale, o se dovremo uniformarci a quanto avviene nella grande maggioranza dei convegni scientifici (compresi quelli dedicati a T_EX) in cui la stesura degli atti viene fatta in un secondo tempo. Noi siamo orgogliosi del nostro sistema, che ci sembra offrire un servizio in più ai partecipanti al meeting, e vorremmo continuare in questa direzione.

Questo ed altro ci impegnerà per i prossimi mesi, e il meeting stesso sarà l'occasione per discutere di questi temi, organizzarci e reclutare volontari (siete autorizzati a considerare le ultime parole come una velata forma di appello).

- ▷ Gianluca Pignalberi
g dot pignalberi at
freesoftwaremagazine
dot com
- ▷ Massimiliano Dominici
mlgdominici at
interfree dot it

Simboli matematici in T_EX e L^AT_EX

Enrico Gregorio

Sommario

Un'introduzione ai comandi primitivi di T_EX per la composizione di formule matematiche e ai corrispondenti comandi di L^AT_EX con esempi d'uso e suggerimenti per definire nuovi simboli in modo appropriato per sfruttare le spaziature automatiche di T_EX.

Abstract

An introduction to the primitive commands of T_EX for the typesetting of mathematical formulas and to the corresponding L^AT_EX commands, with examples and suggestions for defining new symbols in a suitable way in order to exploit the automatic spacing provided by T_EX.

1 Introduzione

Uno dei punti di forza di T_EX nei confronti di altri sistemi di tipografia elettronica è senza dubbio la capacità di comporre formule matematiche senza troppi interventi 'manuali'. Le tradizionali regole di composizione della matematica, che sono molto simili in tutti i paesi occidentali, sono già note al motore tipografico di T_EX e quindi chi lo adopera non è obbligato a conoscerle.

Il classico esempio è dato dalle due formule

$$\begin{aligned}a + b \\ a = b\end{aligned}$$

perché, a un'osservazione attenta, appare chiaro come la spaziatura fra i simboli sia diversa: leggermente più larga nella seconda rispetto alla prima. Infatti T_EX sa che il segno di uguaglianza è un simbolo di relazione, mentre quello di addizione è un simbolo di operazione e la tradizione tipografica, che ha origine nel significato dei simboli, vuole così. Eppure le due formule possono essere scritte semplicemente come `$a+b$` e `$a=b$` rispettivamente; delle spaziature si fa carico T_EX e normalmente non sbaglia.

Un altro classico esempio è quello della formula

$$\log(xy) = \log x + \log y$$

nella quale si può notare uno spazio sottile fra il simbolo della funzione logaritmo e la variabile, assente nel primo membro fra 'log' e la parentesi aperta. Se invece consideriamo

$$e^{x+y} = e^x e^y$$

possiamo notare che nell'operazione a esponente la spaziatura è assente: così dice la pratica tipografica, perché spaziare quella somma introdurrebbe un'ambiguità semantica che è meglio evitare. Possiamo provare a comporre la stessa formula spaziando l'esponente

$$e^{x+y} = e^x e^y$$

e l'effetto è certamente meno attraente.

Scopo di questo articolo non è di insegnare a scrivere correttamente le formule, si veda per questo l'ottimo 'Consigli su come non maltrattare le formule matematiche' (GUIGGIANI e MORI, 2008) se si desidera imparare bene quest'arte. L'articolo è invece orientato ad approfondire gli aspetti T_EXnici della faccenda: ciò dovrebbe essere utile a chi vuole ottenere effetti particolari o capire meglio il funzionamento del sistema senza cadere nei trabocchetti che qualche volta causano grattacapi e affrante richieste di aiuto.

Sebbene l'articolo riguardi L^AT_EX, sarà necessario e utile esaminare a fondo anche i comandi primitivi che regolano la composizione delle formule.

2 Codici matematici

I caratteri che si scrivono in ambiente matematico hanno per T_EX un significato diverso da quello che possiedono nel normale modo di composizione del testo. Ogni carattere ha associato un *codice matematico*, un numero, che viene convenientemente espresso in rappresentazione esadecimale; un codice matematico è un intero compreso fra "0 e "7FFF. Useremo la convenzione del T_EXbook nel quale i numeri preceduti dal simbolo " e scritti in carattere dattilografico sono in base 16. Le cifre sono, come si usa, 0 1 2 3 4 5 6 7 8 9 A B C D E F; la traduzione di "7FFF in rappresentazione decimale è 32 767, cioè $2^{15} - 1$: un codice matematico è un numero a 15 bit. C'è una piccola eccezione: un codice matematico può anche essere "8000 = 32 768, ci torneremo; per il momento ci limiteremo ai codici normali.

Ogni formato (il nucleo di L^AT_EX, per esempio) è responsabile di assegnare ai caratteri un codice matematico; solo alcune convenzioni sono impostate fin dall'inizio, ma modificabili se lo si desidera. Per esempio, il codice matematico di **a** è "7161 e si riconosce nelle ultime due cifre il codice ASCII del carattere **a** (esadecimale). Analogamente, il codice matematico di **1** è "7031. Inizialmente il codice matematico di ciascuna lettera (maiuscola

o minuscola) che abbia codice ASCII x è

$$"7100 + x$$

mentre quello delle cifre (da 0 a 9) è

$$"7000 + y$$

dove y è il codice ASCII della cifra. Tutti gli altri caratteri hanno, all'inizio, codice matematico nullo. Scopriremo fra poco che le prime due cifre del codice matematico sono proprio quelle responsabili di far comporre a T_EX le lettere in corsivo (quello speciale usato nelle formule) e le cifre in tondo. Le ultime due, naturalmente, si riferiscono al carattere da scegliere.

Si assegna un codice matematico a un carattere con comandi del tipo

```
\mathcode'a="7161
\mathcode'1="7031
```

perché un numero esadecimale può essere inserito, quando la sintassi di T_EX richiede un numero, precedendo le cifre con il carattere ".¹

Facciamo una breve digressione. T_EX mantiene un vettore con 256 componenti, il vettore dei codici matematici (ne ha parecchi altri). L'istruzione

```
\mathcode⟨numero₁⟩ = ⟨numero₂⟩
```

dice a T_EX di impostare la componente del vettore dei codici matematici di posto $\langle numero_1 \rangle$ al valore $\langle numero_2 \rangle$. Ovviamente $\langle numero_1 \rangle$ deve essere a 7 bit, mentre $\langle numero_2 \rangle$ deve essere a 15 bit. Gli spazi attorno al simbolo di uguaglianza, che indica l'assegnazione, sono opzionali (sarebbe opzionale anche il segno =, ma lo scriveremo sempre).

Una convenzione utile per scrivere $\langle numero_1 \rangle$ è di usare un carattere preceduto da ' che rappresenta il codice ASCII di quel carattere. In caso di dubbio se il carattere sia speciale e possa non essere compreso così com'è, è possibile precederlo con la barra rovescia: in questo contesto, '\a e 'a sono equivalenti.

Proviamo a vedere che succede scrivendo la formula

```
\[ \mathcode'a="7162 \mathcode'1="7131
a+1=x
\]
```

che produce

$$b + 1 = x$$

L'effetto è alquanto misterioso, ma non troppo: al posto di a appare b , perché il carattere b ha codice ASCII "62; il numero 1 appare come numero minuscolo, perché il font della famiglia Computer Modern usato per i caratteri matematici, che viene scelto quando la seconda cifra del codice matematico è "1, ha al posto delle cifre i numeri minuscoli

1. Si veda alla fine dell'articolo un'importante avvertenza.

o 1 2 3 4 5 6 7 8 9. L'effetto di quelle bizzarre assegnazioni non si propaga, per fortuna, dal momento che le formule costituiscono un gruppo e quindi l'assegnazione di quei codici viene annullata alla fine della formula. Infatti ora possiamo scrivere $\$a+1=x\$$ e ottenere $a + 1 = x$.

Ma torniamo alla teoria generale: un codice matematico è un numero esadecimale a quattro cifre. La prima cifra può essere da "0 a "7, come abbiamo visto, la seconda una cifra qualunque da "0 a "F; le ultime due rappresentano il codice ASCII del carattere da stampare, prendendolo dal font che viene scelto da T_EX esaminando le prime due cifre.

La prima cifra assegna al carattere un *tipo* secondo questa tabella:

0	ordinario	4	apertura
1	operatore	5	chiusura
2	operazione	6	punteggiatura
3	relazione	7	speciale

Questo è il trucco con il quale T_EX decide come trattare i simboli e le spaziature corrispondenti: per esempio, il codice matematico di < è "313C che lo qualifica come simbolo di relazione; quello di , è "613B che qualifica la virgola come segno di punteggiatura in modo che le formule come $\$(x,y)\$$ siano stampate correttamente: (x,y) .

Normalmente il tipo 1 non viene mai assegnato come codice matematico di un carattere: serve per simboli come quello di integrale o di sommatoria, vedremo più avanti perché esiste. Tutti i formati assegnano codici ai caratteri 'stampabili' diversi dalle lettere e dalle cifre² nel modo seguente:

```
\mathcode'. = "013A \mathcode'/ = "013D
\mathcode'| = "026A \mathcode'* = "2203
\mathcode'+ = "202B \mathcode'- = "2200
\mathcode': = "303A \mathcode'< = "313C
\mathcode' = "303D \mathcode'> = "313E
\mathcode'(' = "4028 \mathcode'[' = "405B
\mathcode') = "5029 \mathcode'] = "505D
\mathcode'! = "5021 \mathcode'? = "503F
\mathcode', = "613B \mathcode'; = "603B
\mathcode'' = "8000
```

Queste assegnazioni sono un po' sorprendenti, in certi casi. Il punto viene considerato un simbolo ordinario perché è usato come separatore decimale (nell'uso inglese). La barra è un simbolo ordinario invece che di operazione, perché non si usa spaziarla rispetto a dividendo e divisore; lo stesso vale per la barra dritta che non andrebbe usata per indicare il modulo, preferendo i comandi \lvert e \rvert forniti da amsmath. Asterisco, più e meno sono simboli di operazione; notate qualche stranezza sull'assegnazione del codice matematico al carattere - (trattino)?

2. Per la precisione, il codice matematico è rilevante solo per i caratteri con codice di categoria 11 o 12.

Un'altra piccola sorpresa è il tipo di punto esclamativo e interrogativo: sono simboli di chiusura, come le parentesi: il fatto è che questo fornisce la corretta spaziatura per il fattoriale ($n!$); il punto interrogativo non viene usato in matematica, gli viene assegnato il tipo 5 per analogia con $!$. Ultima osservazione: il carattere 'due punti' è un simbolo di relazione, perché il suo uso più frequente è questo. Vedremo come è possibile trasformarlo in un segno di punteggiatura o in simbolo di operazione.

Un discorso a parte merita il codice assegnato all'apostrofo (o singolo apice): è "8000 ed è questo che permette alla scrittura $\$f'(x)\$$ di essere interpretata come $f'(x)$.

Il tipo 7 è simile al tipo 0 per quanto riguarda la scelta delle spaziature, cioè produce un simbolo considerato ordinario; ma un carattere di tipo 0 sarà *sempre* preso dal font scelto guardando la seconda cifra del codice matematico, per uno di tipo 7 T_EX guarda invece il valore del registro numerico interno `\fam`, secondo una regola che scopriremo più in là.

3 Inserire simboli diversi

I caratteri del codice ASCII certamente non sono sufficienti a coprire il vastissimo spettro dei simboli matematici di uso frequente: solo i simboli di relazione sono alcune decine, poi ci sono quelli di operazione, gli operatori come l'integrale, le lettere greche e tanti altri. L'idea è allora di adoperare numeri a 15 bit per esprimerli; il comando apposito è `\mathchar`. Con

`\mathchar"tfxy`

si istruisce T_EX a stampare il carattere di posto "xy con tipo "t e preso dal font corrispondente alla famiglia "f. Se però il tipo è "7, T_EX opererà in modo analogo ai caratteri inseriti direttamente per quanto riguarda la scelta del font. Naturalmente sarebbe assurdo dover scrivere

`\[ \mathchar"1350_{i=1}^n a_i ]`

per la formula

$$\sum_{i=1}^n a_i$$

e infatti tutti sanno che il comando con cui richiamare il simbolo $\sum$ è `\sum`. Si potrebbe dunque pensare che sia presente nel formato in uso la definizione

`\def\sum{\mathchar"1350 }`

ma T_EX possiede un'istruzione più compatta:

`\mathchardef\sum="1350`

che è agli effetti pratici equivalente a quella con `\def`.³

3. C'è una differenza T_EXnica; il comando definito con `\def` è espandibile, quello definito con `\mathchardef` no.

Vediamo qui finalmente un simbolo di tipo 1, che obbedisce a regole molto particolari: infatti viene stampato verticalmente centrato rispetto all'asse delle formule, una linea immaginaria che è leggermente più alta rispetto alla linea di base e, normalmente, attraversa il simbolo di sottrazione e le linee di frazione, come in $-\frac{a}{b}$.

Ciascun formato (Plain T_EX o L^AT_EX, per esempio) ha una lunga lista di definizioni di simboli matematici come quella precedente. Per poterle scrivere è necessario consultare le tabelle dei font matematici; per fortuna in L^AT_EX sono state definite alcune tabelle standard che permettono la compatibilità fra famiglie di font diverse, ammesso che gli sviluppatori le seguano. Per la verità, L^AT_EX ha un suo sistema per dare quelle definizioni, che però alla fine si riduce a istruzioni elementari di quella forma; le vedremo in una prossima sezione.

4 Famiglie matematiche

Abbiamo dunque visto il significato della prima cifra di un codice matematico, sia per il vettore `\mathcode` che per `\mathchar` o `\mathchardef`. Per capire il significato della seconda, occorre introdurre il concetto di *famiglia*.

T_EX gestisce un massimo di 16 famiglie di font matematici, corrispondenti proprio alla seconda cifra di un codice matematico. Per convenzione, nella famiglia 0 c'è sempre il font con cui si scrive il testo normale; nella famiglia 1 il font per il corsivo matematico, cioè per le lettere da usare comunemente nelle formule; nella famiglia 2 c'è un font per i simboli più frequenti; la famiglia 3 contiene il font per i simboli 'grandi', come parentesi ingrandibili, sommatoria, integrale e altri.

Ci ricordiamo che il codice matematico dei caratteri 0-9 ha infatti "0 come seconda cifra. Questo significa che, normalmente, i caratteri da stampare verranno presi dal font usuale per il testo. *Normalmente*, perché se al momento di comporre questo carattere il parametro `\fam` ha un valore compreso tra 0 e 15, il carattere sarà preso dal font della famiglia `\fam`.

Facciamo un esempio che dovrebbe illustrare il concetto, sperando di non confondere le idee:

`\$ \fam=2 a=z \$`

produce la formula $\mathfrak{a} = \mathfrak{z}$. Infatti, siccome il codice matematico di **a** e **z** è rispettivamente "7161 e "717A, il loro tipo è 7, quindi T_EX sceglie il font appartenente alla famiglia 2 per comporli. Naturalmente nessuno fa così!

Le famiglie 0, 1, 2 e 3 sono sempre predefinite, perché forniscono il minimo indispensabile per comporre le formule. In Plain T_EX vengono definite anche le famiglie con nomi simbolici `\itfam`, `\slfam`, `\bfam` e `\ttfam`; scrivendo `\fam=\itfam` i caratteri con tipo 7 saranno composti nel font corsivo

usato nel testo, perché alla famiglia corrispondono le assegnazioni

```
\textfont\itfam=\tenit
\scriptfont\itfam=\sevenit
\scriptscriptfont\itfam=\fiveit
```

Il loro significato è: se occorre comporre un carattere nella famiglia `\itfam` (questo, lo ricordiamo, è semplicemente un certo numero), T_EX userà il font `\tenit` se gli occorre la grandezza normale, mentre userà `\sevenit` per gli esponenti e i pedici di primo livello, ricorrendo a `\fiveit` per quelli di livello superiore. Il significato di questi comandi di scelta di font dovrebbe essere chiaro.

I simboli matematici (caratteri normali o comandi definiti con `\mathchardef`) di tipo diverso da 7 non vengono influenzati dal valore di `\fam`. Il valore del parametro rispetta la struttura dei gruppi, quindi

```
#{\fam=\bffam a}=a$
```

produce la formula $\mathbf{a} = a$. In Plain però non si scrive così, perché il comando `\bf` è definito con

```
\def\bf{\fam\bffam\tenbf}
```

Fuori dal modo matematico il valore di `\fam` è irrilevante; in modo matematico un comando di scelta di font come `\tenbf` è ignorato.

All'inizio di ogni formula (dopo il `$` o `$$`), il valore di `\fam` è sempre impostato a `-1`, cosicché occorre cambiarne esplicitamente il valore se si desidera che i simboli di tipo 7 siano composti con il font corrispondente alla famiglia scelta. Dunque si può scrivere la formula precedente con `#{\bf a}=a$`. O anche con

```
#{\bf a}=\mit a$
```

perché `\mit` è il comando per scegliere la famiglia 1, cioè quella del normale corsivo matematico. Siccome il codice matematico di `=` è `"303D`, questo simbolo non è influenzato dal valore di `\fam`.

Con L^AT_EX la sintassi per ottenere la stessa cosa è molto diversa; un tempo si poteva in effetti scrivere come in Plain, ma i comandi introdotti con il passaggio alla versione L^AT_EX 2_ε sono più chiari e aiutano a minimizzare gli errori.⁴ È chiaro però che, alla fine, i comandi L^AT_EX eseguono sempre comandi primitivi e scrivere `#{\mathbf{a}}=a$` porta a eseguire qualcosa di molto simile a `#{\fam=<numero> a}=a$`. La situazione è più complicata, perché L^AT_EX deve tener conto del corpo corrente e quindi non ha un'assegnazione fissa dei font appartenenti a una famiglia.

Facciamo un riassunto di quanto visto. Un codice matematico può essere

4. Si possono, a dire il vero, usare comandi come `\bf`, ma è meglio evitarlo; versioni future di L^AT_EX potrebbero disabilitare questi arcaismi.

- assegnato dal vettore `\mathcode`;
- richiamato tramite un comando definito con `\mathchardef`;
- richiamato direttamente come argomento di `\mathchar`.

Un codice matematico è un numero che ha una rappresentazione esadecimale del tipo `"tfx`, dove `"t` è il tipo, `"f` la famiglia di riferimento e `"xy` indica il carattere da scegliere nel font che T_EX impiegherà usando la regola seguente:

1. se il tipo `"t` è minore di 7, il font sarà quello appartenente alla famiglia `"f`;
2. se il tipo `"t` è 7 e il valore attuale di `\fam` non è fra 0 e 15 (compresi), si procede come nel passo precedente;
3. se il tipo `"t` è 7 e il valore attuale di `\fam` è compreso fra 0 e 15, il font è quello appartenente alla famiglia `\fam`.

5 Atomi

Non è finita qui: il tipo determina anche le spaziature rispetto agli altri elementi della formula che T_EX sta componendo. Un codice matematico produce un *atomo* che può essere

0. un simbolo ordinario (tipo 0 o tipo 7);
1. un operatore (tipo 1);
2. un simbolo di operazione (tipo 2);
3. un simbolo di relazione (tipo 3);
4. un delimitatore aperto (tipo 4);
5. un delimitatore chiuso (tipo 5);
6. un segno di punteggiatura (tipo 6).

Per gli atomi, cioè i costituenti elementari delle formule, il T_EXbook usa i nomi **Ord**, **Op**, **Bin**, **Rel**, **Open**, **Close**, **Punct**. La lista non è completa, perché T_EX conosce altri sei tipi di atomo: **Inner**, **Over**, **Under**, **Acc**, **Rad** e **Vcent**. Tuttavia questi sei tipi di atomi vengono alla fine considerati come se fossero **Ord**, eccetto quelli **Inner**.

Esistono comandi (primitivi) che possono trasformare un atomo di qualsiasi tipo o anche una sottoformula in un altro; i nomi dovrebbero spiegarsi da soli:

```
\mathord \mathop \mathbin \mathrel
\mathopen \mathclose \mathpunct \mathinner
```

Leggermente diversi sono i comandi `\overline` e `\underline` che producono atomi di tipo **Over** e **Under** rispettivamente. In L^AT_EX esiste anche `\mathalpha` che corrisponde ai codici matematici di tipo 7, ma va usato solo in particolari casi che esamineremo più avanti.

6 Spaziature in modo matematico

T_EX decide la spaziatura fra due atomi a seconda della loro natura, usando la tabella seguente: in orizzontale è l'atomo di sinistra, cioè il primo, in verticale l'atomo di destra, cioè il secondo. Il significato di 0 è che lo spazio fra i due atomi è nullo, di 1 che lo spazio è sottile, di 2 che lo spazio è medio e di 3 che lo spazio è ampio. Quando il numero è fra parentesi, lo spazio non è aggiunto se gli atomi sono in apici o pedici. L'intestazione delle righe e delle colonne è con i numeri corrispondenti al tipo di atomo, 'I' indica gli atomi di tipo **Inner**.

	0	1	2	3	4	5	6	I
0	0	1	(2)	(3)	0	0	0	(1)
1	1	1	*	(3)	0	0	0	(1)
2	(2)	(2)	*	*	(2)	*	*	(2)
3	(3)	(3)	*	0	(3)	0	0	(3)
4	0	0	*	0	0	0	0	0
5	0	1	(2)	(3)	0	0	0	(1)
6	(1)	(1)	*	(1)	(1)	(1)	(1)	(1)
I	(1)	1	(2)	(3)	(1)	0	(1)	(1)

Torneremo fra poco sul significato dell'asterisco. I parametri di T_EX relativi a queste spaziature si chiamano `\thinmuskip`, `\medmuskip` e `\thickmuskip` che vanno impostati con sintassi del tipo

```
\thinmuskip=3mu
\medmuskip=4mu plus 2mu minus 4mu
\thickmuskip=5mu plus 5mu
```

cioè le dimensioni vanno espresse in *unità matematiche*, in modo analogo alle lunghezze elastiche (*rubber lengths* nel manuale di L^AT_EX): un'ampiezza naturale, un grado di espandibilità e un grado di contraibilità, questi due opzionali. Vale l'uguaglianza $18\mu = 1em$, dove $1em$ è preso dal font della famiglia 2 garantendo così spaziature adatte sia per le parti normali che per apici e pedici di primo e secondo livello. I valori normalmente usati sono proprio quelli indicati, e consigliamo di non modificarli. Come si vede, lo spazio sottile non è espandibile né comprimibile; quello usato attorno ai simboli di operazione è leggermente espandibile e può annullarsi all'occorrenza; quello attorno ai simboli di relazione è solo espandibile. La scelta ha a che fare con le usuali regole di precedenza nelle espressioni matematiche: i simboli di relazione 'legano di meno' rispetto ai simboli di operazione. La differenza fra le ampiezze naturali di questi spazi non è grande, ma sufficiente a distinguere visivamente le funzioni dei simboli nelle formule.

I comandi per usare queste spaziature in unità matematiche sono `\mskip`, analogo di `\hskip`, e `\mkern`, analogo di `\kern`.

Nella tabella il simbolo * significa che la combinazione è impossibile; infatti T_EX cambia d'autorità un atomo **Bin** in **Ord** in certe situazioni, perché

un atomo **Bin** deve essere preceduto e seguito da atomi compatibili con un simbolo di operazione binaria. Per esempio è impossibile che un atomo **Op** sia seguito da un simbolo di operazione: `\log-a` viene risolto come $\log -a$ cambiando il tipo del simbolo di opposto (normalmente un simbolo di operazione) in un atomo **Ord**; in termini di atomi la formula diventa **Op Ord Ord**. Lo stesso avviene nel caso di due simboli di operazione adiacenti: `a++b` diventa $a++b$, il secondo simbolo di operazione diventa un atomo **Ord**. Entrambi gli esempi sono di errato uso dei simboli. Un caso che capita è un atomo **Open** seguito da un atomo **Bin**: `1+(-1)` è una formula accettabile che diventa

Ord Bin Open Ord Ord Close

come si desidera, cioè $1 + (-1)$. Un caso in cui questo può creare problemi è quello del valore assoluto; il codice matematico di `|` è "026A, cioè si tratta di un simbolo ordinario; scrivere `|-1|` produce quindi il risultato scorretto $|-1|$. Soluzione: scrivere `\lvert-1\rvert` oppure `\lvert-1\rvert`, con i comandi di `amsmath`, che danno $|-1|$.

Nel caso si desiderasse usare il simbolo `++` come simbolo unico di operazione si può scrivere

```
$a\mathbin{++}b=\sqrt{a^2+b^2}$
```

ottenendo $a++b = \sqrt{a^2+b^2}$, ne parleremo diffusamente più avanti.

Si noti che fra due atomi di tipo **Rel** non viene inserito alcuno spazio: è questo che permette di scrivere $a := b$ senza spaziature sbagliate. Questa proprietà è usata in altre situazioni, per esempio per ottenere le relazioni negate: in Plain troviamo le definizioni

```
\mathchardef\not="3236
\def\neq{\not=}
```

che si basano sul fatto che il carattere di posto "36 nel font della famiglia 2 ha ampiezza nulla e ciò che viene stampato sta a destra del punto in cui T_EX si trova. Così il carattere 'sembra' sovrapposto al simbolo di uguaglianza.⁵

7 Modificare il tipo di un atomo

Supponiamo di voler usare la lettera *R* come simbolo di relazione, come capita spesso in formule come $a R b$. Potremmo consultare la tabella per vedere quale sia la spaziatura corretta e scrivere

```
$a\mskip\thickmuskip R
\mskip\thickmuskip b$
```

oppure `a\R;b`, visto che `\;` è equivalente a `\mskip\thickmuskip`. Ma c'è un modo molto più efficiente e sicuro:

5. Lo si può controllare con il comando `\the\fontcharwd\textfont2 "36` che produce infatti 0.0pt

`\mathrel{R}b`

che ha anche il pregio di essere più chiaro sintatticamente. Allo stesso modo qualsiasi oggetto può diventare un atomo **Op** tramite il comando `\mathop` il quale però ha una particolarità che richiede una certa attenzione: *se l'argomento di `\mathop` risulta essere un solo carattere, questo sarà centrato verticalmente rispetto all'asse delle formule*. Vediamo infatti il risultato di `\mathop{A}A` che produce $A A$. Eventuali comandi di scelta di font matematici sono eseguiti, ma non cambiano il comportamento: lo si vede con la formula `\mathop{\fam=0 A}A` che produce $A A$. Qui si nota anche come le graffe attorno all'argomento di `\mathop` costituiscono un gruppo, infatti l'assegnazione del valore a `\fam` non influenza il secondo carattere. La situazione con L^AT_EX è identica: la formula `\mathop{\mathrm{A}}A` produce $A A$. Il trucco per evitare questo comportamento è di inserire qualcosa di innocuo che renda la sottoformula non composta da un solo carattere: `\mathop{\kern0pt\fam=0 A}A` produce la formula 'corretta' $A A$.

Caricando il pacchetto `amsmath` (più precisamente il modulo `amsopn`), si ha a disposizione con L^AT_EX il comando `\operatorname` che solleva l'utente dal problema: con

`\[ \operatorname{A}A \]`

si ottiene la formula $A A$. Torneremo più avanti sugli atomi **Op**.

La specificazione di apici e pedici di un atomo non ne cambia la natura: per esempio le spaziature in `$(a+_1b)+_2c$` sono corrette, come si vede componendo la formula, $(a+_1b)+_2c$. Non è lo stesso se si usa un accento: una costruzione `\hat{+}` infatti produce un atomo **Acc** che, secondo le regole, è trattato come se fosse **Ord** per quanto riguarda le spaziature. Perciò per avere $a \hat{+} b$ come simbolo di operazione occorre

`\[ a \mathbin{\hat{+}} b \]`

che infatti dà le spaziature corrette

$$a \hat{+} b$$

che non si avrebbero con `\mathop{\hat{+}}b` che dà $a \hat{+} b$. Ovviamente un utente saggio definirebbe un comando personale che in L^AT_EX sarebbe

`\newcommand*{\hplus}{\mathbin{\hat{+}}}`

e scriverebbe `\hplus`.

Ogni atomo può essere trasformato in **Ord** con il comando `\mathord`. Per esempio, volendo definire la 'sottrazione pitagorica'

$$a + - + b = \sqrt{a^2 - b^2}$$

non si può scrivere

`\[ a \mathbin{+-+} b = \sqrt{a^2 - b^2} \]`

che darebbe

$$a + - + b = \sqrt{a^2 - b^2}$$

Infatti, secondo le regole sugli atomi **Bin** i due segni di addizione vengono trasformati in **Ord** e quello di sottrazione rimane **Bin**, con conseguenza disastrosa. Tuttavia non è necessario ricorrere alla mostruosità

`\mathbin{\mathord{+}\mathord{-}\mathord{+}}`

perché basterebbe

`\mathbin{+\mathord{-}+}`

sempre per le stesse regole sugli atomi **Bin**. In realtà c'è un modo ancora più semplice: una sottoformula, cioè una parte della formula scritta tra graffe, è sempre considerata come un **Ord** ai fini delle spaziature rispetto agli oggetti attorno. Perciò

`\mathbin{+{-}+}`

è sufficiente: la successione **Bin Ord Bin** viene trasformata in **Ord Ord Ord**.

Questo trucco permette di usare la virgola come separatore decimale in modo non troppo complicato: scrivere `$2,5$` produrrebbe 2,5, mentre basta `$2{,}5$` per avere 2,5. Purtroppo la virgola ha usi diversi che rendono impossibile rendere automatica la spaziatura. Almeno un pacchetto cerca di ovviare all'inconveniente, richiedendo però che la virgola usata come segno di interpunzione sia seguita da uno spazio. Più semplice, forse, è definire

`\def\comma{\mathord{,}}`

se si usa poco la virgola come separatore decimale; oppure

`\mathcode' ,="013B`
`\def\comma{\mathpunct{,}}`

se si usa molto la virgola come separatore decimale e poco come segno di punteggiatura (in modo matematico).

8 Atomi di tipo Op

Segue dalle regole già viste che un codice matematico del tipo `"1fxy` produce un carattere che viene centrato rispetto all'asse delle formule. Non solo: ciò che è specificato come apice e pedice va, se si è in una formula in `\displaystyle`, sopra e sotto il carattere. La stessa cosa accade con un atomo generato tramite `\mathop`. Per esempio,

`\[ \mathop{\mathrm{log}}_{10} 2`
`\approx 0.30103 \]`

verrebbe reso scorrettamente con

$$\log_2 \approx 0.30103$$

Esistono tre comandi primitivi che si occupano di questo: `\displaylimits`, `\limits` e `\nolimits`. Se non si specifica alcuno di questi dopo l'atomo **Op**, il comportamento è quello determinato dal primo: apici e pedici sopra e sotto solo in `\displaystyle`, altrimenti in posizione normale. Il secondo forza in ogni caso, anche non in `\displaystyle`, apici e pedici sopra e sotto; il terzo invece forza apici e pedici in posizione normale in qualsiasi caso. Dunque la definizione corretta dell'operatore 'logaritmo' sarebbe

```
\newcommand*{\log}{%
  \mathop{\mathrm{log}}\nolimits}
```

che è (circa) la definizione data dal nucleo di L^AT_EX. La definizione senza `\nolimits` andrebbe bene per operatori come `min` e `max`. La definizione del simbolo di integrale è indiretta: in Plain è

```
\mathchardef\intop="1352
\def\int{\intop\nolimits}
```

e in L^AT_EX è molto simile. Questo perché in generale i limiti di integrazione vanno posti a fianco del simbolo. Nel caso si volesse mettere i limiti sopra e sotto, basta specificare `\limits`: se ci sono più comandi del tipo in discussione subito dopo l'atomo **Op**, vale l'ultimo. Perciò, per esempio, volendo scrivere gli estremi di integrazione sopra e sotto (probabilmente perché si tratta di un integrale di linea), si potrà scrivere

```
\[
\int\limits_{\cammino}f(z)\,dz =
\sum_{i=1}^n\int_{\gamma_i}f(z)\,dz
\]
```

(dando a `\cammino` un'opportuna definizione) per ottenere

$$\int_{\gamma_1 \sqcup \gamma_2 \sqcup \dots \sqcup \gamma_n} f(z) dz = \sum_{i=1}^n \int_{\gamma_i} f(z) dz$$

Si noti che nell'integrale di sinistra si è usato `\limits`, in quello di destra no: è questione di gusti. Come appare evidente, l'indicazione `\limits` annulla l'effetto di `\nolimits` nella definizione di `\int`.⁶

Gli atomi di tipo **Op** possono anche essere 'abusati' per la loro proprietà di porre i limiti sopra e sotto. Ne è un esempio la definizione in Plain di `\buildrel`

6. Il lettore attento noterà che la funzione integranda è più vicina al segno di integrale; è stato infatti inserito nella formula un comando `\mspace{-24mu}` che la rende più gradevole. Si lascino sempre all'ultimo questi aggiustamenti.

```
\def\buildrel#1\over#2{\mathrel{%
  \mathop{\kern0pt#2}\limits^{#1}}}
```

La sintassi non è familiare per chi è abituato a L^AT_EX, dove esiste il comando `\stackrel` che fa, più o meno la stessa cosa:

```
\def\stackrel#1#2{%
  \mathrel{\mathop{#2}\limits^{#1}}}
```

Per ottenere un simbolo di relazione nuovo, una x sormontata da un punto esclamativo, in Plain e L^AT_EX si scriverebbe, rispettivamente,

```
\buildrel !\over x      \stackrel{!}{x}
```

e il risultato sarebbe (a sinistra quello in Plain, a destra quello in L^AT_EX)

$$A \overset{!}{x} B \qquad A \stackrel{!}{x} B$$

Il lettore attento noterà una lieve differenza: a causa di `\kern0pt` nella definizione di Plain, la x non è centrata rispetto all'asse delle formule, come infatti deve essere. Dunque è meglio evitare l'uso di `\stackrel`, preferendo invece `\overset`: ecco le due formule di prima, ma a destra si è adoperato quest'ultimo comando, messo a disposizione da `amsmath`

$$A \overset{!}{x} B \qquad A \overset{!}{x} B$$

Come si vede, sono uguali (e corrette): il codice è, per la formula a sinistra quello già descritto con `\buildrel`, per l'altra si è usato `\mathrel{\overset{!}{x}}`.

Vediamo come funziona `\stackrel`: viene creato un atomo **Rel** a partire da un atomo **Op** costituito dal secondo argomento sopra il quale, usando `\limits` viene imposto il primo argomento. Nella definizione di `\buildrel` l'atomo **Op** non è mai costituito da un solo carattere, per via di `\kern0pt`. È importante ricordarsi che `\stackrel` crea un atomo **Rel**: se si desidera impiegarlo per costruire un simbolo ordinario occorre mettere l'intera costruzione tra graffe o come argomento di `\mathord`, altro motivo per non usarlo.

Il comando `\overset` di `amsmath` invece cerca di creare un atomo del tipo corretto basandosi sulla natura del secondo argomento: riconosce **Bin** e **Rel**, trattando tutto il resto come **Ord**; può quindi essere necessaria la menzione esplicita del tipo di atomo desiderato tramite uno dei comandi appositi come `\mathrel` e simili. La definizione di `\overset` è assai istruttiva e ingegnosa (dovuta a Mike Spivak):

```
\def\binrel@#1{\begingroup
  \setbox0=\hbox{\thinmuskip 0mu
    \medmuskip -1mu \thickmuskip 1mu
    \setbox2=\hbox{${#1}$}\kern-\wd2
    ${#1}$}%
  \edef\@tempa{\endgroup
```

```
\let\noexpand\binrel@@
\ifdim\wd0<0pt \mathbin
\else\ifdim\wd0>0pt \mathrel
\else \relax\fi\fi}%
\@tempa}
\let\binrel@@\relax
\newcommand{\overset}[2]{\binrel@{#2}%
\binrel@@{\mathop{\kern\z@#2}%
\limits^{#1}}}
```

Si tratta di una definizione del tutto simile a quella di `\buildrel`, solo che la menzione esplicita di `\mathrel` è sostituita da `\binrel@@` che riceve un valore dall'esecuzione di `\binrel@`; questo valore può essere `\mathbin` o `\mathrel` se il secondo argomento è rispettivamente un atomo `\Bin` o `\Rel`, è `\relax` altrimenti, presentando a T_EX, in questo caso, una sottoformula tra graffe che è perciò trattata come simbolo ordinario. Il lavoro di `\binrel@` consiste nell'impostare i valori dei parametri di spaziatura matematica in modo piuttosto curioso:

```
\thinmuskip = 0mu
\medmuskip = -1mu
\thickmuskip = 1mu
```

A questo punto si compone una *box* temporanea: prima, con

```
\setbox0=\hbox{\thinmuskip 0mu
\medmuskip -1mu \thickmuskip 1mu
\setbox2=\hbox{$\#1$}\kern-\wd2
$\#1$}
```

si misura l'ampiezza dell'argomento (che è il secondo di `\overset`) e si 'va a sinistra' di quell'ampiezza. Poi viene composta la formula `$_{\#1}$` che mette il secondo argomento fra due atomi `Ord` vuoti. Se l'argomento è un `Bin`, la spaziatura sarà negativa, quindi la formula risulterà più 'stretta' dell'ampiezza naturale e la *box* temporanea avrà ampiezza negativa. Se l'argomento è un `Rel`, la spaziatura sarà positiva, quindi l'ampiezza della *box* temporanea sarà positiva. Negli altri casi, non verrà aggiunto alcuno spazio e perciò l'ampiezza della *box* temporanea sarà nulla. Il resto della macro serve per dare il valore desiderato a `\binrel@@`, secondo i vari casi. Non è questo il luogo per andare nel dettaglio di come funzioni la definizione con `\edef` e quella particolare struttura di `\begingroup` e `\endgroup`.

Va in ogni caso osservato che un gruppo tra `\begingroup` e `\endgroup` in una formula *non* costituisce un atomo di tipo `Ord` e, se per caso il gruppo non contiene in definitiva alcun atomo, viene addirittura ignorato per quanto riguarda le spaziature matematiche. Il caso in esame è proprio questo: il gruppo viene aperto in modo da usare una nuova istanza locale dei registri `\box0` e `\box2` che sono usati spesso nelle macro per la composizione di formule matematiche.⁷ Si veda

7. La definizione è stata semplificata per eliminare dettagli T_EXnici irrilevanti.

la differenza tra `$a\begingroup+\endgroup b$` e `$a{+}b$`:

$$a + b \quad a + b$$

dove balza agli occhi che il gruppo non cambia la natura dell'atomo `Bin` nel primo caso, a differenza che nel secondo. In genere è consigliabile usare `\begingroup` e `\endgroup` per isolare assegnazioni locali in una formula, quando in altre situazioni il più semplice `{...}` è sufficiente.

Un altro aspetto degli atomi `Op` è stato sfruttato da Claudio Beccari (BECCARI, 2009, p. 144) nell'interessante definizione di una macro per scrivere automaticamente la 'd' del differenziale negli integrali o nelle forme differenziali. La tradizione matematica richiede che questa sia separata da ciò che precede con uno spazio sottile, ma con certe cautele:

$$\int \frac{1}{x} dx = \int \frac{dx}{x}$$

Nel primo integrale lo spazio sottile è necessario, nel secondo è vietato. Vorremmo anche spaziature automatiche in espressioni come

$$\omega = xy \, dx + dy$$

Il trucco è di inserire prima della 'd' un atomo `Op vuoto`! La tabella delle spaziature dice che questo atomo sarà preceduto dalla spaziatura giusta a seconda dell'atomo che precede (nessuna spaziatura se il differenziale compare all'inizio di una formula o sottoformula); poiché l'atomo seguente è `Ord` (la 'd'), sarà seguito da uno spazio sottile che annulliamo con il comando `\!`:

```
\def\diff{\mathop{\}\!d}
```

Chi volesse la 'd' in carattere tondo può modificare la definizione in modo ovvio. Le formule precedenti sono state scritte

```
\int\frac{1}{x}\diff x=
\int\frac{\diff x}{x}
\omega=xy\diff x + \diff y
```

(si ricordi che `\frac` è un comando L^AT_EX e non Plain). Scrivere `\diff{x}` è del tutto equivalente, ma non lo è `\diff{(x+y)}`, perché la parte tra graffe sarebbe trattata come una sottoformula e le spaziature attorno al + non partecipano agli aggiustamenti per la giustificazione. Si potrebbe definire `\diff` con un argomento, non sembra ragionevole, tanto più che spesso occorre $d^2 f$ e una sintassi come `\diff[2]{x}` è pesante e poco intuitiva rispetto a `\diff^2{x}`. Si veda BECCARI (1997) per una definizione dello stesso autore, molto più complicata.

Il trucco funziona perché T_EX inserisce ugualmente lo spazio richiesto dalle regole anche se l'atomo `Op` (vuoto) è seguito da un comando `\!` che significa `\mskip-\thinmuskip`: un comando di questo tipo non è un atomo.

9 Delimitatori

Gli atomi di tipo 4 e 5 sono quelli destinati a funzionare come le usuali parentesi tonde. Ogni atomo può essere trasformato in uno di questi con i comandi `\mathopen` e `\mathclose`. La definizione di `\lvert` di `amsmath` sarebbe essenzialmente equivalente a

```
\def\lvert{\mathopen|}
```

se non fosse che ci occorre una definizione diversa in modo che il comando funzioni anche dopo `\left`. Un caso in cui si possono usare esplicitamente `\mathopen` e `\mathclose` è la notazione (brutta) degli intervalli aperti nella forma $]a, b[$, che è stata scritta

```
\mathopen]a,b\mathclose[
```

Più interessante è scoprire come funziona la scelta dei simboli che compaiono dopo `\left` e `\right` e di come possano ingrandirsi a piacimento.

T_EX ha un altro vettore con 256 componenti simile a `\mathcode` che si chiama `\delcode`: il vettore dei *codici di delimitazione*. Ogni componente di questo vettore è un numero a 24 bit, quindi espresso con sei cifre esadecimali. La maggior parte di questi codici è -1, quelli non negativi sono

```
\delcode' ( = "028300
\delcode') = "029301
\delcode'. = "0
\delcode'/ = "02F30E
\delcode'< = "26830A
\delcode'> = "26930B
\delcode'[ = "05B302
\delcode'] = "05D303
\delcode'| = "26A30C
```

Un codice di delimitazione negativo significa che il carattere non può essere usato dopo `\left` o `\right`; codice nullo significa che il carattere indica un delimitatore vuoto. Quando è non nullo il codice è della forma *ffxyguv* e va letto come due numeri a 12 bit affiancati, molto simili ai codici matematici. La cifra *f* indica la famiglia da cui prendere il carattere di posto *xy*, se la grandezza del delimitatore richiesto è 'normale'; se il delimitatore deve essere ingrandito, verrà preso il carattere dalla famiglia *g* di posto *uv*. Per esempio,

```
$\mathchar"028 \mathchar"300$
```

produce $(($. Il font usato per la famiglia 3 ha le informazioni necessarie per costruire i delimitatori più grandi a richiesta, partendo, nel caso della parentesi aperta, da quello di posto "00.⁸

Il delimitatore vuoto è utile in situazioni nelle quali si vuole delimitare una formula solo da una

8. La formula precedente con le due parentesi non fa esattamente quello che viene mostrato: per motivi T_EXnici i caratteri nella famiglia 3 hanno posizioni strane rispetto alla linea di base.

parte: per scrivere un sistema di equazioni, per esempio. In questo caso si specificherà

```
\left\lbrace \langle equazioni \rangle \right.
```

Ma com'è possibile scrivere `\lbrace` dopo `\left`, visto che `\lbrace` non è un carattere? Se chiediamo a T_EX di mostrare la definizione di questo comando otterremo come risposta che è una macro:

```
macro:->\delimiter "4266308
```

La verità è che `\left` e `\right` devono essere seguiti o da un carattere con codice di delimitazione non negativo oppure da una macro la cui espansione cominci con `\delimiter`. Si potrebbe anche usare direttamente

```
\left\delimiter"<numero a 27 bit>
```

ma nessuno lo fa mai, perché il numero da inserire è, come sempre in questi casi, arcano. Il comando `\delimiter` ha come argomento un numero a 27 bit che possiamo dunque esprimere nella forma *ffxyguv*. La parte *ffxy* serve a T_EX nel caso non si sia dopo un comando `\left` o `\right` quando viene interpretata come un codice matematico; se invece siamo dopo uno dei comandi `\left` o `\right`, T_EX usa la parte *ffxyguv* esattamente come se fosse stato specificato un codice di delimitazione.

Vediamo il caso di `\uparrow`, che è definito con

```
\def\uparrow{\delimiter"3222378 }
```

Se il comando non appare dopo `\left` o `\right`, l'effetto è equivalente a dire `\mathchar"3222`, quindi verrebbe creato un atomo **Rel**. Altrimenti T_EX costruisce un delimitatore (di apertura o di chiusura) secondo la regola vista per i codici di delimitazione: carattere "22 del font della famiglia 2 se è richiesta una misura normale, carattere "78 della famiglia 3 se deve essere ingrandito: lo vediamo con

```
$\mathchar"222 \mathchar"378$
```

che produce $\uparrow$. La seconda freccia è apparentemente più piccola, per via delle regole interne di T_EX sulla costruzione di caratteri estendibili. Si tratta di regole piuttosto complesse che non è il caso di esaminare: basta sapere che funzionano, purché chi definisce il font sappia ciò che fa. Compito di T_EX è misurare la formula da delimitare (se è richiesto usando `\left` e `\right`) e scegliere la misura corretta della 'parentesi' in base anche ai valori dei parametri `\delimiterfactor` e `\delimitershortfall`. Un altro parametro che può interessare è `\nulldelimiterspace`, che è l'ampiezza di una *box* vuota inserita quando viene specificato un codice di delimitazione nullo.

Una parola sul comando `\middle` che è a disposizione quando si usino le estensioni ε -T_EX, in

particolare se si compila con L^AT_EX in una distribuzione aggiornata. Il comando ha le stesse proprietà e richieste di `\left` e `\right`: richiede un delimitatore allo stesso modo e lo ingrandisce con le stesse regole, dovendo però apparire tra quei due comandi. Il delimitatore costruito verrà considerato come un atomo **Ord**; è lasciato all'utente inserire le giuste spaziature. Potremmo per esempio definire

```
\def\bra#1{\left<#1\;\right|}
\def\ket#1{\left|\;\#1\right>}
\def\braket#1#2{%
  \left<#1\;\middle|\;\#2\right>}
```

per la notazione 'bra-ket' di Dirac usata in fisica teorica. Si noti come sia necessario specificare le spaziature attorno (prima o dopo nel caso di `\bra` e `\ket`) alla riga verticale. Un esempio:

```
\bra{\frac{x}{2}}\quad\ket{\frac{y}{3}}
\quad\braket{\frac{x}{2}}{\frac{y}{3}}
```

darebbe

$$\left\langle \frac{x}{2} \middle| \right| \frac{y}{3} \right\rangle \quad \left\langle \frac{x}{2} \middle| \frac{y}{3} \right\rangle$$

Il pacchetto `braket` si occupa di questo (risolvendo le spaziature in modo diverso, però).

Ci si può aspettare che i comandi delle serie `\big...` e `\Big...` siano definiti in termini di `\left` e `\right`: la congettura è corretta. Vediamone uno, `\bigl`, che è definito in Plain tramite

```
\def\bigl{\mathopen\big}
\def\big#1{\{\hbox{$\left#1$}
  \vbox to8.5pt{\}\right.$}}}
```

(la definizione è semplificata). Quando scriviamo `\bigl(`, T_EX sostituisce `\mathopen\big`, quindi produce un atomo **Open** che consiste di una parentesi aperta adatta a delimitare una formula che sia alta 8.5 pt. Il valore è adatto a caratteri di corpo 10, e occorre modificarlo se il corpo usato è diverso. In L^AT_EX la definizione è la stessa e infatti questi comandi non funzionano nel modo che ci si aspetterebbe. Il pacchetto `amsmath` infatti li modifica in modo che le misure siano adatte al corpo corrente. Per esempio,

```
$\bigl(b+(c+d)\bigr)$,
{\Large$\bigl(b+(c+d)\bigr)$}
```

produce

$$a(b + (c + d)), a(b + (c + d))$$

ma, senza `amsmath`, si otterrebbe

$$a(b + (c + d)), a(b + (c + d))$$

che è, evidentemente, sbagliato.

10 Atomi di tipo Inner

Un atomo di tipo **Inner** è una sottoformula delimitata con `\left` e `\right`. Si guardi con attenzione la formula inserita con

```
\[ x\Bigl(\frac{a}{b}\Bigr) +
  x\left(\frac{a}{b}\right) ]
```

che produce

$$x\left(\frac{a}{b}\right) + x\left(\frac{a}{b}\right)$$

In effetti abbiamo barato, aumentando la dimensione dello spazio sottile che, secondo la tabella, viene inserito tra un atomo **Ord** e un atomo **Inner**, cioè il valore del parametro `\thinmuskip`. Se riproduciamo la formula senza questo aumento

$$x\left(\frac{a}{b}\right) + x\left(\frac{a}{b}\right)$$

la differenza si può notare ugualmente. Per un'espressione fra parentesi così grande la cosa può anche andare bene, ma è certamente scorretta nella formula `$a\left(b+c\right)$`; a sinistra della riga metteremo la formula scorretta, a destra quella scritta correttamente `$a(b+c)$`:

$$a(b+c) \quad a(b+c).$$

Alcuni front-end per T_EX e L^AT_EX inseriscono automaticamente `\left` e `\right` prima delle parentesi: meglio disabilitare questa funzione. Oppure è possibile, per quanto visto prima, mettere l'espressione delimitata con `\left` e `\right` tra graffe che la trasformeranno da **Inner** a **Ord**, ciò che non è sempre desiderabile.

11 Stili matematici

Il titolo di questa sezione non tragga in inganno: non si discuterà di come scrivere la matematica, ma di un argomento T_EXnico. Infatti T_EX conosce ben otto stili matematici, che usa nelle varie situazioni: due per le formule in mostra, due per le formule in testo, due per apici e pedici di primo livello e gli ultimi due per apici e pedici di secondo livello. Il T_EXbook li indica con D , D' , T , T' , S , S' , SS e SS' . La differenza tra gli stili X e X' riguarda essenzialmente il posizionamento degli esponenti e non la tratteremo.

A ogni coppia di stili corrisponde un comando di scelta di quello stile: `\displaystyle`, `\textstyle`, `\scriptstyle` e `\scriptscriptstyle`. Si noti che questi comandi sono *dichiarazioni*, il loro effetto è valido in tutto il gruppo in cui si trovano: l'intera formula o una sottoformula tra graffe o la coppia `\left-\right`. Questi comandi *non* obbediscono ai gruppi tra `\begingroup` e `\endgroup`.

La differenza fondamentale tra stile D e T riguarda la scelta della grandezza di certi simboli, per esempio `\sum`. Questo simbolo viene preso dal font della famiglia 3 che possiede informazioni in

modo da sceglierne uno più grande se lo stile di formula in vigore è D invece che T :

```
\[{\textstyle\sum}\sum\]
```

sceglie due simboli diversi, cioè

$$\Sigma \sum$$

Infatti nella sottoformula lo stile in vigore è T , mentre fuori dalle graffe lo stile in vigore è D , dal momento che siamo in una formula in mostra. L'utente non deve fare nulla, di solito: la decisione sulla grandezza del simbolo usato spetta a TEX. Il motivo della differenza è che nelle formule in corpo un simbolo grande turberebbe la spaziatura verticale delle righe, problema che non si pone con le formule fuori corpo.

Normalmente l'assegnazione dello stile è automatica; sarà D quando TEX deve cominciare a comporre una formula fuori corpo, diventerà T durante l'elaborazione del numeratore di una frazione che appaia in stile D o D' (e sarà T' per il denominatore); sarà invece S per il numeratore (e S' per il denominatore) di una frazione che appaia in stile T . Se un radicale compare in stile D , il radicando viene composto in stile D' . Un esponente di un atomo in stile D , D' , T o T' viene composto in stile S , un pedice invece in stile S' . Ma, come abbiamo visto, è possibile forzare la scelta di uno stile con le dichiarazioni apposite. Per esempio, la definizione della macro `\dfrac` di `amsmath` è più o meno equivalente a

```
\def\dfrac#1#2{%
  {\displaystyle\frac{#1}{#2}}}
```

(si noti la coppia di graffe in più per confinare l'effetto della dichiarazione).

È spesso impossibile conoscere in anticipo in che stile verrà composta una formula o un certo simbolo, oppure è necessario definire questo simbolo in modo diverso nei vari stili. Per questo TEX ha la primitiva `\mathchoice` che prende quattro argomenti: le liste di token da eseguire nei quattro stili principali. Se scriviamo

```
\[ \def\x{\mathchoice{0}{1}{2}{3}}
\x\frac{x^{\x^{\x}}}{a} \]
```

otteniamo la strana formula

$$0 \frac{1^3}{a}$$

che mostra appunto le scelte fatte da TEX nei vari stili.

Supponiamo che non si gradiscano troppo i simboli del tipo `\bigoplus` che vengono usati in stile D , perché troppo grandi. Un modo possibile di ovviare al problema è di usare `\mathchoice`:

```
\newcommand*\boplus{\mathop{%
  \mathchoice{\textstyle\bigoplus}
    {\bigoplus}{\bigoplus}{\bigoplus}}}
```

In questo modo LATEX userà anche in stile D lo stesso simbolo che usa in stile T . Si noti che l'effetto non si propaga, perché l'argomento di `\mathop` tra graffe costituisce un gruppo. Ovviamente, cambiare uno dei simboli richiede di cambiare anche tutti quelli simili; vogliamo una macro che faccia il lavoro con il minimo sforzo. Eccola qui, adoperando la macro `\csletcs` del pacchetto `etoolbox`:

```
\newcommand*\reducebigsymbol[1]{%
  \csletcs{@@#1}{#1}%
  \@namedef{#1}{\mathop{\mathchoice
    {\textstyle\@nameuse{@@#1}}
    {\@nameuse{@@#1}}
    {\@nameuse{@@#1}}
    {\@nameuse{@@#1}}}}}
```

Naturalmente va data in un ambiente in cui `@` sia una lettera. Dopo questo, avremo che

```
\reducebigsymbol{\bigoplus}
\reducebigsymbol{\bigotimes}
\reducebigsymbol{\bigvee}
\reducebigsymbol{\bigwedge}
```

ridefinirà `\bigoplus` e gli altri indicati. Nel caso si dovesse tornare alle definizioni originali, basterà commentare le righe con `\reducebigsymbol`. Si tratta di una versione astratta della definizione precedente, con la differenza che questa salva il significato originale di `\bigoplus` (per esempio) nel comando `\@@bigoplus` e ridefinisce il comando stesso. Dunque il documento potrà essere scritto come sempre.

Più maneggevole è la macro `\mathpalette` che ha due argomenti; il primo dei due deve a sua volta essere una macro che prende due argomenti, il primo dei quali sia una delle dichiarazioni di stile. Per esempio,

```
\mathpalette\ab
```

si espande a

```
\mathchoice{\a\displaystyle\b}
{\a\textstyle\b}
{\a\scriptstyle\b}
{\a\scriptscriptstyle\b}
```

ed è compito della macro `\a` sapere che fare dei due argomenti. Tipicamente `\a` sarà definita con

```
\def\ab#1#2{{#1...#2...}}
```

cioè comporrà una sottoformula nello stile specificato (il primo argomento sarà sempre una delle dichiarazioni di stile) facendo qualcosa con il secondo argomento. Si possono trovare parecchi esempi in rete.

12 LATEX

Uno dei problemi fondamentali di LATEX è di fornire definizioni sufficientemente generiche che siano

adatte a diverse famiglie di caratteri e codifiche. Quando viene compilato il formato con $\text{\LaTeX}$ vengono letti quattro file: `latex.ltx` che contiene le definizioni del nucleo di $\text{\LaTeX}$, `fonttext.ltx`, `fontmath.ltx` e `preload.ltx`. Ci interessa in particolare il secondo che si occupa delle definizioni normali dei simboli matematici e dei font da usare. Pacchetti caricati dall'utente nel proprio documento potranno modificare queste definizioni (per esempio `fourier` o `mathpazo`).

Non studieremo tutto il file `fontmath.ltx`, ma solo le parti fondamentali. Come Plain definisce le famiglie matematiche 'obbligatorie' (da 0 a 3), anche $\text{\LaTeX}$ lo deve fare, ma in modo indiretto in modo che le attribuzioni di font alle famiglie dipendano dal corpo in uso. Perciò, invece di assegnare

```
\textfont0=\tenrm
\scriptfont0=\sevenrm
\scriptscriptfont0=\fiverm
```

come fa Plain, in $\text{\LaTeX}$ si scrive

```
\DeclareSymbolFont{operators}
  {OT1}{cmr}{m}{n}
```

Infatti la famiglia 0 riceve il nome simbolico `operators`; analogamente

```
\DeclareSymbolFont{letters}
  {OML}{cmm}{m}{it}
\DeclareSymbolFont{symbols}
  {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}
  {OMX}{cmex}{m}{n}
```

definiscono i font da usare per le famiglie 1, 2 e 3. Ricordiamo che in $\text{\LaTeX}$ ogni font è caratterizzato da quattro attributi: codifica, famiglia, peso e forma. Normalmente a ogni combinazione di codifica e famiglia corrisponde un file di descrizione (per esempio `ot1cmr.fd` o `omlcmm.fd`) dove vengono stabilite le corrispondenze delle combinazioni dei vari attributi con i 'veri' font disponibili. Queste famiglie vanno dichiarate in quest'ordine preciso, perché $\text{\TeX}$ ha bisogno che nelle famiglie da 0 a 3 ci siano font con certe caratteristiche.

Le codifiche OT1, OML, OMS e OMX sono esattamente quelle descritte nel $\text{\TeX}$ book per i font `cmr10`, `cmmi10`, `cmsy10` e `cmex10` e tutti i font matematici disponibili usano essenzialmente queste tabelle.

I comandi come `\mathrm` vengono definiti con

```
\DeclareSymbolFontAlphabet
  {\mathrm}{operators}
```

e questo fa in modo che una chiamata come `\mathrm{A}` sia sostituita da una sostanzialmente equivalente a

```
{\fam=\symoperators A}
```

perché il nome simbolico `operators` corrisponde alla costante `\symoperators`; analogamente per le altre famiglie:

Nome	Famiglia
<code>operators</code>	<code>\symoperators = 0</code>
<code>letters</code>	<code>\symletters = 1</code>
<code>symbols</code>	<code>\symsymbols = 2</code>
<code>largesymbols</code>	<code>\symlargesymbols = 3</code>
<code>AMSA</code>	<code>\symAMSA = 4</code>
<code>AMSb</code>	<code>\symAMSb = 5</code>

Le ultime due righe denotano famiglie definite dal pacchetto `amsfonts` (caricato automaticamente da `amssymb`); i valori mostrati non sono fissi, eccetto i primi quattro, perché altri pacchetti caricati prima potrebbero modificarli e l'uso di nomi simbolici evita di doverli conoscere.

Il file `fontmath.ltx` prosegue definendo i font matematici per la versione `bold`:

```
\SetSymbolFont{operators}{bold}
  {OT1}{cmr}{bx}{n}
\SetSymbolFont{letters}{bold}
  {OML}{cmm}{b}{it}
\SetSymbolFont{symbols}{bold}
  {OMS}{cmsy}{b}{n}
```

La sintassi è simile, ma il comando ha un argomento in più e può essere usato solo se sono già state definite la *versione* e la *famiglia*. Le versioni predefinite sono `normal` e `bold`, questa viene scelta con la dichiarazione

```
\mathversion{bold}
```

Di fatto l'introduzione dei nuovi comandi di $\text{\LaTeX}$ 2_ε evita quasi sempre il ricorso alle contorsioni necessarie per ottenere formule in nero: si carichi il pacchetto `bm`, se servono simboli in nero. Una nuova versione è definita con il comando

```
\DeclareMathVersion{nome}
```

che non ha altro effetto che di abilitare l'uso del comando `\SetSymbolFont` per questa versione. Se una particolare famiglia non ha associato un font per la versione scelta, viene adoperato quello associato alla versione `normal`, cioè quello dichiarato con `\DeclareSymbolFont`.

Successivamente `fontmath.ltx` definisce

```
\DeclareMathAlphabet{\mathbf}
  {OT1}{cmr}{bx}{n}
\DeclareMathAlphabet{\mathsf}
  {OT1}{cmsy}{m}{n}
```

e simili comandi per `\mathit` e `\mathtt`. Occorre moderazione nell'uso di questi 'alfabeti matematici', perché la prima apparizione di `\mathbf`, o di uno degli altri comandi dello stesso tipo, ordina l'allocazione di una nuova famiglia. Il numero di famiglie disponibili è, come sappiamo, sedici di cui quattro già impegnate per le esigenze minime. Per questo $\text{\LaTeX}$ non dice

```
\DeclareMathAlphabet{\mathrm}
{OT1}{cmr}{m}{n}
```

altrimenti anche l'uso di `\mathrm` allocherebbe una nuova famiglia: inutile, visto che la famiglia apposita è già disponibile, e dannoso perché getterebbe via una risorsa preziosa. Perciò il comando diverso

```
\DeclareSymbolFontAlphabet
{\mathrm}{operators}
```

che abbiamo già visto.

Il comando `\DeclareMathAlphabet` definisce i font per la versione `normal`; se si volesse definire il font corrispondente a `\mathsf` per la versione `bold` si potrebbe usare

```
\SetMathAlphabet{\mathsf}{bold}
{OT1}{cms}{bx}{n}
```

Il comando è del tutto analogo a `\SetSymbolFont`.

I comandi successivi in `fontmath.ltx` si occupano della corrispondenza del corpo con gli stili matematici. Per esempio, la riga

```
\DeclareMathSizes{10}{10}{7}{5}
```

dice che quando il corpo corrente è 10 pt, i simboli in stile *D* e *T* saranno di corpo 10, quelli in stile *S* di corpo 7 e quelli in stile *SS* in corpo 5.

LATEX ricalcola le associazioni dei font alle famiglie ogni volta che deve comporre una formula, basandosi sul corpo corrente. È una procedura che fa perdere un po' di tempo, ma non è possibile fare altrimenti, se si vuole che le formule si adattino al corpo in uso. Il comando interno per questa associazione è `\check@mathfonts`, che usa una delle dichiarazioni precedenti per dare un significato ai comandi `\tf@size`, `\sf@size` e `\ssf@size`. Se non ne trova di adatte, perché si usa un corpo non standard, esegue i calcoli necessari secondo uno schema ragionevole. Per esempio, a corpo 80 eseguirebbe

```
\def\tf@size{80}
\def\sf@size{56}
\def\ssf@size{40}
```

cioè il corpo in stile *S* è 8/10 del corpo principale, quello in stile *SS* è la metà. Soprattutto con corpi piccoli è consigliabile usare quelli standard, in modo che il calcolo non porti a font illeggibili. Per esempio LATEX definisce

```
\DeclareMathSizes{5}{5}{5}{5}
\DeclareMathSizes{6}{6}{5}{5}
```

proprio per evitare che venga usato il corpo 4 per gli esponenti e pedici di primo livello e addirittura 2.5 per quelli di secondo livello. Ma, naturalmente, scrivere a corpo 5 è da evitare comunque.

La sintassi usata da `fontmath.ltx` per dare valori al vettore `\mathcode` è assai particolare:

```
\DeclareMathSymbol{a}{\mathalpha}
{letters}{'a}
```

...

```
\DeclareMathSymbol{Z}{\mathalpha}
{letters}{'Z}
```

```
\DeclareMathSymbol{0}{\mathalpha}
{operators}{'0}
```

...

```
\DeclareMathSymbol{9}{\mathalpha}
{operators}{'9}
```

```
\DeclareMathSymbol{!}{\mathclose}
{operators}{'21}
```

```
\DeclareMathSymbol{*}{\mathbin}
{symbols}{'03}
```

...

```
\DeclareMathSymbol{.}{\mathord}
{letters}{'3A}
```

```
\DeclareMathSymbol{:}{\mathrel}
{operators}{'3A}
```

...

Si tratta essenzialmente di un'operazione equivalente (per la prima riga) a

```
\mathcode'a="7\symletters'a
```

dove `\symletters` e `'a` vanno pensati nella loro rappresentazione esadecimale. Di ogni carattere sono specificati nell'ordine il tipo, la famiglia da usare e il codice ASCII da impiegare. Si tratta solo di un modo più esplicito di impostare i codici rispetto a quello compatto ma oscuro direttamente con `\mathcode`.

Per il vettore `\delcode` e per i comandi definiti in Plain con `\delimiter` la sintassi è simile:

```
\DeclareMathDelimiter{(){\mathopen}
{operators}{'28}{largesymbols}{'00}
\DeclareMathDelimiter{\uparrow}{\mathrel}
{symbols}{'22}{largesymbols}{'78}
```

Nel caso il primo argomento sia un carattere, questo equivale a impostare il valore di `\delcode` e il secondo argomento è ignorato; se il primo argomento è un comando, questo è definito mediante `\delimiter`. Perciò il primo esempio equivale a⁹

```
\delcode' (=
"\symoperators28\symlargesymbols00
```

mentre il secondo a

```
\def\uparrow{\delimiter
'3\symsymbols22\symlargesymbols78}
```

Per verificarlo, scriviamo

```
\the\delcode' (
\meaning\uparrow
```

9. Qui si suppone sempre che `\sym...` sia tradotto nella sua rappresentazione esadecimale, non è possibile scrivere così quelle definizioni e infatti la definizione in LATEX non è questa, ma quella riportata prima.

per avere il significato (questo articolo è composto per L^AT_EX e quindi il risultato dipende dalle definizioni riportate):

```
164608
macro:->\delimiter "322378
```

che è corretto, essendo $164608 = 28300$.

Potremmo usare `\mathchoice` e, implicitamente, `\check@mathfonts` nel caso avessimo bisogno di un simbolo particolare di un font, diciamo $\odot$, come simbolo di un'operazione (associativa, vista la regola che segue):

$$a \odot (b \odot c) = (a \odot b) \odot c.$$

Possiamo definirlo senza sprecare una famiglia. Il simbolo si trova al posto "A9 del font Marvosym, accessibile tramite la famiglia mvs. Il pacchetto `pifont` ci permette di usare simboli arbitrari da ogni font di cui conosciamo la tabella:

```
\usepackage{pifont}
\newcommand*\Smile{
  \mathbin{
    \mbox{\Pisymbol{mvs}{A9}}
  }}

```

tratta il comando esattamente come desideriamo. Tuttavia questo simbolo non cambia grandezza se messo a esponente; possiamo allora sfruttare `\mathchoice`:

```
\usepackage{pifont}
\newcommand*\Smilesym[1]{%
  \mbox{\fontsize{#1}{#1}%
    \Pisymbol{mvs}{A9}}}
\makeatletter
\newcommand*\Smile{%
  \mathbin{\mathchoice
    {\Smilesym\tf@size}
    {\Smilesym\tf@size}
    {\Smilesym\tf@size}
    {\Smilesym\tf@size}}}
\makeatother

```

e abbiamo la possibilità di usare il simbolo anche a esponente:

$$x^{a \odot b} = x^a \odot x^b$$

(è un'operazione molto peculiare, $\odot$).¹⁰

Una definizione più generale potrebbe essere

```
\usepackage{pifont}
\makeatletter
\newcommand\Pimathsymbol[3][\mathord]{%
  #1{\@Pimathsymbol{#2}{#3}}}
\def\@Pimathsymbol#1#2{\mathchoice
  {\@Pimathsymbol{#1}{#2}\tf@size}
  {\@Pimathsymbol{#1}{#2}\tf@size}
  {\@Pimathsymbol{#1}{#2}\sf@size}
  {\@Pimathsymbol{#1}{#2}\sf@size}}
\makeatother

```

10. Dopo l'impostazione di `\fontsize` non è necessario dare `\selectfont` perché questo è implicito in `\Pifont`.

```
{\@Pimathsymbol{#1}{#2}\ssf@size}}
\def\@Pimathsymbol#1#2#3{%
  \mbox{\fontsize{#3}{#3}%
    \Pisymbol{#1}{#2}}}
\makeatother

```

data la quale potremmo dire

```
$a\Pimathsymbol[\mathbin]{mvs}{A9}b$
```

per avere la formula $a \odot b$. Ovviamente il tutto andrà in una definizione del tipo

```
\newcommand*\Smile{%
  \Pimathsymbol[\mathbin]{mvs}{A9}}

```

Unica limitazione è che il font sia disponibile nella codifica U (cioè 'generica'): se il nome della famiglia è `xyz`, occorre che nel sistema sia presente il file `uxyz.fd`, che di solito succede per i font di simboli. Il comando `\Pimathsymbol` ha come argomento opzionale il tipo di atomo che desideriamo (valore normale `\mathord`), come primo argomento obbligatorio la famiglia e come secondo argomento il byte da adoperare, esattamente come per `\Pisymbol`. Altro esempio: un simbolo per indicare una contraddizione potrebbe essere

```
$\Pimathsymbol{mvs}{45}$
```

che darebbe ! . È possibile produrre la tabella di un font di cui si conosca il nome della famiglia compilando con L^AT_EX il seguente file `testfam.tex`:

```
\let\noinit!
\input{nfssfont}
\fontencoding{U}
\fontfamily{mvs}
\selectfont
\table\bye

```

Chi è pratico della linea di comando può sostituire in quel file la chiamata esplicita della famiglia con la riga

```
\fontfamily{\test}
```

Lanciando la compilazione con

```
> latex "\def\test{pzd}\input{testfam}"
```

si ricaverebbe la tabella del font Zapf Dingbats. Si ricordi che i comandi in cui si usa il doppio apice " vanno dati prima di caricare `babel`.

Per chi non vuole fare le cose a mano, c'è il pacchetto `fonttable` di Peter Wilson che permette di realizzare tabelle di font e di impostare il font anche con gli attributi di L^AT_EX: il comando `\xfonttable`, a differenza del comando principale `\fonttable` del pacchetto, ha quattro argomenti; nell'ordine (1) la codifica, (2) il nome della famiglia, (3) il peso, (4) la forma.¹¹ La tabella del Computer

11. Il comando `\xfonttable` è stato aggiunto su suggerimento dell'autore con il codice concepito scrivendo questo articolo.

Modern nero corsivo in codifica T1 si otterrebbe con

```
\xfonttable{T1}{cmr}{bx}{it}
```

mentre quella del font Marvosym che ci interessava per il simbolo © si avrebbe con

```
\xfonttable{U}{mvs}{m}{n}
```

Il vantaggio di usare questo comando è che la tabella riporta anche i valori decimali con cui richiamare il carattere, come si vede nella tabella 1 e questo permette di definire i comandi precedenti con il riferimento al valore decimale, eliminando il problema di `babel`. La tabella non mostra tutti i caratteri del font, per questioni di spazio.

13 Altre tecniche

Certi autori preferiscono usare un simbolo simile a $\times$ per indicare prodotti di famiglie di insiemi, ma occorre che sia nello stile degli operatori grandi:

$$\bigtimes_{i=1}^n A_i$$

Il pacchetto `dlftxbcodetips` di Lars Madsen fornisce una definizione adatta:¹²

```
\newcommand\dlf@b@gtimes[1]{%
  \vcenter{\hbox{%
    #1$\m@th\mkern-2mu
    \times\mkern-2mu$}}
\newcommand\dlf@b@bigtimes{%
  \mathchoice{\dlf@b@gtimes\huge}
    {\dlf@b@gtimes\LARGE}
    {\dlf@b@gtimes}
    {\dlf@b@gtimes\footnotesize}
}
\newcommand\bigtimes{%
  \mathop{\dlf@b@bigtimes}\displaylimits}
```

Come in altre situazioni si impiega `\mathchoice` in modo che il simbolo sia della grandezza giusta nei quattro stili matematici. La particolarità di questa definizione è nell'uso di `\vcenter` che crea un atomo **Vcent** (che è trattato poi come se fosse **Ord**). La sintassi di `\vcenter` è la stessa di `\vbox`, ma questo comando può essere usato solo in modo matematico e la `vbox` costruita viene centrata rispetto all'asse delle formule, come richiesto per gli operatori grandi. Si usa, come si vede, il simbolo `\times` a un font più grande (o più piccolo). Un limite della definizione adottata da Madsen è che non funziona se la formula appare in un contesto nel quale sia valida una dichiarazione come `\small` o `\footnotesize`. La si potrebbe modificare nel modo seguente:

12. La definizione è dovuta all'autore, risultato di una discussione su `comp.text.tex`.

```
\makeatletter
\@ifundefined{@currsizeindex}%
  {\RequirePackage{relsize}%
   \let\@bl@rger\relsize}
  {\def\@bl@rger#1{\larger[#1]}}
\def\bigtimes{
  \mathop{\@bigtimes}\displaylimits}
\def\@bigtimes{
  \mathchoice{\@bigt@mes4}{\@bigt@mes3}
    {\@bigt@mes0}{\@bigt@mes{-1}}}
}
\def\@bigt@mes#1{
  \vcenter{\hbox{\@bl@rger{#1}%
    $\m@th\mkern-2mu\times\mkern-2mu$}}}
\makeatother
```

Si tratta di una definizione molto simile, solo che le variazioni di grandezza sono relative al corpo attuale, sfruttando il comando `\relsize` del pacchetto `relsize`. Solo nel caso in cui sia definito il comando `\@currsizeindex` la definizione è diversa, perché la classe del documento è una di quelle dell'AMS nelle quali è definito un comando `\larger` che ammette come argomento opzionale un 'salto'. In queste classi `\larger[4]` è equivalente a `\relsize{4}`.

Come esercizio si analizzino le definizioni di `\nuparrow` e `\ndownarrow`

```
\RequirePackage{amssymb,graphicx}
\def\@nrotarrow#1#2{%
  \setbox0=\hbox{$\m@th#1\uparrow$}%
  \dimen0=\dp0
  \setbox0=\hbox{%
    \reflectbox{\rotatebox[origin=c]{90}%
      {$\m@th#1\mkern2.22mu #2$}}}%
  \dp0=\dimen0 \box0 \mkern2.3965mu
}
\def\nuparrow{\mathrel{
  \mathpalette\@nrotarrow\nrightarrow}}
\def\ndownarrow{\mathrel{
  \mathpalette\@nrotarrow\nleftarrow}}
```

I simboli prodotti sono le negazioni di `\uparrow` e `\downarrow`:

$$\begin{array}{cc} a \uparrow b & a \downarrow b \\ a \nuparrow b & a \ndownarrow b \end{array}$$

Naturalmente non è possibile usare `\nuparrow` e `\ndownarrow` dopo `\left` e `\right` come invece è ammesso con `\uparrow` e `\downarrow`.

14 Casi speciali

TEX considera il codice matematico solo dei caratteri di categoria 11 o 12. Gli altri vengono trattati al modo solito e anche l'espansione delle macro avviene del tutto normalmente, compresa quella dei caratteri attivi. Fanno eccezione i caratteri di categoria 10 (spazio) che in modo matematico sono

TABELLA 1: Tabella del font Marvosym ottenuta con `\xfonttable{U}{mvs}{m}{n}` dando `\fontrange{"40}{AF}`.

	'0	'1	'2	'3	'4	'5	'6	'7	
'10x	@ ₆₄	Ⓐ ₆₅	Ⓜ ₆₆	Ⓒ ₆₇	€ ₆₈	⚡ ₆₉	Ⓢ ₇₀	— ₇₁	"4x
'11x	☺ ₇₂	Ⓜ ₇₃	Ⓢ ₇₄	☕ ₇₅	Ⓜ ₇₆	77	□ ₇₈	□ ₇₉	
'12x	— ₈₀	∞ ₈₁	--- ₈₂	Ⓢ ₈₃	☎ ₈₄	⌚ ₈₅	✓ ₈₆	✂ ₈₇	"5x
'13x	✂ ₈₈	☯ ₈₉	☎ ₉₀	/ ₉₁	✂ ₉₂	≡ ₉₃	≡ ₉₄	/ ₉₅	
'14x	Ⓢ ₉₆	☼ ₉₇	Ⓢ ₉₈	€ ₉₉	€ ₁₀₀	€ ₁₀₁	Ⓢ ₁₀₂	— ₁₀₃	"6x
'15x	☼ ₁₀₄	Ⓢ ₁₀₅	☼ ₁₀₆	☼ ₁₀₇	↓ ₁₀₈	☼ ₁₀₉	☼ ₁₁₀	☼ ₁₁₁	
'16x	— ₁₁₂	✂ ₁₁₃	--- ₁₁₄	✂ ₁₁₅	Ⓢ ₁₁₆	Ⓢ ₁₁₇	☼ ₁₁₈	☼ ₁₁₉	"7x
'17x	Ⓢ ₁₂₀	Ⓢ ₁₂₁	Ⓢ ₁₂₂	○ ₁₂₃	♂ ₁₂₄	♀ ₁₂₅	♀ ₁₂₆	♂ ₁₂₇	
'20x	♂ ₁₂₈	♀ ₁₂₉	□ ₁₃₀	♂ ₁₃₁	♀ ₁₃₂	♀ ₁₃₃	† ₁₃₄	♂ ₁₃₅	"8x
'21x	† ₁₃₆	☼ ₁₃₇	☼ ₁₃₈	☼ ₁₃₉	♥ ₁₄₀	@ ₁₄₁	Ⓢ ₁₄₂	□ ₁₄₃	
'22x	☼ ₁₄₄	● ₁₄₅	● ₁₄₆	■ ₁₄₇	■ ₁₄₈	● ₁₄₉	— ₁₅₀	□ ₁₅₁	"9x
'23x	□ ₁₅₂	Ⓢ ₁₅₃	Ⓢ ₁₅₄	○ ₁₅₅	Ⓢ ₁₅₆	Ⓢ ₁₅₇	Ⓢ ₁₅₈	Ⓢ ₁₅₉	
'24x	Ⓢ ₁₆₀	β ₁₆₁	Ⓢ ₁₆₂	Ⓢ ₁₆₃	€ ₁₆₄	Ⓢ ₁₆₅	\$ ₁₆₆	☼ ₁₆₇	"Ax
'25x	☼ ₁₆₈	☼ ₁₆₉	Ⓢ ₁₇₀	Ⓢ ₁₇₁	Ⓢ ₁₇₂	Ⓢ ₁₇₃	Ⓢ ₁₇₄	Ⓢ ₁₇₅	
	"8	"9	"A	"B	"C	"D	"E	"F	

del tutto ignorati; ne segue che anche un carattere di fine riga (categoria 5) convertito in spazio sarà ignorato e quindi una formula può cominciare in una riga del documento `.tex` e finire in una successiva. Tuttavia non è possibile lasciare righe vuote in un ambiente matematico, perché una riga vuota equivale a un comando `\par` che in modo matematico è illegale.

Un carattere di categoria 11 o 12 può avere codice matematico "8000 che lo rende molto speciale: viene infatti trattato come se fosse un carattere attivo (cioè una macro) e `TeX` ne deve avere una definizione proprio come carattere attivo. L'esempio classico è quello dell'apostrofo:

```
\mathcode'\='8000
{\catcode'\='active
 \gdef'\bgroup\prim@s}}
```

Non ci interessa qui analizzare la definizione, quanto di segnalare questa possibilità. Una definizione ingenua potrebbe essere

```
{\catcode'\='active
 \gdef'\prime}}
```

che però non sarebbe affatto efficiente; ciò che conta è che l'apostrofo in modo matematico si comporta come una macro.

Esistono altri caratteri con codice matematico "8000:

```
\mathcode'\='8000 % \space
\mathcode'\='8000 % ^\prime
\mathcode'\_='8000 % \_
{\catcode'\_='active
 \global\let\_=_}}
```

Queste sono le definizioni in Plain, in `LATEX` sono equivalenti. In realtà, a parte l'apostrofo, servono solo a trattare casi particolari: certe macro potrebbero rendere lo spazio e il carattere `'_` di categoria 11 o 12, con queste definizioni lo spazio verrebbe trattato come uno spazio (il carattere spazio di categoria 13 è definito come `\space`, quindi alla fine sarebbe ignorato). Invece `'_` attivo stampa il carattere stesso: è chiaro, se lo rendiamo 'stampabile', è perché non lo vogliamo più come indicatore dei pedici.

Uno dei vantaggi di questo tipo di 'caratteri matematicamente attivi' è che la loro espansione avviene solo durante la composizione tipografica, non quindi durante la scrittura di file ausiliari e quindi sono automaticamente robusti. Tuttavia non è facile immaginare applicazioni significative di questa possibilità, se non quella di riservare il carattere `'?` per gli allineamenti in colonna di numeri:

```
\mathcode'?'="8000
\begin{group}\lccode'\_?' \lccode'0='0
 \lowercase{\endgroup\def~{\phantom{0}}}
```

La tabella seguente sfrutta questa definizione da mettere nel preambolo: è codice un po' oscuro, ma la magia ha le sue regole.

Unità	Valore (sp)
punto	65 536
millimetro	186 467
punto Didot	70 124
pollice	4 736 286

È in generale molto complicato allineare in colonna numeri sotto un'intestazione più larga dei numeri stessi, se vogliamo che l'intestazione sia centrata. La soluzione consiste nel definire la colonna con allineamento al centro e rendere tutti i numeri della stessa ampiezza. Nel preambolo si è data la definizione richiesta del punto interrogativo come attivo in modo matematico e la tabella è inserita con

```
\begin{tabular}{l>{$}c<{$}}
\toprule
\multicolumn{1}{c}{Unità} &
$Valore (sp)$\\
\midrule
punto & ?\,?65\,536\\
millimetro & ?\,186\,467\\
punto Didot & ?\,?70\,124\\
pollice & 4\,736\,286\\
\bottomrule
\end{tabular}
```

Si è usato il pacchetto `array` per specificare che la seconda colonna è in modo matematico, i caratteri \$ servono, nell'intestazione, ad annullare l'effetto di quelli inseriti a inizio e fine della cella. Il punto interrogativo inserisce uno spazio equivalente a una cifra; naturalmente occorre usare un font in cui tutte le cifre abbiano la stessa larghezza, ma questa regola vale in generale per tabelle con dati numerici. Il prezzo da pagare è l'impossibilità di lasciare i dati così come sono; la rinuncia al carattere ? in modo matematico non sembra una grave perdita.

Non è l'unico modo, ma ha il pregio di essere automatico. La stessa tabella si può ottenere senza usare il trucco di prima, eseguendo alcune misure possibili solo quando si conoscano già i dati della tabella: qui si è calcolata la differenza tra l'ampiezza dell'intestazione e quella del numero maggiore; divisa per due dice di quanto spostare l'intestazione verso destra e altrettanto spazio va inserito in più tra prima e seconda colonna che, però, avrà un allineamento a destra.

Unità	Valore (sp)
Punto	65 536
Millimetro	186 467
Punto Didot	70 124
Pollice	4 736 286

Il codice, nella parte iniziale, è

```
\setlength{\dimen0}{
(\widthof{Valore (sp)}-
\widthof{4\,736\,286})/2
}
\begin{tabular}{lr!{\hspace{\dimen0}}}{
\toprule
...

```

che usa le operazioni messe a disposizione dal pacchetto `calc`.

Un secondo uso dei caratteri attivi in modo matematico è per poter usare la virgola come delimitatore della parte decimale di un numero. Come abbiamo visto, il codice matematico della virgola corrisponde a un segno di punteggiatura, perché l'uso maggiore è, almeno nei paesi di lingua inglese, per formule del tipo (x, y) . In Italia si usa invece la virgola per dividere la parte intera da quella decimale; il trucco più semplice per ingannare TEX è di scrivere `1{,}05`, ma questo può essere noioso e porta a errori di composizione se ci si dimentica delle graffe. Una possibile soluzione è di usare il pacchetto `icomma` di Walter Schmidt il cui contenuto è, con modifiche inessenziali,

```
\AtBeginDocument{%
\mathchardef\mathcomma\mathcode'\,
\mathcode'\,="8000 }
{\catcode'\,=\active
\gdef{\futurelet\@let@token\sm@rtcomma}}
\def\sm@rtcomma{%
\ifx\@let@token\@sptoken\else
\mathord\fi\mathcomma}
```

Non entriamo nel dettaglio di `\futurelet`; il funzionamento è in sostanza questo: quando TEX incontra una virgola in modo matematico, osserva anche il *token* che la segue (il carattere o comando successivo, per intenderci); se questo *token* non è uno spazio, ciò che verrà eseguito è `\mathord\mathcomma`, mentre se il token è uno spazio (`\@sptoken` è gergo LATEX per uno spazio) verrà eseguito solo `\mathcomma`. Questo comando è definito usando il codice matematico della virgola, lo si fa a inizio del documento per evitare problemi nel caso si usino pacchetti che assegnano alla virgola un codice matematico insolito.

Il pacchetto dunque permette di scrivere `$1,05$` e di ottenere 1,05; se si vuole una virgola come segno di punteggiatura, occorre lasciare uno spazio (o un fine riga), come in `$(a, b)$` per avere (a, b) . Un piccolo difetto di questo approccio è che richiede una grande disciplina nella scrittura del documento `.tex`, ma è certamente conveniente se si hanno parecchi numeri decimali da scrivere in modo matematico.

Avvertenza

Si faccia attenzione quando si usa `babel`, perché quasi tutti i suoi moduli per le lingue rendono attivo il carattere ", producendo misteriosi errori quando si vogliono passare a TEX numeri in formato esadecimale. In queste note abbiamo finto di scrivere "1234, dove in effetti abbiamo scritto `\string"1234`, che evita il problema. Un modo diverso di evitarlo è di scrivere il codice tra

```
\shorthandoff{"}
...
\shorthandon{"}
```

oppure usare un gruppo come in

```
{\shorthandoff{"}...}
```

Nel preambolo del documento il problema non c'è, perché " è attivato solo nell'ambiente `document`, purché non si sia dato il comando

```
\selectlanguage{italian}
```

o altro simile, che però non andrebbe mai dato nel preambolo. L'uso delle tabelle di font prodotte con i comandi di `fonttable` permette di conoscere il codice decimale dei caratteri da usare. Dunque il doppio apice può dare problemi solo se si dovesse usare nel documento un comando esplicito come `\mathchar"613B`, cosa del tutto sconsigliabile: meglio definire una macro nel preambolo.

Riferimenti bibliografici

BECCARI, C. (1997). «Typesetting mathematics for science and technology according to ISO 31/XI». *TUGboat*, **18** (1), pp. 39–48.

BECCARI, C. (a cura di) (2009). *Introduzione all'arte della composizione tipografica*. GfT. URL <http://www.guit.sssup.it>.

GUIGGIANI, M. e MORI, L. F. (2008). «Consigli su come *non* maltrattare le formule matematiche». *ArsT_EXnica*, **6**, pp. 5–14.

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

Produrre grafica vettoriale di alta qualità programmando *Asymptote*

Agostino De Marco

Sommario

Asymptote è un potente sistema grafico programmabile, distribuito con licenza GNU e particolarmente adatto a produrre disegni tecnici. Il suo linguaggio di programmazione è di alto livello ed è basato su sofisticate funzionalità matematiche. *Asymptote* permette di comporre con L^AT_EX le annotazioni delle figure (semplici etichette ed oggetti testuali più complessi) garantendo una resa tipografica di alta qualità.

Questo articolo non offre una panoramica completa su *Asymptote* ma, piuttosto, si propone di introdurre gradualmente, con degli esempi mirati, gli elementi salienti del suo linguaggio di programmazione e di porre in risalto gli aspetti importanti per l'utilizzatore L^AT_EX. Data la vastità degli argomenti legati alla programmazione di un linguaggio di alto livello, il lettore interessato agli approfondimenti sulle potenzialità di questo sistema grafico è rimandato ai riferimenti citati ed allo studio del codice sorgente delle numerose funzioni predefinite.

Abstract

Asymptote is a powerful programmable graphics system, distributed with a GNU license. It provides a high level descriptive programming language, which is based on advanced mathematical functions. *Asymptote* is particularly suited to produce technical drawings. It allows users to compose labels and more complicated textual objects with L^AT_EX and this feature guarantees a high-quality typographical performance.

This article does not give a complete overview of *Asymptote*, rather it has the aim at introducing gradually, with appropriate examples, the main elements of its programming language emphasizing the aspects which are of interest for the L^AT_EX user. Given the breadth of topics related to a high level programming language, the reader interested in the details and in the potential of this graphics system is advised to read carefully the references cited and to study the source code of the many pre-defined functions.

1 Introduzione

Gli utenti di L^AT_EX hanno spesso bisogno d'inserire nei loro documenti disegni, schemi e figure di varia complessità. Sui precedenti numeri di *Ar_ST_EX*nica

sono comparsi diversi articoli che trattano questo argomento. Si vedano: CASCHILI (2006) per un'introduzione all'uso del pacchetto `picture`, CASCHILI (2007) per un'interessante panoramica sul pacchetto `pstricks`, ed inoltre DE MARCO (2008) per una discussione sulla gestione avanzata delle figure in L^AT_EX (con esempi d'uso dei pacchetti `psfrag`, `tikz` e `pgfplots`).

Un'ottima descrizione delle numerose possibilità offerte da L^AT_EX tramite i suoi pacchetti di estensione per la produzione di materiale grafico è data nell'articolo di BECCARI (2007) apparso sulla rivista *PracT_EX Journal*. Questo lavoro si concentra sulle tecniche adatte a creare disegni direttamente nel sorgente L^AT_EX di origine (senza l'uso di programmi esterni o limitandosi, al più, a qualche conversione di formato).

Un'introduzione all'uso del programma *Sketch* è presentata in DE MARCO (2007). È questo un esempio in cui un sistema esterno, programmabile attraverso un linguaggio grafico descrittivo, produce in uscita dei frammenti di codice `pstricks` o `tikz`, da includere in un sorgente L^AT_EX. Questo articolo è orientato, in particolare, alla produzione di illustrazioni di grafica tridimensionale.

Gli utenti L^AT_EX hanno anche la possibilità di lavorare con strumenti di disegno esterni, comunemente detti 'L^AT_EX-aware'. Essi sono cioè capaci di comporre le annotazioni testuali all'interno delle figure utilizzando una distribuzione pre-installata di L^AT_EX. Questa caratteristica assicura la coerenza tipografica delle immagini create con questi strumenti con i documenti L^AT_EX destinati ad includerle. Ne risulta un alto livello di qualità complessiva della composizione.

Un esempio molto interessante di strumento di disegno L^AT_EX-aware è costituito dal programma *Ipe*, disponibile sia per Linux che per Windows¹. *Ipe* è un'applicazione dotata di interfaccia grafica (*graphical user interface*, GUI). Essa si presenta come un *editor* visuale di disegni ed è stata sviluppata per la creazione di immagini direttamente in formato PDF (*Portable Document Format*) o EPS (*Encapsulated PostScript*). L'utente può inserire nei disegni delle annotazioni testuali attraverso un'apposita GUI, nella quale viene richiesto di digitare il corrispondente codice L^AT_EX. Ciascuna annotazione viene compilata in un processo separa-

1. <http://tclab.kaist.ac.kr/ipe/> (sito principale), <http://luaforge.net/projects/ipe/> (*repository*), <http://melusine.eu.org/lab/ipe> (galleria di esempi)

to dall'applicazione principale ed il risultato viene importato come oggetto grafico all'interno della finestra di lavoro. Ciò risulta praticamente trasparente all'utente, che ha l'impressione di lavorare con una comune applicazione grafica con il vantaggio di ottenere delle annotazioni create con L^AT_EX. *Ipe* permette di aggiungere delle funzionalità personalizzate attraverso il meccanismo dei *plug-in* detti *ipelet* (cioè delle librerie a collegamento dinamico che l'utente deve sviluppare in C++).

Va osservato che anche *Inkscape*, il popolare programma di grafica vettoriale², è espandibile con dei *plug-in* sviluppati in linguaggio Python. Tra questi va annoverata l'estensione *textext*, che permette di elaborare oggetti testuali mediante pdfL^AT_EX e di renderli nel *viewport* dell'applicazione come tracciati³.

Nel seguito del presente articolo illustrerò le caratteristiche principali del programma di grafica vettoriale *Asymptote* (BOWMAN, 2009) disponibile per Linux, Windows e Mac. Questo strumento di disegno, essendo anch'esso L^AT_EX-aware, è concettualmente simile ad *Ipe*. A differenza di quest'ultimo, *Asymptote* è anche un interprete di comandi e dispone di un linguaggio di programmazione di alto livello orientato alla matematica. Ciò lo rende un potente strumento, adatto alla produzione di disegni tecnici e di immagini a carattere matematico.

Esistono molte informazioni su internet su *Asymptote*. Una breve introduzione all'uso è data in HAMMERLINDL e BOWMAN (2007). Il forum *Art of Problem Solving*⁴ ha una sezione dedicata a questo programma e ne fornisce un *tutorial* di livello base. Di notevole interesse sono i due articoli scritti da John Bowman, lo sviluppatore di *Asymptote* insieme a Andy Hammerlindl e Tom Prince, pubblicati sulla rivista *TUGboat*: il primo (BOWMAN e HAMMERLINDL, 2008) presenta le caratteristiche principali, il secondo (BOWMAN e SHARDT, 2009) discute gli algoritmi su cui si basano le funzionalità orientate alla rappresentazione di scene tridimensionali.

Inoltre, un'ampia galleria di esempi d'uso (corredati di sorgenti) è disponibile sul sito ufficiale di *Asymptote*⁵. Il sito contiene anche dei collegamenti a molte risorse esterne⁶, tra cui due ulteriori gallerie di esempi⁷ ed una eccellente guida passo passo scritta da Philippe Ivaldi, un utilizzatore esperto che si è distinto per la sua produttività⁸.

2. <http://www.inkscape.org>

3. <http://www.elisanet.fi/ptvirtan/software/textext/>

4. [http://www.artofproblemsolving.com/wiki/index.php/Asymptote_\(Vector_Graphics_Language\)](http://www.artofproblemsolving.com/wiki/index.php/Asymptote_(Vector_Graphics_Language))

5. <http://asymptote.sourceforge.net/gallery/>

6. <http://asymptote.sourceforge.net/links.html>

7. <http://piprim.tuxfamily.org/asymptote/index.html> e <http://marris.org/asymptote>

8. <http://piprim.tuxfamily.org/asymptote/generale/index.html>

2 Come funziona *Asymptote*?

Lo strumento di lavoro si presenta all'utilizzatore come un'applicazione *command-driven*. Questo significa che all'utente è richiesto di raccogliere la sequenza di comandi necessari a realizzare un dato disegno in un file sorgente (*script* di estensione *.asy*), da compilare con *Asymptote* per ottenere un'immagine vettoriale in formato PostScript o PDF. Questa modalità di funzionamento è chiamata *batch mode*. Detto *test.asy* un file sorgente, l'utente digiterà il comando *asy -V test* al *prompt* di una finestra dei comandi ottenendo il file *test.eps* (l'opzione *-V* sta per *view* e permette di aprire automaticamente il file di output con un'applicazione predefinita).

Di default *Asymptote* produce un output in formato PostScript, ma può anche generare un'uscita in qualsiasi formato che il pacchetto *ImageMagick*⁹ sia in grado di produrre (opzione *-outformat <formato>*). È possibile configurare opportunamente l'applicazione impostando per default il formato di uscita PDF e pdfL^AT_EX come compositore delle annotazioni testuali (opzione *-tex pdflatex*).

Per gli utenti Windows con un ambiente di lavoro opportunamente configurato è sufficiente fare doppio click su un file di estensione *.asy* per lanciare un processo di compilazione con *Asymptote* (tramite l'eseguibile *asy.exe*) ed ottenere automaticamente un'immagine nel formato desiderato.

L'utilizzo di *Asymptote* in modalità *batch* è consigliabile. Esso si contrappone alla modalità *interattiva* per la quale si rimanda alla consultazione del manuale d'uso (HAMMERLINDL *et al.*, 2009).

Nella distribuzione ufficiale di *Asymptote* è compresa una rudimentale applicazione, denominata *xasy*, dotata di interfaccia grafica ed utile ad apportare semplici modifiche a disegni precedentemente generati tramite uno *script*. Questo programma, attualmente in fase di sviluppo, è destinato a diventare una vera e propria applicazione interattiva, capace di generare direttamente dei nuovi oggetti grafici. Ciò renderà *Asymptote* accessibile all'utente medio, permettendo a questo strumento di disegno di raggiungere il livello di semplicità d'uso di *Ipe* (almeno per gli utenti interessati al metodo di lavoro esclusivamente visuale).

Una caratteristica ulteriore, di non trascurabile importanza, è che il codice *Asymptote* può essere incluso all'interno di un sorgente L^AT_EX mediante il pacchetto di estensione *asymptote* (presente nella distribuzione ufficiale) che mette a disposizione l'ambiente *asy*. Si veda più avanti in questo articolo per maggiori dettagli.

9. <http://www.imagemagick.org/>

3 Caratteristiche del linguaggio di programmazione *Asymptote*

Asymptote offre un potente linguaggio descrittivo di grafica vettoriale pensato per la creazione di disegni tecnici. Esso dispone di numerose funzioni matematiche che permettono di ottenere risultati di notevole qualità e precisione.

Gli autori del linguaggio dichiarano di averlo concepito allo scopo di creare uno strumento standard per la creazione di figure matematiche, così come LATEX rappresenta lo standard *de facto* per la creazione di documenti matematici. Attualmente *Asymptote* è considerato il moderno successore di METAPOST (BOWMAN e HAMMERLINDL, 2008; HOBBY, 2009).

Uno dei vantaggi principali di *Asymptote* rispetto ad altri pacchetti grafici è che esso mette a disposizione un vero e proprio linguaggio di programmazione di alto livello. Ciò è vero solo per poche altre applicazioni di grafica, quasi tutte commerciali.

Asymptote ha la caratteristica di essere un ambiente matematico di disegno, basato sull'uso naturale di sistemi di coordinate. Il suo linguaggio di programmazione è ispirato a METAPOST, ma con una sintassi molto più pulita e potente, quasi del tutto simile a quella del C++.

Asymptote è stato progettato soprattutto per la creazione di figure matematiche che hanno bisogno di composizione tipografica. Gli utenti hanno comunque la possibilità di scrivere *moduli di estensione* (raccolte di funzioni) che aggiungono nuove funzionalità all'ambiente grafico e lo adattano alle specifiche esigenze. Molte delle nuove caratteristiche di questo sistema grafico, che si vanno migliorando o aggiungendo ad ogni nuova *release*, sono realizzate nel suo stesso linguaggio, sotto forma di funzioni raccolte in moduli.

L'estensione del linguaggio tramite funzioni definite dall'utente rientra fra le tecniche di programmazione tradizionali, messe a disposizione da tutti i linguaggi procedurali. Più avanti vedremo con un semplice esempio come *Asymptote*, al pari del C++, supporta il paradigma di programmazione a oggetti (STROUSTRUP, 2000). Esso non solo permette al programmatore di definire strutture dati personalizzate che inglobano anche delle *funzioni membro* (tipi *concreti*), ma supporta anche il meccanismo di ereditarietà che consente di costruire gerarchie di strutture dati (funzioni membro *virtuali* e tipi *derivati*).

Nelle distribuzioni recenti di *Asymptote* sono già disponibili dei moduli di estensione sviluppati da alcuni utenti esperti, che l'utilizzatore generico può importare all'occorrenza nel suo codice sorgente. Va segnalato il modulo `geometry.asy` di Philippe Ivaldi¹⁰.

10. http://svnweb.tuxfamily.org/filedetails.php?repname=piprim%2Fgeometrydoc&path=%2Fbranches%2Fen%2Fgeometry_en.pdf

4 Disegnare programmando

In *Asymptote*, ogni comando deve essere seguito da un punto e virgola (;). Questo permette all'interprete di separare un comando dal successivo. Pertanto, due o più comandi non hanno bisogno di comparire su righe separate. Gli spazi bianchi prima e dopo i comandi sono ignorati da *Asymptote*, quindi si può fare uso dell'indentazione dei comandi per migliorare la leggibilità del codice.

Il testo di commento, che non va ad influenzare l'output, può essere inserito nel codice secondo le regole del C++, facendolo iniziare con la doppia barra // o circondandolo con la coppia `/* ... */`.

Un esempio di codice minimale che disegna un segmento è il seguente:

```
/* Disegniamo un segmento dal punto
(0,0) al punto (50,50) */
draw((0,0)--(50,50)); // fatto!
```

La funzione di disegno `draw` è discussa più avanti.

Come avviene spesso nei testi introduttivi dei linguaggi di programmazione, è bene anche qui mostrare la semplicità del codice *Asymptote* necessario a stampare la frase tormentone "Hello world!". Esso consiste nella sola istruzione:

```
label("Hello world!", (0,0), N);
```

e fa uso della funzione `label`. La funzione è stata invocata richiedendo di posizionare la stringa di testo desiderata a Nord (N) del punto di coordinate (0,0). L'aspetto importante da sottolineare qui è che la stringa di testo viene processata di default dal programma `latex` oppure, a richiesta, da `pdflatex`. L'esempio seguente mostra la possibilità di personalizzare il meccanismo di composizione del testo con dei comandi LATEX che si inserirebbero nel preambolo di un documento:

```
usepackage("guil");
usepackage(
    "siunitx", // nome pacchetto
    "decimalsymbol=comma"); // opzione
texpreamble("\newcommand{\anno}{2009}")
);
label("\Large\GuITmeeting
[style=display, year=\anno]",
(0,0), N);
label(
"$\ell=\SI{4.5}{\centi\meter}$",
```

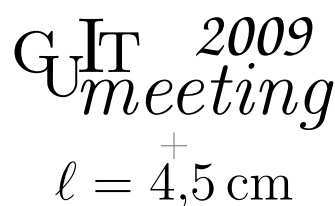


FIGURA 1: Comporre etichette in *Asymptote* con l'ausilio di pacchetti di estensione LATEX, attraverso le funzioni `usepackage` e `texpreamble`.

(0,0),S);

Questo codice realizza l'immagine della figura 1. La funzione predefinita `usepackage` permette di caricare un pacchetto di estensione di L^AT_EX nel contesto del processo di composizione delle annotazioni (ed eventualmente di specificare delle opzioni). La funzione `texpreamble` permette di inserire una determinata stringa nel preambolo. Nel caso precedente si è definita la macro `\anno`. Nel caso in cui si ha l'esigenza di utilizzare un preambolo più articolato e di maggiore complessità sarà conveniente preparare un file, ad esempio `my_preamble.tex`, salvarlo nella stessa cartella di lavoro in cui risiede il sorgente *Asymptote* ed invocare la funzione come segue:

```
texpreamble("\input{my_preamble}");
```

Una delle caratteristiche più utili del linguaggio *Asymptote* è la possibilità di definire nuove funzioni. Il seguente esempio risulterà senz'altro comprensibile anche ai programmatori meno esperti:

```
void drawcross(real x1, real y1,
               real x2, real y2)
{
    draw((x1,y1)--(x2,y2));
    draw((x1,y2)--(x2,y1));
}
```

Nelle intenzioni del programmatore la funzione `drawcross` potrebbe voler disegnare due segmenti che si incrociano. La parola chiave `void` significa che la funzione non ritorna nessun valore (è una procedura). La parola chiave `real` determina il tipo degli argomenti di scambio. I tipi di dati predefiniti in *Asymptote* sono discussi più avanti.

4.1 Definire il sistema di coordinate

I comandi di disegno in *Asymptote* si basano sulla definizione di un opportuno sistema di coordinate. Ogni punto della "tela" (*canvas*) del disegno è inteso come una coppia (x, y) di coordinate. Ci sono diversi modi per scegliere un sistema di coordinate cartesiane nel piano. Bisogna scegliere il posizionamento dell'origine e la scala di ciascuno degli assi. Per default, l'unità di lunghezza in entrambe le direzioni x ed y è il cosiddetto *PostScript bigpoint* (bp), che corrisponde ad $1/72$ di pollice. Quindi, se non si modifica la scala del disegno, i punti $(0,0)$ e $(72,0)$ distano esattamente un pollice quando vengono disegnati in *Asymptote*.

Il disegno in unità *bigpoint* può risultare scomodo, ad esempio se si vogliono disegnare figure ed oggetti di dimensioni definite naturalmente in centimetri. La funzione `unitsize` può essere utilizzata per specificare l'unità di lunghezza lungo ciascun lato dell'immagine. Questa funzione richiede fino a 3 argomenti: l'immagine che si vuole scalare, (se questa non è specificata, significa che si vuole scalare la figura corrente), l'unità di lunghezza nella direzione x , l'unità di lunghezza in direzione y .

Se la funzione è invocata con un solo argomento numerico si avrà che per entrambe le direzioni si avrà la stessa unità di lunghezza. In tal modo il comando `unitsize(72)` dirà ad *Asymptote* che, da quel punto in poi, l'unità di lunghezza è di 1 pollice in entrambe le direzioni coordinate.

Asymptote offre la possibilità di operare con diverse unità predefinite: il punto (`pt`, pari a $1/72,27$ pollici), il pollice (`inch`), il centimetro (`cm`), il millimetro (`mm`). Un comando equivalente a quello precedente è dunque: `unitsize(1inch)`.

Un'altra funzione utile è `size` che specifica la larghezza e l'altezza finali del disegno. La stessa funzione può essere invocata con un solo argomento, ottenendo che sia la larghezza che l'altezza dell'immagine corrisponderanno a questa dimensione. Per esempio, il comando `size(3cm,3cm)` o semplicemente `size(3cm)` permette di scalare il disegno in modo che rientri di una casella di dimensioni $3\text{ cm} \times 3\text{ cm}$, a prescindere da quanto specificato da eventuali chiamate alla funzione `unitsize`.

Se si scrive:

```
unitsize(1cm)
// size(15mm,10mm)
draw((0,0)--(5,5));
```

si otterrà un'immagine di dimensioni $5\text{ cm} \times 5\text{ cm}$. Basta scomentare la riga con il comando che invoca la funzione `size` per ottenere la scalatura del disegno ed un'immagine finale di dimensioni $15\text{ mm} \times 10\text{ mm}$.

Se uno degli argomenti di `size` è 0 l'algoritmo di scalatura renderà uno due lati dell'immagine di lunghezza pari a quella specificata; ad esempio `size(0,3cm)` permette di scalare il disegno in modo che l'altezza sia pari a 3 cm, dimensionando proporzionalmente la dimensione corrispondente all'argomento nullo.

Nella figura 2 è riportato un esempio di codice sorgente che genera il disegno di un sistema di assi cartesiani. Le funzioni utilizzate a configurare le dimensioni degli assi, le etichette, le tacche principali e quelle secondarie, eccetera, sono `xaxis` ed `yaxis`. Un esempio delle funzionalità introdotte dal modulo `geometry.asy` è mostrato nella figura 4. Si esamini l'uso dei nuovi tipi di dato, `point` e `vector`, introdotti da questo modulo. Il codice usato per generare il disegno mostra anche esempi di conversione implicita di tipi di variabili (*casting*), da `point` a `vector` oppure da `point` a `pair`.

4.2 Variabili e tipi di dati

L'interprete di *Asymptote* attua un meccanismo di *parsing* del codice sorgente ispirato a quello dei compilatori C++. Ciò consente agli utenti di utilizzare diversi tipi di dati predefiniti e di crearne di nuovi (secondo il paradigma della programmazione a oggetti).

```
import graph;
unitsize(1cm);
// Una curva
path curva=(.5,2){dir(-30)}..{0}(2.5,-2)..{dir(30)}(5,3);
draw(curva,red);
dot(curva,black);
// Configurazione degli assi
xaxis(Label("$x$",position=EndPoint, align=SE),
xmin=-1.5,
Ticks(scale(.7)*Label(align=W),
NoZero,
endlabel=false,
Step=1,step=.25,
end=false,
Size=1mm, size=.5mm,
pTick=black,ptick=gray),
Arrow);
yaxis(Label("$y$",position=EndPoint, align=NE),
ymin=-3.5,ymax=3.5,
Ticks(scale(.7)*Label(),
NoZero,
Step=1,step=.25,
Size=1mm, size=.5mm,
pTick=black,ptick=gray),
Arrow);
```

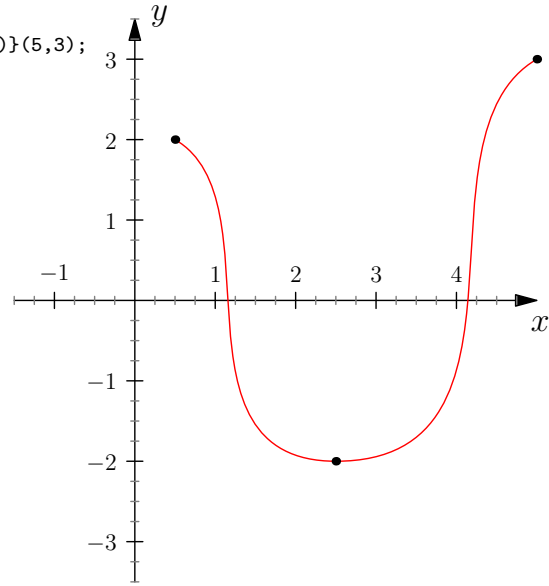


FIGURA 2: Disegnare un riferimento cartesiano e configurare lo stile degli assi.

4.2.1 Variabili numeriche e coordinate

Introduciamo con un esempio la dichiarazione e l'assegnamento di variabili:

```
unitsize(1mm);
pair A, B, C, D;
A = (0, 0); B = (50,50);
int yc = 50;
real xd = 50.0;
C = (0,yc); D = (xd, 0);
draw (A--B);
draw (C--D);
```

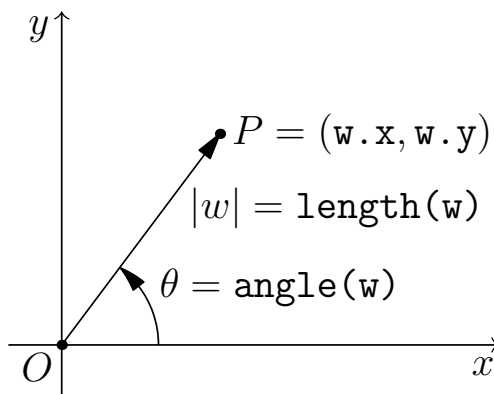
In *Asymptote* l'assegnamento di valore ad una variabile può anche avvenire contestualmente alla sua dichiarazione come nel frammento di codice seguente:

```
real d=50;
pair A=(0,0), B=(d,d);
```

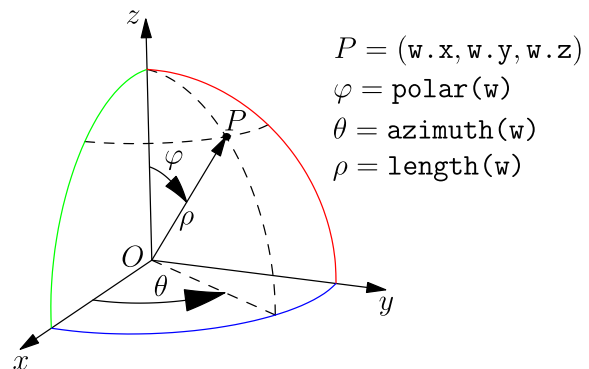
`draw(A--B);`

cioè il programmatore può inizializzare esplicitamente una variabile.

Oltre ai tipi numerici `int` e `real`, comuni ad altri linguaggi di alto livello, nei frammenti di codice precedenti si riconosce la dichiarazione di variabili di tipo `pair`. Una variabile `pair` non è altro che una coppia di coordinate nel sistema di riferimento specificato (nel caso dell'esempio, il sistema di coordinate di default). In *Asymptote* una `pair` è trattata come numero complesso, cioè una coppia ordinata (x, y) di elementi reali. Definita una variabile w appartenente a questo tipo, corrispondente al numero complesso w , la sua parte reale e la sua parte immaginaria sono date da $w.x$ e $w.y$. Le istruzioni `length(w)` e `degree(w)` ritornano valori reali corrispondenti, rispettivamente, al modulo $|w|$ ed all'angolo di fase (in gradi) nell'intervallo



(a) Un punto P del piano complesso.



(b) Un punto P dello spazio tridimensionale.

FIGURA 3: Interpretazione di una variabile di tipo `pair` e di tipo `triple`.

```
import geometry;
size(7cm,0);
usepackage("amsmath");
show("$0$",
    "\boldsymbol{i}$",
    "\boldsymbol{j}$", currentcoordsys);
vector u=(1.5,0.5), v=rotate(90)*u;
point O1=(5,3);
coordsys R=cartesiansystem(O=O1,i=u,j=v);
show("$0_1$",
    "\boldsymbol{u}$",
    "\boldsymbol{v}$", R, xpen=invisible);
point M=(1,0.5);
dot("$M$", M);
point P=point(R, (1,1));
dot("$P$", P);
vector w=P;
show("$\boldsymbol{w}$", w);
point Q=M+w;
dot("$Q=M+\boldsymbol{w}$", align=Relative(N),Q);
draw((0,0)--M,Arrow);
draw("$\boldsymbol{w}$",M--Q,RightSide,EndArrow);
```

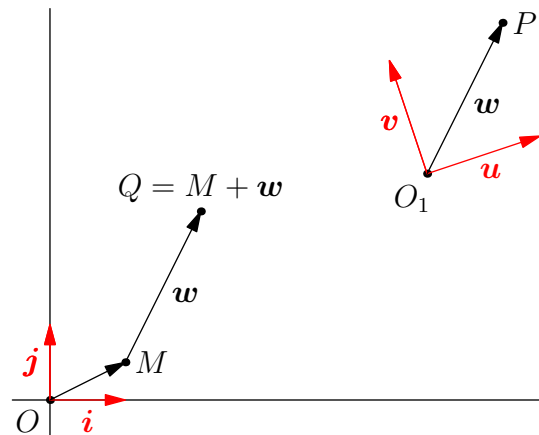


FIGURA 4: Introdurre un nuovo riferimento, definire un vettore e sommare un punto ad un vettore con le funzioni definite dal modulo `geometry.asy`.

$[0, 360[$ (la funzione ritorna un *warning* se w coincide con l'origine). Una funzione analoga a `degree` è `angle` che ritorna un angolo in radianti nell'intervallo $[0, 2\pi[$. Questi concetti sono resi in forma grafica nella figura 3a.

Il tipo `triple` è analogo a `pair` e serve per definire terne di coordinate (x, y, z) . La figura 3b rappresenta un'interpretazione grafica delle funzioni `length(triple)`, `polar(triple)` e `azimuth(triple)`. Esse estraggono le coordinate polari di un punto P rappresentato dalla variabile w di tipo `triple`. Funzioni che ritornano valori in gradi analoghe a `polar` e `azimuth` sono, rispettivamente, `colatitude` e `latitude`. Si rimanda al manuale utente per i dettagli sui costruttori e sulle funzioni di manipolazione di variabili `triple`.

I tipi di dati predefiniti più comunemente usati in *Asymptote* sono riportati nella tabella 1. Nel prossimo paragrafo, che introduce le funzioni di disegno più importanti, verranno presentati alcuni esempi che fanno uso dei tipi `path` e `guide`, necessari per la gestione di tracciati.

4.2.2 Tipi definiti da utente

A titolo di esempio di programmazione a oggetti in *Asymptote*, si riporta un estratto dal file `geometry.asy` che mostra la definizione del tipo `point`. Si tratta di un tipo che non compare tra i tipi predefiniti del linguaggio base. Esso viene definito nell'ambito del modulo `geometry` che attualmente fa parte della distribuzione ufficiale di *Asymptote*. Il tipo `point` è usabile dopo aver dato l'istruzione `import geometry`. La definizione si serve della parola chiave `struct` (analogo alla parola chiave `class` del C++) e fornisce un tipo di dato che contiene informazioni su una `pair` e sul riferimento cartesiano in cui vengono assegnate le

coordinate. Il codice che lo definisce assomiglia al seguente:

```
// Estratto da geometry.asy
struct point
{
    // variabili membro
    coordsys coordsys; // pubblica
    restricted pair coordinates; //
    privata
    restricted real x, y; // private
    // funzione membro di
    // inizializzazione
    void init(coordsys R,
        pair coordinates, real mass)
    {
        this.coordsys=R;
        this.coordinates=coordinates;
        this.x=coordinates.x;
        this.y=coordinates.y;
        this.m=mass;
    }
    /* ...
    coordsys, defaultcoordsys,
    e currentcoordsys
    definite nello stesso modulo */
}
```

Il lettore che ha familiarità con la programmazione a oggetti riconoscerà da questo frammento di codice l'utilizzo dei concetti di *funzioni membro* e di *variabili membro* (pubbliche e private) e della parola chiave `this` (come in C++ o in Java).

Sempre nel file `geometry.asy` è definito uno dei costruttori del tipo `point`:

```
// Un costruttore
point point(coordsys R, pair p,
```

TABELLA 1: Tipi di dati predefiniti del linguaggio *Asymptote*.

<i>Tipo</i>	<i>Descrizione</i>	<i>Esempi</i>
bool	tipo logico, vero o falso	true, false, 1>2 (false)
string	stringa di caratteri, delimitati da doppi apici	"Ciao", "Questa e' una stringa"
int	un numero intero	-2, -1, 0, 1, 2, 3
real	un numero reale	1.0, 5.48, 12345.6789
pair	una coppia di numeri reali o interi, delimitati da parentesi tonde	(0,2), (-30,42.5)
triple	una terna di numeri reali o interi, delimitati da parentesi tonde	(1,2,3), (-2.5,5,4)
path	una <i>spline</i> cubica non modificabile	(0,0)--(5,0), (0,1)..(1,0)..(1,1)--cycle
guide	una <i>spline</i> cubica, simile a un path ma aggiustabile se raccordata ad altre curve dello stesso tipo	(0,0)--(5,0), (0,1)..(1,0)..(1,1)--cycle, (0,1)--(1,0)--(1,1)--cycle
picture	una tela (<i>canvas</i>) su cui disegnare nel sistema di coordinate corrente	currentpicture, qualsiasi insieme di oggetti
frame	una tela (<i>canvas</i>) su cui disegnare in coordinate PostScript	currentpicture, qualsiasi insieme di oggetti
transform	una trasformazione del piano, applicabile a qualsiasi oggetto disegnabile usando l'operatore *	scale(2), xscale(2), rotate(30), rotate(15,(-1,3)), shift(2,4)
void	usato per dichiarare funzioni che non hanno argomenti	

```

        real m=1) // val. di
    default
{
    point op;
    op.init(R, p, m);
    return op;
}

```

In base a queste definizioni una variabile di questo tipo può essere inizializzata come segue:

```

// CS, rif. definito precedentemente
point pt=point(CS, (2,5));

```

ed, eventualmente, si può fare un controllo sul sistema di coordinate al quale si riferiscono i valori delle variabili private *x* e *y*:

```

if (pt.coordsys == defaultcoordsys){
    // riferimento predefinito
    ...
} else {
    // abbiamo un nuovo riferimento
    ...
}

```

Per un ulteriore esempio d'uso del tipo **point** si veda anche la definizione del sistema cartesiano **R** e del punto **P** nel codice che genera il disegno della figura 4. In quell'esempio il punto **M** è un **point** definito nel sistema di riferimento di default (che è quello corrente) come:

```

point M=(1,0.5); // cast da pair a
point

```

mentre **P** è definito assegnando coordinate nel nuovo riferimento **R**:

```

point P=point(R, (1,1)); // costruttore

```

4.2.3 Array di variabili

Aggiungendo **[]** nella dichiarazione di variabili predefinite o in variabili di tipo definito da utente si possono costruire degli array (matrici di dati di una o più dimensioni). L'istruzione:

```

real[] A={-0.5,1.5,2};

```

dichiara ed inizializza l'array **A**. Si può accedere all'elemento *i*-mo della matrice unidimensionale **A** con l'operazione **A[i-1]** (come in C, il primo elemento dell'array **A** è l'elemento **A[0]**).

Per default, i tentativi di accesso o di assegnamento che utilizzano un indice negativo generano un errore. Anche l'accesso ad un elemento di un array con un indice che va oltre la dimensione della matrice genera un errore. Tuttavia, l'assegnazione che fa uso di un indice maggiore del riempimento corrente dell'array provoca il ridimensionamento della matrice in modo da allocare il nuovo elemento. L'istruzione **A[6] = -1.0;** provoca il ridimensionamento di **A**, che avrà 6 elementi di cui il quarto e quinto nulli.

Gli array di *Asymptote* hanno caratteristiche che li rendono simili ai contenitori **vector** della libreria standard del C++. Si veda il manuale utente per le funzioni di manipolazione degli array e per i dettagli sulle matrici a più dimensioni.

```
int b=6; // b>3
for(int a=0; a<b-2; ++a) {
    draw ((a,a)--(a,a+1)--(a+1,a+1));
    label(format("$u_{i}$",a),(a+.5,a+1),N);
}
for(int a=b-2; a<b; ++a) {
    draw ((a,a)--(a,a+1)--(a+1,a+1),dashed);
}
for(int a=0; a<2; ++a) {
    draw ((b+a,b+a)--(b+a,b+a+1)--(b+a+1,b+a+1));
    if (a != 0)
        label(format("$u_{n+i}$",a),
            (b+a+.5,b+a+1),N);
    else label("$u_n", (b+a+.5,b+a+1),N);
}
```

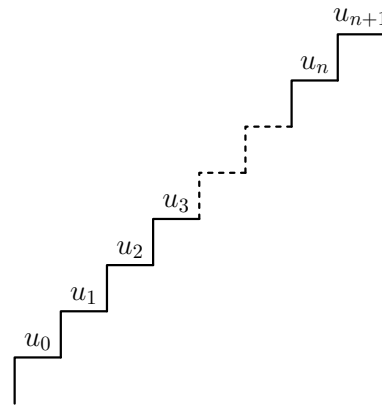


FIGURA 5: Esempio di disegno ottenuto con un ciclo iterativo.

4.3 Costrutti

Ecco un breve esempio che dimostra la somiglianza delle strutture di controllo del linguaggio *Asymptote* con quelle del C, del C++ e di Java:

```
// Una variabile reale
real x=1.0;
// Un test condizionale
if (x == 1.0) {
    // stampa un messaggio
    // sulla console
    write("x uguale a 1.0");
} else {
    write("x diverso da 1.0");
}
// Un loop di 10 iterazioni
for(int i=0; i<10; ++i){
    write(i);
}
```

Asymptote mette a disposizione anche i costrutti **while do**, **break** e **continue**, proprio come in C++. Inoltre, come in Java, il linguaggio permette di scrivere elegantemente dei cicli su tutti gli elementi di un *array*:

```
// Itera sugli elementi di
// un array
int[] myarray={1,1,2,3,5};
for(int k : myarray) {
    write(k);
}
```

Un esempio più articolato di disegno basato su un ciclo iterativo è mostrato nella figura 5. Esso utilizza la funzione `format` per formattare stringhe destinate alla composizione tipografica.

4.4 Funzioni di disegno

Tutte le capacità grafiche di *Asymptote* si basano su quattro funzioni primitive (comandi). I tre comandi PostScript **draw**, **fill** e **clip** servono ad aggiungere oggetti ad un disegno. Il quarto comando **label** viene utilizzato per aggiungere annotazioni testuali (e immagini EPS esterne). Una

quinta funzione molto utile è **shipout** e si usa tipicamente come ultimo comando, dopo aver creato il disegno, per aggiustare alcune impostazioni di default sul formato finale. Anche se i comandi di disegno accettano molte opzioni, queste ultime hanno tutte delle assegnazioni di default ragionevoli e richiedono un controllo esplicito solo in caso di particolari esigenze di disegno.

I quattro comandi di disegno fondamentali di *Asymptote* sono ampiamente documentati nel manuale d'uso. Nel prossimo paragrafo vedremo qualche dettaglio sull'uso della funzione `draw`.

4.4.1 Punti, segmenti e stili

Una delle funzioni predefinite più importanti del linguaggio è **draw**. Dall'esame dei frammenti di codice riportati fino a questo punto si comprende che il suo utilizzo è molto intuitivo.

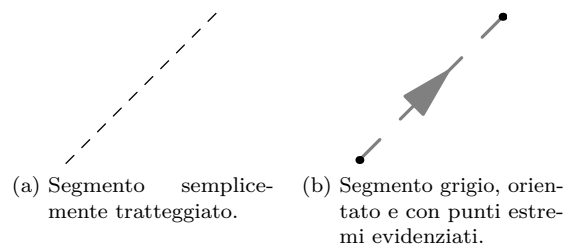


FIGURA 6: Disegnare un segmento in modi diversi.

Consideriamo un semplice esempio: disegnare un segmento che va dal punto A al punto B del piano. In *Asymptote* si possono definire due variabili di tipo `pair` che memorizzano le coordinate degli estremi del segmento, ad esempio A e B , utilizzandole direttamente nella chiamata alla funzione `draw`. Nel disegno del segmento AB l'istruzione fondamentale è `draw(A--B)`. In questo comando viene passato a `draw` un solo argomento, cioè il risultato dell'operazione $A--B$.

L'operatore `--`, così come accade in `METAPOST` oppure in `pgf`, determina l'unione di due punti del piano attraverso una linea. In *Asymptote* questa

operazione produce un tracciato che può essere memorizzato nella sua interezza come tipo di dato. Un possibile tipo predefinito che memorizza il tracciato `A--B` è il tipo `path`. Disporre di una variabile che memorizza un tracciato consente di riusare quest'oggetto in tutte le funzioni che concettualmente ne fanno uso, che siano esse predefinite o definite dall'utente.

Volendo gestire il segmento come un `path`, si può scrivere:

```
pair A=(0,0), B=(50,50);
path p=A--B;
draw(p,dashed);
```

ed ottenere il disegno della figura 6a. Alla funzione `draw` viene passato un secondo argomento (`dashed`) che serve a specificare lo stile della linea.

Il segmento della figura 6b è ottenuto con le istruzioni:

```
draw(p,                // tracciato
     gray+dashed+1.2pt, // pennino
     MidArrow);         // freccia
dot(p);
```

Il secondo argomento mostra come sia intuitivo definire lo stile della linea utilizzando l'operatore `+` per sommare logicamente una lista di attributi. In questo caso si è specificato: il colore grigio (`gray`), il tratteggio (`dashed`) e lo spessore (`1.2pt`).

Le diverse chiamate a `draw` mettono in evidenza la possibilità offerta da *Asymptote* di definire funzioni invocabili con un numero variabile di argomenti. Ciò è possibile perché molte funzioni in *Asymptote* sono definite assegnando agli argomenti dei valori di default (ad esempio gli argomenti che controllano lo stile dei tracciati o il posizionamento delle etichette). Gli argomenti hanno spesso dei nomi predefiniti che consentono il loro passaggio *per nome*, oltre che per posizione, semplificando il lavoro del programmatore. Un esempio evidente è dato dalla chiamata:

```
coordsys R=cartesiansystem(
    0=01,i=u,j=v);
```

effettuata nel codice sorgente riportato nella figura 4. Le variabili `01`, `u` e `v` sono state precedentemente definite e vengono passate alla funzione `cartesiansystem` come gli argomenti di nome `0`, `i` e `j`, rispettivamente.

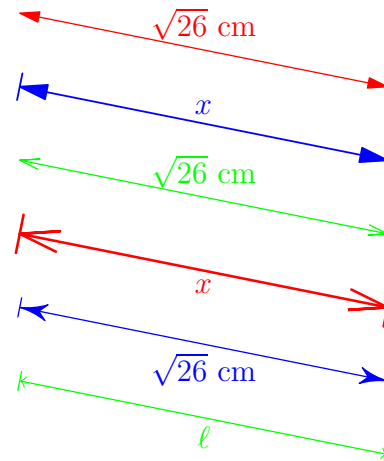
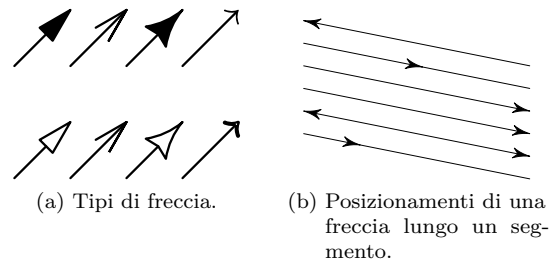
La funzione `dot` serve a disegnare punti. Nell'esempio che genera la figura 6b essa viene invocata come `dot(p)`, passandole come argomento un `path` ed ottenendo il disegno di tutti i punti (nodi) che definiscono il percorso memorizzato nella variabile `p`. Questa funzione può essere anche chiamata, come è naturale che sia, passando una `pair`, ad esempio con l'istruzione `dot((20,0))`, per disegnare un semplice puntino. In realtà, come vedremo più avanti, grazie alla rappresentazione generica di un tracciato in *Asymptote*, un `path` definito da un solo nodo (un percorso degenerare, di lunghezza nulla) si identifica con quel punto stesso. È lecito

cioè dire: `path p=(20,0)`. Pertanto, il disegno di un singolo puntino corrisponde all'applicazione di `dot` ad un `path` degenerare.

L'ultimo esempio mostra il passaggio della specifica `MidArrow` come terzo argomento della funzione `draw`. Lo stesso risultato si ottiene con l'istruzione

```
draw(p,gray+dashed+1.2pt,
     Arrow(Relative(0.5)));
```

che utilizza le funzioni `Arrow` e `Relative`. Quest'ultima serve a controllare la posizione lungo il tracciato in cui si vuole disegnare la freccia (in questo caso nella posizione intermedia).



(c) Dimensioni di un segmento.

FIGURA 7: Diversi stili di freccia.

Esempi di diversi stili di freccia sono riportati nella figura 7. Il seguente frammento di codice realizza il disegno della figura 7a:

```
// 4 tipi di freccia
draw((0,0)--(1,1),2bp+black,
     Arrow);
draw((1,0)--(2,1),2bp+black,
     Arrow(SimpleHead));
draw((2,0)--(3,1),2bp+black,
     Arrow(HookHead));
draw((3,0)--(4,1),2bp+black,
     Arrow(TeXHead));
// Varianti (option NoFill)
draw((0,-2)--(1,-1),2bp+black,
     Arrow(NoFill));
draw((1,-2)--(2,-1),2bp+black,
     Arrow(SimpleHead,NoFill));
draw((2,-2)--(3,-1),2bp+black,
```

```

    Arrow(HookHead,NoFill));
draw((3,-2)--(4,-1),2bp+black,
    Arrow(TeXHead,NoFill));

```

Il disegno della figura 7b si ottiene con la seguente sequenza di istruzioni:

```

DefaultHead=HookHead; // N.B.
draw((0,1)--(5,0),BeginArrow());
draw((0,.5)--(5,-.5),MidArrow());
draw((0,0)--(5,-1),EndArrow());
draw((0,-.5)--(5,-1.5),Arrow());
draw((0,-1)--(5,-2),Arrows());
draw((0,-1.5)--(5,-2.5),
    Arrow(Relative(.25)));

```

Il disegno della figura 7c si ottiene come segue:

```

unitsize(1cm);
draw("$\sqrt{26}$ cm",
    (0,5)--(5,4),
    N,red,Arrows);
draw("$x$",
    (0,4)--(5,3),
    N,.7bp+blue,Arrows,Bars);
draw("$\sqrt{26}$ cm",
    (0,3)--(5,2),
    N,green,Arrows(SimpleHead));
draw("$x$",
    (0,2)--(5,1),
    S,1bp+red,Arrows(SimpleHead),
    Bars);
draw("$\sqrt{26}$ cm",
    (0,1)--(5,0),
    S,blue,Arrows(HookHead),
    Bars);
draw("$\ell$",
    (0,0)--(5,-1),
    S,green,Arrows(TeXHead), Bars);

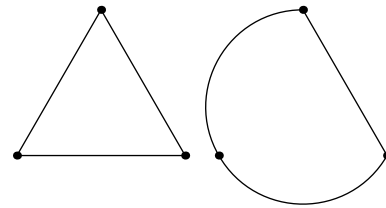
```

4.4.2 Tracciati

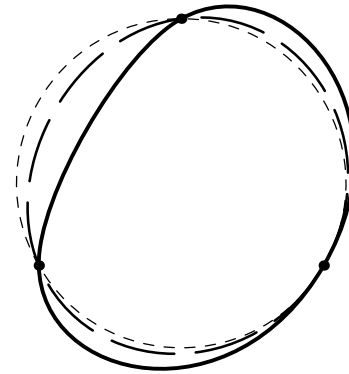
Un percorso di tipo **path** (o **guide**) è rappresentato internamente in *Asymptote* come una curva *spline* cubica a tratti di parametro t , con t che varia tra 0 ed n (numero di *nodi*). Questa è una rappresentazione abbastanza generale da permettere di creare e manipolare un gran numero di tracciati diversi.

Un caso particolarissimo di tracciato è quello di lunghezza nulla, costituito da un solo nodo (punto). Come già detto al paragrafo precedente, l'assegnazione **path** $p=(0,0)$ è perfettamente lecita. Inoltre, due punti (o percorsi di lunghezza nulla) possono essere semplicemente uniti collegandoli con una linea retta. Pertanto, detto anche q un percorso degenero definito come **path** $q=(1,1)$, il semplice tracciato dato dal segmento che unisce i due punti è realizzato dall'operazione: $p--q$.

L'operatore di doppio trattino $--$ unisce, in generale, due tracciati con una linea retta. È ciò che accade nel disegno del triangolo della figura 8a ottenuta con il codice seguente:



(a) Due tracciati chiusi definiti con gli stessi nodi.



(b) Alterazione della tensione della curva nel tratto tra i primi due nodi.

FIGURA 8: Tracciati ed operazioni su di essi.

```

pair p1=(0,0), p2=(1/2,sqrt(3)/2),
    p3=(1,0);
path t1, t2;
t1=p1--p2--p3--cycle;
t2=p1..p2..p3..cycle;
t2=shift(1.2*right)*t2;
draw(t1); dot(t1);
draw(t2); dot(t2);

```

Si noti come sia naturale in *Asymptote* utilizzare funzioni matematiche per la definizione di punti particolari come $p2$. La parola chiave **cycle** è usata per chiudere un tracciato unendo l'ultimo nodo con il primo.

Il tracciato $t2$ dell'ultimo listato mostra un esempio di collegamento tra punti tramite curve di Bezier ottenuto con l'operatore di doppio punto ($p..q$). Si rimanda al manuale di *Asymptote* ed alla letteratura ivi citata per un approfondimento sugli algoritmi di disegno dei tracciati curvi.

La figura 8b mostra come sia possibile modificare la tensione della curva *spline* che unisce tre punti. Il disegno è ottenuto con il codice seguente:

```

path t1, t2, t3;
t1=p1..p2..p3..cycle;
t2=p1..tension 1.1 ..p2..p3..cycle;
t3=p1..tension 1.5 ..p2..p3..cycle;
draw(t1,dashed); dot(t1,black+4bp);
draw(t2,linetype("24 8")+1bp);
draw(t3,black+1.5bp);

```

Ogni tratto costruito con l'operatore $..$ ha un valore di tensione predefinito pari ad 1. Con la parola chiave **tension** si può controllare questo

parametro che accetta valori maggiori o uguali di 0,75. Nell'esempio precedente la curva iniziale viene modificata variando semplicemente la tensione del tratto che unisce i primi due nodi, `p1` e `p2`. Più è alto il valore della tensione e più quel tratto di tracciato si avvicina ad una linea retta.

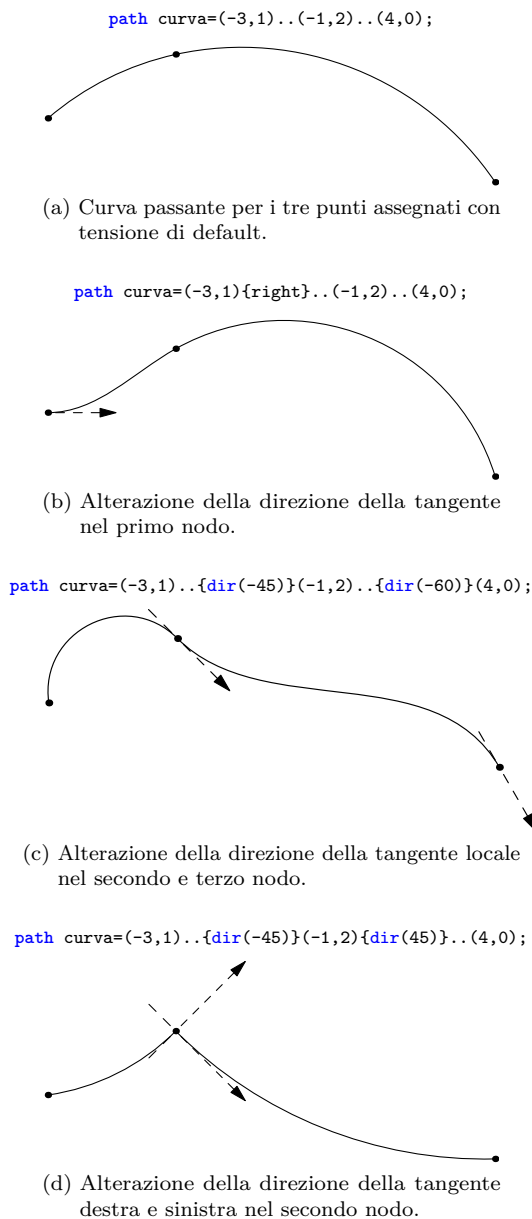


FIGURA 9: Controllo delle tangenti ad una curva nei nodi.

La figura 9 mostra esempi di controllo delle tangenti ad un tracciato in corrispondenza dei nodi. Nei disegni viene riportato il comando che definisce la variabile `curva`. Si osservi l'uso della funzione `dir` per la specifica della pendenza in gradi della tangente.

La differenza tra i tipi di tracciato `path` e `guide` si comprende quando ci si pone il problema di unire due curve definite inizialmente in maniera separata. Si consideri il disegno della figura 10a che rappresenta due linee spezzate. Esse sono generate dal codice seguente:

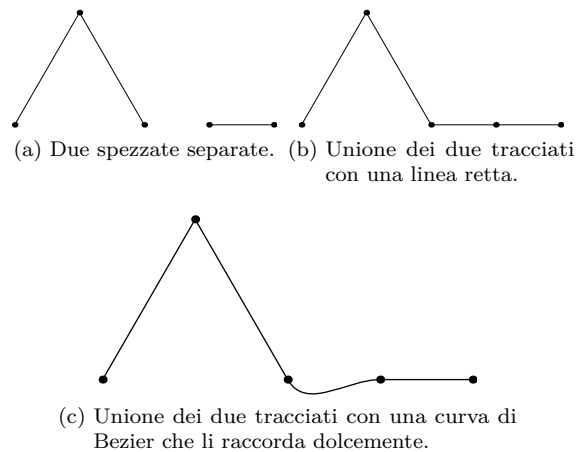


FIGURA 10: Tracciati ed operazioni su di essi.

```
path t1, t2;
t1=(0,0)--(1/2,sqrt(3)/2)--(1,0);
t2=(1.5,0)--(2,0);
draw(t1); dot(t1);
draw(t2); dot(t2);
```

La semplice unione dei due tracciati `t1` e `t2`, entrambi di tipo `path`, con un segmento è ottenuta con l'operazione `t1--t2`. Il codice seguente genera la figura 10b:

```
path t3=t1--t2;
draw(t3); dot(t3);
```

Il raccordo con una curva di Bezier che sia tangente ad entrambi i tracciati si ottiene con l'operazione `t1..t2`. Il codice seguente genera la figura 10c:

```
path t4=t1..t2;
draw(t4); dot(t4);
```

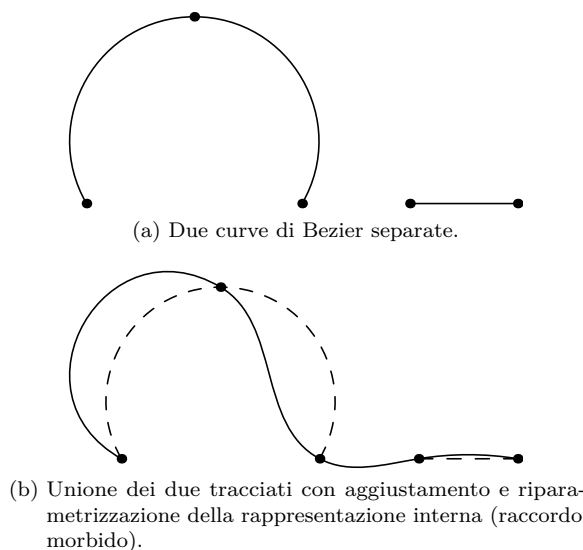


FIGURA 11: Tracciati aggiustabili (`guide`).

Nelle precedenti operazioni di unione di due tracciati le curve iniziali si comportano come se fossero rigide. Se queste fossero definite come `guide`, a

seguito di operazioni di unione esse sarebbero soggette ad una riparametrizzazione con conseguente modifica dell'aspetto.

Un esempio di guide è rappresentato dalla figura 11. I due tracciati della figura 11a sono definiti come segue:

```
guide t1, t2;
t1=(0,0)..(1/2,sqrt(3)/2)..(1,0);
t2=(1.5,0)..(2,0);
```

Il raccordo delle due guide con una curva di Bezier è effettuato ancora una volta dall'operazione di concatenazione `t1..t2`. Il risultato è quello della figura 11b che è generato dal codice

```
path t3=t1..t2; // cast guide<-path
draw(t3); dot(t3);
```

L'esempio che segue dimostra l'uso della funzione predefinita `intersectionpoint`. Essa restituisce in generale un array di `pair` corrispondenti ai punti di intersezione tra due tracciati. L'esempio proposto qui è molto semplice, ed è il seguente:

```
unitsize(1cm); size(6cm,0);
usepackage("siunitx",
"decimalsymbol=comma");
path segm1=(-1,1)--(5,2),
      segm2=(0,3)--(2,-0.5);
draw(segm1^segm2,blue);
pair pX=intersectionpoint(segm1,segm2)
;
dot(pX,4bp+red);
string sX= // formatta e concatena
format("$\SI{%.3f}{\centi\meter}",
pX.x)+
format("\SI{%.3f}{\centi\meter}}$",
pX.y);
label(sX,pX,0.9S+2E);
dot((0,0));
label("$ (0,0) $", (0,0),N);
```

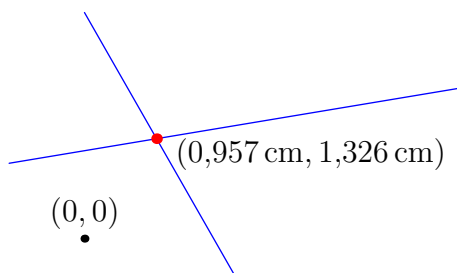


FIGURA 12: Esempio di intersezione tra due tracciati ed estrazione delle coordinate.

Il risultato è il disegno della figura 12 in cui si è ricercata l'intersezione tra due segmenti. L'esempio mostra anche un'interessante applicazione della funzione `format`, utilizzata per la composizione dell'annotazione che mostra le coordinate, non note *a priori*, del punto di intersezione `pX`.

4.4.3 Trasformazioni

Molte delle funzioni predefinite di *Asymptote* fanno ampio uso di trasformazioni affini (mutuate dalle analoghe funzioni di trasformazione del linguaggio PostScript).

Le trasformazioni possono essere applicate a variabili di tipo `pair`, `guide`, `path`, `pen`, `string`, `frame` e `picture`. Una trasformazione è applicata sotto forma di moltiplicazione a sinistra, attraverso l'operatore `*`. Le trasformazioni possono essere composte l'una con l'altra e memorizzate in variabili di tipo `transform`. L'inizializzazione implica di tali variabili assegna ad esse la trasformazione predefinita `identity` (di ovvio significato). Una variabile `T` di tipo `transform` può trasformare, ad esempio, un tracciato `crv` (`path` o `guide`) attraverso l'operazione `T*crv`.

Il codice che segue è quello che realizza il disegno della figura 13:

```
size(4cm,0);
import markers;
// Due punti, A e B
pair pA=(0,0),pB=(4,2);
dot("$A$",pA,SW); dot("$B$",pB,SE);
// B1 per rotazione di B intorno ad A
// di un angolo di 60 gradi
pair pB1=rotate(60,pA)*pB;
// disegno
dot("$B'$",B1,N,red);
draw(pB--pA--pB1,green);
// marcatura dell'angolo
markangle(scale(1.5)*"$+60^\circ$",
radius=50,
pB,pA,pB1,ArcArrow(2mm),
1mm+red);
```

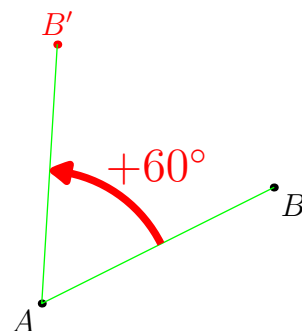


FIGURA 13: Esempio di trasformazione di rotazione applicata ad un punto e di scalatura di una casella di testo.

Per la rotazione di 60° del punto B intorno al punto A viene usata la funzione `rotate`. L'ingrandimento dell'annotazione `" $+60^\circ$ "` è prodotto applicando la funzione `scale` direttamente alla stringa passata alla funzione `markangle`.

Una utile funzione di trasformazione è quella che permette di proiettare un dato punto su una retta. Il codice seguente genera il disegno della figura 14:

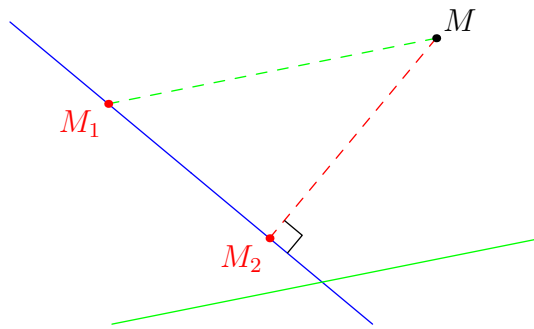


FIGURA 14: Proiettare un punto ortogonalmente ad una retta e secondo una direzione prefissata.

```
import geometry;
size(7cm,0);
// i punti assegnati
point A=(2,1),B=(-1,3.5), // direz. r
      C=(3,1.5),D=(.5,1), // direz. d
      M=(3,4); // punto da proiettare
// linee
line r=line(A,B), d=line(C,D);
// risultati delle proiezioni su r
pair M1=projection(r,d)*M, // lungo d
     M2=projection(r)*M; // ortogonale
// Costruzioni
draw(r,blue); draw(d,green);
draw(segment(M,M1),dashed+green);
draw(segment(M,M2),dashed+red);
perpendicularmark(
    line(M,M2),line(A,B),quarter=2);
dot("$M$",M,NE);
dot("$M_1$",M1,SW,red);
dot("$M_2$",M2,SW,red);
addMargins(1cm,1cm);
```

In questo esempio sono stati usati tipi e funzioni messi a disposizione dal modulo `geometry`: i tipi `line` e `segment` e le funzioni `projection` e `perpendicularmark`.

4.4.4 Pennini e tipi di tratto

In Asymptote, è possibile fornire un contesto per i quattro comandi di disegno di base attraverso un tipo predefinito detto `pen` (pennino).

Una variabile `pen` può essere utilizzata per specificare i seguenti attributi di disegno: il colore, il tipo di linea, lo spessore della linea, l'estremità della linea, il tipo di linea di unione, regole di riempimento di regioni piane. Inoltre è possibile agire con una `pen` su alcuni aspetti legati alla composizione delle etichette testuali come: l'allineamento del testo, il font e la sua dimensione.

Il pennino utilizzato dalla routine di disegno, quando non vi è il controllo esplicito da parte dell'utente, è chiamato `currentpen`. L'inizializzazione implicita di una variabile `pen` assegna il valore `defaultpen`.

Gli attributi di un pennino possono essere assegnati cumulandoli attraverso l'uso dell'operatore non associativo binario `+`. Così, si può ottenere un'impostazione del pennino corrispondente ad un tratteggio rosso di spessore pari a 2pt specificando il valore `dashed+red+2pt`.

L'operatore binario `*` può essere usato per scalare un colore con un numero reale, fino alla saturazione di una o più componenti. Per le funzionalità di manipolazione del colore si vedano, tra le altre, le funzioni `gray`, `rgb`, `cmk` nel manuale utente di *Asymptote*.

Una funzione utile ad impostare il tipo di tratteggio è `linetype(string s, real offset=0)`, che ritorna un valore di tipo `pen`. Il primo argomento è il più importante e specifica tramite una stringa il tipo di tratteggio. Ad esempio, il codice seguente mostra come creare uno stile personalizzato di "tratto e punto":

```
// Tratteggio:
// 16 continuo, 8 vuoto,
// 1 continuo, 8 vuoto
pen mydashdotted=linestyle(
    "16 8 1 8");
draw((0,0)--(5,5),mydashdotted);
```

Lo spessore di un tracciato è impostabile con una chiamata alla funzione `linewidth(real w)`.

La dimensione del font è specificata in punti con la funzione `fontsize(real size, real lineskip=1.2*size)`, che ritorna un `pen`. La dimensione del font di default, 12pt, può essere modificata con la funzione `defaultpen(pen p)`. Per impostare una dimensione non standard dei caratteri è richiesto il comando `import fontsize`; all'inizio del file sorgente (questo richiede anche l'uso del pacchetto LATEX `fix-cm` che è supportato dalle recenti distribuzioni di LATEX). Si rimanda al manuale di *Asymptote* per un riferimento su tutte le funzioni di gestione dei caratteri e della composizione del testo.

Il linguaggio offre una comoda interfaccia con i font PostScript standard attraverso le funzioni `AvantGarde`, `Bookman`, `Courier`, `Helvetica`, `NewCenturySchoolBook`, `Palatino`, `TimesRoman`, `ZapfChancery`, `Symbol`, `ZapfDingbats`. Un esempio di composizione che fa uso del font Helvetica è riportato nella figura 15.

5 Integrazione di codice *Asymptote* in sorgenti LATEX

Per poter usare del codice *Asymptote* in un documento LATEX esiste l'ambiente `asy` messo a disposizione dal pacchetto di estensione `asymptote`. Quest'ultimo è implementato nel file `asymptote.sty` e viene distribuito ufficialmente insieme al resto del sistema grafico.

Per usufruire di questa funzionalità, tutto quello che si deve fare è includere il coman-

```
unitsize(1cm,1cm);
real i=0,j=0;
pen p1=Helvetica(series="m",shape="n");
pen p2=Helvetica(series="m",shape="it");
pen p3=Helvetica(series="b",shape="n");
// funzione di stampa
void testpolice(string phrase, pen police, real d=0) {
    label(phrase,(i,j),police);
    label("ABCDEFGHIJK",(i+3,j-d),police);
    label("abcdefghijk",(i+3,j-.5-d),police);
    label("0123456789",(i+3,j-1-d),police);
    j-=2+d;
}
testpolice("Helvetica normal",p1,0.5); // stampa
testpolice("Helvetica Italic",p2,0.5);
testpolice("Helvetica bold",p3,0.5);
```

Helvetica normal

ABCDEFGHIJK
abcdefghijk
0123456789

Helvetica Italic

ABCDEFGHIJK
abcdefghijk
0123456789

Helvetica bold

ABCDEFGHIJK
abcdefghijk
0123456789

FIGURA 15: Gestione del font con il tipo `pen`.

```
import labelpath;
usepackage("guit");
unitsize(1cm);
string t="\Large\GuITmeeting[style=inline,year=2009]";
path c=(0,0)..(2,2)..{dir(10)}(7,0.5);
labelpath(t,c);
draw(c, red);
```



FIGURA 16: Disegnare un testo lungo un tracciato.

do `\usepackage{asymptote}` nel preambolo del documento L^AT_EX. Un esempio minimale è il seguente:

```
% preambolo
\usepackage%
    %[pdftex] se si usa pdflatex
    {graphicx}
\usepackage{asymptote}
% ...
% dopo \begin{document}
% ...
\begin{figure}
\centering
\begin{asy}
size(3cm);
draw(unitcircle);
\end{asy}
\caption{Sembra facile!}
\label{fig:asymptote:embedded}
\end{figure}
```

Si deve osservare che la dichiarazione di chiusura `\end{asy}` deve figurare su una linea a sé stante, senza spazi iniziali o finali.

Se il frammento di documento precedente appartiene ad un file di nome `doc.tex` la sequenza dei comandi che serve a creare il documento finale è la seguente:

```
latex doc
asy doc
latex doc
```

Se si preferisce usare PDFL^AT_EX basta specificare l'opzione `pdftex` per il pacchetto `graphicx` ed invocare il programma `pdflatex` al posto di `latex`.

Si rimanda alla guida TEIXEIRA (2007) per maggiori dettagli sull'uso di questo metodo di lavoro.

La figura 16 mostra un esempio dell'uso di `labelpath` per disegnare un testo lungo un tracciato.

6 Miscellanea

Il disegno nella figura 17 fa parte della galleria di immagini del sito ufficiale di *Asymptote* ed il sorgente è ottenibile all'indirizzo <http://asymptote.sourceforge.net/gallery/CAD1.asy>.

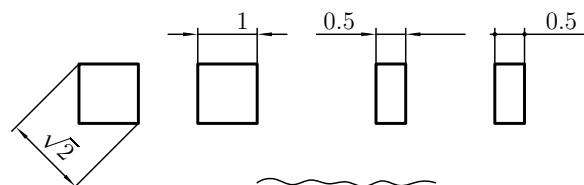


FIGURA 17: Esempio realizzato mediante il modulo di estensione denominato CAD.

Compilando il seguente programma si ottiene l'immagine riportata nella figura 18.

```
import graph3;
size(7.5cm,0);
texpreamble("\usepackage{guit}");
currentprojection=perspective
    (100,100,150);
```

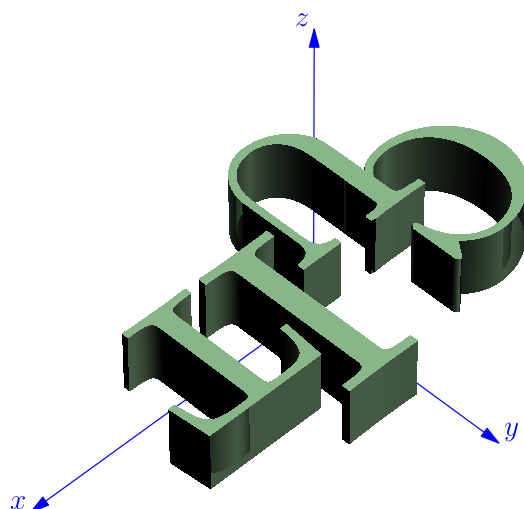


FIGURA 18: Esempio realizzato mediante il modulo di estensione denominato **graph3**.

```
draw(extrude("\GuIT{ }",3Z),palegreen);
limits(0,15X+10Y+15Z);
xaxis3(Label("$x$",1),blue,arrow=
    Arrow3);
yaxis3(Label("$y$",1),blue,arrow=
    Arrow3);
zaxis3(Label("$z$",1),blue,arrow=
    Arrow3);
```

7 Conclusioni

Questo articolo si è proposto di offrire una graduale introduzione ad *Asymptote*. Il sistema grafico è stato presentato con una sequenza di esempi, spesso semplificati, che si sono focalizzati sui diversi aspetti del linguaggio di programmazione. Si sono evidenziate le potenzialità e le possibilità grafiche offerte da questo software libero.

Per motivi di brevità sono diversi gli argomenti che si sono lasciati alla cura del lettore. Si pensi al riempimento di figure chiuse con *pattern* e tratteggi vari, alla gestione dei colori e degli algoritmi di *shading*, alle librerie per il disegno di grafici scientifici, alle potenti funzioni di gestione di scene tridimensionali. Tutto questo, ed altro ancora, può essere approfondito nei manuali citati ed è dimostrato efficacemente nelle gallerie di esempi disponibili in rete.

Riferimenti bibliografici

- BECCARI, C. (2007). «Graphics in L^AT_EX». *PracT_EX Journal*, (1). URL <http://www.tug.org/pracjourn/2007-1/beccari/>.
- BOWMAN, J. (2009). «Webpage of Asymptote: The vector graphics language». URL <http://asymptote.sourceforge.net/>.
- BOWMAN, J. C. e HAMMERLINDL, A. (2008). «Asymptote: A vector graphics language». *TUG-BOAT: The Communications of the T_EX Users Group*, **29** (2), pp. 288–294. URL <http://www.tug.org/TUGboat/Contents/contents29-2.html>.
- BOWMAN, J. C. e SHARDT, O. (2009). «Asymptote: Lifting T_EX to three dimensions». *TUGBOAT: The Communications of the T_EX Users Group*, **30** (1), pp. 58–63. URL <http://www.tug.org/TUGboat/Contents/contents30-1.html>.
- CASCHILI, M. (2006). «Semplici figure con l'ambiente picture». *ArsT_EXnica*, (1), pp. 20–28. URL <http://www.guit.sssup.it/arstexnica.php>.
- (2007). «Introduzione a PSTricks». *ArsT_EXnica*, (4), pp. 25–44. URL <http://www.guit.sssup.it/arstexnica.php>.
- DE MARCO, A. (2007). «Illustrazioni tridimensionali con Sketch/L^AT_EX/PSTricks/TikZ nella didattica della Dinamica del Volo». *ArsT_EXnica*, (4), pp. 51–68. URL <http://www.guit.sssup.it/arstexnica.php>.
- (2008). «Gestione avanzata delle figure in L^AT_EX». *ArsT_EXnica*, (6), pp. 10–27. URL <http://www.guit.sssup.it/arstexnica.php>.
- HAMMERLINDL, A. e BOWMAN, J. (2007). «Asymptote: The vector graphics language». URL <http://asymptote.sourceforge.net/intro.pdf>.
- HAMMERLINDL, A., BOWMAN, J. e PRINCE, T. (2009). «Asymptote: The vector graphics language». URL <http://asymptote.sourceforge.net/asymptote.pdf>.
- HOBBY, J. D. (2009). *METAPOST A User Manual*. URL <http://www.tug.org/tutorials/mp/mpman.pdf>.
- STROUSTRUP, B. (2000). *C++ Linguaggio, libreria standard, principi di programmazione*. Addison-Wesley Publishing, Pearson Education Italia (collana Professionale).
- TEIXEIRA, D. (2007). «Asymptote and L^AT_EX: An Integration Guide». URL <http://www.dse.nl/~dario/projects/asylatex/asylatex.pdf>.

▷ Agostino De Marco
Università degli Studi di Napoli
“Federico II”
Dipartimento di Ingegneria
Aerospaziale
agostino dot demarco at unina
dot it

T_EX as an ebook reader

Kaveh Bazargan

Abstract

An important advantage of eBook readers is their ability to modify text size and page orientation, for the most comfortable reading configuration. The eBook reader has to reformat the text on the fly and with minimum delay. Current eBook readers (e.g. Stanza on the iPhone) can do this reformatting, but cannot deal with complicated text such as mathematics. We have been experimenting with using T_EX as the formatting engine. Of course it can handle complex mathematics, but it also creates the best line breaks of any eBook reader. We will report our experiments with using T_EX as an ebook reader on the iPhone.

Sommario

Un importante vantaggio dei lettori di eBook è la loro capacità di modificare le dimensioni del testo e l'orientazione della pagina, in modo da trovare la configurazione ottimale per una lettura confortevole. Il lettore di eBook deve reimpaginare il testo al volo e nel minor tempo possibile. I lettori di eBook attualmente in commercio (per esempio Stanza, su iPhone) possono effettuare questa reimpaginazione, ma non sono in grado di gestire situazioni complicate, come quando compaiono formule matematiche. Abbiamo fatto degli esperimenti, usando T_EX come motore per la reimpaginazione. Naturalmente può gestire formule complesse, ma anche la divisione in righe dei paragrafi è migliore rispetto a quella ottenuta da qualsiasi lettore di eBook. Nell'articolo riportiamo i risultati dei nostri esperimenti.

1 Problems in the eBook world

1.1 Format wars

There is a 'format war' going on in the world of eBooks. Several formats are proprietary, and several openly specified. This has produced confusion for the publishers, as they don't know which format to distribute their books on, and don't know whether or not they will have to redistribute in the future, in other formats. The XML format was originally envisaged for just this sort of format war. By using a single format which embodies all the logical structure and the content of the document, any other format, e.g. PDF, can be created automatically.

ePub is a comprehensive XML format for eBooks. In particular it supports MathML for mathematical text. So the ideal scenario would be that an

eBook is saved in only one format, namely ePub, and an eBook reader would render this on the fly and present a readable view. We'll come back to this after discussing the workflow in our company.

1.2 Ugly output

One of the most compelling features of eBooks is that they can reflow the text to the user's taste. But the line breaking engine on these small devices is not that sophisticated, and the output does not look professionally typeset. For example, eBook readers on the iPhone show lots of bad breaks and large word gaps when the font size is increased. Again, we will come back to this.

2 Evolution of the workflow at River Valley Technologies

The main activity at our company is typesetting mathematical text. To explain most simply, we need to go from a T_EX file submitted by the author, to a PDF file. Before the need for generation of XML, this was a straightforward process. We would put the T_EX file into style, typeset it, proofread, etc. A few years ago, publishers started requesting XML, and rightly so. As we deal with mathematical material, we needed to use MathML in order to keep the structure for future re-use of XML.

But here is a problem with XML. It is easy to produce a file that parses and validates, but it is not easy to check the content. Checking the PDF, on the other hand is simple—it is called reading the document, or proofreading it. We had to find a way of guaranteeing the fidelity of the XML and PDF. The only solution we could think of was an automated way of generating the PDF from the XML. But this was a non-trivial task. XML deals only with content and logical structure. It does not deal with visual style, placement of figures, hyphenation, etc. There are other anomalies. For instance an XML file might list all figures at the top of the file, but the PDF will have them in the order that they appear.

Here is the workflow we came up with:

Author T_EX → Structured T_EX →
XML → Slave T_EX → PDF

The structured T_EX is not for typesetting, but simply uses T_EX as markup for tagging elements; for instance `\firstname{John}` or

\journalyear{1985}. This file is then transformed automatically into the XML. The program we use for this is a highly configured version of TEX4ht, written by the late Eitan Gurari. But for the purposes of the present discussion, it is the stages from the XML onwards that are important.

The XML file is transformed to a TEX file using XSLT. This is what we call a ‘slave’ TEX file which is not normally looked at. It is simply created in order to produce the final PDF file. The important point is that the translation of the XML to PDF is done with 100% automation. And the intermediate TEX file is not modified in any way, but simply run through TEX to produce the final PDF.

So we have two fully automated processes, one going from a TEX file to an XML file, and one from the XML to a second TEX file. The filters for these two processes were written independently, and checked rigorously. So let us suppose a proof-reader is comparing the output from the author file and that from the final TEX file, and checks that the mathematical symbols match. It is then almost certain that the XML also matches. So the method gives us a high degree of confidence in the content of the XML.

3 Back to eBooks

So how does all this relate to eBooks? Well, we said in the section about ‘format wars’ that ideally ePub should be the one and only format, and it should be rendered on the fly by the eBook reader. So we need a rendering engine that renders ePub in near-real time. Well, our XML → Slave TEX → PDF is just such an engine. So we decided to try to implement this on an eBook reader. The only device which was accessible to us and had a software development kit available was the iPhone from Apple.

So there are two elements to the full process—transforming XML to TEX, and typesetting the TEX. The second seemed the most challenging and we sought the help of Jonathan Kew. Within a short time he managed to port the full TEX program to the iPhone. We decided to create a DVI file rather than a PDF, for speed and for compact file size. Jonathan also wrote a DVI reader for the device. The program runs very quickly indeed, and turning the iPhone on its side instantly reformats the output to the new page aspect ratio.

The transformation from ePub to the slave TEX has not yet been done, but it seems to be the easier part of the problem.

3.1 Quality of output

Regarding the ‘ugly output’ referred to above, we compared our output through TEX for a purely textual file, to the output from the other eBook readers on the iPhone and we think TEX does much better. Of course TEXies should not be surprised by this. We all know that the line breaking algorithm

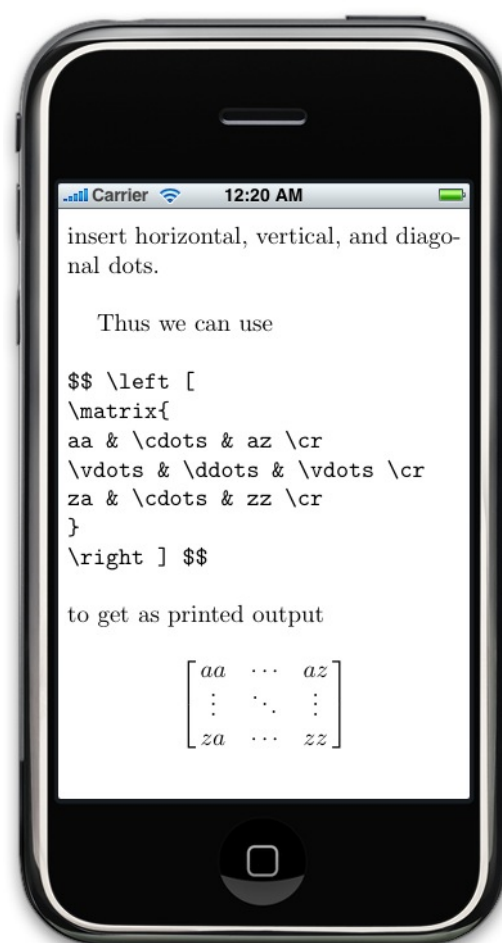
of TEX is the best!

4 Conclusions

Our automated workflow, from XML to PDF, can be modified and applied to eBook readers. The output looks good with good spacing and hyphenation, and only one format, ePub, need be produced by the publisher.

5 Acknowledgements

I am merely the ‘Steve Jobs’ of our company, others do the clever work and I take to the stage and take the credit! Here the main credit must go to Jonathan Kew who did the work reported here. Credit of course also to Radhakrishnan CV for discussions on the overall concept.



First published in *TUGboat* 30:2 (2009).
Reprinted with permission.

▷ Kaveh Bazargan
River Valley Technologies
<http://www.river-valley.com>
kaveh at river-valley dot com

La composizione di tabelle con larghezza specificata

Claudio Beccari

Sommario

Questo tutorial si propone di esaminare il funzionamento elementare dei comandi del nucleo di $\text{\LaTeX}$ che servono per produrre le tabelle con larghezza specificata; ne esamina pregi e difetti e, come esercizio, ne propone di risolvere alcuni difetti con macro apposite.

Abstract

This tutorial examines the $\text{\LaTeX}$ kernel basic commands necessary to typeset tabular material with specified width; pros and cons are discussed and, as an exercise, this tutorial suggests to correct some glitches with suitable macros.

1 Introduzione

Gli utenti di $\text{\LaTeX}$ usano l'ambiente `tabular` più o meno frequentemente e ne apprezzano la funzionalità anche se spesso non ne sfruttano appieno le potenzialità; per altro il sistema $\text{\TeX}$ ormai viene distribuito con numerosi pacchetti di estensione per la composizione delle tabelle, dei quali quello maggiormente usato sembra essere `array`.

Molto più raramente gli utenti compongono tabelle di larghezza prefissata mediante l'ambiente `tabular*`; questo non solo dipende dal fatto che le tabelle di larghezza prefissata sono meno frequenti, ma anche perché i risultati ottenibili mediante l'ambiente `tabular*` talvolta sono ben lontani dall'ottimo.

Si consideri la tabella 1 che contiene una tabella della sua larghezza naturale in base ai parametri compositivi che $\text{\LaTeX}$ usa per la composizione delle tabelle.

La tabella in questione contiene anche i filetti verticali; sono stati usati apposta per descrivere il funzionamento dei comandi, anche se non è considerato buono stile quello di usare sia i filetti verticali sia i filetti orizzontali¹. Per altro si vede che dentro ogni colonna i contenuti delle varie celle sono separati dai filetti di un piccolo spazio il cui ammontare *minimo* è specificato dal parametro interno `\tabcolsep` che è predefinito al valore di 6 pt, pari a circa 2 mm. La stessa tabella composta senza questo spazio di separazione apparirebbe come nella tabella 2 e, come si vede, è di una bruttezza inaccettabile.

1. I filetti orizzontali per iniziare e finire la tabella e per separare la righe delle intestazioni dal corpo della tabella sono ammessi.

Se si desiderasse comporre la stessa tabella specificandone la larghezza, questa, ovviamente, non potrebbe essere specificata di valore inferiore a quello della tabella 2, ma, oltre ad usare l'ambiente `tabular*`, bisogna anche provvedere ad un allungamento variabile in ogni cella in modo che $\text{\LaTeX}$ possa estendere ogni casella quanto basta perché l'intera tabella abbia la larghezza complessiva specificata. Questo, dice il manuale $\text{\LaTeX}$, LAMPORT (1994), si ottiene mediante una *@-espressione* che specifichi l'ammontare di questo spazio allungabile e di quanto esso possa venire allungato. Il manuale $\text{\LaTeX}$ ricorda all'utente anche che questo spazio allungabile viene inserito a sinistra del contenuto di ogni colonna a partire dalla colonna che segue quella che contiene la *@-espressione*. Non ha importanza se la *@-espressione* sia collocata come separatore di sinistra o di destra della cella; l'importante è che la cella delimitata dalla *@-espressione* non viene toccata da questa definizione di spazio aggiuntivo allungabile.

Rivediamo allora che cosa succede se specifichiamo l'allungamento come separatore di sinistra dalla prima colonna di ogni riga, dando come direttiva di apertura della tabella il comando

```
\begin{tabular*}{\textwidth}{%  
  @{\extracolsep{\fill}}\vline}*5{c|}}
```

Il risultato si vede nella tabella 3 e di nuovo è di una bruttezza incredibile; ma che cosa è andato storto? Eppure si sono seguite le istruzioni indicate nel manuale $\text{\LaTeX}$.

2 Gli inconvenienti della composizione delle tabelle

Gli esempi mostrati nell'introduzione usano esclusivamente i comandi del nucleo di $\text{\LaTeX}$; i risultati, specialmente per la tabella estensibile 3 non sono entusiasmanti, anzi sono pessimi.

Seguendo le raccomandazioni contenute in PANTIERI (2009) e PANTIERI (2008), oltre a quelle riportate nella guida GUIT (2009), si possono trovare molte buone idee e raccomandazioni per ovviare agli inconvenienti visti. Fra tutte, entro certi limiti, è bene tenere conto della possibilità di comporre la tabella secondo le impostazioni di default, senza cercare di estenderla ad una larghezza specificata, e poi ingrandirla o rimpicciolirla mediante il comando `\resizebox` messo a disposizione dal pacchetto `graphicx`; infatti, quando la larghezza naturale con i parametri di default è molto simile a quella desiderata, la soluzione migliore, forse, è

TABELLA 1: Tabella composta con i parametri predefiniti del nucleo di L^AT_EX

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

TABELLA 2: Tabella composta con `\tabcolsep=0pt`

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

TABELLA 3: Tabella composta con la larghezza specificata pari alla giustezza, ma con un errore relativo alla prima colonna

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

TABELLA 4: Tabella composta con la larghezza specificata pari alla giustezza senza l'errore relativo alla prima colonna

Pippo	Pluto	Paperino	Topolino	Minnie
Tizio	Caio	Sempronio	Giulio	Cesare

proprio quella di ingrandire o rimpicciolire la tabella per quella via. Negli esempi usati in questo articolo la larghezza naturale è piuttosto piccola rispetto a quella desiderata, perciò la soluzione suggerita non sarebbe adeguata.

Tuttavia qui ci concentriamo sulle possibilità offerte dai comandi contenuti nel nucleo di L^AT_EX oppure dai comandi più o meno primitivi del linguaggio T_EX.

I brutti risultati visti prima dipendono da due fattori; (a) un errore facile da commettere (e qui espressamente commesso) e (b) una incapacità di L^AT_EX di comporre le tabelle estensibili in modo accettabile. Usando pacchetti esterni, per esempio `tabularx.sty`, si potrebbero forse avere risultati migliori, ma non con una tabella così semplice come quella degli esempi appena illustrati.

L'errore (espressamente) commesso consiste nell'aver "dimenticato" che la *@-espressione* sostituisce completamente il separatore fra le colonne; nel nostro caso essa sostituisce il "separatore" del margine a sinistra della prima colonna; come si vede il contenuto della prima cella della prima colonna non è separato dal filetto verticale alla sua sinistra; ci si è (volutamente) dimenticati di reinserire lo spazio di separazione fra il filetto verticale `\vline`² e il contenuto della prima colonna. La prima riga dell'ambiente `tabular*` avrebbe dovuto essere stata scritta correttamente in questo modo:

```
\begin{tabular*}{\textwidth}{%
  @{\extracolsep{\fill}}\vline
  \hspace{\tabcolsep}*5{c|}}
```

Il risultato 4 continua a non essere accettabile, anche se ora il contenuto della prima colonna è correttamente distanziato dal filetto alla sua sini-

stra. Si vede però che l'allungamento estensibile inserito con la parte di *@-espressione* che recita

```
\extracolsep{\fill}
```

mostra il suo effetto solo a partire dalla seconda colonna e solo a sinistra del contenuto della seconda e delle restanti colonne.

Come provvedere a questa bruttezza?

Per quanto riguarda l'allungamento estensibile, basterebbe inserire delle colonne fittizie in modo che queste colonne contengano solo l'allungamento. Il codice di composizione della tabellina che abbiamo finora usato come esempio:

```
\begin{tabular*}{\textwidth}{%
  @{\extracolsep{\fill}}\vline
  \hspace{\tabcolsep}*5{c|}}
\hline
Pippo & Pluto & Paperino & Topolino
& Minnie \\
Tizio & Caio & Sempronio & Giulio
& Cesare \\
\hline
\end{tabular*}
```

dovrebbe venire modificato così:

```
\setlength{\tabcolsep}{0pt}%
\begin{tabular*}{\textwidth}{%
  @{\extracolsep{\fill}}c|*5{cc|}}
\hline
&Pippo && Pluto && Paperino && Topolino
&& Minnie& \\
&Tizio && Caio && Sempronio && Giulio
&& Cesare& \\
\hline
\end{tabular*}
```

La tabella 5 è ora composta correttamente; i filetti verticali compaiono tutti esattamente equidistanziati dai contenuti delle varie colonne, per cui

2. All'interno delle *@-espressioni* bisogna usare per il filetto verticale il comando `\vline` invece della più semplice specificazione `|`.

TABELLA 5: Tabella estensibile contenente colonne fittizie

Pippo Tizio	Pluto Caio	Paperino Sempronio	Topolino Giulio	Minnie Cesare
----------------	---------------	-----------------------	--------------------	------------------

TABELLA 6: Tabella composta con `\tabcolsep=9.87mm`

Pippo Tizio	Pluto Caio	Paperino Sempronio	Topolino Giulio	Minnie Cesare
----------------	---------------	-----------------------	--------------------	------------------

TABELLA 7: Tabella composta con `widetable`

Pippo Tizio	Pluto Caio	Paperino Sempronio	Topolino Giulio	Minnie Cesare
----------------	---------------	-----------------------	--------------------	------------------

questi contenuti sono tutti correttamente centrati fra i filetti verticali.

Tuttavia il “trucco” di azzerare la spaziatura standard (`\setlength{\tabcolsep}{0pt}`) e di aggiungere le colonne fittizie (portandole ad un totale di 11 colonne, contro le 5 effettivamente necessarie) sembra un artificio veramente acrobatico, anche se in realtà il T_EXbook, KNUTH (1996), mostra esempi di tabelle composte con i comandi primitivi del sistema T_EX, o di quelli del plain T_EX, che attribuiscono deliberatamente una colonna ad ogni separatore; se si usa plain T_EX la cosa è accettabile, ma se si usa L^AT_EX bisognerebbe evitare di ricorrere a una programmazione di così basso livello; tuttavia non scordiamoci che questa programmazione è la più efficace in quanto permette una maggiore elasticità.

Per esempio, se si volesse comporre la stessa tabella senza ricorrere ai filetti verticali ma specificando comunque una larghezza pari alla giustezza, sarebbe forse preferibile evitare gli spazi allungabili a sinistra della prima colonna e a destra dell’ultima; in questo caso, avendo specificato una colonna apposta per i separatori non sarebbe difficile modificare il codice presentato prima; il risultato sarebbe proprio quello voluto.

Tuttavia non dobbiamo scordarci che prima di comporre qualunque tabella sarebbe meglio studiarla attentamente per scegliere il tipo di composizione più appropriato, senza pretendere di comporre delle brutte tabelle al fine di rispettare specifiche di progetto tipo: “tutte le tabelle devono avere una larghezza pari alla giustezza”. Non è accettabile un approccio simile, mentre è accettabile “ogni tabella deve avere la larghezza che più le si addice in base al suo contenuto”.

3 Per evitare le acrobazie

Il titolo di questa sezione potrebbe sembrare una presa in giro. Infatti qui ci si propone di indicare come ottenere una tabella di larghezza specificata senza ricorrere alle colonne fittizie. Vedremo che le acrobazie sarà necessario farle in altri modi, quindi,

comunque sia, le tabelle di larghezza specificata richiedono sempre qualche sforzo in più per venire composte in modo decente.

3.1 Prima soluzione manuale

Se riprendiamo la tabella 2, che, conviene ricordarlo, non è composta con l’ambiente `tabular*` ma con il semplice `tabular`, e ne misuriamo la larghezza, possiamo determinare quanta larghezza manca affinché essa sia pari alla larghezza specificata; infatti questa larghezza mancante deve essere distribuita fra i vari separatori `\tabcolsep` che compaiono nella tabella stessa; essa ha 5 colonne, e ognuna ha una spaziatura `\tabcolsep` a sinistra e una a destra; dunque ogni `\tabcolsep` deve essere 1/10 della larghezza mancante. Nel nostro caso la larghezza mancante vale circa 99mm, per cui ogni `\tabcolsep` deve valere circa 9,9mm. Il “circa” sta a indicare che con un righello millimetrato è difficile essere più precisi; questo non è un vero problema: basta infatti specificare `\tabcolsep=9.9mm` e poi eventualmente correggere in più o in meno di qualche centesimo di millimetro; dopo due o tre passate di bozze si vede che `\tabcolsep=9.87mm` è perfettamente adeguato e il risultato è mostrato nella tabella 6.

Se si deve comporre una sola tabella con questa “acrobazia” manuale, il lavoro extra da fare non è poi così grande; ed è perfettamente accettabile. Se lo si deve fare per molte tabelle, allora sarebbe meglio disporre di una macro.

3.2 Seconda soluzione automatica

Nessuna macro è onnipotente, ma è possibile eseguire i calcoli con tutta la precisione di cui T_EX, il programma, è capace, purché si disponga di una macro adeguata.

A mano si è misurata la larghezza mancante e la si è divisa per il numero di `\tabcolsep` presenti nella tabella. In realtà la tabella potrebbe contenere altre *@-espressioni* e il numero di `\tabcolsep` potrebbe non essere uguale al doppio del numero delle colonne.

Conviene allora far sì che T_EX stesso calcoli quanti `\tabcolsep = t` contribuiscano alla larghezza della tabella; con qualche acrobazia si potrebbe ottenere che T_EX stesso conti il numero di segni & che compaiono in ogni riga, ma la presenza di comandi `\multicolumn` potrebbe rendere incerto il calcolo.

Per evitare questa incertezza conviene allora misurare le due larghezze l_0 ed l_1 che la tabella assume quando `\tabcolsep` vale zero o, rispettivamente, un valore convenzionale t_1 , per esempio di 6 pt. Un semplice calcolo di proporzioni permette allora di dire: se con $t = 0$ ottengo l_0 e con $t = t_1$ ottengo l_1 quanto deve valere t per ottenere la larghezza specificata l_{spec} ? Un semplice esercizio di proporzioni permette di calcolare che

$$t = t_1 \frac{l_{\text{spec}} - l_0}{l_1 - l_0} \quad (1)$$

Il calcolo è banale, ma la macro deve comporre la tabella tre volte, le prime due delle quali servono solo per determinare l_0 ed l_1 ; dopo queste due prime composizioni deve eseguire due sottrazioni di lunghezze, una divisione fra lunghezze, e una moltiplicazione di una lunghezza per il quoziente precedentemente calcolato. Tutto semplicissimo, salvo il fatto che T_EX esegue solo divisioni fra numeri interi e ne può calcolare il rapporto, ma in aritmetica intera, quindi senza determinare la parte fratta del quoziente.

Per quante richieste il sottoscritto abbia fatto al L^AT_EX Team, a tutt'oggi non esiste un algoritmo definito mediante un comando primitivo che consenta a T_EX di calcolare un quoziente fratto. Esistono solo delle macro (abbondantemente usate nel nucleo di L^AT_EX, specialmente per le operazioni che riguardano la selezione dei font) che hanno pregi e difetti; il sottoscritto ha la presunzione di ritenere che l'algoritmo di "long division", come la chiamano gli americani, cioè della divisione che ci hanno insegnato a fare alla scuola elementare, sia l'algoritmo migliore da programmare con T_EX.

Nelle pagine 46 e 47 è riportato il listato del pacchetto che contiene tutte le macro che servono per eseguire le tre composizioni e per svolgere la divisione cifra per cifra; le prime poche righe riguardano appunto l'algoritmo della divisione; il comando ad argomenti delimitati `\dividi` (BECCARI, 2008) richiede due lunghezze, una per il dividendo (prima di `\per`) e una per il divisore (dopo `\per` e prima di `\in`) e, infine, il quoziente che viene messo nel terzo argomento (dopo `\in`) e deve essere una sequenza di controllo o un carattere attivo, insomma un solo token che possa ricevere una definizione.

Il lettore non si spaventi se vuole cercare di seguire il processo; in realtà si tratta di assegnare le lunghezze dividendo e divisore a due contatori interi (operazione lecita e in questi contatori va a finire il numero intero di "scaled points" corrispondente a ciascuna lunghezza); dopo una prima

divisione intera fra questi due contatori al fine di determinare la parte intera del quoziente, e dopo avere aggiunto un punto decimale a questa stringa, si entra in una procedura iterativa che moltiplica il resto per 10 ed esegue una divisione aggiungendo il quoziente parziale (di una cifra) alla stringa che contiene il quoziente determinato fino a quel momento; è inutile proseguire oltre la sesta iterazione, perché le cifre eventualmente calcolabili non avrebbero nessun significato quando si consideri il modo di rappresentare internamente le lunghezze adottato da T_EX. Se il divisore fosse nullo, al quoziente verrebbe attribuito il valore "infinito", che per T_EX, quando si tratta di lunghezze, vale $(2^{30} - 1)/2^{16} \approx \text{\maxdimen}$.

Il resto del pacchetto definisce l'apertura dell'ambiente `\widetable` e della sua chiusura `\endwidetable` con comandi primitivi, perché bisogna leggere e memorizzare in registri token (BECARI, 2007) sia la larghezza desiderata, sia la descrizione delle colonne, sia il corpo della tabella; questo è definito come tutto quanto sia compreso nel file sorgente fra la fine del gruppo che contiene i descrittori delle colonne e la fine dell'ambiente `\widetable`. Tuttavia non basta fermarsi al comando `\end`, ma bisogna verificare che il suo argomento valga davvero `\widetable`; questo è quello che succede normalmente, ma potrebbe succedere che una cella contenga a sua volta un ambiente non racchiuso fra graffe³, quindi bisogna davvero fare questa verifica e nel caso che l'argomento di `\end` non sia `\widetable` bisogna arrestare la composizione ed emettere un avviso che spieghi il motivo dell'arresto e ponga una tabellina fittizia al posto di quella che la macro non riesce a comporre. Non capita quasi mai una situazione del genere, ma è meglio prevederla.

Merita anche notare che, se la larghezza specificata è di poco inferiore alla larghezza naturale della tabella, $l_0 < l_{\text{spec}} < l_1$, il valore predefinito di `\tabcolsep` ne viene ridotto e la tabella può ancora risultare composta in modo relativamente gradevole; se invece la larghezza specificata è decisamente inferiore alla larghezza naturale, $l_{\text{spec}} < l_0$, `\tabcolsep` viene ad assumere un valore negativo, come si vede chiaramente dall'espressione (1), di modo che le varie colonne risultano parzialmente sovrapposte le une alle altre; certamente non sarebbe difficile inserire un confronto in modo da segnalare l'inconveniente così da evitare questa sgradevole situazione; questo, però, è lasciato per esercizio al lettore. . .

Va anche notato che l'ambiente `\widetable`, così come è stato definito, non consente di specificare il valore facoltativo dell'allineamento verticale della

3. Se il contenuto di una cella di una tabella è racchiuso fra graffe, il suo contenuto viene analizzato solo al momento della composizione, quindi può tranquillamente contenere un altro ambiente, senza che il programma lo isoli come tale.

```

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{widetable}
\dimendef\wt@Numer=2
\dimendef\wt@Denom=4
\countdef\wt@Num=2
\countdef\wt@Den=4
\countdef\wt@I=6
\def\wt@segno{ }

\def\dividi#1\per#2\in#3{%
  \begingroup
  \wt@Numer #1\relax \wt@Denom #2\relax
  \ifdim\wt@Denom<\z@ \wt@Denom -\wt@Denom \wt@Numer -\wt@Numer\fi
  \ifdim\wt@Numer<\z@ \def\wt@segno{-}\wt@Numer -\wt@Numer\fi
  \ifdim\wt@Denom=\z@
    \edef\wt@Q{\ifdim\wt@Numer<\z@-\fi\strip@pt\maxdimen}%
  \else
    \wt@Num=\wt@Numer \wt@Den=\wt@Denom \divide\wt@Num\wt@Den
    \edef\wt@Q{\number\wt@Num.}%
    \advance\wt@Numer -\wt@Q\wt@Denom \wt@I=6
    \@whilenum \wt@I>\z@ \do{\wt@dividiDec\advance\wt@I\m@ne}%
  \fi
% \xdef#3{\wt@segno\wt@Q}\endgroup
\edef\x{\noexpand\endgroup\noexpand\def\noexpand#3{\wt@segno\wt@Q}}
\x
}

\def\wt@dividiDec{%
  \wt@Numer=10\wt@Numer \wt@Num=\wt@Numer \divide\wt@Num\wt@Den
  \edef\wt@Q{\number\wt@Num}\edef\wt@Q{\wt@Q\wt@Q}%
  \advance\wt@Numer -\wt@Q\wt@Denom}

\newdimen\wt@width
\def\wt@starttabular{\expandafter\table\expandafter{\wt@preamble}}
\def\widetable#1#2{%
  \def\@tempC{widetable}\setlength{\wt@width}{#1}%
  \def\wt@preamble{#2}\wt@getTable}
\newif\ifwt@scartare\wt@scartarefalse
\def\endwidetable{%
  \ifwt@scartare
    \noindent\null
  \else
    \tabcolsep=\z@
    \setbox\z@=\hbox{\wt@starttabular\the\toks@\endtabular}%
    \tabcolsep=1cm\relax
    \setbox\tw@=\hbox{\wt@starttabular\the\toks@\endtabular}%
    \advance\wt@width-\wd\z@
    \@tempdimb=\wd\tw@
    \advance\@tempdimb-\wd\z@
    \dividi\wt@width\per\@tempdimb\in\@tempA
    \tabcolsep=\@tempA\tabcolsep
    \wt@starttabular\the\toks@\endtabular
  \fi
  \ignorespacesafterend
}

\def\wt@finetabella{\end{widetable}}%

```

```

\def\wt@getTable#1\end#2{\def\@tempB{#2}%
  \ifx\@tempB\@tempC
    \toks@={#1}%
    \expandafter\wt@finetabella
  \else
    \PackageWarning{widetable}{%
      La tabella contiene l'ambiente '\@tempB' non chiuso%
    }
    \MessageBreak
    tra graffe. Questo e' espressamente vietato!%
    \MessageBreak
    La tabella non viene composta ma sostituita con%
    \MessageBreak
    un rettangolo}
    \noindent\framebox[\wt@width]{Tabella non composta perch'e
      contiene \texttt{\char'\end} in una o pi'u celle}\par
    \expandafter\wt@finishTable
  \fi
}

\def\wt@finishTable#1\end#2{%
  \def\@tempB{#2}%
  \ifx\@tempB\@tempC
    \wt@scartaretrue\expandafter\wt@finetabella
  \else
    \expandafter\wt@finishTable
  \fi
}
\endinput

```

tabella che ne viene composta; non si è ritenuto indispensabile consentire l'uso di questo parametro facoltativo, benché l'ambiente `tabular*` vi possa ricorrere; non sarebbe complicato eseguire la modifica necessaria per estendere la funzionalità di `widetable`, ma lo si lascia al lettore; lo scopo di questo tutorial, infatti, non è quello di creare un nuovo ambiente, ma di mostrare alcune delle vie che permettono di ovviare agli inconvenienti che $\text{\LaTeX}$ presenta quando si compongono tabelle di larghezza specificata.

Questa seconda soluzione potrebbe sembrare ancora più acrobatica di quanto esposto nelle sezioni precedenti; di fatto, copiato il codice in un file dal nome, per esempio, `widetable.sty`, basta invocare questo pacchetto e usare l'ambiente `widetable` al posto di `tabular*` mantenendo esattamente la stessa sintassi di quest'ultimo.

4 Conclusione

Ho voluto qui discutere a livello didattico moderatamente basso i problemi della composizione di tabelle delle quali si vuole specificare la larghezza, e ho indicato alcuni modi per superare gli inconvenienti che si manifestano.

Tutto è migliorabile, ovviamente, comprese le soluzioni che ho indicato; talvolta bisogna ingegnarsi un poco per trovare una soluzione accettabile ai problemi di composizione che $\text{\LaTeX}$ non risolve in modo ottimale da solo; questo tutorial vorrebbe sollecitare la fantasia dei lettori a manifestarsi nel-

la ricerca di soluzioni adeguate ai pochi problemi che $\text{\LaTeX}$ non riesce a risolvere da solo.

Riferimenti bibliografici

- BECCARI, C. (2007). «I registri *token*: questi sconosciuti». *ArsTeXnica*, (4).
- (2008). «Macroistruzioni con argomenti delimitati». *ArsTeXnica*, (5).
- GUI T (2009). «Introduzione all'arte della composizione tipografica con $\text{\LaTeX}$ ». v 0.73.
- KNUTH, D. E. (1996). *The TEXbook*. Addison Wesley, Reading, Mass., 16^a edizione.
- LAMPORT, L. (1994). *A document preparation system — $\text{\LaTeX}$ — User's guide and reference manual*. Addison Wesley, Reading, Mass., 2^a edizione.
- PANTIERI, L. (2008). *$\text{\LaTeX}$ per l'impaziente — Un'introduzione all'Arte di scrivere con $\text{\LaTeX}$* . Gruppo Utilizzatori Italiani di $\text{\TeX}$ e $\text{\LaTeX}$.
- (2009). *L'arte di scrivere con $\text{\LaTeX}$* . URL http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.

▷ Claudio Beccari
Villarbasce (TO)
claudio dot beccari at gmail
dot com

L'arte di gestire la bibliografia con biblatex

Lorenzo Pantieri*

Sommario

In questo articolo, basato su MORI (2009) e su LEHMAN (2009), cui si rimanda per ogni approfondimento, vengono presentati i concetti fondamentali del pacchetto biblatex, che offre all'utente una soluzione generale per la gestione e la personalizzazione della bibliografia in un documento composto con L^AT_EX. L'articolo presuppone una conoscenza di base del funzionamento del programma B^IB^TE_X (spiegato, per esempio, in PANTIERI, 2009).

Abstract

The purpose of this work is to provide the basic concepts of the biblatex package, which offers a general solution to manage and customize the bibliography in a L^AT_EX document. The article requires a basic knowledge of B^IB^TE_X.

Introduzione

Il principale limite di B^IB^TE_X è il fatto che, per poter avere un controllo totale dello stile bibliografico, è necessario imparare un linguaggio *ad hoc*, notevolmente diverso da quello cui è abituato l'utente di L^AT_EX.

Il pacchetto biblatex, invece, offre all'utente una soluzione complessiva per gestire e personalizzare gli stili bibliografici e di citazione. Infatti, oltre a includere le funzioni di molti pacchetti, tramite comandi L^AT_EX permette di effettuare modifiche anche profonde di un particolare stile bibliografico.

1 Le basi di biblatex

1.1 Installazione e uso

Il pacchetto biblatex è ancora in fase di sviluppo ma, pur non essendo incluso nelle distribuzioni ufficiali, è comunque reperibile su CTAN.¹ L'installazione di biblatex non presenta particolari difficoltà. Si scarica il file biblatex.zip da CTAN e, dopo averlo decompresso, si copiano tutti i file presenti nella cartella latex dell'archivio nella sottocartella /tex/latex/biblatex/ dell'albero personale (vedi PANTIERI, 2009), eventualmente creando tale

*Ringrazio i membri dello Staff del Gruppo Utilizzatori Italiani di T_EX e L^AT_EX, in particolare il Prof. Claudio Beccari, Massimiliano Dominici e il Prof. Enrico Gregorio, e tutti coloro che hanno discusso con me sul forum del Gruppo, in particolare Tommaso Gordini: senza il loro eccellente contributo questo articolo non avrebbe mai assunto la forma attuale. Grazie mille, ragazzi!

1. <http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/>.

cartella, se non ci fosse già; bisogna anche copiare il contenuto della sottocartella bibtex dell'archivio nella cartella /bibtex/bst/ dell'albero personale.

Il pacchetto biblatex richiede il pacchetto etoolbox, anch'esso in fase di sviluppo, ma ugualmente scaricabile da CTAN.² Neppure l'installazione di etoolbox presenta difficoltà. Si scarica il file etoolbox.zip da CTAN e, dopo averlo decompresso, si copiano i file etoolbox.sty ed etoolbox.def nella sottocartella /tex/latex/ dell'albero personale.

L'uso di babel e di csquotes (con l'opzione babel), anche se non indispensabile, è raccomandato per il buon funzionamento di biblatex.

```
\usepackage[⟨lingue⟩]{babel}
\usepackage[babel]{csquotes}
\usepackage[⟨opzioni⟩]{biblatex}
```

Ovviamente, biblatex non può essere usato assieme ai pacchetti che si propone di sostituire.

1.2 Basi di dati di riferimenti bibliografici

Una base di dati dei riferimenti bibliografici è un file, con estensione .bib, che contiene un certo numero di record, come i seguenti (i comandi \Ars e \GuIT richiedono il pacchetto guit):

```
@book{bringhurst:elementi,
  author   = {Bringhurst, Robert},
  title    = {Gli Elementi dello Stile
              Tipografico},
  publisher = {Sylvestre Bonnard},
  year     = {2001},
  location = {Milano}
}
```

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

```
@manual{gregorio:breveguida,
  author = {Gregorio, Enrico},
  title  = {Breve guida ai pacchetti di uso
              più comune},
  year   = {2008},
  url    = {http://profs.sci.univr.it/}
```

2. <http://www.ctan.org/tex-archive/macros/latex/contrib/etoolbox/>.

```

}

@manual{gregorio:programmazione,
  author = {Gregorio, Enrico},
  title = {Appunti di programmazione in
    \LaTeX{} e \TeX{}},
  year = {2008},
  url = {http://profs.sci.univr.it/}
}

@online{guit,
  title = {Gruppo Utilizzatori di \TeX{}
    e \LaTeX{}},
  year = {2009},
  url = {http://www.guit.sssup.it/},
  sortkey = {guit},
  label = {\GUIT},
}

```

Come con BIBTEX, ogni record corrisponde a un'opera il cui tipo è indicato per primo, subito dopo il carattere @. Si tratta in questo caso di un libro (contenuto quindi in un record di tipo `book`), di un articolo (contenuto in un record di tipo `article`), di due manuali (record di tipo `manual`) e di un sito Web (record di tipo `online`).

Si notino i campi non standard `journaltitle`, `location`, `sortkey` (che sostituiscono rispettivamente i campi `journal`, `address` e `key` di BIBTEX), `url` e `label` (che non hanno un analogo standard in BIBTEX e che permettono rispettivamente di scrivere l'indirizzo Internet dell'opera e di contrassegnarla con un'etichetta; vedi il paragrafo 1.4 nella pagina seguente), e il record non standard `online`, che indica una risorsa disponibile su Internet. Si noti anche l'assenza delle parentesi graffe attorno ai comandi di LATEX: se si usano gli stili predefiniti di *biblatex*, esse non sono più necessarie (vedi il paragrafo 1.5 a pagina 51).

Successivamente si indica una chiave che servirà a identificare l'opera nelle basi di dati e a citarla nei documenti come argomento dei comandi `\cite` (o analoghi). Il formato della chiave è libero, ma di solito si usa il nome dell'autore seguito da : e da una parola che caratterizza l'opera. Le chiavi devono essere *uniche* nelle basi di dati.

Poi si riempie una serie di campi che definiscono l'opera (autore, titolo...). Per un dato tipo d'opera, alcuni campi sono obbligatori e altri facoltativi. Ogni campo assume la forma

`<nome del campo> = {<contenuto del campo>}`

Come sempre, i campi sono separati tra di loro e dalla chiave con delle virgole.

1.3 I diversi tipi di record

Ogni record contiene uno o più campi. Un campo può essere:

obbligatorio, ovvero indispensabile affinché *biblatex* generi la bibliografia;

opzionale, cioè non indispensabile ma usato, se specificato, per completare le informazioni relative all'opera in questione.

Di seguito vengono riportati *i principali tipi di record* riconosciuti da *biblatex*. Per ogni tipo sono indicati i campi obbligatori e *i principali campi opzionali*; il loro significato verrà spiegato nel paragrafo successivo. I record e i campi disponibili con *biblatex* sono numerosissimi e rispondono alle esigenze più disparate; per il loro elenco completo si rimanda alla documentazione del pacchetto.

article

Articolo apparso in una rivista o in un giornale.

Campi obbligatori: `author`, `title`, `journaltitle`, `year`.

Campi opzionali: `editor`, `series`, `volume`, `number`, `month`, `pages`, `note`, `url`.

book

Libro regolarmente pubblicato da una casa editrice.

Campi obbligatori: `author`, `title`, `year`.

Campi opzionali: `editor`, `volume`, `series`, `number`, `publisher`, `location`, `note`, `url`.

booklet

Libro che viene distribuito senza un editore.

Campi obbligatori: `author` o `editor`, `title`, `year`.

Campi opzionali: `howpublished`, `type`, `note`, `location`, `chapter`, `pages`, `url`.

collection

Raccolta di contributi indipendenti di autori diversi.

Campi obbligatori: `editor`, `title`, `year`.

Campi opzionali: `volume`, `series`, `number`, `publisher`, `location`, `note`, `chapter`, `pages`, `url`.

inbook

Parte di un libro con un titolo proprio.

Campi obbligatori: `author`, `title`, `booktitle`, `year`.

Campi opzionali: `editor`, `volume`, `series`, `number`, `publisher`, `location`, `chapter`, `note`, `pages`, `url`.

incollection

Parte di una raccolta con un titolo proprio.

Campi obbligatori: `author`, `title`, `booktitle`, `year`.

Campi opzionali: `volume`, `series`, `number`, `publisher`, `location`, `note`, `chapter`, `pages`, `url`.

inproceedings

Articolo nei lavori di una conferenza.

Campi obbligatori: `author`, `editor`, `title`,

booktitle, **year**.

Campi opzionali: **volume**, **series**, **number**, **note**, **organization**, **publisher**, **location**, **chapter**, **pages**, **url**.

manual

Documentazione tecnica. I campi **author** o **editor** possono in realtà essere omessi (vedi il paragrafo 2.2 a pagina 55).

Campi obbligatori: **author** o **editor**, **title**, **year**.

Campi opzionali: **type**, **version**, **series**, **number**, **organization**, **note**, **publisher**, **location**, **chapter**, **pages**, **url**.

misc

Record da usare quando nessuno degli altri è appropriato. I campi **author** o **editor** possono in realtà essere omessi (vedi il paragrafo 2.2 a pagina 55).

Campi obbligatori: **author** o **editor**, **title**, **year**.

Campi opzionali: **howpublished**, **type**, **organization**, **location**, **note**, **date**, **url**.

online

Risorsa disponibile su Internet. I campi **author** o **editor** possono in realtà essere omessi (vedi il paragrafo 2.2 a pagina 55).

Campi obbligatori: **author** o **editor**, **title**, **year**, **url**.

Campi opzionali: **note**, **organization**, **date**.

proceedings

Lavori di una conferenza. Il campo **editor** può in realtà essere omissso (vedi il paragrafo 2.2 a pagina 55).

Campi obbligatori: **editor**, **title**, **year**.

Campi opzionali: **volume**, **series**, **number**, **note**, **organization**, **publisher**, **location**, **chapter**, **pages**, **url**.

report

Relazione pubblicata da un'università, una scuola o un'altra istituzione.

Campi obbligatori: **author**, **title**, **type**, **institution**, **year**.

Campi opzionali: **number**, **note**, **location**, **chapter**, **pages**, **url**.

thesis

Tesi di laurea o di dottorato. Il campo **type** permette di specificare il tipo di tesi.

Campi obbligatori: **author**, **title**, **type**, **institution**, **year**.

Campi opzionali: **note**, **location**, **chapter**, **pages**, **url**.

unpublished

Documento con un autore e un titolo, ma che non è stato pubblicato.

Campi obbligatori: **author**, **title**, **year**.

Campi opzionali: **howpublished**, **note**, **date**, **url**.

Si noti che il profilo di alcuni record (per esempio **book** e **inbook**) è leggermente diverso da quello standard di BIBTEX (LEHMAN, 2009, p. 8). Altri record (come **online**, **report** e **thesis**) sono sconosciuti a BIBTEX standard. Infine, alcuni record standard di BIBTEX (per esempio **mastersthesis** e **phdthesis**, entrambi sostituiti da **thesis**, e **techreport**, sostituito da **report**) non sono previsti da biblatex, anche se i loro nomi sono conservati come alias, per mantenere una compatibilità con gli stili tradizionali di BIBTEX.

1.4 I diversi tipi di campi

Di seguito è riportato l'elenco dei principali tipi di campi riconosciuti da biblatex.

author

Il nome e il cognome dell'autore (o degli autori, nel formato descritto più avanti).

booktitle

Indica il titolo dell'opera principale, per lavori che sono parte di una pubblicazione più ampia.

chapter

Il numero del capitolo (o di una qualunque parte del documento).

editor

Il nome e il cognome del curatore (o dei curatori).

howpublished

Il genere di pubblicazione.

institution

Il nome dell'università o dell'istituzione.

journaltitle

Il nome della rivista o del giornale.

label

È un'etichetta da usare per le citazioni, nel caso in cui manchino i dati necessari per formare l'etichetta "regolare" (vedi il paragrafo 2.2 a pagina 55).

location

Indica l'indirizzo dell'editore (**publisher**) oppure dell'istituzione (**institution**).

month

Il mese di pubblicazione dell'opera. Dev'essere un numero intero; per esempio, non si scrive **month={January}**, ma **month={1}**.

note

Informazioni supplementari che possono aiutare il lettore a identificare meglio l'opera.

number	Il numero della rivista, del giornale o della raccolta.
organization	L'organizzazione di patrocinio della conferenza o che pubblica il documento.
pages	Uno o più numeri di pagina.
publisher	Il nome dell'editore.
series	Il nome della collezione di cui fa parte l'opera.
sortkey	Imposta l'ordinamento alfabetico delle voci bibliografiche prive dell'indicazione dell'autore o del curatore (vedi il paragrafo 2.1 a pagina 54).
title	Il titolo dell'opera.
type	Il tipo di manuale, relazione o tesi.
url	L'indirizzo Internet dell'opera.
volume	Il volume dell'opera.
year	L'anno di pubblicazione.

Rispetto ai campi standard disponibili con `BIBTEX`, ce ne sono di nuovi: `journaltitle`, `label`, `location`, `sortkey` e `url` (dei quali si è già detto) e `booktitle` (che non ha un analogo standard). Alcuni campi non sono più previsti, anche se i loro nomi sono conservati come alias: oltre ai già menzionati `journal`, `address` e `key` (sostituiti rispettivamente da `journaltitle`, `location` e `sortkey`), è il caso di `school`, sostituito da `institution`.

1.5 Alcune precisazioni

Com'è noto, per alcuni tipi di record (articoli, lavori di conferenze...), gli stili standard di `BIBTEX` mettono automaticamente in minuscolo il titolo dell'opera; ciò può risultare fastidioso nel caso di nomi, acronimi, sigle o altro che si vogliono in maiuscolo. Questo problema non si presenta con gli stili predefiniti di `biblatex`, per cui si può scrivere semplicemente

```
title = {TCP-IP e lo Zen di Confucio}
```

e le maiuscole del titolo appariranno esattamente come sono scritte. Se però si usa uno stile personalizzato che mette in minuscolo il titolo dell'opera,

per evitarlo è sufficiente (come con `BIBTEX`) mettere tra parentesi graffe le parole che non si vogliono in minuscolo:

```
title = {{TCP-IP} e lo {Zen} di {Confucio}}
```

Altrimenti, per fare in modo che le maiuscole del titolo appaiano come sono scritte, basta racchiudere tutto il titolo dentro un'ulteriore coppia di parentesi graffe:

```
title = {{TCP-IP e lo Zen di Confucio}}
```

Se un campo contiene un comando di `LATEX`, con gli stili predefiniti di `biblatex` (a differenza di quelli di `BIBTEX`) non è necessario metterlo tra parentesi graffe:

```
title = {L'arte di scrivere con \LaTeX}
```

Se però si usa uno stile personalizzato, può essere necessario (come con `BIBTEX`) mettere i comandi di `LATEX` tra parentesi graffe, per non confondere `biblatex`:

```
title = {L'arte di scrivere con {\LaTeX}}
```

Quando un campo `author` o `editor` contiene più nomi, i nomi devono essere separati tra loro dalla parola `and`:

```
author = {Mori, Lapo Filippo and  
Himmelman, Maurizio}
```

Se l'elenco dei nomi degli autori o dei curatori è troppo lungo, può essere concluso da `and others`, che di regola viene reso da `biblatex` come *“et al.”*:

```
author = {Gregorio, Enrico and  
Mori, Lapo Filippo and  
Pantieri, Lorenzo and others}
```

I cognomi multipli si scrivono separando i cognomi e il nome in quest'ordine:

```
author = {Levi Montalcini, Rita}
```

Per i cognomi preceduti dalla particella “von” o “van” c'è una peculiarità: se l'autore è, per esempio, “Ludwig van Beethoven”, gli stili standard di `biblatex` (a differenza di quelli di `BIBTEX`), ai fini dell'ordine alfabetico, considerano il cognome come “Beethoven” e non come “van Beethoven”. Se si vuole evitarlo, seguendo lo stile comunemente usato nelle bibliografie in italiano, basta caricare `biblatex` con l'opzione `useprefix`:

```
\usepackage[...]{useprefix}{biblatex}
```

In questo modo, `biblatex` considera il cognome come “van Beethoven”. (Con qualsiasi stile, l’opzione `useprefix=false`, predefinita negli stili standard, permette di ottenere l’ordinamento considerando il cognome come “Beethoven”).

Se il cognome è invece preceduto da una particella con iniziale maiuscola, esso viene sempre ordinato alfabeticamente da `biblatex` comprendendo la particella nel cognome. Per esempio “Alcide De Gasperi” viene sempre ordinato automaticamente come “De Gasperi”:

```
author = {De Gasperi, Alcide}
```

1.6 Stili bibliografici e schemi di citazione

Con `biblatex`, i formati dei riferimenti bibliografici e delle citazioni sono contenuti in file `.bbx` (*stili bibliografici*) o `.cbx` (*schemi di citazione*). Il formato viene specificato tramite un’opzione del pacchetto:

```
\usepackage[style=alphabetic]{biblatex}
```

oppure

```
\usepackage[bibstyle=authortitle,%
  citestyle=verbose-trad1]{biblatex}
```

Nel primo caso il valore `alphabetic` è assegnato sia a `bibstyle` sia a `citestyle`.

Il pacchetto `biblatex` fornisce alcuni stili bibliografici predefiniti: numerico (`numeric`), alfabetico (`alphabetic`), autore-anno (`authoryear`), autore-titolo (`authortitle`). Essi differiscono per il tipo di etichette stampate nella bibliografia e per il formato complessivo dell’elenco dei riferimenti bibliografici.

numeric

Opere in ordine alfabetico, contrassegnate da numeri.

alphabetic

Opere contrassegnate da etichette del tipo “[Mor07]” formate a partire dal nome dell’autore e dall’anno di pubblicazione.

authoryear

Opere in ordine alfabetico, senza alcuna etichetta. Questo stile differisce dagli altri per il fatto che l’anno non è stampato alla fine della voce ma subito dopo l’autore o il curatore.

authortitle

Opere in ordine alfabetico, senza alcuna etichetta.

A ciascuno di questi stili bibliografici è possibile associare uno schema di citazione appropriato. A seconda dello schema scelto è possibile avere riferimenti numerici, con etichetta alfanumerica,

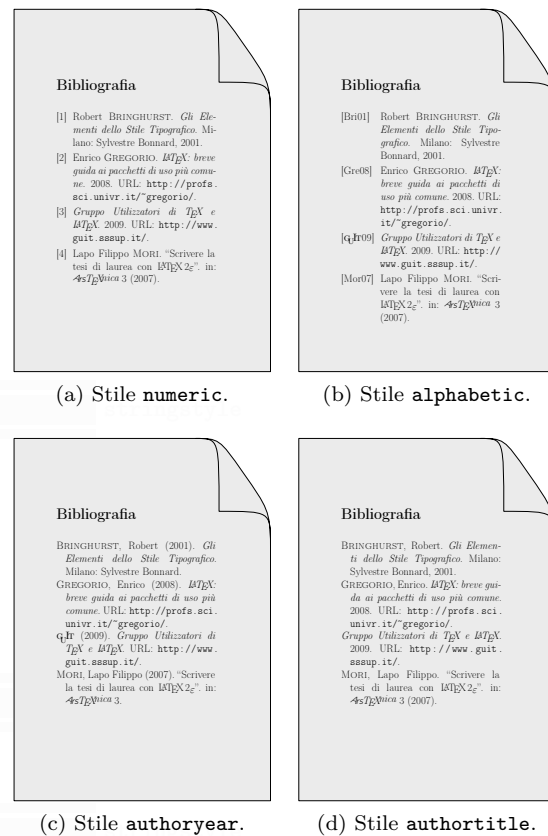


FIGURA 1: Esempi di stili bibliografici.

autore-anno, autore-titolo, in nota, eccetera. Di seguito sono riportati alcuni degli schemi di citazione predefiniti di `biblatex`.

numeric, numeric-comp

Producono riferimenti numerici, come per esempio “[1]”. Vanno usati assieme a uno stile bibliografico numerico, che stampa le etichette corrispondenti nella bibliografia. Con `numeric-comp`, una variante compatta dello schema `numeric`, le citazioni numeriche multiple sono automaticamente ordinate e compresse: per esempio si ha “[2–4, 8]” al posto di “[4, 2, 8, 3]” (è uno schema analogo al risultato che si ottiene con l’opzione `sort&compress` del pacchetto `natbib` nel modo numerico).

alphabetic

Riferimenti contrassegnati da etichette del tipo “[Mor07]” formate a partire dal nome dell’autore e dall’anno di pubblicazione. Va usato assieme a uno stile bibliografico alfabetico.

authoryear, authoryear-comp

Voci bibliografiche menzionate citando l’autore e l’anno di pubblicazione, come “(Mori, 2007)”. Questi schemi vanno usati assieme a uno stile bibliografico autore-anno. Con `authoryear-comp`, una variante compatta dello schema `authoryear`, le citazioni multiple con lo stesso autore (ed eventualmente lo stesso anno) sono automaticamente ordinate e

TABELLA 1: Alcuni esempi di comandi per le citazioni, con lo schema autore-anno compatto.

<code>\cite{mori:tesi}</code>	MORI, 2007
<code>\cite[p.~7]{mori:tesi}</code>	MORI, 2007, p. 7
<code>\cite[vedi] []{mori:tesi}</code>	vedi MORI, 2007
<code>\cite[vedi] [p.~7]{mori:tesi}</code>	vedi MORI, 2007, p. 7
<code>\parencite{mori:tesi}</code>	(MORI, 2007)
<code>\parencite[p.~7]{mori:tesi}</code>	(MORI, 2007, p. 7)
<code>\parencite[vedi] []{mori:tesi}</code>	(vedi MORI, 2007)
<code>\parencite[vedi] [p.~7]{mori:tesi}</code>	(vedi MORI, 2007, p. 7)
<code>\textcite{mori:tesi}</code>	MORI (2007)
<code>\textcite[p.~7]{mori:tesi}</code>	MORI (2007, p. 7)
<code>\textcite[vedi] []{mori:tesi}</code>	vedi MORI (2007)
<code>\textcite[vedi] [p.~7]{mori:tesi}</code>	vedi MORI (2007, p. 7)
<code>\citeauthor{mori:tesi}</code>	MORI
<code>\citeyear{mori:tesi}</code>	2007
<code>\parencite{mori:tesi,% gregorio:breveguida,% gregorio:programmazione}</code>	(GREGORIO, 2008 <i>a,b</i> ; MORI, 2007)

comprese: per esempio si ha Gregorio 2003; Gregorio 2008*a,b*; Mori 2007 invece di Gregorio 2008*b*; Mori 2007; Gregorio 2008*a*; Gregorio 2003 (analogamente all'opzione `sort` di `natbib` nel modo autore-anno).

`authortitle`, `authortitle-comp`, `verbose`

Opere menzionate citando l'autore e il titolo della pubblicazione, del tipo “Mori, *Tito*lo”. Questi schemi vanno usati assieme a uno stile bibliografico autore-titolo. A differenza dei precedenti, *si tratta di schemi pensati per citazioni date in note a piè di pagina*. Con `authortitle-comp`, una variante compatta dello schema `authortitle`, le citazioni multiple con lo stesso autore sono automaticamente ordinate e compresse: per esempio si ha Mori, *Primo titolo*, *Secondo titolo* al posto di Mori, *Primo titolo*; Mori, *Secondo titolo*. Lo schema `verbose`, che usa la citazione completa alla prima occorrenza e una forma abbreviata le volte successive, prevede alcune varianti (per esempio, `verbose-trad1`) in cui vengono usate, a seconda del contesto, le formule convenzionali *idem*, *ibidem*, *op. cit.* e *loc. cit.*

Per l'elenco completo degli schemi di citazione messi a disposizione da *biblatex* si rimanda a LEHMAN (2009, p. 12).

Quale schema (e quindi quale stile) usare dipende dal tipo di lavoro che si sta scrivendo. Nei documenti scientifici, quando si cita un'opera per riferirsi a un risultato che non si riporta per intero, spesso non ha molta importanza che il lettore sappia chi ha scritto il lavoro e quando: ciò fa sì che lo schema numerico sia molto diffuso nelle discipline scientifiche. Nei documenti umanistici, invece, il riferimento a un autore o a un anno invece che a un altro ha di per sé significato ed è importante che il

lettore possa ottenere queste informazioni direttamente dal testo: per questo motivo, nelle materie umanistiche è assai diffuso lo schema autore-anno, che però si può usare solo a condizione che si tratti di una bibliografia *moderna*. Una via di mezzo tra lo schema numerico e quello autore-anno è dato dallo schema alfabetico; i risultati di questo schema sono dubbi, dato che non è sintetico come lo schema numerico e al tempo stesso non riporta sufficienti informazioni nel caso che autore e anno di pubblicazione risultino essenziali per la comprensione della citazione. Lo schema autore-titolo si usa esclusivamente nei documenti umanistici (se nel testo si cita un'opera, una nota corrispondente ne fornisce il riferimento): questo schema dà al lettore la comodità di sapere subito a quale opera ci si riferisce, ma ha lo svantaggio di affollare la pagina di note, faticose da leggere.

C'è uno stile bibliografico adatto per ogni schema di citazione. Si noti che se si sceglie come valore di `bibstyle` uno schema di citazione, viene caricato lo stile bibliografico corrispondente. Per esempio, il codice

```
\usepackage[style=authoryear-comp]{biblatex}
```

è equivalente a

```
\usepackage[bibstyle=authoryear,%  
citetstyle=authoryear-comp]{biblatex}
```

A differenza degli stili standard di *BIBTEX*, che ipotizzano che la bibliografia sia scritta in inglese — per esempio scrivendo nell'elenco dei riferimenti qualcosa come “Mori, L. F. and Himmelmann, M.” o “*PhD Thesis*” (per una tesi di dottorato), oppure producendo citazioni come “(MORI and HIMMELMANN, 2007)” — gli stili e gli schemi predefiniti di

bibtex si adattano automaticamente alla lingua specificata da babel.

1.7 Comandi per le citazioni

Il pacchetto bibtex fornisce comandi per le citazioni che possono essere usati in diversi contesti. Oltre ai classici `\cite` e `\nocite` sono disponibili `\textcite`, da usare quando la citazione è parte integrante del discorso, `\parencite`, che racchiude la citazione fra parentesi, `\footcite`, che inserisce la citazione in nota, `\supercite` (solo per schemi numerici) per citazioni in esponente, e `\fullcite` che inserisce l'intera voce bibliografica.³

Nella tabella 1 nella pagina precedente sono riportati alcuni esempi d'uso di questi comandi, con lo schema autore-anno compatto. Nella tabella 2 nella pagina successiva sono riproposti gli stessi esempi, con lo schema numerico compatto.

1.8 Comandi per la bibliografia

Il pacchetto bibtex si discosta dall'uso tradizionale di L^AT_EX per quanto riguarda l'inserimento dei riferimenti bibliografici. Come visto in precedenza, lo stile bibliografico viene specificato come opzione del pacchetto, senza usare il comando `\bibliographystyle`. Il comando `\bibliography`, da scrivere nel preambolo, ha la sola funzione di specificare le basi di dati da usare per recuperare le voci bibliografiche (come sempre, è opportuno che il percorso dei file inclusi non contenga spazi), ma non inserisce alcun elenco. Tale elenco deve essere inserito con il comando apposito `\printbibliography`.

Come con BIB_TE_X, anche in questo caso bisogna includere nell'indice la voce relativa alla bibliografia con il comando `\addcontentsline`, perché non viene fatto automaticamente né da L^AT_EX né da bibtex:

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

per le classi book o report o, in alternativa,

```
\addcontentsline{toc}{section}{\refname}
```

per la classe article (aggiungendo eventualmente `\clearpage` prima di `\addcontentsline` se la bibliografia inizia in una nuova pagina).

Riassumendo, a parte i comandi `\cite` (o analoghi) disseminati nel documento, l'inclusione della bibliografia si fa così:

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

3. L'opzione `natbib` di bibtex permette di mantenere una compatibilità quasi totale con i documenti scritti usando il pacchetto `natbib`: la maggior parte dei nomi dei comandi per le citazioni di `natbib`, come `\citep` e `\citet`, sono conservati come alias.

```
\printbibliography
```

per le classi book o report o, in alternativa,

```
\addcontentsline{toc}{section}{\refname}
\printbibliography
```

(con eventualmente `\clearpage` prima del comando `\addcontentsline` se la bibliografia inizia in una nuova pagina) per la classe article.

Infine, come con BIB_TE_X, per generare la bibliografia e includerla nel documento, bisogna:

1. compilare il documento con L^AT_EX;
2. eseguire BIB_TE_X sul documento;
3. ricompilare due volte con L^AT_EX per includere la bibliografia e aggiornare tutti i riferimenti.

Il file `.bbl` che si ottiene dalla compilazione con BIB_TE_X non contiene, in questo caso, l'ambiente `thebibliography`, ma una serie di istruzioni che contengono i dati delle singole voci della bibliografia. La formattazione vera e propria viene eseguita al momento di comporre i riferimenti bibliografici, seguendo le istruzioni contenute nello stile bibliografico prescelto.

1.9 Riferimenti finali

L'opzione `backref` di bibtex permette di avere, nelle voci bibliografiche, l'indicazione delle pagine in cui esse sono citate:

```
\usepackage[⟨...⟩,backref]{bibtex}
```

1.10 Riferimenti cliccabili

Per rendere cliccabili le citazioni e i riferimenti finali è sufficiente caricare bibtex con l'opzione `hyperref` (bisogna anche caricare il pacchetto `hyperref`):

```
\usepackage[⟨...⟩,hyperref]{bibtex}
\usepackage[⟨...⟩]{hyperref}
```

2 Specialità

In questo paragrafo vengono presentati alcuni argomenti avanzati connessi con l'uso di bibtex.

2.1 Campi speciali

I campi elencati in questo paragrafo non contengono dati stampabili, ma servono per scopi differenti. Si possono applicare a tutti i record bibliografici.

hyphenation

Permette di impostare la lingua della voce bibliografica; il valore dev'essere il nome di una lingua nota al pacchetto `babel` (vedi il paragrafo 2.3 a fronte per maggiori dettagli).

TABELLA 2: Alcuni esempi di comandi per le citazioni, con lo schema numerico compatto.

<code>\cite{mori:tesi}</code>	[3]
<code>\cite[p.~7]{mori:tesi}</code>	[3, p. 7]
<code>\cite[vedi] []{mori:tesi}</code>	vedi [3]
<code>\cite[vedi] [p.~7]{mori:tesi}</code>	vedi [3, p. 7]
<code>\parencite{mori:tesi}</code>	[3]
<code>\parencite[p.~7]{mori:tesi}</code>	[3, p. 7]
<code>\parencite[vedi] []{mori:tesi}</code>	[vedi 3]
<code>\parencite[vedi] [p.~7]{mori:tesi}</code>	[vedi 3, p. 7]
<code>\textcite{mori:tesi}</code>	MORI [3]
<code>\textcite[p.~7]{mori:tesi}</code>	MORI [3, p. 7]
<code>\textcite[vedi] []{mori:tesi}</code>	vedi MORI [3]
<code>\textcite[vedi] [p.~7]{mori:tesi}</code>	vedi MORI [3, p. 7]
<code>\citeauthor{mori:tesi}</code>	MORI
<code>\citeyear{mori:tesi}</code>	2007
<code>\parencite{mori:tesi,% gregorio:breveguida,% gregorio:programmazione}</code>	[1–3]

sortkey

Permette di impostare l'ordinamento alfabetico degli elementi della bibliografia: serve come chiave di ordinamento nei record privi dell'indicazione dell'autore o del curatore. Sostituisce il campo standard `key` di *BIBTEX*.

keywords

Un elenco, separato da virgole, di parole chiave, che permettono a *biblatex* di eseguire dei *filtri* sulle voci bibliografiche da stampare (per un esempio, vedi il paragrafo 2.4).

2.2 Campi omissibili

Alcuni dei campi indicati come obbligatori nel paragrafo 1.3 a pagina 49 non sono in realtà sempre strettamente necessari. Per esempio, tra le voci bibliografiche del proprio documento ci può essere un libro di un autore anonimo, oppure la raccolta dei lavori di una conferenza senza un curatore, o ancora un manuale d'uso di un software privo di un autore esplicito. Di regola, ciò non costituisce un problema nella composizione della bibliografia, ma può rappresentarlo nelle citazioni: per esempio, uno schema di citazione autore-anno richiede sempre un campo `author/editor` e un campo `year`.

Nei casi in cui mancano i dati necessari per formare l'etichetta “regolare”, si può ricorrere al campo `label` per fornire un sostituto per ogni dato mancante. Il modo in cui il campo `label` è impiegato dipende dallo schema di citazione usato. Negli schemi autore-anno, per esempio, il campo `label` è adoperato come sostituto del campo `author/editor` oppure del campo `year` nelle opere che ne sono prive. Negli schemi numerici, invece, il campo `label` non viene usato, poiché in questo caso il formato delle citazioni è indipendente dai dati della bibliografia. Anche negli schemi autore-titolo il campo `label` viene di regola ignorato, poiché

il solo titolo è in genere sufficiente per formare una citazione univoca (di regola, ogni opera ha un titolo).

2.3 Bibliografia multilingue

Se è stata attivata l'opzione `babel=hyphen`,

```
\usepackage[...]{babel=hyphen}{biblatex}
```

oppure, in alternativa, l'opzione `babel=other`,

```
\usepackage[...]{babel=other}{biblatex}
```

il pacchetto *biblatex* controlla, per ogni voce della bibliografia, se è stata specificata una lingua tramite il campo `hyphenation` e, in caso affermativo, usa per quella voce le regole di sillabazione (se è stato scelto il valore `hyphen`) ed eventualmente (se è stato scelto il valore `other`, che mette la voce in un ambiente `otherlanguage`) compie la traduzione di termini quali “curatore” o “volume”, eccetera, associate a tale lingua.

Di seguito è riportato un esempio di record in cui è stata specificata la lingua:

```
@book{lamport:latex,
  author   = {Lamport, Leslie},
  title    = {\LaTeX: a document
              preparation system},
  publisher = {Addison-Wesley},
  year     = {1994},
  location = {Reading (Massachusetts)},
  hyphenation = {english}
}
```

2.4 Suddividere una bibliografia

Talvolta può essere necessario suddividere una bibliografia in base a certi criteri. Per esempio, si può

voler suddividere una bibliografia in letteratura di base e di approfondimento, oppure elencare separatamente le risorse cartacee e quelle online, oppure separare le opere citate esplicitamente nel documento da quelle suggerite non citate nel lavoro. Il pacchetto `biblatex` consente di farlo facilmente.

Supponiamo, per esempio, di avere una base di dati di riferimenti bibliografici `Bibliografia.bib` definita come a pagina 48. Si consideri il seguente codice:

```
\documentclass{article}
\usepackage[italian]{babel}
\usepackage[babel]{csquotes}
\usepackage[style=alphabetic]{biblatex}
\usepackage{guit}
\bibliography{Bibliografia}

\DeclareBibliographyCategory{basi}
\DeclareBibliographyCategory{approf}

\addtocategory{basi}{mori:tesi}
\addtocategory{approf}{bringhurst:elementi,%
  gregorio:programmazione}

\defbibheading{basi}%
  {\subsection*{Bibliografia essenziale}}
\defbibheading{approf}%
  {\subsection*{Letture d'approfondimento}}

\begin{document}
Il riferimento a-\cite{mori:tesi} appare
nella bibliografia essenziale.

Il riferimento a-\cite{bringhurst:elementi}
appare nelle letture di approfondimento.
Si possono includere negli approfondimenti
anche delle opere non esplicitamente citate
nel documento.
\nocite{gregorio:programmazione}

\section*{\refname}
\printbibliography[heading=basi,%
  category=basi]
\printbibliography[heading=approf,%
  category=approf]
\end{document}
```

Si tratta di un articolo con due bibliografie separate, una per la letteratura di base e l'altra per gli approfondimenti.

- Nel preambolo sono definite due *categorie bibliografiche* (mediante il comando `\DeclareBibliographyCategory`), una per la letteratura di base, l'altra per gli approfondimenti. Le singole voci vengono assegnate alle rispettive categorie mediante il comando `\addtocategory`.
- Il comando `\defbibheading` permette di definire il titolo delle due bibliografie (letteratura di base e approfondimenti). Le dimensioni del font dei titoli delle due bibliografie sono quelle

Il riferimento a [Mor07] appare nella bibliografia essenziale.
 Il riferimento a [Bri01] appare nelle letture di approfondimento. Si possono includere negli approfondimenti anche delle opere non esplicitamente citate nel documento.

Riferimenti bibliografici

Bibliografia essenziale

[Mor07] Lapo Filippo MORI. "Scrivere la tesi di laurea con $\text{\LaTeX}$ 2_ε", in: *ArsTeXnica* 3 (2007).

Letture d'approfondimento

[Bri01] Robert BRINGHURST. *Gli Elementi dello Stile Tipografico*. Milano: Sylvestre Bonnard, 2001.

[Gre08] Enrico GREGORIO. *Appunti di programmazione in $\text{\LaTeX}$ e $\text{\TeX}$* . 2008. URL: <http://profs.aci.univr.it/~gregorio/>.

FIGURA 2: Una bibliografia suddivisa per argomenti.

usate per comporre i sottoparagrafi; questa è la prassi, in un articolo. (Se si sta scrivendo un libro o una relazione con una bibliografia suddivisa, nel codice precedente bisogna sostituire i comandi `\subsection*` con `\section*` e l'istruzione `\section*{\refname}` con `\chapter*{\bibname}`.)

- L'argomento opzionale del comando `\printbibliography` permette di inserire una serie di istruzioni di controllo. L'opzione `heading=<nome>` imposta il titolo della bibliografia, come specificato con `\defbibheading`. L'opzione `category=<categoria>` realizza un *filtro bibliografico* nelle voci da stampare: vengono stampate solo le voci appartenenti alla *<categoria>* specificata.

Al posto dei due comandi `\printbibliography`, con i relativi argomenti opzionali, si può scrivere un semplice

```
\bibbycategory
```

che equivale a dare un `\printbibliography` per ogni categoria, nell'ordine in cui esse sono state dichiarate.

Compilando il codice precedente si ottiene il risultato visibile nella figura 2.

Il pacchetto `biblatex` offre un metodo alternativo per risolvere il problema precedente. Innanzitutto, bisogna modificare i record bibliografi-

ci della base di dati `.bib` introducendo il campo speciale `keywords`, specificando una *parola chiave* per la letteratura di base e un'altra per gli approfondimenti (per esempio, si può scrivere `keywords=basi` nelle opere che andranno nella letteratura di base e `keywords=approf` nelle opere che andranno negli approfondimenti). Poi, nel documento principale, si usa l'argomento opzionale del comando `\printbibliography` per eseguire un filtro sulle voci da stampare: l'opzione `keyword=<parola chiave>` stampa solo le opere il cui campo `keywords` contiene la *<parola chiave>*, mentre l'opzione `notkeyword=<parola chiave>` stampa solo le opere il cui campo `keywords` non contiene la *<parola chiave>*. Nel caso in questione, basta scrivere

```
\printbibliography[heading=basi,keyword=basi]
\printbibliography[heading=approf,%
keyword=approf]
```

(dove i titoli delle bibliografie sono definiti con `\defbibheading`, come nell'esempio precedente). Anche se il risultato è lo stesso, in questo modo è necessario modificare uno per uno i record bibliografici del file `.bib`; pertanto è preferibile usare le categorie bibliografiche, con il metodo visto prima.

Consideriamo un altro esempio. Supponiamo ancora di avere una base di dati di riferimenti bibliografici `Bibliografia.bib` definita come a pagina 48. Si consideri il seguente codice:

```
\documentclass{article}
\usepackage[italian]{babel}
\usepackage[babel]{csquotes}
\usepackage[style=alphabetic]{biblatex}
\usepackage{guit}
\bibliography{Bibliografia}

\defbibheading{cartaceo}%
{\subsection*{Manuali cartacei}}
\defbibheading{web}%
{\subsection*{Siti Web consultati}}

\begin{document}
I riferimenti a~\cite{bringhurst:elementi} e
a~\cite{mori:tesi} appaiono nell'elenco dei
manuali cartacei.

Il riferimento a~\cite{guit} appare
nell'elenco dei siti Web consultati.

\section*{\refname}
\printbibliography[heading=cartaceo,%
nottype=online]
\printbibliography[heading=web,type=online]
\end{document}
```

Si tratta di un articolo con due bibliografie separate, una per i manuali cartacei e l'altra per i siti Web consultati (una "sitografia").

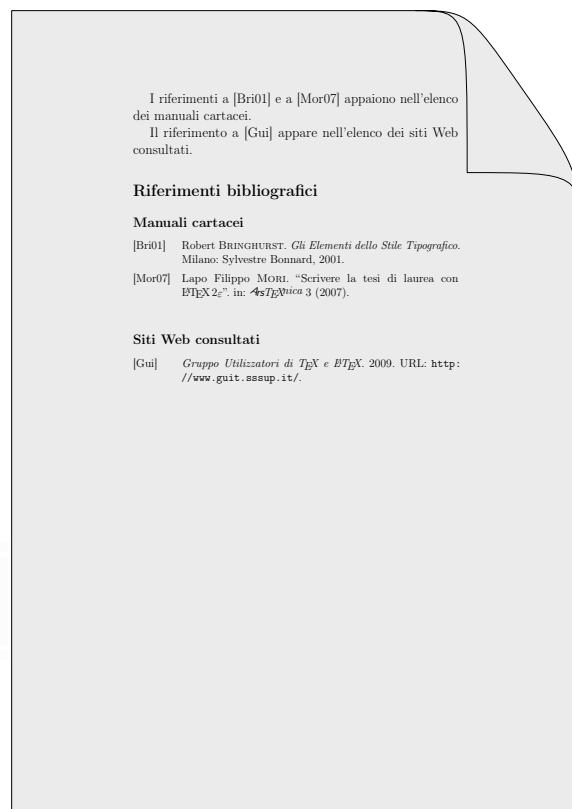


FIGURA 3: Una bibliografia suddivisa per tipo di fonte.

- Il comando `\defbibheading` permette di definire il titolo delle due bibliografie (manuali cartacei e siti Web consultati).
- L'argomento opzionale del comando `\printbibliography` consente di eseguire un filtro sulle voci da stampare: l'opzione `type=<tipo>` di `\printbibliography` stampa solo le voci bibliografiche il cui campo `type` è *<tipo>*, mentre l'opzione `nottype=<tipo>` stampa solo i record bibliografici il cui campo `type` non è *<tipo>*.

Compilando il codice precedente si ottiene il risultato visibile nella figura 3.

Consideriamo un altro esempio. Supponiamo di voler scrivere un articolo con due bibliografie separate, una per le opere citate esplicitamente nel documento e l'altra contenente un elenco di letture suggerite non citate nel lavoro. Supponiamo di avere una base di dati di riferimenti bibliografici `Bibliografia.bib` definita come a pagina 48. Si consideri il seguente codice:

```
\documentclass{article}
\usepackage[italian]{babel}
\usepackage[babel]{csquotes}
\usepackage[style=alphabetic]{biblatex}
\usepackage{guit}
\bibliography{Bibliografia}

\defbibheading{citazioni}%
{\subsection*{Opere citate}}
```

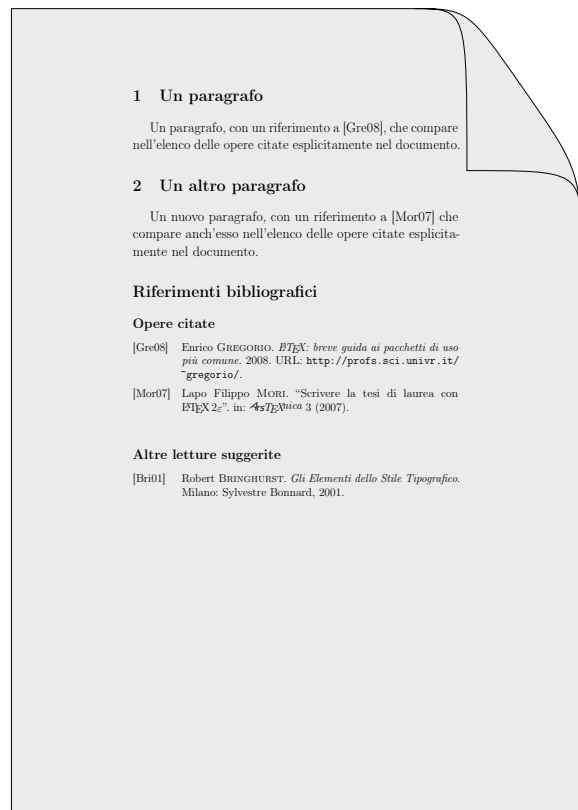


FIGURA 4: Una bibliografia suddivisa tra opere citate e altre letture suggerite.

```
\defbibheading{altro}%
{\subsection*{Altre letture suggerite}}

\defbibfilter{altro}{\not \segment{1}}

\begin{document}

\begin{refsegment}
\section{Un paragrafo}
Un paragrafo, con un riferimento
a-\cite{gregorio:brevguida}, che compare
nell'elenco delle opere citate esplicitamente
nel documento.

\section{Un altro paragrafo}
Un nuovo paragrafo, con un riferimento
a-\cite{mori:tesi} che compare anch'esso
nell'elenco delle opere citate esplicitamente
nel documento.
\end{refsegment}

\section*{\refname}
\printbibliography[heading=citazioni,%
segment=1]

\nocite{bringhurst:elementi}
\printbibliography[heading=altro,%
filter=altro]
\end{document}
```

- Il comando `\defbibheading` definisce il titolo delle due bibliografie (opere citate e altre

letture suggerite).

- Il comando `\defbibfilter{<nome del filtro>}{<filtro>}` definisce un nuovo filtro bibliografico: l'opzione `filter=<nome del filtro>` del comando `\printbibliography` consente di stampare solo le opere che soddisfano i requisiti del `<filtro>` specificato. In questo caso, vengono stampate solo le opere che *non* appartengono al primo (e in realtà unico) *segmento di riferimenti bibliografici* del documento (definito con l'ambiente `refsegment`).

Compilando il codice precedente si ottiene il risultato visibile nella figura 4.

2.5 Bibliografie separate per sezione

Talvolta può essere necessario realizzare una bibliografia separata per ogni capitolo di un libro o di una relazione, oppure per ogni paragrafo di un articolo. Per esempio, in una raccolta di articoli di diversi autori, come un volume dei lavori di una conferenza, è molto comune avere una bibliografia per ogni articolo invece di una bibliografia globale per l'intera raccolta.

Il pacchetto `biblatex` risolve questo problema, raccogliendo i riferimenti di ogni bibliografia *locale*, relativa cioè a ogni unità di sezionamento (capitolo o paragrafo), in file ausiliari che vengono elaborati da `BIB $\TeX$` .

Supponiamo, per esempio, di avere la solita base di dati di riferimenti bibliografici `Bibliografia.bib` definita come a pagina 48. Si consideri ora il seguente codice:

```
\documentclass{article}
\usepackage[italian]{babel}
\usepackage[babel]{csquotes}
\usepackage[style=alphabetic,%
refsection=section]{biblatex}
\usepackage{guit}
\bibliography{Bibliografia}

\defbibheading{bibliography}%
{\subsection*{Riferimenti bibliografici
del paragrafo-\thesection}}

\begin{document}
\section{Un paragrafo}
Un paragrafo, con un riferimento
a-\cite{bringhurst:elementi}, che
appare nella relativa bibliografia locale.

Si possono includere nella bibliografia
locale anche delle opere non esplicitamente
cite nel documento.
\nocite{gregorio:brevguida}

\printbibliography

\section{Un altro paragrafo}
Un nuovo paragrafo, con un riferimento
```

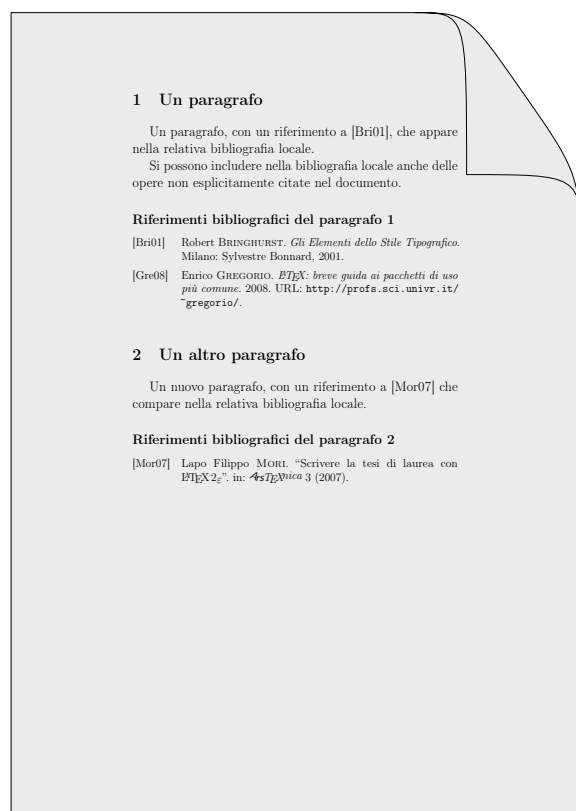


FIGURA 5: Bibliografie separate per sezione.

`a~\cite{mori:tesi}`, che compare nella relativa bibliografia locale.

```
\printbibliography
\end{document}
```

Si tratta di un articolo diviso in due paragrafi, ciascuno dei quali ha la propria bibliografia.

- Il pacchetto *biblatex* è caricato con l'opzione `refsection=section`. Questa opzione comincia automaticamente una nuova sezione di riferimenti bibliografici (`refsection`) a ogni nuovo paragrafo (`section`). (L'opzione `refsection=chapter` inizia una nuova sezione di riferimenti a ogni capitolo.)
- Il comando `\defbibheading` definisce i titoli delle bibliografie locali. Il parametro `\thesection` è il contatore dei paragrafi, gestito automaticamente da *LATEX*. (Analogamente, `\thechapter` è il contatore dei capitoli.)
- La posizione della bibliografia locale all'interno di ogni sezione si specifica con il comando `\printbibliography`.

Quando si compila con *LATEX* un documento come il precedente, per ogni unità di sezionamento viene generato un file ausiliario a sé stante (il cui nome è formato dal nome del documento principale seguito da un numero progressivo e dal suffisso `-blx.aux`);

ciascuno di questi file deve essere elaborato da *BIBTEX*. Nell'esempio in questione, se `prova.tex` è il nome del documento, si ottengono dunque due file ausiliari, chiamati `prova1-blx.aux` e `prova2-blx.aux`, da elaborare con *BIBTEX* (si dà `bibtex prova1-blx.aux` e `bibtex prova2-blx.aux` dalla linea di comando dopo essersi portati nella stessa cartella dei file). Per generare le bibliografie è necessario compilare successivamente il documento principale con *BIBTEX*; servono infine altre due compilazioni con *LATEX*.

Schematizzando, per generare le bibliografie locali con *biblatex* e includerle nel documento bisogna:

1. compilare il documento con *LATEX*;
2. eseguire *BIBTEX* su ciascuno dei file ausiliari;
3. eseguire *BIBTEX* sul documento principale;
4. ricompilare due volte con *LATEX* per includere le bibliografie e aggiornare tutti i riferimenti.

Il risultato di queste operazioni applicate all'esempio considerato è visibile nella figura 5.

2.6 L'indicizzazione automatica delle voci bibliografiche

(Questo paragrafo presuppone una conoscenza di base degli strumenti per generare un indice analitico con *LATEX* presentati, per esempio, in PANTIERI, 2009).

Un'interessante funzione di *biblatex* è l'indicizzazione automatica delle voci bibliografiche. L'indice analitico è utile al lettore di qualsiasi libro e di norma è presente in tutte le pubblicazioni scientifiche. L'opzione `indexing` di *biblatex* permette di inserire nell'indice analitico tutti gli autori citati nella bibliografia e nel testo. Il pacchetto *biblatex* si appoggia, per creare l'indice analitico, al pacchetto *makeidx* (per le funzioni di base) o al pacchetto *index* (per le funzioni avanzate, come gli indici multipli). Di seguito è riportato un semplice esempio di indicizzazione automatica dei nomi, rimandando a LEHMAN (2009, p. 41 e p. 103) per una spiegazione più dettagliata.

Supponendo di avere una base di dati di riferimenti bibliografici *Bibliografia.bib* definita come a pagina 48, il seguente codice per il documento principale permette di provare le funzioni di indicizzazione automatica di *biblatex*.

```
\documentclass{article}
\usepackage[italian]{babel}
\usepackage[babel]{csquotes}
\usepackage[style=alphabetic,%
  indexing]{biblatex}
\usepackage{guit}
\usepackage{makeidx}
\makeindex
```

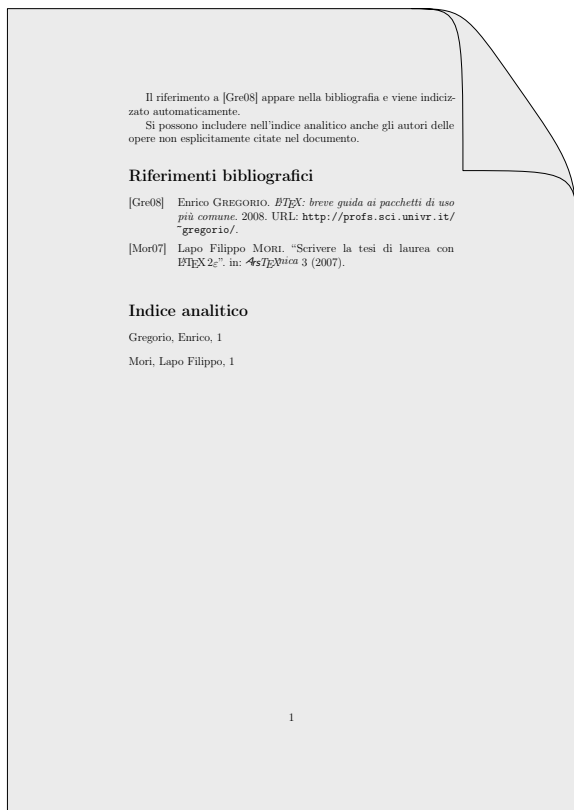


FIGURA 6: L'indicizzazione automatica delle voci bibliografiche.

```
\bibliography{Bibliografia}

\begin{document}
Il riferimento a-\cite{gregorio:breveguida}
appare nella bibliografia e viene indicizzato
automaticamente.

Si possono includere nell'indice analitico
anche gli autori delle opere non
esplicitamente citate nel documento.
\nocite{mori:tesi}

\printbibliography
\printindex
\end{document}
```

A questo punto, si procede nel modo seguente:

1. si compila il documento con $\text{\LaTeX}$;
2. si esegue $\text{\BIBTeX}$ sul documento;

3. si ricompila un'altra volta con $\text{\LaTeX}$;
4. si esegue $\text{\MakeIndex}$ sul documento;
5. si compila un'ultima volta con $\text{\LaTeX}$.

In questo modo si ottiene un documento con la normale bibliografia, seguita dall'indice analitico contenente tutti i nomi citati esplicitamente nel testo (con uno dei comandi di citazione presentati nel paragrafo 1.7 a pagina 54) e in bibliografia (con il comando $\text{\nocite}$). (Specificando $\text{\indexing=cite}$ è possibile inserire nell'indice analitico *solo* i nomi citati esplicitamente nel testo.)

Un problema che sorge è che $\text{\biblatex}$ inserisce automaticamente nell'indice analitico anche i *titoli* delle opere, cosa questa del tutto inusuale. Una possibile soluzione è ridefinire nel preambolo il campo utilizzato da $\text{\biblatex}$ per gestire l'indicizzazione dei titoli, come segue:

```
\DeclareIndexFieldFormat{indextitle}{\}\{\}
```

Il risultato di tutte queste operazioni applicate all'esempio considerato è visibile nella figura 6.

Riferimenti bibliografici

ECO, U. (1980). *Come si fa una tesi di laurea*. Bompiani, Milano.

LEHMAN, P. (2009). *The biblatex package*. <http://tug.ctan.org/tex-archive/macros/latex/exptl/biblatex/doc/>.

MORI, L. F. (2009). «Gestire la bibliografia con $\text{\LaTeX}$ ». *ArsTeXnica*, (6). URL <http://www.guit.sssup.it/arstexnica/>.

PANTIERI, L. (2009). *L'arte di scrivere con LaTeX*. Un'introduzione a $\text{\LaTeX} 2_{\epsilon}$, http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.

▷ Lorenzo Pantieri
www.lorenzopantieri.net
sit6113@iperbole.bologna.it

L^AT_EX e CSV

Massimiliano Dominici

Sommario

Questo articolo presenta alcune tecniche e alcuni esempi per gestire, all'interno di documenti L^AT_EX, dati organizzati in tabelle di *comma separated values*. In particolare l'attenzione verrà focalizzata su due pacchetti scritti appositamente per facilitare questo compito: `datatool` e `pgfplots`.

Abstract

In this paper we will present some techniques and a few examples about handling data in comma separated values. We will focus mainly on two packages specifically aimed at this purpose: `datatool` e `pgfplots`.

1 Introduzione

CSV (*comma separated values*) è un particolare formato di file adatto a rappresentare strutture di dati in forma tabulare. Un file CSV è a tutti gli effetti un file di testo, con alcuni vincoli che stabiliscono l'interpretazione del suo contenuto. Questi vincoli sono i seguenti:

1. I dati sono organizzati in record e ogni record è suddiviso in *campi* o *colonne*;
2. ogni record è delimitato da un'interruzione di riga (l'ultimo record può anche esserne privo);
3. la prima riga può contenere un'*intestazione*, ovvero il nome assegnato a ciascuna colonna;
4. ogni campo all'interno di un record, o dell'intestazione, deve essere separato da una virgola; il numero di campi deve essere lo stesso per ogni record e per l'intestazione; un campo può anche essere vuoto, ovvero non contenere dati;
5. gli spazi, anche iniziali e finali, fanno parte del campo e devono essere preservati;
6. i doppi apici (") funzionano da carattere di *escaping*: se un campo contiene virgole, interruzioni di riga o doppi apici, deve essere racchiuso tra doppi apici; i doppi apici contenuti in un campo devono sempre essere raddoppiati.

In base a quanto detto sopra, il codice seguente è una corretta rappresentazione del contenuto di un file CSV:

Autore, Titolo, Anno, Casa Editrice
C. E. Gadda, La meccanica, 1958, "Garzanti, Milano"

A questo punto, sono però necessarie alcune precisazioni. La struttura di un file CSV non è mai stata formalizzata in uno standard. I vincoli descritti sopra sono quelli riportati in un documento informale dello IETF (International Engineering Task Force) per il tipo MIME `text/csv` (IETF, 2005). Tuttavia non tutte le applicazioni che gestiscono file CSV implementano lo stesso insieme di specifiche: in particolare quasi tutte disattendono quanto stabilito al punto 5 e scartano automaticamente gli spazi a inizio e fine campo. Se si vogliono preservare questi spazi è quindi bene racchiudere il campo tra doppi apici. Inoltre quasi tutte le applicazioni sono in grado di usare (in lettura e scrittura) caratteri diversi per i delimitatori di campo. Per esempio "punto e virgola" o "TAB", al posto della virgola.¹

Poiché un CSV (nel senso esteso specificato nella nota 1) è molto adatto a rappresentare dati di qualsiasi tipo in forma tabulare, i campi di applicazione in cui viene usato sono numerosi e di ambito differente. In particolare in questo articolo prenderemo in considerazione la gestione di semplici basi di dati e la manipolazione di tabelle di valori numerici provenienti da registrazioni di dati sperimentali o simulazioni numeriche. I dati organizzati nel CSV potranno essere stati scritti manualmente, o, più verosimilmente, provenire da un'applicazione. In quest'ultimo caso è probabile che si tratti di un gestore di basi di dati (Oracle, MySQL, PostgreSQL, ecc.), di un foglio di calcolo (MS Excel, OpenOffice Calc, ecc.) oppure di un programma per il calcolo numerico o simbolico (Matlab, Octave, Mathematica, Sagemath, ecc.).

2 Gestire semplici basi di dati

Il formato CSV si presta abbastanza bene a rappresentare semplici basi di dati di uso quotidiano in applicazioni da ufficio, come rubriche di indirizzi o registrazioni di punteggi scolastici. I dati possono essere compilati manualmente, oppure essere estratti da un foglio di calcolo o da una base di dati.

È bene far notare che il formato CSV è un formato "appiattito", ovvero tutti i dati sono contenuti in un'unica tabella. Chi ha dimestichezza

1. In questo caso, bisognerebbe piuttosto parlare di "Fielded Text" (<http://www.fieldedtext.org/>). Ma siccome il "Fielded Text" copre casi più generali del CSV, e le varie implementazioni di quest'ultimo esistono ormai da molti anni, si continua a considerare CSV anche un insieme di dati separati da caratteri diversi dalla virgola.

con le basi di dati sa che invece, in genere, la loro struttura è più complessa, organizzata in diverse tabelle coordinate tra loro. Il modello più diffuso è il cosiddetto *Relational Database Management System* (RDBMS), alla base di numerosi prodotti commerciali, *open source* o *free software*. Non è questa la sede adatta per illustrare tale modello; basterà dire che di solito le applicazioni che implementano questo tipo di basi di dati offrono la possibilità di esportare in formato CSV, quindi in un'unica tabella, dati provenienti dalle varie tabelle interne, opportunamente organizzati (per esempio filtrati secondo le indicazioni passate al programma, o ordinati in base a una o più colonne scelte dall'utente).

Per quanto riguarda i fogli di calcolo, invece, i dati sono solitamente già "appiattiti" all'origine e potrà essere necessario, al limite, solo l'ordinamento in base ad una colonna data.

Lo strumento migliore per gestire un CSV di questo tipo all'interno di un documento L $\TeX$ è il pacchetto `datatool`.

2.1 Generalità su `datatool`

Il pacchetto `datatool` (TALBOT, 2009), successore di `csvtools` (TALBOT, 2007), è stato disegnato per poter creare, modificare, caricare e analizzare delle semplici basi di dati. Per semplice intendiamo sia "dalla struttura semplificata" (con tutti i dati, generalmente, contenuti in un'unica tabella) che di ridotte dimensioni. T $\TeX$ infatti non è particolarmente efficiente nel gestire questo tipo di operazione.

Nel seguito dell'articolo ci occuperemo principalmente della gestione di file CSV creati esternamente, quindi non accenneremo, se non di sfuggita, ai comandi che si occupano delle prime due operazioni possibili sui *database*. Esamineremo, invece, nei dettagli quanto attiene alle ultime due operazioni.

I primi due comandi che l'utente deve conoscere sono quelli relativi al caricamento e alla visualizzazione dei *database*.

```
\DTLloaddb[<opzioni>
  {<nome_database>}{<nome_file>}
```

carica il *database* contenuto nel file `nome_file` assegnandogli il nome `nome_database`. Se non vengono specificate opzioni il comando assume implicitamente che la prima riga del file CSV contenga un'intestazione e di conseguenza contrassegnerà ogni colonna con la stringa contenuta nel corrispondente campo della prima riga. Le `<opzioni>` servono appunto a modificare questo comportamento:

noheader segnala che la prima riga non contiene l'intestazione ma valori normali;

keys accetta un insieme di valori, separati da virgole, e li usa per contrassegnare le varie colonne, sovrascrivendo quelle eventualmente presenti nell'intestazione;

headers accetta un insieme di valori, separati da virgole, e li usa per definire l'intestazione del *database* in fase di visualizzazione.

Immaginiamo di avere un ipotetico file `messaggi.csv` in cui sia riportato, mese per mese, il numero di messaggi postati nelle principali categorie del Forum G Υ R nel periodo novembre 2008/febbraio 2009, ripartito per ogni singola categoria presa in considerazione:

```
TeX e LaTeX,235,212,289,321
ConTeXt,0,0,0,0
Altri programmi,24,18,19,31
Tipografia,13,11,18,22
Corsi e didattica,3,2,4,4
Edizioni critiche,2,0,5,3
```

Come si può vedere, il CSV non ha un'intestazione e, senza opzioni, `\DTLloaddb` interpreterebbe scorrettamente la prima riga. Il seguente comando, risolve il problema:

```
\DTLloaddb[noheader,
  keys={cat,nov,dic,gen,feb},
  headers={Categoria, Novembre 2008,
    Dicembre 2008,Gennaio 2009,
    Febbraio 2009}]{messaggi}{messaggi.csv}
```

Il file `messaggi.csv` segue lo standard per quanto riguarda l'uso dei separatori di campo, ma, se così non fosse, si potrebbe usare `datatool`, che fornisce i comandi `\DTLsetseparator{<char>}` e `\DTLsettabseparator` per cambiare separatore in maniera opportuna.² Può anche rendersi necessario impostare il carattere che delimita i campi (in mancanza di specifiche: `"`); è possibile farlo con il comando `\DTLsetdelimeter{<char>}`.

Infine, nel caso che il file CSV contenga caratteri che L $\TeX$ interpreta in maniera speciale ($\$, \&, \%$, ecc.), il *database* va caricato con il comando `\DTLloadrawdb`, che ha la stessa sintassi e fa in modo che tali caratteri vengano interpretati correttamente.³

Per visualizzare in maniera, semplice il precedente *database*, è sufficiente dare l'istruzione

```
\DTLdisplaydb{messaggi}
```

che dispone i dati in una tabella, come quella riportata nella figura 1. In caso di *database* corposi, la tabella può essere particolarmente lunga e occupare più pagine. Sarà opportuno, allora, usare una `longtable`, tramite il comando

```
\DTLdisplaylongdb[<opzioni>
  {<nome_database>}
```

dove le `<opzioni>` servono a impostare l'aspetto dei vari elementi dell'ambiente `longtable`.

2. Il primo dei due comandi serve per impostare il separatore a un carattere generico (per esempio `"`); il secondo nel caso speciale che il separatore sia il carattere `TAB`.

3. È perfettamente lecito introdurre nel file CSV comandi L $\TeX$, che verranno correttamente interpretati, per cui, per esempio, il primo campo della prima riga del file `messaggi.csv` può essere riscritto così: `"\TeX{} e \LaTeX"`.

Categoria	Novembre 2008	Dicembre 2008	Gennaio 2009	Febbraio 2009
T E X e L A T E X	235	212	289	321
ConT E Xt	0	0	0	0
Altri programmi	24	18	19	31
Tipografia	13	11	18	22
Corsi e didattica	3	2	4	4
Edizioni critiche	2	0	5	3

FIGURA 1: Visualizzazione automatica del database `messaggi`.

Tornando alla figura 1, si può notare che le varie colonne sono allineate automaticamente in considerazione del tipo di dato contenuto. Il testo, per esempio, è allineato a sinistra, mentre i numeri sono allineati a destra. `datatool`, infatti, è in grado di riconoscere quattro diversi tipi di dati: stringhe di testo, numeri interi, numeri reali, valuta.⁴ L'utente può modificare gli allineamenti predifiniti usando comandi della forma `\dtl<tipo_dato>align`, dove `<tipo_dato>` è uno dei seguenti: `string`, `int`, `real`, `currency`. Molti altri aspetti della tabella così ottenuta possono essere modificati in maniera limitata (presenza di filetti orizzontali o verticali, aspetto dell'intestazione, applicazione di particolari istruzioni a colonne contenenti un certo tipo di dato, ecc.). Tutto ciò è adeguatamente descritto nel manuale.

Questo modo di visualizzare un database è comodo solo se non si deve operare sui dati, per organizzarli, filtrarli, manipolarli, ecc. A questo scopo è preferibile usare il comando `\DTLforeach`, che permette di scorrere tutti i record e operare sui campi ivi contenuti. La sua sintassi è la seguente:

```
\DTLforeach[<condizioni>]{<nome_database>}{
  <assegnazioni>}{<istruzioni>}
```

Le `<condizioni>` sono, appunto, opzioni di filtro da applicare ai dati (ad esempio ci interessa visualizzare solo record che contengano, per un dato campo, valori maggiori di zero). Le `<assegnazioni>` costituiscono un modo, invece, per filtrare le colonne. Possiamo infatti essere interessati a visualizzare solo una parte dei campi riportati nel database; assegnando un particolare campo a un comando, possiamo richiamare il valore di quel campo, per ogni record, tramite il relativo comando. Le `<istruzioni>`, infine, determinano ciò che vogliamo fare con i dati in questione.

Possiamo, ad esempio, ricostruire (approssimativamente) la tabella illustrata nella figura 1, con l'aggiunta di una ulteriore colonna con i totali, per mezzo del seguente codice:

```
\begin{tabular}{lrrrrr}
\bfseries Categoria &
```

4. `datatool` è inoltre corredato di una serie di comandi, accessibili all'utente, che permettono di identificare, comparare e manipolare questi quattro tipi di dati. In questo articolo non prenderemo in esame tali comandi, se non dove è necessario per la comprensione degli esempi. Per una panoramica completa si rimanda al manuale (TALBOT, 2009).

```
\bfseries Novembre 2008 &
\bfseries Dicembre 2008 &
\bfseries Gennaio 2009 &
\bfseries Febbraio 2009 &
\bfseries Totale%
\DTLforeach*{messaggi}{%
  \cat=cat,\nov=nov,\dic=dic,%
  \gen=gen,\feb=feb}{%
  \\\
  \cat \gdef\tot{0} &
  \nov \DTLgadd{\tot}{\nov}{\tot} &
  \dic \DTLgadd{\tot}{\dic}{\tot} &
  \gen \DTLgadd{\tot}{\gen}{\tot} &
  \feb \DTLgadd{\tot}{\feb}{\tot} &
  \tot
}
\end{tabular}
```

Il risultato è visibile nella figura 2. Per generare i totali dell'ultima colonna, usiamo il comando `\DTLgadd`⁵ i cui tre argomenti obbligatori rappresentano rispettivamente: il registro in cui viene immagazzinato il risultato della somma e i due addendi. All'inizio di ogni riga azzeriamo il valore del totale. Poiché ci troviamo all'interno di una cella l'assegnazione deve essere globale (`\gdef`). Si noterà che `\DTLforeach` è chiamato, in questo caso, nella versione con asterisco. Questo significa che il database viene usato in modalità di sola lettura. In precedenza, infatti, è stato già accennato che `datatool` permette anche di creare e modificare un database. Se si vuole essere sicuri che nessuna modifica venga apportata al file è bene usare la versione con asterisco dei comandi che manipolano i database.

A questo punto, avendo esaminato gli strumenti da usare, possiamo introdurre un paio di esempi concreti.

2.2 Primo esempio: *mail merging*

Le maggiori distribuzioni di TEX contengono più di una classe disegnata per comporre lettere formali (DEKKER, 2008; KOHM e MORAWSKI, 2009; LAMPORT *et al.*, 2008; MEZZETTI, 2006; THOMPSON, 2009). La maggior parte di queste forniscono anche meccanismi per produrre automaticamente le etichette con gli indirizzi, e qualcuna anche la possibilità di creare numerose istanze della stessa lettera verso diversi destinatari, il cui indirizzo viene estratto da un file esterno (*mail merging*). In caso queste funzionalità non venissero fornite

5. `\DTLgadd` fa parte di una libreria di comandi per effettuare calcoli aritmetici in virgola fissa inclusa in `datatool` e basata sul pacchetto `fp`.

Categoria	Novembre 2008	Dicembre 2008	Gennaio 2009	Febbraio 2009	Totale
T _E X e L ^A T _E X	235	212	289	321	1,057
ConTeXt	0	0	0	0	0
Altri programmi	24	18	19	31	92
Tipografia	13	11	18	22	64
Corsi e didattica	3	2	4	4	13
Edizioni critiche	2	0	5	3	10

FIGURA 2: Aggiunta di una colonna alla visualizzazione del database `messaggi`.

dalla classe in questione, si possono usare pacchetti dedicati (BRAAMS, 1994; EMMEL, 2006; RAHTZ *et al.*, 2003; VEYTSMAN, 1997).

Il problema è che, in ogni caso, il formato accettato da tutti questi pacchetti per il file che contiene la rubrica, non è compatibile con il formato CSV. In alcuni casi è possibile costringere l'applicazione in cui originariamente si trovano i dati ad esportare nel formato richiesto, ma in genere sarebbe più comodo poter lavorare con un file CSV. Grazie a `datatool` questo è possibile.

L'esempio che proporremo usa la classe `guitletter`. Questa è semplicemente una versione della classe standard `letter` personalizzata in modo da inserire nella testatina e nel piè di pagina il logo del G_UT_EX e altre informazioni relative al gruppo.⁶ Tutto ciò che si può fare con `guitletter` si può fare anche con `letter`. In particolare vedremo come preparare automaticamente una lettera di benvenuto nel gruppo per diversi destinatari e come generare automaticamente le etichette con gli indirizzi da apporre sulle buste.

Per comodità dividiamo in più parti l'analisi del codice, cominciando dal preambolo:

```
\documentclass[12pt,color]{guitletter}
\usepackage[italian]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{datatool}

\DTLloaddb{indirizzi}{indirizzi.csv}

\signature{Massimiliano Dominici}
\city{Pisa}
\date{\today}
\subject{Adesione al \GuITtext}

\makelabels
```

Conosciamo già il comando `\DTLloaddb` che serve, in questo caso, a caricare la nostra rubrica degli indirizzi:

```
Nome,Cognome,Via,Numero,Città,Prov,CAP
Mario,Bianchi,Piazza Cavour,7,Novi Ligure,AL,15067
Carlo,Rossi,Via Roma,8,Noto,SR,96017
Franco,Verdi,Corso Garibaldi,3,Senigallia,AN,60017
```

Le altre istruzioni riguardano la composizione delle lettere e delle etichette: `\signature`, `\city`, `\today` e `\subject` fanno parte delle informazioni su mittente, oggetto, ecc., mentre `\makelabels` specifica che vogliamo anche stampare le etichette con gli indirizzi dei destinatari.

6. La classe non è disponibile pubblicamente.

Il corpo del documento, è invece il seguente:

```
\begin{document}

\DTLforeach*{indirizzi}{%
  \Nome=Nome,\Cognome=Cognome,%
  \Via=Via,\Numero=Numero,%
  \Citta=Città,\Prov=Prov,\CAP=CAP}{%
\begin{letter}{\Nome\ \Cognome\
  \Via, \Numero\
  \CAP\ -- \Citta\ (\Prov)}

\opening{Caro socio,}

<testo della lettera>

\closing{Cordialmente,}

\end{letter}
}

\end{document}
```

Ogni singola lettera deve essere racchiusa all'interno di un ambiente `letter`. Mentre `\opening` e `\closing` si limitano ad inserire le formule di saluto, formattandole adeguatamente, le informazioni da variare per ciascuna lettera si trovano nell'argomento obbligatorio dell'ambiente. Proprio perché tali informazioni devono variare, le inseriamo sotto forma di *registri*. Di volta in volta, leggendo i vari record della rubrica, il comando `\DTLforeach` assegna a tali registri il valore del campo corrispondente, generando tre lettere (una per record).

Le etichette vengono generate automaticamente, al momento della chiamata `\end{document}`. L'istruzione `\makelabels` che abbiamo visto in precedenza, infatti, abilita la scrittura delle informazioni relative ai vari indirizzi sul file `.aux`, dove `\end{document}` può leggerle e poi provvedere a stamparle.

2.3 Secondo esempio: produrre schede bibliografiche

In questo secondo esempio mostreremo che è possibile, fino a un certo punto, operare con diversi file CSV come se fossero le tabelle di un database relazionale. Attenzione: questo tipo di approccio è, in genere, sconsigliabile, ed è preferibile, come detto in precedenza, organizzare appropriatamente i dati al momento dell'esportazione in CSV dall'applicazione in cui i dati sono contenuti, e lavorare quindi su un'unica tabella. Pianificare questa operazione in funzione di ciò che si vuole ottenere è fondamentale per avere buoni risultati.

Tuttavia, allo scopo di esplorare le potenzialità di `datatool`, e in considerazione del fatto che non sempre l'utente è in grado di organizzare da sé tali dati, ma può trovarsi nella condizione di dover lavorare su materiale fornito da altri, daremo anche un esempio dell'uso contemporaneo di più file CSV in relazione l'uno con l'altro.

Una possibile applicazione di questo principio è la costruzione di schede bibliografiche (per esempio per una biblioteca personale) a partire da tre file in cui siano contenute, rispettivamente, le informazioni riguardanti gli *autori*, le *case editrici* e i singoli *titoli*. Uno degli scopi di un database relazionale è quello di evitare la ridondanza dei dati: è più comodo, infatti, dover gestire le informazioni relative, ad esempio, a un autore in una singola collocazione e richiamare poi queste informazioni tramite un riferimento.

I primi due file CSV riportati nella figura 3 non contengono riferimenti a dati presenti in altri file, ma il terzo CSV, invece, contiene due campi (`RefAutore` e `RefEditrice`) che prevedono un riferimento agli altri due file. Il riferimento si effettua tramite il codice dell'autore, o della casa editrice corrispondenti, per come sono specificati nell'apposito campo del relativo file. Così, nella seconda riga di `titoli.csv`, il terzo campo (`RefAutore`) riporta il valore 003 che corrisponde, nel file `autori.csv` all'autore Francesco de Sanctis. Nella sesta riga di `titoli.csv`, il quinto campo (`RefEditrice`) riporta il valore 002 che corrisponde, nel file `case_editrici.csv` alla casa editrice Garzanti.

Con `datatool` è possibile usare filtri sui valori di tali campi per connettere i dati contenuti nei vari file CSV.

Il codice seguente produce come risultato una serie di schede bibliografiche (una per pagina), come quelle riportate nella figura 4.

```
\newcounter{scheda}
\setcounter{scheda}{0}

\DTLforeach*{autori}{%
  \AutId=Id,\Nome=Nome,\Cognome=Cognome}{%
  \DTLforeach*[AutId=RefAutore]{titoli}{%
    \RefAutore=RefAutore,%
    \RefEditrice=RefEditrice,\Titolo=Titolo,%
    \Anno=Anno,\Note=Note,\Codice=Codice}{%
    \DTLforeach*[EdId=RefEditrice]{editori}{%
      \EdId=Id,\NomeEditore=Nome,\Luogo=Luogo}{%
        \vspace*{\stretch{1}}
        \stepcounter{scheda}
        \centering
        \fbox{
          \begin{tabular}[t]{t}{%
            >\columncolor[gray]{.8}}r
            >\raggedright\arraybackslashp{3.5cm}
          }
          \multicolumn{2}{l}{Scheda n° \thescheda}\\
          \midrule
          Autore: &
            \Nome
            \CheckEmpty{\Nome}{\ }
            \Cognome\\
```

```
\midrule
Titolo: & \Titolo\\
\midrule
Editore: &
  \NomeEditore
  \CheckEmpty{\NomeEditore}{ }
  {\CheckEmpty{\Luogo}{ }{, }}
  \Luogo\\
\midrule
Anno: & \Anno\\
\midrule
Note: & \Note\\
\midrule
Codice: & \Codice\\
\midrule
\end{tabular}
}
\vspace*{\stretch{1}}
\clearpage
}
}
```

Il risultato è stato ottenuto annidando tre diverse chiamate a `\DTLforeach` (una per file CSV) e imponendo la condizione, nei cicli interni, che i valori dei campi di riferimento del file CSV principale (`titoli.csv`) coincidano con il valore del campo di identificazione degli altri due file CSV. Ogni scheda ha un numero progressivo di identificazione, ottenuto con `\thescheda` che visualizza il valore corrente del contatore che tiene traccia dell'avanzamento nella "produzione" di schede.

Le schede sono ordinate per autore. `datatool` consente infatti di imporre ai database caricati un ordinamento, con il comando `\DTLsort`. Le impostazioni usate nel nostro caso sono state le seguenti:

```
\DTLloaddb{autori}{autori.csv}
\DTLloaddb{editori}{case_editrici.csv}
\DTLloaddb{titoli}{titoli.csv}

\DTLsort*{Cognome,Nome}{autori}
\DTLsort*{Nome,Luogo}{editori}
\DTLsort*{Titolo,Anno,Codice}{titoli}
```

Come si nota dal codice precedente, l'ordinamento può interessare anche più campi. Ovviamente, nel nostro caso, l'ordinamento principale è quello imposto dal ciclo più esterno.

Il comando `\CheckEmpty`, costruito a partire da *macro* fornite dal pacchetto `datatool`, serve ad evitare la presenza di spazi o punteggiatura spuri. La sua definizione è la seguente:

```
\newcommand\CheckEmpty[3]{%
  \ifthenelse{\DTLisseq{#1}{ }{#2}{#3}%
}
```

Infine, il codice sopra esposto richiede di caricare i pacchetti `booktabs` e `colortbl` per le rifiniture della tabella e i tocchi di colore.

3 Tabelle di dati numerici

Come abbiamo accennato nell'introduzione, i vari formati di esportazione sotto forma di matrici di

```

Id,Nome,Cognome
001,Carlo Emilio,Gadda
002,Ryunosuke,Akutagawa
003,Francesco,De Sanctis
004,Louis,Stevenson
005,Jan,Tschichold
006,Matthew P.,Shiel
007,Giorgio,Manganelli
008,Galileo,Galilei
009,,Erodoto

Codice,Titolo,RefAutore,Anno,RefEditrice,Note
001,Storia della letteratura italiana,003,2006,005,
002,Weir di Hermiston,004,2000,006,
003,Le storie,009,2006,001,2 volumi
004,La Meccanica,001,1999,002,
005,Un fulmine sul 220,001,2005,002,
006,La nube purpurea,006,2004,003,
007,Rashomon e altri racconti,002,2008,004,
008,Opere,008,2005,001,
009,Teatro,007,2008,007,
010,La forma del libro,005,2003,008,

```

FIGURA 3: File CSV rappresentanti tabelle di un database bibliografico.

Scheda n° 1 Autore: Ryunosuke Akutagawa Titolo: Rashomon e altri racconti Editore: TEA, Milano Anno: 2008 Note: Codice: 007	Scheda n° 2 Autore: Francesco De Sanctis Titolo: Storia della letteratura italiana Editore: BUR, Milano Anno: 2006 Note: Codice: 001	Scheda n° 3 Autore: Erodoto Titolo: Le storie Editore: UTET, Torino Anno: 2006 Note: 2 volumi Codice: 003
Scheda n° 4 Autore: Carlo Emilio Gadda Titolo: La Meccanica Editore: Garzanti, Milano Anno: 1999 Note: Codice: 004	Scheda n° 5 Autore: Carlo Emilio Gadda Titolo: Un fulmine sul 220 Editore: Garzanti, Milano Anno: 2005 Note: Codice: 005	Scheda n° 6 Autore: Galileo Galilei Titolo: Opere Editore: UTET, Torino Anno: 2005 Note: Codice: 008
Scheda n° 7 Autore: Giorgio Manganelli Titolo: Teatro Editore: Bompiani, Milano Anno: 2008 Note: Codice: 009	Scheda n° 8 Autore: Matthew P. Shiel Titolo: La nube purpurea Editore: Adelphi, Milano Anno: 2004 Note: Codice: 006	Scheda n° 9 Autore: Louis Stevenson Titolo: Weir di Hermiston Editore: Mondadori, Milano Anno: 2000 Note: Codice: 002
Scheda n° 10 Autore: Jan Tschichold Titolo: La forma del libro Editore: Edizioni Sylvestre Bonnard, Milano Anno: 2003 Note: Codice: 010		

FIGURA 4: Schede bibliografiche.

valori, da parte di numerose applicazioni di calcolo numerico, possono essere assimilati, agli effetti pratici, a dei CSV. La loro utilità sta nel fatto che possono essere usati come formato di interscambio tra applicazioni diverse e, per ciò che ci riguarda più da vicino, per tracciare grafici con LATEX.

Di norma i vari software che abbiamo citato nell'introduzione (vedi sezione 1) hanno anche la capacità di generare grafici (tramite funzionalità interne, o appoggiandosi a programmi come GNUPLOT). Tuttavia la possibilità di poter controllare l'aspetto tipografico delle illustrazioni, in modo da avere una stretta integrazione con il testo, è fondamentale per ottenere un risultato estetico di ottima qualità (si veda a questo proposito DE MARCO (2008)).

3.1 Il pacchetto pgfplots

Il pacchetto `pgfplots` (FEUERSÄNGER, 2009a) si appoggia al sistema grafico PGF/TikZ (TANTAU, 2008) per tracciare grafici a partire da tabelle di coordinate; queste possono essere passate alle *macro* che si occupano del tracciamento secondo diverse modalità: esplicitamente, implicitamente sotto forma di espressione matematica, delegando a GNUPLOT⁷ il calcolo esplicito, caricandole da un file esterno.

L'ambiente principale è `axis`, che traccia assi e griglia secondo le indicazioni dell'utente, e all'interno del quale è possibile aggiungere i singoli grafici tramite il comando `\addplot`. I tipi di grafici supportati sono numerosi e comprendono istogrammi, diagrammi di dispersione, diagrammi cumulativi, ecc.

Come tutti i pacchetti che si appoggiano a PGF, e come PGF stesso, l'uso di `pgfplots` è configurabile praticamente all'infinito. In questa sede ci limiteremo a mostrare solo alcune possibili modifiche alle impostazioni predefinite, rimandando ad un'attenta lettura del manuale chi volesse acquistare dimestichezza con le varie opzioni.

3.2 Esempio: importare dati da GNU Octave

GNU Octave⁸ è un programma libero per l'analisi numerica, parzialmente compatibile con Matlab e corredato di funzionalità per il calcolo matriciale. Il programma avvia una shell all'interno della quale è possibile passare le istruzioni all'interprete, il quale restituisce (implicitamente o esplicitamente) un *output* sotto forma matriciale. Le istruzioni possono essere anche preparate in un file esterno, sotto forma di funzioni o di *script*, ed essere poi richiamate dalla shell. I risultati ottenuti possono essere visualizzati sotto forma di grafici oppure esportati in file esterni sotto diversi formati. Quello che ci interessa è il formato ASCII. I valori vengono

espressi in notazione scientifica, che `pgfplots` è in grado di comprendere, e organizzati in tabelle.

Questo, per esempio, è l'inizio di un file del genere:

```
0.00000000e+00 1.00000000e+00 5.00000000e-01
5.66993145e-03 1.00797044e+00 4.99726732e-01
4.51639430e-02 1.06532387e+00 4.98401027e-01
8.51085701e-02 1.12669546e+00 4.98125368e-01
1.25796018e-01 1.19282722e+00 4.99007906e-01
1.67235284e-01 1.26407660e+00 5.01190251e-01
2.06413960e-01 1.33517539e+00 5.04527388e-01
2.43682108e-01 1.40627123e+00 5.08935002e-01
```

Il file immagazzina i risultati di un'analisi numerica effettuata su equazioni di Lotka-Volterra (note anche come equazioni preda-predatore, in quanto forniscono un modello matematico per un ecosistema in cui interagiscono due specie animali: una come predatore, l'altra come preda).

Si noterà che il formato differisce da quello di un CSV standard: infatti il separatore di campo è in questo caso uno spazio (o più spazi successivi). Tuttavia, come premesso nell'introduzione, in questo articolo consideriamo un formato CSV in senso esteso (si veda la nota 1). `pgfplots`, d'altra parte, è in grado di interpretare correttamente anche CSV veri e propri ed altri formati con separatori non standard. Per una lista completa si consulti il manuale. Le righe precedute da `#` o `%` vengono automaticamente saltate (per mantenere compatibilità con file generati da GNUPLOT), e la prima riga viene considerata come intestazione che definisce i nomi delle colonne se contiene almeno un valore non numerico.⁹

Gli script di Octave usati per generare i dati sono i seguenti:

```
t0 = 0;
tf = 20;
init_vals = [1; 0.5];
options=odeset('AbsTol',1.e-12,'RelTol',
1.e-9,'InitialStep',2,'MaxStep',2);
[t,x] = ode45(@pred_prey_odes,[t0,tf],
init_vals,options);
A = [t,x];
save -ascii PredPrey.dat A

function deriv_vals = pred_prey_odes(t,x)
deriv_vals = zeros(size(x));
deriv_vals(1) = 2*x(1) - 1.2*x(1).*x(2);
deriv_vals(2) = -1*x(2) + 0.9*x(1).*x(2);
```

e sono stati adattati a partire da quelli presenti su questa pagina web: http://paws.wcu.edu/emcnelis/NCSI_Houston08.html. L'adattamento è consistito nel sostituire le istruzioni per generare i grafici con quelle per salvare i dati nel file `PredPrey.dat`.

9. Appoggiandosi sul pacchetto `pgfplotstable` (FEUERSÄNGER, 2009b) per la lettura di tabelle di valori da file esterni, `pgfplots` riconosce i quattro tipi di dati che hanno senso per il primo: stringhe di caratteri alfanumerici, interi, reali e date. Tuttavia è in grado di usare realmente solo gli ultimi tre. Come si vede dall'esempio riportato sopra, i reali possono essere espressi anche in notazione scientifica.

7. <http://www.gnuplot.info/>.

8. <http://www.gnu.org/software/octave/>.

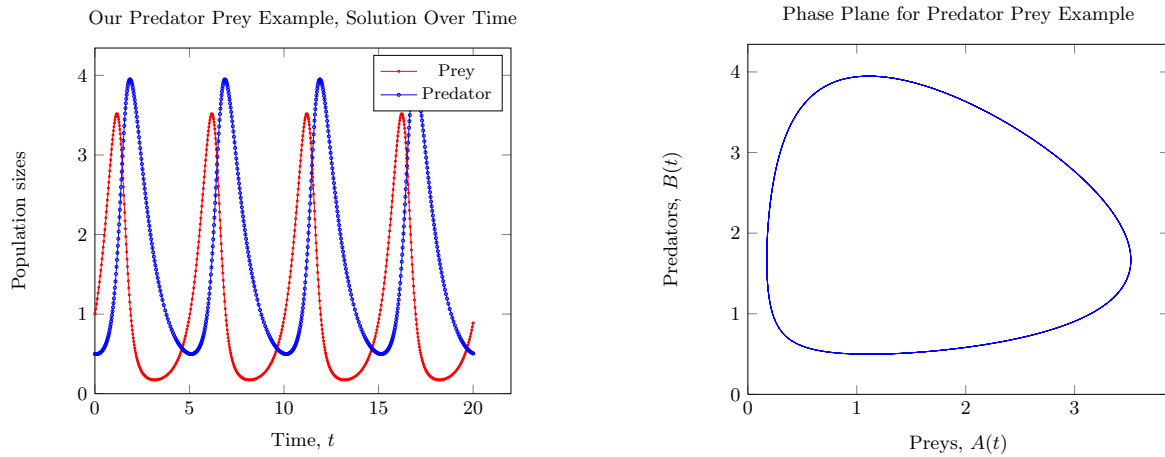


FIGURA 5: Equazioni di Lotka-Volterra

Tali dati sono stati poi importati in un file `.tex`, in cui era stato precedentemente caricato il pacchetto `pgfplots`.

In primo luogo tracciamo il grafico con l'andamento delle soluzioni nel tempo:

```
\begin{tikzpicture}
  \begin{axis}[ymin=0,xmin=0,
    xlabel={Time, $t$},
    ylabel={Population sizes},
    title={Our Predator Prey Example,
      Solution Over Time},
    legend entries={Prey,Predator}]
    \addplot[red,mark=asterisk,
      mark options={scale=.35}]
      plot table[x index=0,
        y index=1] {PredPrey.dat};
    \addplot[blue,mark=o,
      mark options={scale=.35}]
      plot table[x index=0,
        y index=2] {PredPrey.dat};
  \end{axis}
\end{tikzpicture}
```

I grafici vengono tracciati, come abbiamo detto sopra, tramite il comando `\addplot`. Poiché ci troviamo di fronte a un file che contiene più di due colonne di dati, dobbiamo aggiungere la specificazione `table`, che ci permette di scegliere di volta in volta le colonne da usare per le coordinate.¹⁰ Le colonne possono essere identificate da una stringa in un'intestazione, oppure, se il file ne è privo, da un indice progressivo, come in questo caso. Scegliamo per il grafico della prima soluzione le colonne 1 e 2 (indici 0 e 1) e per il grafico della seconda le colonne 1 e 3 (indici 0 e 2). Le altre opzioni di `\addplot` servono a personalizzare l'aspetto visuale del grafico (colore, stile e dimensioni dei marcatori, ecc.). Altre personalizzazioni sono state effettuate sull'aspetto degli elementi "esterni" della figura: assi, etichette, legenda, e sono contenute nell'argomento opzionale dell'ambiente `axis`.

10. Altrimenti avremmo potuto usare la specificazione `file`.

Tracciamo anche, in una illustrazione a parte, il diagramma nel piano delle fasi, scegliendo le colonne 2 e 3 (indici 1 e 2):

```
\begin{tikzpicture}
  \begin{axis}[ymin=0,xmin=0,
    xlabel={Preys, $A(t)$},
    ylabel={Predators, $B(t)$},
    title={Phase Plane for Predator Prey Example},
    line width=.1pt]
    \addplot[blue,mark=none]
      table[x index=1,y index=2] {PredPrey.dat};
  \end{axis}
\end{tikzpicture}
```

In questo caso abbiamo ommesso i marcatori (`mark=none`) e modificato lo spessore delle linee. Dopo aver aggiunto, nel preambolo, alcune impostazioni globali per cambiare le dimensioni dei caratteri

```
\pgfplotsset{
  tick label style={font=\small},
  label style={font=\small},
  title style={font=\small},
  legend style={font=\footnotesize},
}
```

otteniamo finalmente le due illustrazioni contenute nella figura 5.

4 Raffronto tra `datatool` e `pgfplots`

Come si può arguire dalla scelta degli esempi, benché entrambi i pacchetti siano disegnati per gestire tabelle di dati organizzati in record e campi, essi sono tuttavia indirizzati a diversi ambiti di applicazioni. Questo nonostante entrambi, accanto alle funzionalità fondamentali (visualizzazione e manipolazione di basi di dati per `datatool` e tracciamento di grafici per `pgfplots`) consentano anche, tramite pacchetti "ancillari" distribuiti insieme a quello principale, di eseguire compiti secondari che finiscono per sovrapporsi. `datatool`, infatti, permette di visualizzare il contenuto di un database anche sotto la forma di un semplice grafico, o di

un istogramma o di un diagramma circolare (grafico a torta), grazie ai pacchetti `dataplot`, `databar` e `datapie`. D'altra parte `pgfplotstable` può caricare, manipolare e visualizzare una tabella contenente dati di diverso tipo (testi, numeri interi e reali, date).

Il fattore discriminante nell'uso dell'uno o dell'altro pacchetto rimane il tipo di dati con i quali si deve lavorare e la loro organizzazione. Se i dati sono costituiti da matrici di valori numerici da visualizzare con grafici, `pgfplots` è la scelta più opportuna, scelta che diventa addirittura obbligata se i dati sono forniti in notazione scientifica, che `datatool` non è in grado di riconoscere. D'altra parte, se lo scopo è quello di manipolare e visualizzare dati di diverso tipo organizzati in un database, `datatool` risulta essere lo strumento più adatto, grazie alla flessibilità di comandi come `\DTLforeach`, alla libreria di funzioni per confrontare stringhe di caratteri di ogni tipo, e, conseguentemente, alla possibilità di applicare filtri ai dati da visualizzare.

5 Ringraziamenti e note

L'impulso iniziale a scrivere questo articolo lo devo a Maurizio W. Himmelmann e all'esigenza di trovare un sistema "comodo" per gestire in via `TEXnica` la corrispondenza del `GUT`. Per i sensibili miglioramenti rispetto alla versione iniziale dell'articolo devo ringraziare l'anonimo recensore e la redazione di *ArsT_EXnica*.

È superfluo far notare, infine, che tutti i dati esposti nella sezione 2, tranne quelli relativi alle schede bibliografiche, sono inventati di sana pianta e che ogni eventuale riferimento a fatti e persone realmente esistenti è puramente casuale.

Riferimenti bibliografici

- BRAAMS, J. (1994). «Creating a mailing». <http://www.ctan.org/get/macros/latex/contrib/mailling/mailling.pdf>.
- DE MARCO, A. (2008). «Gestione avanzata delle figure in L^AT_EX». *ArsT_EXnica*, (6), pp. 10–27. URL <http://www.guit.sssup.it/arstexnica.php>.
- DEKKER, W. (2008). «The isodoc class». <http://www.ctan.org/get/macros/latex/contrib/isodoc/isodoc.pdf>.
- EMMEL, T. (2006). «Making labels, visiting cards, pins and flash-cards with L^AT_EX». <http://www.ctan.org/get/macros/latex/contrib/ticket/doc/manual.pdf>.
- FEUERSÄNGER, C. (2009a). «Manual for Package PGFPLOTS». <http://www.ctan.org/>

[get/graphics/pgf/contrib/pgfplots/doc/latex/pgfplots/pgfplots.pdf](http://www.ctan.org/get/graphics/pgf/contrib/pgfplots/doc/latex/pgfplots/pgfplots.pdf).

— (2009b). «Manual for Package PGFPLOTS». <http://www.ctan.org/get/graphics/pgf/contrib/pgfplots/doc/latex/pgfplots/pgfplotstable.pdf>.

IETF (2005). «RFC4180». <http://tools.ietf.org/html/rfc4180>.

KOHM, M. e MORAWSKI, J.-U. (2009). «Koma Script. The Guide». <http://www.ctan.org/get/macros/latex/contrib/koma-script/scrguide.pdf>.

LAMPORT, L., MITTELBAACH, F. e SCHÖPF, R. (2008). «Standard Letter Document Class for L^AT_EX 2_ε version». <http://www.ctan.org/get/macros/latex/base/letter.dtx>.

MEZZETTI, G. (2006). «The C.D.P. Bundle». <http://www.ctan.org/get/macros/latex/contrib/cdpbundl/cdpbundl.dtx>.

RAHTZ, S., BARROCA, L., GUSTAFSON, G. e GILBEY, J. (2003). «A package for making sticky labels in L^AT_EX». <http://www.ctan.org/get/macros/latex/contrib/labels/labels.pdf>.

TALBOT, N. (2007). «csvtools v1.24 : A L^AT_EX 2_ε Package Providing Access to Data Saved in a CSV File». <http://www.ctan.org/tex-archive/obsolete/macros/latex/contrib/csvtools/doc/csvtools.pdf>.

— (2009). «datatool v 2.02: Databases and data manipulation». <http://www.ctan.org/get/macros/latex/contrib/datatool/datatool.pdf>.

TANTAU, T. (2008). «The TikZ and PGF Packages. Manual for version 2.00». <http://www.ctan.org/get/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf>.

THOMPSON, P. A. (2009). «A New Letter, Fax, Memo Document Class for L^AT_EX 2_ε». <http://www.ctan.org/get/macros/latex/contrib/newlrm/manual.pdf>.

VEYTSMAN, B. (1997). «Printing Envelopes and Labels in L^AT_EX 2_ε: EnvLab Package. User Guide». <http://www.ctan.org/get/macros/latex/contrib/envlab/envlab.pdf>.

▷ Massimiliano Dominici
mlgdominici at interfree dot
it

combelow: abbasso i segni diacritici di serie B

Gianluca Pignalberi

Sommario

Romeno e lettone devono essere considerate lingue di serie B nel mondo di $\text{\TeX}$? Se finora lo potevano essere, questo piccolo pacchetto tenta di riportarle al livello giusto con il semplice uso del segno diacritico corretto. Niente più cediglia al posto del *comma below*.

Abstract

Should Romanian and Latvian be considered second choice languages in the $\text{\TeX}$ world? Though up to now they could have been, this small package tries to put them back at the right level, just using the correct diacritic mark. No more cedilla instead of *comma below*.

1 Introduzione

Se avete modo di vedere, o leggere se ne siete in grado — io purtroppo no —, delle pubblicazioni in lingua romena, vi accorgerete senz'altro che spesso le 't' e le 's' hanno un segno strano. Questo segno diacritico, un modificatore di suono, è una virgola centrata sotto il carattere in questione.

Se provate a riscrivere un testo cercando di riprodurre fedelmente quel segno diacritico, vi troverete presto frustrati dal fatto che $\text{\LaTeX}$ non ha un comando per comporlo.

Guardando più attentamente i giornali romeni noterete che, in base alla testata e al font in uso, quella specie di virgola viene sostituita con una cediglia (¸). Anch'io, dovendo scrivere un volantino in romeno (copiando bovinamente quanto mi è stato scritto), ho dovuto sottostare alla sostituzione, con ampio senso di insoddisfazione, nonostante le rassicurazioni di madrelingua romeni.

Per determinare da quale parte stava la correttezza, ho fatto ricorso alle tabelle Unicode (KORPELA, 2006; UNICODE CONSORTIUM, 2009a) e a Wikipedia (WIKIPEDIA (2009b); e il più dettagliato corrispondente italiano WIKIPEDIA (2009a)). Queste mi hanno detto alcune cose non proprio armoniche, di cui proviamo a ricostruire la natura. Le tabelle di Unicode (UNICODE CONSORTIUM, 2009c,d) non prevedono caratteri con *comma below*, ma solo con cediglia. La forma della cediglia però varia a seconda della lettera a cui è applicata, e non riesco a capire la logica di quest'applicazione. Si può comunque notare che la cediglia sotto 's' e 't' è a forma di uncino (la forma caratteristica del simbolo, somigliante alla parte inferiore del numero 5), mentre sotto le altre lettere per cui

è prevista la cediglia questa è a forma di virgola. In UNICODE CONSORTIUM (2009d) viene detto che 'š' e 'ţ' sono varianti tipografiche preferibili a 's' e 't' per il romeno. In ogni caso, il simbolo '¸' compare da solo, come *comma below*, nella tabella delle composizioni degli accenti (UNICODE CONSORTIUM, 2009b).

Wikipedia, invece, è un po' più dettagliata, spiegando in quali lingue si usa la cediglia con la lettera 's' (ad esempio in turco, tartaro e turkmeno) e la proposta di Ambroise Firmin-Didot per usarla con la 't'. Spiega il suo uso in lettone, dicendo che tipograficamente il simbolo è una virgola, e ha un caso particolare di posizionamento che vedremo nel paragrafo 3; precisa infine la questione della cediglia in romeno. Fornisce inoltre un link molto interessante al Diacritics Project (TYPO.CZ e DESIGNIQ.CZ, 2009). Qui il simbolo in questione viene chiamato *comma accent*.

Tutte le informazioni raccolte mi hanno permesso la scrittura di un pacchetto (veramente più di uno, come leggerete nel paragrafo 4), rilasciato da pochi giorni su CTAN, chiamato *combelow*.

2 $\text{\LaTeX}$ e cediglia

Abbiamo accennato al fatto che gli utenti $\text{\LaTeX}$ hanno sempre approssimato il *comma below* con la cediglia in caso di testi in lingua romena. La stessa cosa avviene con *babel* caricando l'opzione per la lingua romena (BRAAMS, 2006). Siamo abituati a vedere la cediglia sotto la lettera 'c' in francese. Abbiamo visto anche che metterla sotto 's' e 't' non crea problemi dal punto di vista tipografico: abbiamo sempre considerato lettere con profondità praticamente pari a zero.

La lingua lettone prevede l'uso del *comma below* con 'g', 'k', 'l', 'n' e 'r'. Possiamo sempre approssimare il *comma below* con la cediglia? Vediamo che succede: g, k, l, n, r. La cediglia non è adatta a essere usata con lettere discendenti in quanto non tiene conto della profondità della lettera a cui si applica (il fatto viene anche citato a pagina 47 di ELJKHOUT (1993)). Oltre a non essere il simbolo corretto nei casi esaminati. Volendo poi essere pignoli, non viene neanche posto bene sotto la 'r': andrebbe allineato all'asta.

3 Comma below: casi normali e speciali

Il segno *comma below* ha un caso particolare di applicazione, proprio nella lingua lettone, con la 'g',

а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ь ы э ю я

FIGURA 1: Il segno comma below applicato all’alfabeto russo: anche in questo caso il rilevamento automatico dei discendenti funziona sempre, così come quello della diversa altezza degli ascendenti. Non mi risulta che il comma below sia applicato all’alfabeto cirillico

l’unica lettera discendente. Mentre nei casi normali il segno diacritico si trova centrato sotto la lettera a cui si applica, nel caso della ‘g’ questo deve essere posto centrato al di *sopra* del carattere, e ruotato di 180°. In questo caso cambia anche il nome del segno diacritico, diventando *turned comma above* (‘[^]’).

Il comando introdotto dal pacchetto `combelow` è `\cb`, e prende come unico argomento il carattere a cui bisogna applicare il segno diacritico. In caso il carattere sia Unicode multibyte, questo *deve* essere posto in parentesi graffe. Quello che si ottiene applicando il nuovo segno diacritico ai caratteri visti delle lingue romena e lettone è questo: ġ, ķ, ļ, ņ, ŀ, ŕ, ŝ, ŧ.

Per come avevo scritto il codice in due versioni precedenti del pacchetto (mai rilasciate), il comando non era in grado di riconoscere le lettere discendenti, potendo quindi discriminare il solo caso della ‘g’; per tacere del maggiore spreco di memoria dovuto alla minor efficienza dell’algoritmo usato. Ora, come vedremo nel paragrafo 5, il pacchetto riconosce tutti i casi, anche per alfabeti diversi dal latino. Vediamo un esempio di applicazione del comma below alle lettere dell’alfabeto russo nella figura 1, sebbene l’esempio abbia solo valore estetico: non c’è questo segno in cirillico.

4 Peripezie di un codice

La mia “conoscenza” di TEX e LATEX mi ha permesso di scrivere un pacchetto che ritenevo quasi definitivo (versione 0.99), ma che non ha resistito ai suggerimenti di Claudio Beccari in fase di giudizio. Siccome è una persona *veramente* competente, ha trovato immediatamente il suo pregio e i suoi difetti. La mia affezione al codice e al numero mi ha convinto a lavorare a versioni parallele del pacchetto (una sorta di `fork`), e a contrassegnare le versioni con una lettera aggiuntiva. Nelle sezioni seguenti descriviamo tutte le versioni prodotte, a partire dalla prima versione, quella radiografata dal comitato scientifico, e definita in seguito 0.99a. Si tenga presente che alcune idee per eliminare i difetti sono tratte da un codice di esempio di `combelow` scritto da Claudio Beccari. La qualità del suo codice è semplicemente annichilente.

5 combelow 0.99a

Il codice del pacchetto viene riportato qui di seguito, riformattato per esigenze di impaginazione e senza commenti di qualunque natura. Allo stesso modo saranno riformattati tutti gli altri codici.

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{combelow}
3   [2009/08/23 v0.99a comma below
   command]
4 \RequirePackage{graphicx}
5 \RequirePackage{ifthen}
6 \newdimen\fcxpos
7 \newdimen\fcypos
8 \newdimen\ftpos
9 \newcommand{\cb}[1]{%
10 \setbox1=\hbox{#1}%
11 \setbox2=\hbox{,}%
12 \fcxpos=.5\wd1%
13 \advance\fcxpos by .5\wd2%
14 \ftpos=.5\wd1%
15 \advance\ftpos by -.5\wd2%
16 \ifthenelse{\lengthtest{\dp1>
17   .1\ht1}}%
18 {\fcypos=\ht1%
19 \advance\fcypos by .6ex%
20 \raisebox{0pt}{#1\hspace
21   {-\fcxpos}\raisebox{\fcypos}%
22   {\rotatebox[origin=c]{180}{,}%
23     \hspace{\ftpos}}}}}%
24 {\fcypos=\dp1%
25 \advance\fcypos by .4ex%
26 \raisebox{0pt}{#1\hspace
27   {-\fcxpos}\raisebox{-\fcypos},
28   \hspace{\ftpos}}}}%
29 }
30 \endinput

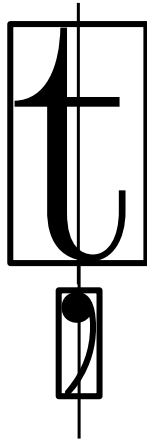
```

5.1 Commento al codice

Le righe da 1 a 8 specificano la versione minima di LATEX prevista, caricano i pacchetti necessari al funzionamento e definiscono alcune variabili usate per posizionare il segno diacritico e il testo dopo la lettera accentata; le righe 9–29 definiscono il comando `\cb`.

In due box (righe 10 e 11) poniamo il carattere da accentare e la virgola (il segno diacritico). Siccome il segno diacritico deve essere centrato sopra o sotto la lettera (le mediane sono allineate verticalmente), la posizione del suo estremo sinistro sull’asse *x* corrisponderà alla metà del box della lettera, traslato a sinistra della metà del proprio box (righe 12 e 13); vedi la figura 2.

Le righe 14 e 15 calcolano la posizione del testo dopo la lettera accentata: questo si troverà a destra del box del carattere, separato da esso da uno spazio pari a metà del box della virgola. Se non è il valore esatto, è senz’altro un’ottima approssimazione (vedi la figura 3).

FIGURA 2: Allineamento dei box nel comando `\cb`

București

FIGURA 3: Nome originale della capitale romena composto con `combelow`

Le righe 16–28 si occupano del posizionamento del segno diacritico: se la profondità del carattere è più del 10% della sua altezza (valore arbitrario, righe 16 e 17) allora bisogna porre un turned comma above, superando di poco (0.6 ex) l'altezza della lettera (righe 18–23), altrimenti si pone il segno diacritico normale sotto la lettera, lasciando il solito “spazio di rispetto” (posto arbitrariamente a 0.4 ex, righe 24–28). In entrambi i casi si lascia lo spazio orizzontale discusso per riprendere la normale composizione del testo seguente.

5.2 Pro e contro

Il pregio del pacchetto appena presentato è che è indipendente dalla codifica, cioè funziona sempre senza doversi preoccupare di quello che carichiamo in `fontenc`. A fronte di questo pregio presenta una pletora di difetti non da poco: l'algoritmo di posizionamento adottato fa perdere le informazioni sulla crenatura, e anche la sillabazione non sarà più possibile per la parola accentata. Vediamo un esempio di questo problema nelle figure 4 e 5. Rimandiamo al paragrafo 9 una breve discussione su quest'ultimo punto.

Ruotare la virgola di 180° è sostanzialmente inutile: si carica il pacchetto `graphicx` solo per ottenere un simbolo che è identico alla virgoletta singola aperta (‘). Un altro problema riscontrabile con questa versione del pacchetto è che se una lettera discendente con comma below deve essere inclinata (corsivo o inclinato — slanted) il segno diacritico non viene più centrato sulla lettera. Usando la virgoletta singola aperta al posto della virgola ruotata il problema scompare (vedi il paragrafo 6). Come già accennato nel paragrafo 2, anche queste due versioni della pacchetto non posiziona bene

l'accento sotto la ‘r’: la centratura mal si accorda con una lettera con asta decentrata.

Ulteriore problema è che il comando, così come è definito, non può essere usato in didascalie di figure e tabelle. La soluzione di questo problema consiste nel sostituire, nella definizione di `\cb`, il comando `\newcommand` con `\DeclareRobustCommand` (stessa sintassi). Infine, l'ultimo problema è quello relativo alla grandezza della virgola usata come segno diacritico. Molti font usati dai giornali mostrano il comma below più piccolo della virgola, come se fosse composto con un font inferiore. Ecco, `combelow 0.99a` non è in grado di farlo senza dare problemi in composizione.

6 combelow 0.99b

Tenuto conto delle osservazioni fatte nel paragrafo 5 la versione 0.99b diventa così:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{combelow}
3 [2009/08/23 v0.99b comma below
  command]
4 \RequirePackage{graphicx}
5 \RequirePackage{ifthen}
6 \newdimen\fcxpos
7 \newdimen\fcypos
8 \newdimen\ftpos
9 \DeclareRobustCommand{\cb}[1]{%
10 \setbox1=\hbox{#1}%
11 \setbox2=\hbox{,}%
12 \fcxpos=.5\wd1%
13 \advance\fcxpos by .5\wd2%
14 \ftpos=.5\wd1%
15 \advance\ftpos by -.5\wd2%
16 \ifthenelse{\lengthtest{\dp1>
17 .1\ht1}}{%
18 {\fcypos=\ht1%
19 \advance\fcypos by -.7ex%
20 \raisebox{0pt}{#1\hspace
21 {-\fcxpos}\raisebox{\fcypos}
22 {\hspace{\ftpos}}}%
23 {\fcypos=\dp1%
24 \advance\fcypos by .4ex%
25 \raisebox{0pt}{#1\hspace
26 {-\fcxpos}\raisebox{-\fcypos}
27 ,\hspace{\ftpos}}}%
28 }
29 \endinput

```

La differenza di resa sulla *g* corsiva è notevole, e visibile nella figura 6. Nel codice la differenza sostanziale sta nelle righe 19–22.

7 combelow 0.99c: un fantastico passo indietro

Su consiglio di Claudio Beccari, promosso sul campo mio mentore in quest'avventura (partita come autarchica, tanto orgogliosamente quanto avventa-

Mutu: „Amenda îmi afectează viața privată“

„În afara terenului nu sunt deloc în apele mele, nici nu pot să fiu. Acea amendă este de un lucru particular care îmi afectează viața privată... Totuși, când intru pe teren este ca o ușurare su-	fletească“, a precizat Mutu la postul de televiziune italian Sky. Tribunalul de Arbitraj Sportiv (TAS) a anunțat, la 31 iulie, că a respins memoriul formulat de Adrian Mutu la decizia Camerei	pentru Rezolvarea Disputelor (DRC) din cadrul FIFA, care a hotărât, la 7 mai 2008, că jucătorul trebuie să plătească clubului Chelsea despăgubiri în valoare de 17 milioane de euro. Că-	Rezolvarea Disputelor (DRC) din cadrul FIFA a decis, la 7 mai 2008, că Mutu, acum la AC Fiorentina, trebuie să plătească fostului său club, Chelsea, despăgubiri în valoare de 17.173.990 de euro, pen-	tru încălcarea contractului, iar internaționalul român a cerut TAS să anuleze hotărârea FIFA și să stabilească faptul că nu trebuie plătită nicio compensație.
--	---	--	---	--

FIGURA 4: Stralcio di testo in romeno. Il segno diacritico usato è una cediglia, che consente la sillabazione (vedi la parola *internaționalul* in quinta colonna). Testo tratto da Gazeta Românească Nr. 32, An VII. Sono state corrette le virgolette del testo originale (") con quelle per la lingua romena („ “)

Mutu: „Amenda îmi afectează viața privată“

„În afara terenului nu sunt deloc în apele mele, nici nu pot să fiu. Acea amendă este de un lucru particular care îmi afectează viața privată... Totuși, când intru pe teren este ca o ușurare su-	fletească“, a precizat Mutu la postul de televiziune italian Sky. Tribunalul de Arbitraj Sportiv (TAS) a anunțat, la 31 iulie, că a respins memoriul formulat de Adrian Mutu la decizia Camerei	pentru Rezolvarea Disputelor (DRC) din cadrul FIFA, care a hotărât, la 7 mai 2008, că jucătorul trebuie să plătească clubului Chelsea despăgubiri în valoare de 17 milioane de euro. Că-	Rezolvarea Disputelor (DRC) din cadrul FIFA a decis, la 7 mai 2008, că Mutu, acum la AC Fiorentina, trebuie să plătească fostului său club, Chelsea, despăgubiri în valoare de 17.173.990 de euro, pen-	tru încălcarea contractului, iar internaționalul român a cerut TAS să anuleze hotărârea FIFA și să stabilească faptul că nu trebuie plătită nicio compensație.
--	---	--	---	--

FIGURA 5: Stesso testo in romeno. Il segno diacritico usato è un comma below, che purtroppo non consente la sillabazione nel caso analogo: “internaționalul” esce fuori margine

ĝ ĝ

FIGURA 6: La ĝ corsiva resa con la versione 0.99a (a sinistra) e 0.99b (a destra)

tamente; povero illuso), una versione intermedia non poteva non tenere conto del codice plain TEX usato da Knuth per posizionare gli accenti.

Dal file `plain.tex` ho preso il codice di `\c #1` e, prima di procedere similmente a Beccari, ho provato `\accent`. Mai l'avessi fatto, pur sapendo-

lo essere solo un esperimento. Il codice seguente, molto compatto, produce un output improponibile (figura 7):

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{combelow}
3 [2009/08/23 v0.99c comma below
  command]
4 \DeclareRobustCommand{\cb}[1]{%
5 \setbox\z@\hbox{#1}%
6 \ifdim\dp\z@=\z@{\accent44 #1}%
7 \else{\accent96 #1}%
8 \fi}%

```

g k l n r s t

FIGURA 7: Il comma below prodotto con la primitiva `\accent` è insoddisfacente per colpa del posizionamento verticale

9 `\endinput`

Non c'è possibilità di controllare il posizionamento verticale, in questo modo. Se l'accento fosse disegnato appropriatamente non ci sarebbe questo problema, e il codice sarebbe perfetto. Un metodo del genere è dipendente dalla codifica. In caso nella codifica scelta ci fossero caratteri diversi da ' , ' in posizione 44 e ' ' ' in posizione 96, sarebbe diverso anche il carattere accentato risultante.

8 `combelow 0.99d`

L'ultima versione prodotta è molto più simile al codice di Beccari, ed è l'altra strada di `plain` relativamente al codice di `\c`. Il codice è molto compatto ma non elimina ogni eventuale problema di crenatura, così come la versione immediatamente precedente. Il sorgente è:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{combelow}
3   [2009/08/23 v0.99d comma below
   command]
4 \DeclareRobustCommand{\cb}[1]{%
5   \setbox\z@\hbox{#1}%
6   \ifdim\dp\z@=\z@
7     \oalign{\unhbox\z@\crrc
8       \hidewidth\lower.4ex
9       \hbox{,}\hidewidth}%
10  \else\oalign{\unhbox\z@
11    \crrc\hidewidth\raise.3ex
12    \hbox{'}\hidewidth}%
13  \fi}%
14 \endinput

```

ed è quello usato per produrre il testo della figura 5.

9 Sillabazione e crenatura

Abbiamo accennato al problema di sillabazione scaturente dall'uso di `combelow` nel paragrafo 5.2. TUG UK (2009), e soprattutto BECCARI (2009), dedicano alcune pagine all'argomento; ne riassumiamo qui i punti salienti.

L'algoritmo di sillabazione di TEX non sillaba se trova un comando `\accent`. La cosa si può evitare per i pacchetti di macro di TEX usando un font codificato adeguatamente, ad esempio T1 di `fontenc` nel caso di LATEX (se codifichiamo il testo nella figura 4 in OT1 anziché T1 otteniamo un risultato uguale, a meno del segno diacritico, a quello della figura 5, che dimostra la veridicità di quanto appena detto). Questo succede perché in questa codifica

(e in altre, ma non OT1 a 128 caratteri) una lettera accentata è mappata nel glifo corrispondente, e questo permette all'algoritmo di procedere alla sillabazione. In base a KNUTH (2001) (pag. 46), le legature non soffrono di questo problema, e non confondono l'algoritmo di sillabazione di TEX .

La speranza di eliminare ogni problema legato alla codifica è riposta nei candidati successori di TEX (ad esempio LuaTEX), basati su Unicode. Ciò dovrebbe dare la ragionevole certezza che ogni carattere LATEX (almeno quelli esistenti in uno o più alfabeti) sarà mappato in un glifo valido e con tutte le informazioni relative alla sillabazione e alla crenatura.

Una soluzione alternativa, sempre propostami da Claudio Beccari, prevede l'uso dei font virtuali. Questi forniscono un mezzo per generare un singolo glifo mettendo insieme altri glifi, regole e comandi di composizione e informazioni per posizionare il tutto, in modo da specificare esattamente come tutto viene assemblato. Per iniziare a saperne di più sui font virtuali si può fare riferimento all'articolo di KNUTH (1993) ad essi relativo.

Un discorso simile può essere fatto per la crenatura, di cui il carattere composto non ha informazione. Se una codifica avesse le lettere con comma below, TEX potrebbe fare crenature esattamente come quelle delle lettere con la cediglia in codifica T1.

10 Carattere vietato in Unicode

Dopo aver scritto la prima bozza del volantino di cui al paragrafo 1, e prima di decidere di scrivere un pacchetto, ho pensato: «Perché non inserire il carattere adatto da un editor Unicode?». Detto, fatto. Ho compilato il volantino con l'opzione `utf8x` di `inputenc`, ma non ho ottenuto il risultato sperato.

Per ammissione dei due autori di MITTELBACH e ROWLEY (2006), il pacchetto in questione fornisce un supporto Unicode che non è tale. Di fatto fornisce un meccanismo per leggere i caratteri multibyte e una tabella di corrispondenze tra caratteri multibyte e comandi LATEX per ottenere i caratteri corrispondenti. Ecco perché non si ottiene alcun carattere correttamente accentato nel caso allo studio: semplicemente perché ancora non esiste, né esiste un simbolo almeno corrispondente all'accento comma below.

11 Conclusioni

Questo pacchetto, nella sua versione più robusta, potrebbe espandersi nel tempo e diventare un contenitore di tutti i segni diacritici "ignorati" da LATEX (un nome possibile è `diacrit`). Già ora si può (e si dovrebbe) modificare `babel` per usare `combelow`. In realtà proprio `babel` dovrebbe integrare il comando proposto, e gli eventuali altri di futura

realizzazione, per caricarli in base alla lingua selezionata, eventualmente sostituendo i comandi di default: se il comma below è una cediglia, si faccia in modo che `\c` venga ridefinito con il contenuto di `\cb`.

12 Sviluppi futuri

Nell'immediato futuro il pacchetto deve subire le seguenti modifiche, intenzionalmente migliorative: usare i comandi `LATEX \DeclareTextAccent` e `\DeclareTextCompositeCommand` per generare le lettere accentate, e trovare un modo (l'ideale sarebbe un algoritmo generale) per decentrare il segno diacritico in base alle lettere da accentare (come nel caso della 'r').

13 Ringraziamenti

Per questo lavoro devo ringraziare, per chi ancora non l'avesse capito, Claudio Beccari per le sue osservazioni, i suoi suggerimenti e curiosità. L'articolo non sarebbe stato molto significativo senza tutte le informazioni che mi ha proposto di studiare e aggiungere.

Un altro ringraziamento va agli anonimi revisori che hanno ulteriormente migliorato la qualità del lavoro.

Riferimenti bibliografici

- BECCARI, C. (2009). *Introduzione all'arte della composizione tipografica con L^AT_EX*. G_UI_T, Pisa. URL <http://www.guit.sssup.it/download/#manual>. Versione 0.82.
- BRAAMS, J. (2006). *Babel, a multilingual package for use with L^AT_EX's standard document classes*. URL <http://www.ctan.org>.
- ELJKHOUT, V. (1991–1993). *T_EX by Topic. A T_EXnician reference*. Addison-Wesley, UK. URL <http://www.ctan.org/tex-archive/info/texbytopic/>. L'autore ha riottenuto i diritti nel 2001, e da allora ha reso il libro disponibile sul web.
- KNUTH, D. E. (1993). «Virtual fonts: More fun for Grand Wizards». *TUGboat*, **11** (1). URL <http://www.tex.ac.uk/tex-archive/info/knuth/virtual-fonts>. Text version.
- (2001). «The T_EXbook». In *Computers & typesetting / A*, Addison-Wesley, Boston, MA.
- KORPELA, J. K. (2006). *Unicode explained*. O'Reilly, Sebastopol, CA.
- MITTELBAACH, F. e ROWLEY, C. (2006). *Providing some UTF-8 support via inputenc*. URL <http://www.ctan.org>.
- TUG UK (2009). «Faq». URL <http://www.tex.ac.uk/faq>.
- TYPO.CZ e DESIGNIQ.CZ (2009). «Diacritics project». URL <http://diacritics.typo.cz/>.
- UNICODE CONSORTIUM (2009a). «Code charts». URL <http://www.unicode.org/charts>.
- (2009b). «Combining Diacritical Marks». URL <http://www.unicode.org/charts/PDF/U0300.pdf>.
- (2009c). «Latin Extended A». URL <http://www.unicode.org/charts/PDF/U0100.pdf>.
- (2009d). «Latin Extended B». URL <http://www.unicode.org/charts/PDF/U0180.pdf>.
- WIKIPEDIA (2009a). «Cediglia». URL <http://it.wikipedia.org/wiki/Cediglia>.
- (2009b). «Cedilla». URL <http://en.wikipedia.org/wiki/Cedilla>.

▷ Gianluca Pignalberi
Free Software Magazine
g dot pignalberi at
freesoftwaremagazine dot com

Uso del comando `\write18` per comporre l'indice analitico in modo sincrono

*Claudio Beccari**

Sommario

Si mostra uno dei possibili usi del comando `\write18` per rendere l'indice analitico assolutamente sincrono con il testo del documento a cui esso si riferisce, evitando di svolgere le tre o quattro compilazioni asincrone di **makeindex** e **latex** oppure **pdflatex** abitualmente necessarie quando non è disponibile il comando `\write18`.

Abstract

One of the possible uses of the `\write18` command is described in order to compose the index in an absolutely synchronous way relative to the document it refers to. The three or four asynchronous runs of **makeindex** and **latex** or **pdflatex** are therefore avoided, as it is not the case when the command `\write18` is not available.

1 Introduzione

Chiunque abbia composto un indice analitico con **L^AT_EX** conosce benissimo l'inconveniente che si manifesta quando, dopo aver pazientemente marcato tutti i lemmi dell'indice, si dimentica di eseguire l'ultima esecuzione multipla dei programmi necessari alla composizione dell'indice aggiornato alla fine del documento. Normalmente bisogna comporre e ricomporre il documento finché le bozze non mostrano più la necessità di nessuna correzione; successivamente bisogna:

1. lanciare **latex** o **pdflatex** per la compilazione definitiva durante la quale vengono raccolti i lemmi per l'indice, le informazioni per la loro formattazione, il numero della pagina nella quale essi cadono; queste informazioni vengono scritte in un file con l'estensione `.idx` esattamente nell'ordine in cui sono state raccolte;
2. lanciare **makeindex**, eventualmente specificando un file di stile per la composizione dell'indice analitico, in modo da produrre la lista alfabetica dei lemmi, per i quali è specificato il loro "rango" e le pagine dove sono descritti; in realtà queste informazioni vengono scritte in un file con l'estensione `.ind` che contiene tutto

*Quest'articolo è stato tradotto e presentato per la pubblicazione su TUGboat. La versione inglese è ripubblicata per gentile concessione dell'autore e di TUGboat a pagina 111.

l'ambiente `theindex` entro il quale ogni lemma, a seconda del suo "rango", è introdotto con i comandi `\item`, oppure `\subitem` oppure ancora `\subsubitem` insieme alle informazioni per la sua composizione tipografica;

3. lanciare ancora **latex** o **pdflatex** che rielabora l'intero documento, comprensivo dell'indice analitico, introdotto nel file sorgente mediante un comando del tipo `\input{<filename>.ind}` o altri comandi equivalenti forniti da altri pacchetti di estensione o da file di classe.
4. Potrebbe essere necessario lanciare ancora **latex** o **pdflatex** se la classe del documento o le opzioni personali richiedono che l'indice analitico sia indicato anche nell'indice generale.

Ebbene, questi tre o quattro passi possono essere compiuti in un'unica esecuzione di **latex** o **pdflatex** se si usa il comando `\write18{<text>}`.

Infatti le distribuzioni moderne del sistema **T_EX** contengono un file eseguibile e file di formato compilati con opportune opzioni cosicché l'interprete **pdfetex** possa produrre un file di uscita in formato `.dvi` oppure `.pdf`; può usare gli ulteriori comandi dell'interprete esteso **etex** oppure ne può essere interdetto; può eseguire un programma esterno (o può esserne impedito) che consenta al sistema di eseguire un compito particolare per poi, esaurito questo compito, riprendere a lavorare normalmente per continuare la composizione tipografica.

Qui si userà quest'ultima capacità in modo da produrre l'indice analitico effettivo, eseguendo una sola volta il compito dell'interprete dei comandi **L^AT_EX**.

2 I file ausiliari

Quando il comando `\makeindex` viene eseguito durante l'elaborazione del preambolo del documento, viene definito e aperto un nuovo flusso di uscita al fine di scrivere le informazioni per l'indice analitico in un file ausiliario `.idx`; nello stesso tempo il comando rende attivi i vari comandi `\index` sparsi lungo il documento i quali sono i responsabili della scrittura dell'informazione suddetta nel file ausiliario. Il nome di questo flusso di uscita vie-

ne conservato dentro la sequenza di controllo del nucleo di LATEX `\@indexfile`¹.

Questo file ausiliario ha l'estensione `.idx`, ma il suo nome è generalmente lo stesso del file principale `.tex` che genera il documento composto. Il nome di questo file principale è quello del “job” e viene conservato nella sequenza di controllo `\jobname`, che può venire usata da qualunque altra macro che faccia al caso.

Poiché l'indice analitico è verosimilmente l'ultima cosa che viene composta nel documento, il comando `\input{\jobname.ind}` è presumibilmente l'ultimo prima della specificazione `\end{document}`.

Perciò al fine di produrre il file `\jobname.ind` a partire da `\jobname.idx`, subito prima di introdurre con `\input{\jobname.ind}` il file dell'indice analitico alfabetizzato, bisogna eseguire due semplici operazioni:

1. Chiudere il flusso di uscita identificato dalla sequenza `\@indexfile`, ed
2. eseguire il programma ***makeindex*** al fine di produrre la nuova versione aggiornata del file `.ind`.

3 I comandi

Bisogna dunque eseguire *immediatamente* i due comandi seguenti:

```
\immediate\closeout\@indexfile
\immediate\write18{makeindex
-s style.ist \jobname.idx}
```

prima del comando `\input{\jobname.ind}` (o di ogni altra espressione equivalente, come per esempio `\printindex`). Ovviamente se non si usa nessuno stile particolare, ma si compone l'indice analitico secondo le impostazioni di default, allora bisogna omettere l'intera locuzione `-s style.ist`.

Il comando primitivo `\immediate` ordina all'interprete di eseguire immediatamente il comando che segue senza dilazionarlo. Infatti i comandi TEX connessi con i flussi di entrata e, specialmente, di uscita, inclusi i comandi per “leggere da” oppure “scrivere su” file esterni vengono solitamente dilazionati al momento in cui l'interprete usa la sua routine di uscita, a meno che questi comandi di “input/output” non siano preceduti dal comando `\immediate`.

1. Questo è quello che il nucleo di LATEX e le classi standard fanno. Alcune classi non standard possono usare altre sequenze di controllo per conservare l'informazione relativa al flusso di uscita per l'indice analitico: per esempio, la classe ***memoir*** usa un nome formato agglutinando il nome del “job” `\jobname` con il suffisso `@idxfile`, ma, siccome ***memoir*** può costruire diversi indici analitici, ognuno di questi ha un suo flusso di uscita con un nome diverso, il nome che viene specificato come argomento facoltativo del comando `\makeindex`.

Nei due comandi precedenti `\closeout` chiude il flusso di uscita, nel nostro caso quello specificato mediante `\@indexfile`.

Invece il comando `\write18{<text>}` scrive nel flusso di uscita speciale numerato “18”; questo flusso di uscita equivale a scrivere `<text>` in una riga di comando nella finestra comandi della famiglia dei sistemi operativi Windows; oppure nella finestra xterm dei sistemi operativi Linux; oppure nella finestra Terminal dei sistemi operativi Mac OS X. Quando l'applicazione esterna ha terminato di eseguire il suo processo di elaborazione, il programma ***latex*** o ***pdflatex*** riprende il controllo e prosegue dal punto esatto dove si era fermato per lanciare il programma esterno.

La presenza del segno @ nel nome del flusso di uscita `\@indexfile` richiede che si scriva il comando `\makeatletter` prima del primo comando e si scriva `\makeatother` subito dopo. Ma se quei due comandi vengono assegnati come sviluppo di una macro inserita in un file di macro personali `.sty` non c'è alcun bisogno di cambiare codice di categoria al segno @. Definire una macro personale con i due comandi suddetti è anche una buona occasione per aggiungere un po' di “fronzoli” utili: per esempio, per verificare che il flusso di uscita sia davvero aperto e possa quindi venire chiuso; per verificare che il file `\jobname.idx` esista per davvero al fine di elaborarlo; il secondo comando potrebbe avere un argomento facoltativo per consentire di specificare un eventuale file di stile `.ist` così da inserire o da omettere la locuzione concernente il file di stile nella chiamata del programma esterno; si possono aggiungere avvisi di cautela o messaggi d'errore per informare il compositore se c'è qualcosa di inconsistente. Non è qui il caso di insistere oltre; in fondo ogni utente del linguaggio LATEX dovrebbe essere in grado di personalizzare quei comandi secondo le sue necessità.

4 I risultati

Dando quei comandi (o il nuovo comando definito nel proprio file `.sty` personale) subito prima di emettere l'ordine `\input{\jobname.ind}` l'indice analitico viene a contenere esattamente i lemmi e i numeri di pagina raccolti durante l'unica esecuzione di ***latex*** o ***pdflatex***; cade la necessità di ricordarsi di eseguire le ultime tre o quattro esecuzioni ricordate nell'Introduzione, o di aggiornare l'indice dopo ogni correzione dei file sorgente, perché i lemmi e i numeri di pagina sono sempre aggiornati per costruzione.

Ritengo che questi semplici comandi siano utilissimi, e io stesso li uso sempre da quando ho “scoperto” questa possibilità.

Bisogna per altro segnalare che questa possibilità di far sì che ***latex*** o ***pdflatex*** lanci un programma esterno viene per così dire passata un po' sotto silenzio, perché sarebbe possibile far eseguire al si-

stema operativo programmi che potrebbero esporre il compositore al pericolo di eseguire inconsapevolmente del codice maligno; per questo motivo, addirittura, alcune distribuzioni del sistema T_EX non consentono di default di eseguire il comando `\write18`.

Ci sono però buone notizie in arrivo: la prossima distribuzione di T_EXlive 2009 conterrà un versione vincolata di `\write18`, nel senso che in ogni “job” sarà possibile usare questo comando “pericoloso”, ma solo per lanciare alcuni programmi specificati nel file di configurazione `texmf.cfg` e, fra questi programmi leciti, c’è anche ***makeindex***.

Sono certo che esistono molte altre applicazioni di `\write18`; per esempio, sono al corrente che la trasformazione sincrona del formato grafico dei file è già disponibile nel pacchetto `epstopdf.sty`; tocca agli utenti trovarne delle altre!

▷ Claudio Beccari
Villarbasce (TO)
claudio dot beccari at gmail
dot com

Una estensione di luatex: luatex lunatic

Luigi Scarso

Sommario

`luatex lunatic` è una estensione dell'interprete Lua di `luatex`, per consentire l'integrazione di un completo interprete `python`.

Intenzionalmente l'integrazione dell'interprete è limitata alla versione 2.6 di Python e a una versione di `luatex` per sistemi operativi Linux a 32 bit.

Abstract

`luatex lunatic` is an extension of the Lua interpreter of `luatex` to permit an embedding of a complete `python` interpreter.

Intentionally the embedding of interpreter is limited to `python-2.6` release and to a `luatex` release for 32-bit Linux operating system.

1 Introduzione

La parola “`TeX`” assume ai giorni nostri tre diversi significati distinguibili dal contesto:

1. **TEX**: sistema batch di composizione tipografica. Il sistema è disponibile per diverse architetture hardware e software nella ormai consolidata distribuzione `Tex Collection` liberamente disponibile al <http://www.tug.org/texcollection>;
2. **TeX**: un linguaggio di programmazione. È un *macro* linguaggio Turing-complete, la cui semantica riguarda la composizione tipografica;
3. ***tex**: un particolare interprete del linguaggio **TeX**; ve ne sono più d'uno. In generale, l'output di un programma in **TeX** è un formato binario `dvi` (DeVice Independent) oppure `pdf` (Portable Document Format). È da notare che lo standard ISO/IEC 32000-1:2008 riguarda proprio la definizione di un formato `pdf` informalmente noto come `pdf vers. 1.7`.

Data la sua natura, è ovviamente possibile scrivere ed usare pacchetti di macro in **TeX**; per agevolarne il caricamento a tempo di esecuzione da parte dell'interprete, grossi pacchetti vengono usualmente precompilati in un *formato* binario, solitamente indicati dal suffisso `fnt`.

Al giorno d'oggi è molto raro che venga usato il linguaggio **TeX** senza alcun formato: le macro primitive sono troppo a basso livello. Così sono nati pacchetti di macro via via più complessi: `plain`,

`eplain`, `AMS-TeX`, `LATeX`, `ConTeXt`... Naturalmente, un formato corrispondente ad un linguaggio complesso tendenzialmente è potente ma lento da caricare ed eseguire.

☞ Per brevità, con `ConTeXt` (ad esempio) si indica sia il linguaggio sia il formato.

Un secondo programma è praticamente sempre presente nelle installazioni: `mpost`, interprete di `METAPOST`, un linguaggio di tipo procedurale con semantica orientata al disegno vettoriale. È una modifica di `METAFONT`, un linguaggio procedurale con semantica orientata alla progettazione di font.

Esistono diversi pacchetti macro che permettono una stretta integrazione da **TeX** e `METAPOST`, basata nella maggior parte dei casi sul formato `dvi`.

Per quanto riguarda gli interpreti **TeX** è d'obbligo notare che il formato `dvi` viene praticamente usato solo internamente al sistema **TeX**, in quanto il `pdf` è lo standard de facto dell'industria della stampa (ed in misura minore il `PostScript`). Abbiamo, tra gli altri, i seguenti interpreti:

1. **tex**: output in `dvi`, necessita di un convertitore `dvips+ps2pdf`;
2. **etex** (*extendedTeX*): output in `dvi`, necessita di un convertitore `dvips+ps2pdf`;
3. **pdftex**: output in `dvi` e `pdf`;
4. **xetex**: output in `xdv` (*extendeddvi*), necessita di un convertitore `xdvipdfmx`;
5. **luatex**: output in `dvi` e `pdf`.

A volte è possibile trovare il termine **TeX** *the program* per riferirsi all'interprete e **TeX** *the language* per riferirsi al linguaggio-macro.

Ogni interprete viene utilizzato con un pacchetto di macro (solitamente `plain`) e quindi con il relativo formato per garantire un minimo di funzionalità; per i formati **L^ATeX** e **ConTeXt** abbiamo i seguenti interpreti:

- **L^ATeX**: `pdftex`, `xetex`, più raramente `etex`
- **ConTeXt**: `pdftex`, `xetex`, `luatex`; `etex` è praticamente scomparso.

La differenza tra `plain`, `eplain`, **L^ATeX** e **ConTeXt** si può sintetizzare così:

- **plain** è sostanzialmente il formato per il $\text{\TeX}$ book di Knuth: più precisamente **plain** costituisce l'ossatura, **manmac** contiene le macro per il layout che si "appoggiano" su **plain**. **eplain** è una estensione del formato **plain**, una prima generalizzazione di **plain**, non ai livelli di $\text{\LaTeX}$ però.
- $\text{\LaTeX}$ è un pacchetto di macro che nasce per fornire alcuni modelli di documenti (book, article, report, letter) ed è evoluto in modo non coordinato per aggregazione di altri *package*, anche se il nuovo $\text{\LaTeX}$ 3 tenta di rimediare a questo approccio
- Con $\text{\TeX}$ t nasce per affrontare *in generale* il problema della pubblicazione di contenuto, quindi fornisce macro generiche ad alto livello ed una ottima integrazione con METAPOST. Inoltre lo sviluppo è esclusiva di un ristrettissimo team (1-2 persone) ed i pacchetti di terze parti attualmente "vivi" (cioè utilizzati) sono non più di 5-6, in quanto i pacchetti base sono ampiamente sufficienti.

A scanso di equivoci, Con $\text{\TeX}$ t è un formato più lento di $\text{\LaTeX}$: per alcuni compiti (ad esempio typesetting di form) addirittura **eplain** potrebbe essere il formato più adatto, se non fosse in disuso.

Possiamo brevemente tracciare una evoluzione delle estensioni del linguaggio $\text{\TeX}$:

1. $\text{\TeX}$: nessun supporto per sorgente unicode, font type1, font truetype, opentype, direzionalità di composizione;
2. $\varepsilon\text{\TeX}$: nessun supporto per sorgente unicode, font type1, font truetype, opentype; si gioca sulla "e" per **extended** ma si trova anche ε , cioè "minima" nel gergo della matematica. $\varepsilon\text{\TeX}$ è una importante estensione del $\text{\TeX}$, ad esempio introduce la direzionalità di composizione (LR e RL);
3. pdf $\text{\TeX}$: nessun supporto per sorgente unicode, font opentype; nuove primitive per il supporto del formato pdf. Il supporto parziale di Unicode è assicurato dalla codifica utf8x;
4. X $\text{\TeX}$: supporto per unicode, font opentype;
5. Lua $\text{\TeX}$: supporto per unicode, font opentype, scripting in Lua

Possiamo tracciare una catena estensiva in $\text{\TeX}$ $\subset \varepsilon\text{\TeX} \subset \text{pdf}\text{\TeX} \subset \text{Lua}\text{\TeX}$; X $\text{\TeX}$ ha una genesi differente, anche se ora è a buon diritto un motore di impaginazione del $\text{\TeX}$.

Ad esempio pdf $\text{\TeX}$ include $\varepsilon\text{\TeX}$ che include $\text{\TeX}$, ma già con Lua $\text{\TeX}$ l'inclusione con pdf $\text{\TeX}$ non è completa — forse in futuro. Allo stesso modo il programma **pdftex** può essere chiamato in modalità **etex** (output in dvi), quindi in definitiva esiste un solo programma **pdftex** che riunisce 3 diversi interpreti **pdftex**, **etex** e **tex**.

Ancora, $\text{\TeX}$ può essere programmato per il parsing degli *ottetti* dello stream di ingresso, quindi rende teoricamente possibile gestire qualsiasi tipo di codifica, inclusa UTF-8. Quindi il supporto per unicode in codifica UTF-8 di fatto è una realtà almeno a partire da pdf $\text{\TeX}$.

Per riferimenti più puntuali è opportuno consultare l'ultima versione della GuidaGuIT (BEC-CARI), (fare riferimento al sito del $\text{\GjIr}$, perchè è aggiornata con una certa frequenza).

Gli interpreti **tex**, **etex**, **pdftex**, **xetex** sono codificati in un mix di **pascalWEB**, **C**, **C++**; **luatex** ha recentemente (Euro $\text{\TeX}$ 2009) completato la traduzione del codice da **pascalWEB** a **C** (e possibilmente **C++**) con il target **CWEB** (l'attuale linguaggio di programmazione di Knuth). Comunque, non si sbaglia di molto se si afferma che oggi la maggior parte degli utenti $\text{\TeX}$ usa pdf $\text{\TeX}$ + $\text{\LaTeX}$ seguito X $\text{\TeX}$ + $\text{\LaTeX}$ e pdf $\text{\TeX}$ +Con $\text{\TeX}$ t; $\varepsilon\text{\TeX}$ e $\text{\TeX}$ hanno un uso via via sempre più ristretto.

Lua $\text{\TeX}$ merita una menzione particolare:

- nasce come fork dal codice di pdf $\text{\TeX}$ (a volte si parla di **pdftex** 2.0);
- *per design* utilizza **lua** come *glue language* tra le librerie **C**, permettendo di scrivere macro in **lua** nel sorgente ***tex** ed in generale facilitando l'accesso allo stato interno del programma grazie ad una nutrita serie di librerie. Per far ciò Lua $\text{\TeX}$ ha al suo interno un interprete **lua** customizzato;
- viene sviluppato in sincronia con Con $\text{\TeX}$ t: in questo modo si implementano le *feature* più importanti per un immediato utilizzo/test da parte degli utenti esperti;
- è staticamente linkato con **mplib**, fornendo una completa integrazione con METAPOST. Parallelamente allo sviluppo di Lua $\text{\TeX}$ il team ha infatti deciso di riprendere il programma **mpost**, tradurre il codice completamente in **CWEB** e trasformarlo in una libreria **mpilib** (**mpost** utilizza ora la libreria **mpilib**) proprio per integrarla in Lua $\text{\TeX}$. Il team ha calcolato che per alcuni job il guadagno in termini di tempo è di circa 1000, cosa ragionevole se si pensa ai costi di setup di un task runtime.

Con Lua $\text{\TeX}$, il problema della presentazione del contenuto può essere "aggredito" su tre fronti: con un linguaggio macro come $\text{\TeX}$, con un ordinario linguaggio di programmazione come Lua e con un linguaggio grafico ad alto livello come METAPOST, tutti e tre altamente integrati.

Allo stato attuale nessuno degli interpreti $\text{\TeX}$ visti offre queste caratteristiche.

Il manuale di riferimento per programmare in Lua è IERUSALIMSKY (2006).

2 LuaTeX

Come detto nel paragrafo precedente, LuaTeX è sviluppato in sincronia con ConTeXt. Tuttavia, ConTeXt può usare come motore di impaginazione pdfTeX e XeTeX. Come è gestita la scelta del motore?

L'evoluzione di ConTeXt passa attraverso 4 *mark*-fasi con una certa continuità, cfr. HAGEN *et al.* (2009):

- **markI** : è il formato che usa (e)tex e pdfTeX, con comandi in olandese; praticamente scomparso, rimane qualche traccia in qualche pacchetto;
- **markII** (*mkii*) : è il formato attuale che usa (e)tex (uso non supportato), pdfTeX e xetex e che è *frozen*;
- **markIII** : è un termine riservato. Non corrisponde ad un formato e talvolta viene usato per indicare una possibile futura customizzazione di xetex, per separarlo da *mkii*, oppure per indicare una evoluzione di *mkii* che usa esclusivamente etex. Un'altra ragione del non-uso è un possibile fraintendimento con L^ATeX3, che è in intenso sviluppo;
- **markIV** (*mkiv*) : è il formato che usa esclusivamente luatex come motore. Condivide poco col formato *mkii*: anzi, il codice TeX per ConTeXt-*mkii* è indicato col suffisso **mkii*, quello per ConTeXt-*mkiv* è indicato col suffisso **mkiv* e **lua*.

Infine a runtime:

- `texexec nomefile` seleziona ConTeXt-*mkii* con pdfTeX;
- `texexec -xtx nomefile` seleziona ConTeXt-*mkii* con xetex;
- `context nomefile` seleziona ConTeXt-*mkiv* con luatex.

Quindi ConTeXt-*mkiv* è in sostanza il modo migliore per usare LuaTeX, ed è questo che verrà usato in questo articolo; LuaTeX con *plain* è sicuramente vantaggioso in termini di tempo per testare singole parti.

L'altra caratteristica è la possibilità, almeno sulla carta, di sfruttare l'interprete lua per estendere luatex a runtime. Lua mette a disposizione la funzione `package.loadlib()` che permette il caricamento dinamico di librerie **.so* o **.dll*: una volta caricata, una libreria appare come una normale *table* di Lua, quindi utilizzabile in modo trasparente — posto che si conosca l'API (Application Program Interface) della relativa libreria.

Ad esempio, sarebbe possibile caricare runtime la libreria *libgs.so* o *libgs.dll* ed interagire con l'interprete postscript di Ghostscript — ancora una volta, bisogna conoscere l'API relativa, deducibile dalla documentazione e soprattutto dal codice C di qualche programma.

Il più delle volte è necessario del codice Lua di interfacciamento: questo codice costituisce il Lua *binding* della libreria.

Così, proseguendo su questa scia, risulta comprensibile il lavoro di Gustavo Niemeyer nel suo “laboratorio personale” (NIEMEYER, 2009): **Lunatic python**.

Semplicemente, è un programma in C che permette di caricare la libreria *libpython.so* e di usare l'interprete Python tramite Lua, traducendo oggetti Python in oggetti Lua. È, per così dire, un “ponte” (*bridge*) tra Lua e Python; a volte si usa l'espressione “embedding” o “hosting”.

☞ **Lunatic python** è in realtà qualcosa di più: permette a Python di usare l'interprete Lua, quindi è un *two-way bridge*

luatex lunatic è quindi il *bridge runtime* tra LuaTeX e Python.

Ma è possibile costruire questo ponte?

E se sì, a quale prezzo?

Prima di rispondere a queste domande è bene sottolineare quanto segue: **è evidente che non è possibile parlare di luatex lunatic nel modo di cui sopra per gli altri interpreti TeX; detto in altro modo luatex lunatic è possibile solo grazie a luatex.**

3 Motivazioni e obiettivi

TeX è un sinonimo di “portabilità” (è facile implementare/adattare TeX *the program*) e “stabilità” (TeX *the language* viene modificato solo per correggere errori, anche se ormai è congelato e gli eventuali bug sono considerati caratteristiche).

Possiamo riassumere dicendo che “la composizione con TeX aspira ad essere ovunque e per sempre”.

Queste caratteristiche sono un po' inusuali nell'odierno scenario di sviluppo di applicazioni: nessuno si stupisce se un programma esiste per un solo sistema operativo (e a volte perfino per sistemi operativi non più sviluppati, date le varie tecniche di virtualizzazione) e soprattutto nessuno si stupisce dell'uscita di una nuova versione di un programma, il che significa, in effetti, correzione di errori e implementazione di nuove funzioni (da notare che l'inverso è di solito un fatto negativo: nessuna nuova uscita implica che lo sviluppo del programma è stato interrotto).

Naturalmente, se prendiamo in considerazione il sistema-L^ATeX, ovvero L^ATeX e i suoi pacchetti più usati, lo sviluppo di questo non è affatto congelato: basta controllare ogni giorno <http://www.mail-archive.com/ctan-ann@dante.de>. Inoltre pdfTeX viene costantemente modificato perché tenga il passo con le varie versioni del PDF.

Per luatex lunatic ho deciso di adottare questo punto di vista: ConTeXt-*mkiv* come strumen-

to per la pubblicazione di contenuti, con alcune funzioni di *content management* integrate.

In quanto strumento, non c'è da stupirsi se ci sono “frequenti” (per il mondo di T_EX) nuove uscite, che possono essere dovute a un aggiornamento di LuaT_EX, di ConT_EXt-mkiv, di Python, o di una libreria di cui Python fornisce i relativi *binding*. E, naturalmente, non c'è da sorprendersi che tutto ciò rischi di diventare rapidamente ingestibile, se non viene trattato *cum grano salis*.

Il prezzo da pagare è una potenziale **perdita di stabilità**: lo stesso documento (con gli stessi font e le stesse immagini) compilato con una versione più recente può produrre un risultato differente (ecco perché sarei contrario a farne una funzionalità fondamentale di LuaT_EX).

Per quanto riguarda la portabilità, per motivi dovuti a mancanza di risorse non rientra tra i miei progetti immediati sviluppare per sistemi operativi diversi da Linux; quindi è riscontrabile anche una potenziale **perdita di portabilità**.

Possiamo riassumere dicendo che “comporre con LuaT_EX-lunatic è qui e ora”, dove “qui” significa “per uno specifico sistema operativo” e “ora” significa “con questa versione”. In questo momento “qui” significa “Ubuntu Linux 32 bit” e “ora” significa `luatex-snapshot-0.42.0.tar.bz2` con ConT_EXt-mkiv `current` 2008.07.17; entrambi, probabilmente, saranno già sorpassati nel momento in cui quest'articolo andrà in stampa.

Naturalmente queste penalizzazioni devono essere inquadrare nell'ottica del sistema T_EX: ovvero *stabilità e continuità*. L'autore ha realizzato la prima versione di `luatex lunatic` poco più di due anni fa, e sostanzialmente le differenze con quello attuale sono minime e dovute al notevole cambiamento (questo sì) del codice di Luatex e del processo di building.

Durante lo sviluppo di `luatex lunatic` è emersa un'ulteriore motivazione: la possibilità di usare ConT_EXt-mkiv come una specie di strumento per il *literate programming* adattabile a uno specifico contesto, una sorta di *meta-literate programming*.

È noto che CWEB è un sistema per “intrecciare” insieme un programma scritto in uno *specifico* linguaggio (C, in questo caso) e la sua documentazione, redatta in un linguaggio di markup *semplice*, T_EX; ConT_EXt-mkiv può essere usato per “intrecciare” insieme un programma scritto in un *qualsiasi* linguaggio e la sua documentazione, redatta con un linguaggio di markup *altamente estensibile*, ConT_EXt-mkiv.

Detto in maniera diversa: oggi un'applicazione “chiama” T_EX o L^AT_EX (cioè crea un processo) per ottenere un risultato a partire da un frammento di codice `tex` (per esempio una formula matematica); invece ConT_EXt-mkiv si estende runtime “incorporando” l'applicazione tramite caricamento dinamico (senza creare cioè un nuovo processo) per



FIGURA 1: Etichetta con codice a barre.

ottenere il risultato da inserire nel flusso di token analizzato da `luatex`.

Per esempio, durante la redazione di un testo di matematica, l'autore può usare ConT_EXt-mkiv per comporre e *valutare* una formula matematica, tramite un *python binding* con una opportuna libreria di calcolo simbolico (ne esistono diverse, alcune sono in puro python come `sympy`).

Infine, risulta interessante costruire dei formati dedicati per compiti specifici, che magari richiedono l'integrazione di una/due particolari applicazioni esterne, senza però il bisogno di caricare funzionalità di alto livello sul lato T_EX del sistema (METAT_EX in ConT_EXt-mkiv¹).

Si consideri ad esempio il problema di stampare etichette (dette moduli di produzione) come quelle di figura 1.

Si tratta essenzialmente di una tabella con un codice a barre e due o tre font (meglio se a spaziatura fissa), generalmente in bianco e nero. ConT_EXt-mkiv già contiene il necessario per le tabelle e perfino un meccanismo di *layer* per posizionare oggetti in coordinate (x,y); `luatex lunatic` può fornire quanto serve per il codice a barre con il *binding* per Ghostscript ed il programma `barcode.ps`. Non c'è nessun bisogno di colori, interazioni, indici o divisioni del testo, eventuali esigenze grafiche anche complesse possono essere affrontate con il binding con l'interprete Postscript e/o con METAPOST.

Un'altra applicazione interessante per questo tipo di formati può essere quella che concerne programmi come Sage o R. Questi, attualmente, aggiungono ai rispettivi costrutti di calcolo matematico o statistico alcune istruzioni per ottenere report formattati con L^AT_EX.

1. ConT_EXt permette di creare dei formati minimali di questo tipo a partire da `metatex.tex`. Citando dalla documentazione del file:

This format is just a minimal layer on top of the luaT_EX engine and will not provide high level functionality. It can be used as basis for dedicated (specialized) macro packages.

A format is generated with the command:

`luatools -make -compile metatex`

In fondo assomiglia molto all'idea del formato `eplain...`

Con *luatex lunatic* ha senso ribaltare l'ottica: creare dei formati MetaTEXSage e MetaTEXR ad-hoc per elaborare a runtime codice *tex* che contiene parti di codice R o Sage da interpretare. Questa prospettiva è interessante se abbinata ad un sistema CMS come Plone al fine di creare un "laboratorio didattico" online di matematica o statistica di alto livello (ricordiamo che R e Sage sono framework usati per ricerca in ambito accademico).

4 Cenni alla fase di building

Questa sezione, di natura tecnica, è dedicata ad una traccia di installazione e descrizione dei problemi incontrati. Presuppone una certa familiarità con lo sviluppo e compilazione di codice; come già accennato, la piattaforma utilizzata è Linux Ubuntu 8.04 LTS 32 bit su i386. Un buon punto di riferimento è il testo di LEVINE (2000) ed il sito di DREPPER.

La strategia è quella di installare il più possibile un sistema isolato, in modo analogo a Sage Math.

Fissata per convenienza la directory base `HOMEDIR=/opt/luatex/luatex-lunatic`

1. installiamo Python 2.6.1 in

`$HOMEDIR/Python-2.6.1`

```
$> ./configure \
--prefix=/opt/luatex/luatex-lunatic\
--enable-unicode=ucs4 \
--enable-shared \
$> export PATH=\
/opt/luatex/lunatic-python/bin:$PATH
```

2. installiamo `luatex-snapshot-0.42.0` in `$HOMEDIR/luatex-snapshot-0.42.0` e creiamo il link simbolico

```
$> cd $HOMEDIR
$> ln -s luatex-snapshot-0.42.0 luatex
```

Possiamo eventualmente compilare con `build.sh`, giusto per vedere se non ci sono intoppi.

Installiamo ConTEXt-mkiv tramite

`minimals` in `$HOMEDIR/minimals` ed attiviamola con

```
$> . $HOMEDIR/minimals/tex/setuptex
```

3. installiamo `lunatic-python`

```
$> cd $HOMEDIR
$> bzip2 -d lp:lunatic-python
```

modifichiamo `setup.py` per adeguarlo alla nostra installazione:

```
1c1
< #!/usr/bin/python
---
> #!/opt/luatex/luatex-lunatic/bin/python
14,16c14,16
< LUALIBS = ["lua5.1"]
```

```
< LUALIBDIR = []
< LUAINCDIR = glob.glob("/usr/include/lua*")
---
> LUALIBS = ["lua51"]
> LUALIBDIR = ['/opt/luatex/
    luatex-lunatic/
    luatex/build/texk/web2c']
> LUAINCDIR = glob.glob("../luatex/source/texk/
web2c/luatexdir/lua51*")
48a49
>
```

ed infine prepariamo il wrapper `python.lua`

```
loaded = false
func = package.loadlib(
"/opt/luatex/luatex-lunatic/lib/
python2.6/site-packages/python.so",
"luaopen_python")
if func then
    func()
    return
end
if not loaded then
    error("unable to find python module")
end
```

Ora, prima di installare, dobbiamo apportare le necessarie modifiche al codice per abilitare il dynamic loading.

Innanzitutto, dobbiamo permettere *anche* al nostro interprete python il dynamic loading, cioè di poter caricare librerie `*.so` a runtime; modifichiamo `loadlib.c`:

```
luatex/source/texk/web2c/
luatexdir/lua51/loadlib.c
69c69
< void *lib=dlopen(path, RTLD_NOW);
---
> void *lib=dlopen(path, RTLD_NOW|RTLD_GLOBAL);
```

Questo non è sufficiente:

il manuale `luatexref-t.pdf` (LUATEX DEVELOPMENT TEAM, 2009) dice espressamente a pag 23:

Dynamic loading of .so and .dll is disabled on all platforms.

“Abilitare” il dynamic loading non comporta rilevanti modifiche al codice: solo al file `liblua51.am`

```
luatex/source/texk/web2c/luatexdir/am/liblua51.am
12c12
< liblua51_a_CPPFLAGS += -DLUA_USE_POSIX
---
> liblua51_a_CPPFLAGS += -DLUA_USE_LINUX
```

ed al file `Makefile.in`

```
luatex/source/texk/web2c/Makefile.in
98c98
< @MINGW32_FALSE@am__append_14 = -DLUA_USE_POSIX
---
> @MINGW32_FALSE@am__append_14 = -DLUA_USE_LINUX
1674c1674
< $(CXXLINK) $(luatex_OBJECTS) $(luatex_LDADD)
$(LIBS)
---
> $(CXXLINK) $(luatex_OBJECTS) $(luatex_LDADD)
$(LIBS) -Wl,-E -uluaL_openlibs -fvisibility=hidden
-fvisibility-inlines-hidden -ldl
```

In particolare, `-luaL_openlibs` dice al linker di includere il simbolo `luaL_openlibs` anche se questo non è utilizzato da `luatex` standard, ed è proprio questo che permette in ultima analisi di poter utilizzare `loadlib`; l'altro switch `-ldl` permette il link con `libdl.so` per poter risolvere il simbolo `dlopen`. È evidente che gli sviluppatori hanno definito delle funzioni analoghe a `loadlib()` per poter permettere un binding selettivo statico, solo a tempo di compilazione, quindi senza la necessità di usare `dlopen`: si veda ad esempio `luaopen_mplib(L)` in `luastuff.c`:

```
luatex/source/tekk/web2c/
  luatexdir/luastuff.c
void luainterpreter(void)
  lua_State *L;
  L = lua_newstate(my_luaalloc, NULL);
  :
  /* our own libraries */
  luaopen_ff(L);
  luaopen_pdf(L);
  :
  luaopen_tex(L);
  luaopen_mplib(L);
```

L'ultima istruzione dice chiaramente che viene caricata `luaopen_mplib(L)` e linkata staticamente nell'eseguibile `luatex`.

Gli switch `-fvisibility=hidden` e `-fvisibility-inlines-hidden` ci introducono in un serio problema: la collisione tra simboli. Possiamo spiegarla con un esempio: `luatex` è linkato staticamente con `libpng.so vers 1.2.38`, la quale espone il simbolo `png_memcpy_check`, una funzione C. Supponiamo che a runtime l'interprete `python` esegua un binding con una diversa libreria `libpng.so 1.3` e che venga richiamato proprio il simbolo `png_memcpy_check`. Questo verrà risolto verso `libpng.so vers 1.2.38` ed il più delle volte questo causa `segmentation fault` e quindi `abort` del programma. Questo tipo di eccezione non si può gestire, ed è a runtime. Una soluzione consiste quindi nel nascondere i simboli di `luatex` e questo è piuttosto semplice: basta aggiungere al file `build.sh` di `luatex`

```
28a29,36
> CFLAGS="-g -O2 -Wno-write-strings
  -fvisibility=hidden"
> CXXFLAGS="$CFLAGS
  -fvisibility-inlines-hidden"
> export CFLAGS
> export CXXFLAGS
```

Il problema è che, così facendo, anche `loadlib()` viene resa inefficace; quindi bisogna "svelare" i simboli relativi a Lua.

La soluzione si può tranquillamente classificare come una *dirty trick* di basso livello: salviamo l'output della compilazione in `out`:

```
$> cd $HOMEDIR/luatex; ./build.sh &>out
Localizzate le linee sensibili, togliamo il flag
-fvisibility=hidden: ovvero
```

```
gcc
-DHAVE_CONFIG_H -I.
-I../.../source/tekk/web2c
-I../...
-I/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/build/tekk
-I/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/source/tekk
-I../.../source/tekk/web2c/luatexdir/luastuff
-DLUA_USE_LINUX
-g -O2
-Wno-write-strings
-fvisibility=hidden
-Wdeclaration-after-statement
-MT
liblua51_a-lapi.o
-MD -MP -MF
.deps/liblua51_a-lapi.Tpo
-c -o
liblua51_a-lapi.o
'test -f
'luatexdir/luastuff/lapi.c'
|| echo
'../.../source/tekk/web2c/'
  luatexdir/luastuff/lapi.c
mv -f .deps/liblua51_a-lapi.Tpo
.deps/liblua51_a-lapi.Po
```

diventa

```
gcc
-DHAVE_CONFIG_H -I.
-I../.../source/tekk/web2c
-I../...
-I/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/build/tekk
-I/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/source/tekk
-I../.../source/tekk/web2c/luatexdir/luastuff
-DLUA_USE_LINUX
-g -O2
-Wno-write-strings
-Wdeclaration-after-statement
-MT
liblua51_a-lapi.o
-MD -MP -MF
.deps/liblua51_a-lapi.Tpo
-c -o
liblua51_a-lapi.o
'test -f
'luatexdir/luastuff/lapi.c'
|| echo
'../.../source/tekk/web2c/'
'luatexdir/luastuff/lapi.c
mv -f .deps/liblua51_a-lapi.Tpo
.deps/liblua51_a-lapi.Po
```

Ci sono circa una trentina di righe come questa, ed è un lavoro manuale di circa 20 minuti.

Dopo ricompiliamo `luatex`:

```
/bin/bash ./libtool
--tag=CXX
--mode=link
./CXXLD.sh -g -O2
-Wno-write-strings
-fvisibility=hidden
-fvisibility-inlines-hidden
-o luatex
luatex-luatex.o
libluatex.a libff.a
libluamisc.a
libzip.a
libluasocket.a
```

```
liblua51.a
/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/build/libs/
    libpng/libpng.a
/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/build/libs/
    zlib/libz.a
/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/build/libs/
    xpdf/libxpdf.a
/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/build/libs/
    obsdcompat/libopenbsd-compat.a
libmd5.a libmplib.a
lib/lib.a
/opt/luatex/luatex-lunatic/
  luatex-snapshot-0.42.0/build/texk/
    kpathsea/libkpathsea.la
-lm -Wl,-E
-ului_openlibs
-fvisibility=hidden
-fvisibility-inlines-hidden
-ldl
```

Siamo quindi pronti per *luatex lunatic*: copiamo l'eseguibile nella directory standard

```
$> cd $HOMEDIR/luatex/build/texk/web2c/luatex
$HOMEDIR/minimals/tex/texmf-linux/bin
```

e compiliamo

```
$> context --make
$> cp $HOMEDIR/luatex-python
$> python setup.py build \
&& python setup.py install
```

Un test semplice: salviamo

```
\directlua{require "python";
sys = python.import("sys");
tex.print(tostring(sys.version_info))}
\bye
```

in *test.tex* in una directory in cui è presente anche il wrapper *python.lua* ed eseguiamo

```
$> luatex --fmt=plain --output-format=pdf test.tex
```

5 Esempi

Assumeremo una installazione di *luatex lunatic* in *HOMEDIR=/opt/luatex/luatex-lunatic*. Il codice *TeX* non è inteso come didattico per *ConTEXt-mkiv*.

5.1 R

R “*is a free software environment for statistical computing and graphics*” (GENTLEMAN e IHAKA, 2009). Si sta rapidamente diffondendo negli ambienti accademici (COHEN e COHEN, 2008); esiste un binding in python, *rpy2* (GAUTIER, 2009) che possiamo installare nella nostra *\$HOMEDIR*.

Può essere necessario adeguare l'ambiente con

```
$>export R_HOME=/opt/luatex/luatex-lunatic/lib/R
$>export LD_PRELOAD=/opt/luatex/
  luatex-lunatic/lib/R/lib/libR.so
```

È conveniente distinguere la parte Python dalla parte *TeX* in questo modo: si salva il codice seguente in *test-R.py*

```
#test-R.py
import rpy2.robjects as robjects
import rpy2.rinterface as rinterface
class density(object):
    def __init__(self,samples,outpdf,w,h,kernel):
        self.samples = samples
        self.outpdf= outpdf
        self.kernel = kernel
        self.width=w
        self.height=h
    def run(self):
        r = robjects.r
        data = [int(k.strip())
                for k in
                    file(self.samples,'r').readlines()
                ]
        x = robjects.IntVector(data)
        r.pdf(file=self.outpdf,
            width=self.width,
            height=self.height)
        z = r.density(x,kernel=self.kernel)
        r.plot(z[0],z[1],xlab='',ylab='')
        r['dev.off']()
if __name__ == '__main__':
    dens =
        density('u-random-int','test-001.pdf',10,7,'o')
    dens.run()
```

Da notare che si tratta di codice Python e si può testare a riga di comando — non è necessario *ConTEXt-mkiv*: questo ci preserva dal “fastidio” degli spazi, in quanto in Python il livello di indentazione è nella grammatica del linguaggio. La parte di presentazione ha una interfaccia in Lua, una in *TeX*, ed infine la parte Main; il codice Python *test-R.py* è visto come una libreria in Lua e caricato con *pyR = python.import("test-R")*:

```
%% test-R.tex
%%
%% Lua interface
\startluacode
function testR(samples,outpdf,w,h,kernel)
    require("python")
    pyR = python.import("test-R")
    dens = pyR.density(samples,outpdf,w,h,kernel)
    dens.run()
end
\stopluacode
%%
%% TeX interface

\def\plotdensitiy[#1]{%
\getparameters[R][#1]%
\expanded{\ctxlua{testR("u-random-int",
    "\Routpdf",10,7,"\Rkernel")}}}%
%%
%% Main TeX
\setupbodyfont[sans,14pt]
\starttext
\startTEXpage
\plotdensity[outpdf={test-g.pdf},kernel={g}]
\plotdensity[outpdf={test-r.pdf},kernel={r}]
\plotdensity[outpdf={test-t.pdf},kernel={t}]
\plotdensity[outpdf={test-e.pdf},kernel={e}]
\plotdensity[outpdf={test-b.pdf},kernel={b}]
\plotdensity[outpdf={test-c.pdf},kernel={c}]
\plotdensity[outpdf={test-o.pdf},kernel={o}]
```

```

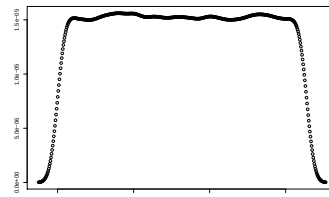
\hsize=400bp
This is a density plot of around {\tt 100 000}
random numbers between
$0$ and $2^{16}-1$ generated
from \type{/dev/urandom}
%
\externalfigure[test-g.pdf][width={400bp}]\par
{\ss gaussian}\par
\externalfigure[test-r.pdf][width={400bp}]\par
{\ss rectangular}\par
\externalfigure[test-t.pdf][width={400bp}]\par
{\ss triangular}\par
\externalfigure[test-e.pdf][width={400bp}]\par
{\ss epanechnikov}\par
\externalfigure[test-b.pdf][width={400bp}]\par
{\ss biweight}\par
\externalfigure[test-c.pdf][width={400bp}]\par
{\ss cosine}\par
\externalfigure[test-o.pdf][width={400bp}]\par
{\ss optcosine}
\stopTEXpage
\stoptext

```

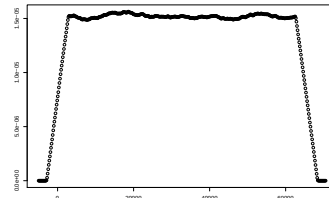
Il risultato si può vedere qui a fianco: sette grafici di densità di probabilità relativi alla stesso set di dati, ma con i sette diversi kernel (gaussian, rectangular, triangular, epanechnikov, biweight, cosine, optcosine) messi a disposizione da R.

Da notare che il codice di cui sopra fornisce il pdf delle dimensioni *esatte* per contenere grafici e testo: una semplice conversione in png o jpg rende il pdf immediatamente fruibile per altri contesti, come ad esempio una pagina html. Appare allora ragionevole pensare ad un processo che, ricevendo in ingresso un set di dati ed un set di kernel, “pubblici” tramite un server web il risultato (ad esempio il pdf; in realtà una sua conversione bitmap) *con qualità tipografica tipica del T_EX e affidabilità del risultato data da R.*

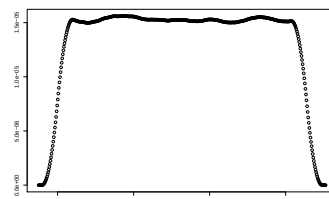
This is a density plot of around 100 000 random numbers between 0 and $2^{16} - 1$ generated from /dev/urandom



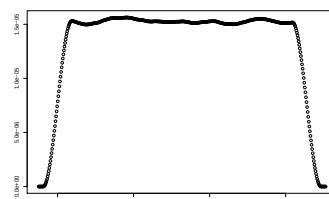
gaussian



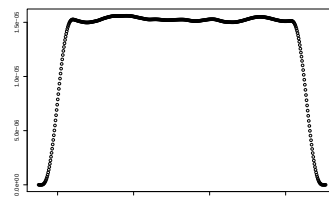
rectangular



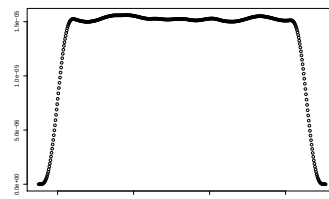
triangular



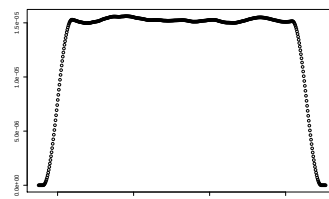
epanechnikov



biweight



cosine



optcosine

5.2 Sage

"Sage is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface. Mission: Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab." (SAGE). Sage è l'acronimo di Software for Algebra and Geometry Experimentation.

In questo caso il setup di *lunatic python* deve far riferimento all'installazione python di SAGE, per cui è necessario modificare in modo adeguato il file di setup e ricompilare *lunatic python*. Anche la variabile d'ambiente \$PATH deve essere modificata per includere python di SAGE.

Ecco il codice python:

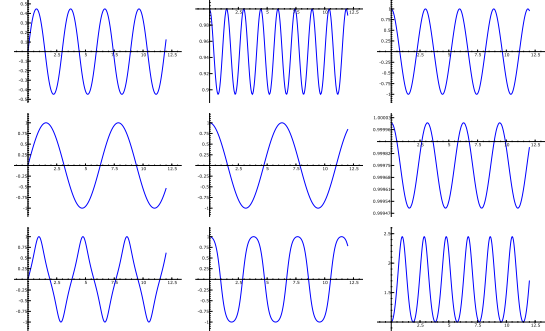
```
from sage.all_cmdline import *
def plot_eqn(diff_eqn,f0,f1,f2):
    T=ode_solver()
    T.algorithm="rk8pd"
    g_1 = eval("lambda t,y:%s" % diff_eqn)
    T.function=g_1
    T.y_0=[0,1,1]
    T.scale_abs=[1e-4,1e-4,1e-5]
    T.error_rel=1e-4
    T.ode_solve(t_span=[0,12],num_points=100)
    f = T.interpolate_solution()
    plot(f,0,12).show(filename=f0)
    f = T.interpolate_solution(i=1)
    plot(f,0,12).show(filename=f1)
    f = T.interpolate_solution(i=2)
    plot(f,0,12).show(filename=f2)
```

E questo è il codice T_EX:

```
\startluacode
function testODE(eqn,f0,f1,f2)
    require("python")
    ode = python.import("test-ode")
    ode.plot_eqn(eqn,f0,f1,f2)
end
\stopluacode
%
\starttext\startTEXpage
\hsize=12cm
\ctxlua{testODE(
    "[y[1]*y[2],-y[0]*y[2],-5*y[0]*y[1]]",
    "y0.pdf","y1.pdf","y2.pdf")}%
\ctxlua{testODE(
    "[y[1]*y[2],-y[0]*y[2],-0.001*y[0]*y[1]]",
    "ya0.pdf","ya1.pdf","ya2.pdf")}%
\ctxlua{testODE(
    "[y[1]*y[2],-y[0]*y[2],5*y[0]*y[1]]",
    "yb0.pdf","yb1.pdf","yb2.pdf")}%
$ y_0'=y_2 \cdot y_3 $
$ y_1'=-y_0 \cdot y_2 $
$ y_2'=\alpha \cdot y_0 \cdot y_1 $
$t=0$ and $t=12$, $\alpha \in \{-5,-0.001,5\}$ \\\
algoritmo \hbox{Runge-Kutta} \hbox{Prince-Dormand}.
\hbox{\externalfigure[y0.pdf] [width=4cm]}
\externalfigure[y1.pdf] [width=4cm]%
\externalfigure[y2.pdf] [width=4cm]}
\par
\hbox{\externalfigure[ya0.pdf] [width=4cm]}
\externalfigure[ya1.pdf] [width=4cm]%
\externalfigure[ya2.pdf] [width=4cm]}
\par
\hbox{\externalfigure[yb0.pdf] [width=4cm]}
\externalfigure[yb1.pdf] [width=4cm]%
\externalfigure[yb2.pdf] [width=4cm]}
\stopTEXpage\stoptext
```

Il risultato è la figura qui sotto: il plot delle 3 diverse soluzioni $(y_0, y_1, y_2)(\alpha)$ per $\alpha \in \{-5, -0.001, 5\}$:

$y_0' = y_2 \cdot y_3$
 $y_1' = -y_0 \cdot y_2$
 $y_2' = \alpha \cdot y_0 \cdot y_1$
 $t = 0$ and $t = 12, \alpha \in \{-5, -0.001, 5\}$
 algoritmo Runge-Kutta Prince-Dormand.



Anche in questo caso è evidente la possibilità di "esplorare" i risultati al variare di α (ad esempio), utilizzando un framework come Sage apprezzato dalla comunità scientifica.

5.3 Ghostscript

Con Ghostscript (DEUTSCH, 2009), noto interprete PostScript, introduciamo una interessante *feature*: l'utilizzo di *ctypes* di Python per la creazione di binding senza dover ricorrere a particolari framework come SWIG (BEAZLEY, 2009b). Nello specifico, l'autore ha abbastanza facilmente creato il wrapper *testgs.py* dal programma *gslib.c* incluso nella distribuzione di Ghostscript, esponendo solo le funzioni per convertire eps/ps in pdf.

```
\startluacode
function testgs(epsin,pdfout)
    require("python")
    gsmodule = python.import("testgs")
    ghost = gsmodule.gs()
    ghost.appendargs('-q')
    ghost.appendargs('-dNOPAUSE')
    ghost.appendargs('-dEPSCrop')
    ghost.appendargs('-sDEVICE=pdfwrite')
    ghost.InFile = epsin
    ghost.OutFile = pdfout
    ghost.run()
end
\stopluacode

\def\epstopdf#1#2{\ctxlua{testgs("#1","#2")}}
\def\EPSfigure[#1]{%lazy way to load eps
\epstopdf{#1.eps}{#1.pdf}%
\externalfigure[#1.pdf]%
}

\starttext
\startTEXpage
{\EPSfigure[golfer]}
{\ss golfer.eps}
\stopTEXpage
\stoptext
```

Il risultato è la classica figura *golfer.eps* che si vede alla pagina successiva: abbiamo *esteso* con un altro formato vettoriale il set delle figure gestite normalmente da LuaT_EX (png, jpg, pdf).



golfer.eps

A prescindere dall'ottimo testo CASSELMAN, una interessante applicazione è la libreria `barcode.ps` (BURTON, 2009):

```
\startluacode
function barcode(text,type,width,
                height,options,savefile)
    require("python")
    gsmodule = python.import("testgs")
    w = width / 25.4
    h = height / 25.4
    H = h / w
    prologue = string.format("%%%\n%s\n",'gsave')
    scale = string.format("%f %f scale\n",w,w)
    barcode_string = \
string.format("1 12 moveto (%s) \
(%s height=%f) %s barcode showpage",
text,options,H,type)
    epilogue = string.format("\n%s\n", 'grestore')
    psfile = string.format("%s.ps",savefile)
    pdffile = string.format("%s.pdf",savefile)
    temp = io.open(psfile,'w')
    temp:write(prologue .. scale ..
tostring(barcode_string) .. epilogue,"\n")
    temp:flush()
    io.close(temp)
    ghost = gsmodule.gs()
    ghost.rawappendargs('-q')
    ghost.rawappendargs(string.format(
'-dDEVICEWIDTHPOINTS=%f', w*72*2.3+4))
    ghost.rawappendargs(string.format(
'-dDEVICEHEIGHTPOINTS=%f', h*72*4+24))
    ghost.rawappendargs('-dFIXMEDIA')
    ghost.rawappendargs('-dNOPAUSE')
    ghost.rawappendargs('-sDEVICE=pdfwrite')
    ghost.rawappendargs(string.format(
'-sOutputFile=%s',pdffile))
    ghost.rawappendargs('barcode.ps')
    ghost.InFile= psfile
    ghost.run()
end
\stopluacode

\def\PutBarcode[#1]{%
\getparameters{bc}[#1%
\ctxlua{barcode("\bctext","\bctype","\bcwidth,
\bcheight, "\bcoptions","\bcsavefile" )}%
\expanded{\externalfigure[\bcsavefile]}%
```

```
}
\starttext
\startTEXpage
{\PutBarcode[text={CODE 39},type={code39},
width=37,height=20,
options={includecheck includetext},
savefile={TEMP1}]]}\
{\ss code39}
\blank
{\PutBarcode[text={CONTEXT},type={code93},
width=37,height=20,
options={includecheck includetext},
savefile={TEMP2}]]}\
{\ss code93}
\blank
{\PutBarcode[text={977147396801},type={ean13},
width=37,height=20,
options={includetext},savefile={TEMP3}]]}\
{\ss ean13}
\stopTEXpage
\stoptext
```



code39



code93



ean13

Vi sono ovviamente (e per fortuna!) numerosi altri siti da esplorare; ad esempio `scipy` (SciPy), oppure `networkx` (HAGBERG, 2009) per calcolo su network, `ROOT` (WIKIPEDIA, 2009) e (THE ROOT TEAM, 2009), il framework per l'analisi dei dati del CERN, il `Natural Language Toolkit`, “*Open source Python modules, linguistic data and documentation for research and development in natural language processing*” (NLTK) e (BIRD *et al.*, 2009). L'autore in SCARSO (2009) ha per esempio una demo di piccolo sistema basato su `Berkeley DB XML` (BRIAN, 2006), `XQuery` (WALMSLEY, 2007) Python e LuaTeX per estrarre e stampare articoli da un dump di `wikiversity` (wikimedia downloads) e (Wikiversity) — un interessante approfondimento di diversi argomenti.

Numerosi i libri su Python e *scientific computing* come LANGTANGEN (2009b) e LANGTANGEN (2009a), mentre per l'aspetto generale della pro-

grammazione certamente il prossimo LUTZ (2009) e l'ottimo reference BEAZLEY (2009a).

6 Conclusioni

Possiamo ritenere soddisfacente il set di patch al codice base, in quanto sono esigue e ben individuabili. La fase di building però è pesantemente influenzata dal problema del symbol collision, e certamente può essere migliorata. Tuttavia la fase attuale di intenso sviluppo non consiglia di proseguire ad ulteriori ottimizzazioni, in quanto potrebbero rivelarsi inutili: non è escluso infatti che `loadlib()` venga resa accessibile dal team di sviluppo, ed in questo caso dovranno essere risolti i problemi relativi alla portabilità (ad esempio quelli su piattaforma Windows/i386), alla stabilità (eccezioni non intercettabili tramite i meccanismi usuali di Lua) ed alla sicurezza (abilitazione di `loadlib` runtime, come `\write18`), decisioni che evidentemente l'autore non può prendere.

Non è sostenibile la posizione di usare Python per sostituire o completare in qualche maniera Lua nella sua funzione di interfaccia con `TEX`: Lua e `TEX` sono un perfect match. Lua è stabile, leggero, veloce, sviluppato in ambiente accademico, saldamente ancorato al C, con una comunità consistente. Non tutte queste caratteristiche sono condivise da Python: ad esempio, Python è più lento, più ingombrante e meno stabile — attualmente circolano le vers. 2.4, 2.5, 2.6, 2.7alpha e 3.1.1, e Google sta studiando una release intermedia tra 2.5 e 2.6.

Un binding di Python ad una libreria è importante solo se non esiste un analogo binding di Lua e non si ritiene conveniente costruirne uno. In definitiva un binding di Python è da privilegiare se è completo, ricco, stabile e mantenuto, oppure perché si utilizza massivamente Python anche in contesti completamente differenti (come ad esempio `SAGE`); altrimenti è opportuno pensare ad un Lua binding, magari ottenuto con `SWIG` (e potrebbe essere il caso di `R`).

È certamente vero che Python “offre” più di Lua, ma questa offerta deve essere calibrata bene, altrimenti si rischia di trasformare `ConTEXt-mkiv` in un tool molto potente che fa tutto — ma di cui non si ha un completo controllo.

Così ad esempio non deve essere sottovalutata la presenza di `METAPOST`: esistono numerosissime librerie di `METAPOST` che coprono praticamente ogni esigenza. Ad esempio, i grafi possono essere disegnati con `Metagraph` in `METAPOST`; se si vuole un binding in Lua per `graphviz` (`Graphviz`) si può usare LEUWER (2009). `NetworkX` o `igraph` (CSÁRDI e NEPUSZ, 2009) sono interessanti se bisogna eseguire *calcoli* sui grafi (per esempio il minimum spanning tree), non meramente per disegnare.

Ancora: nella libreria standard di Python viene fornito `sqlite` (HIPPI, 2009), un embedded relational database molto apprezzato dalla comuni-

tà. Quindi se si ha una sorgente dati in `sqlite`, `luatex lunatic` appare un buon candidato. Ma non è certamente una buona strategia utilizzare `sqlite` per gestire il multipassing, ad esempio; esistono già tecniche consolidate in `LATEX` e `ConTEXt`, e comunque la `table` di Lua è ottima anche per queste problematiche, senza contare la libreria `lpeg` di `ConTEXt-mkiv` — un parser per expression grammar, paragonabile alle Context-free Grammar.

`ConTEXt-mkiv` con `luatex lunatic` è uno strumento estremamente potente — *ma la potenza è nulla senza il controllo*.

7 Ringraziamenti

Desidero ringraziare Taco & Hans per questi primi 3 anni di `LuaTEX` — è questo il “qualcosa di completamente differente”.

Inoltre sono estremamente riconoscente a Massimiliano Max Dominici per aver tradotto in italiano una parte del mio articolo originale in inglese — un pessimo inglese, un'ottima traduzione. Quest'articolo è anche merito suo.

Infine un grazie al professore Claudio Beccari, per il suo lavoro passato e presente — e per quello futuro — ed a Gianluca Pignalberi per il suo paziente lavoro sulla mia bibliografia.

I currently use Ubuntu Linux, on a standalone laptop-it has no Internet connection. I occasionally carry flash memory drives between this machine and the Macs that I use for network surfing and graphics; but I trust my family jewels only to Linux.

— Donald Knuth

Interview with Donald Knuth

By Donald E. Knuth and Andrew Binstock

Apr 25, 2008

<http://www.informit.com/articles/article.aspx?p=1193856>

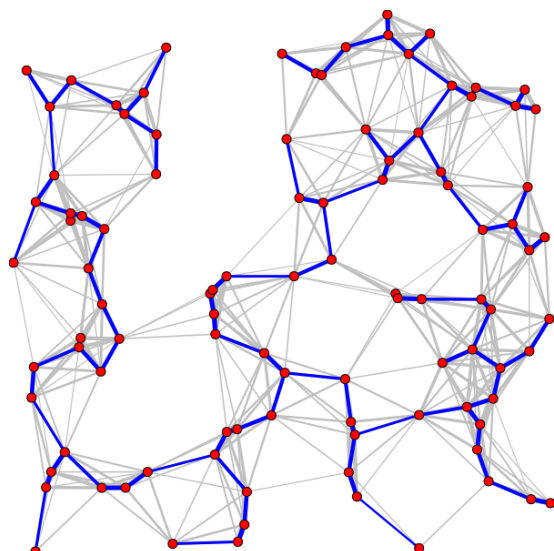
Cool. Thanks for letting me know about this. I've cc'd this to the Sage-devel mailing list.

William

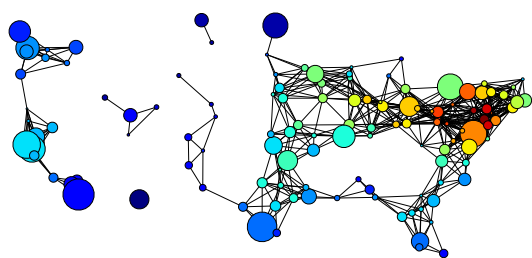
— William Stein Associate Professor of Mathematics
University of Washington

http://groups.google.com/group/sage-devel/browse_thread/thread/b4e9c3f6080e887d#

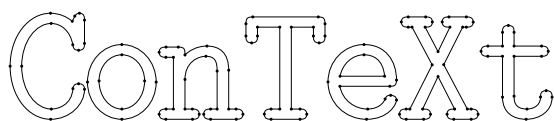
Just for fun (giusto per i fan...)



igraph — Minimum Spanning Tree



NetworkX — miles_dat.txt



fontforge — outlines

Riferimenti bibliografici

BEAZLEY, D. M. (2009a). *Python Essential Reference*. Addison-Wesley Professional, 4^a edizione.

— (2009b). «SWIG». URL <http://www.swig.org>.

BECCARI, C. *Introduzione all'arte della composizione tipografica con L^AT_EX*. URL www.guit.sssup.it/downloads/GuidaGuIT.pdf.

BIRD, S., KLEIN, E. e LOPER, E. (2009). *Natural Language Processing with Python — Analyzing*

Text with the Natural Language Toolkit. O'Reilly Media.

BRIAN, D. (2006). *The definitive Guide to Berkeley DB XML*. Apress.

BURTON, T. (2009). «Barcode Writer in Pure PostScript». URL <http://www.terryburton.co.uk/barcodewriter/>.

CASSELMAN, B. *Mathematical Illustrations A Manual of Geometry and PostScript*. URL <http://www.math.ubc.ca/~cass/graphics/text/www/index.html>.

COHEN, Y. e COHEN, J. (2008). *Statistic and Data with R*. Wiley.

CSÁRDI, G. e NEPUSZ, T. (2009). «The igraph library». URL <http://igraph.sourceforge.net>.

DEUTSCH, L. P. (2009). «Ghostscript». URL <http://ghostscript.com/>.

DREPPER, U. «How to Write Shared Libraries». URL <http://people.redhat.com/drepper/dsohowto.pdf>.

GAUTIER, L. (2009). «A simple and efficient access to R from Python». URL <http://rpy.sourceforge.net/>.

GENTLEMAN, R. e IHAKA, R. (2009). «The R Project for Statistical Computing». URL <http://www.r-project.org/>.

Graphviz (2009). «Graphviz - Graph Visualization Software». URL <http://www.graphviz.org>.

HAGBERG, A. (2009). «NetworkX». URL <http://networkx.lanl.gov>.

HAGEN, H., HENKEL, H. e HOEKWATER, T. (2009). «Lua_T_EX documentation». URL <http://www.lua.org/documentation.html>.

HIPP, D. R. (2009). «SQLite». URL <http://sqlite.org/>.

IERUSALIMSKY, R. (2006). *Programming-in-lua*. 2^a edizione. URL Lua.org.

LANGTANGEN, H. P. (2009a). *A Primer on Scientific Programming with Python*. Springer.

— (2009b). *Python Scripting for Computational Science*. Springer.

LEUWER, H. (2009). «LuaGRAPH - Reference Graph Programming with Lua». URL <http://luagraph.luaforge.net/graph.html>.

LEVINE, J. (2000). *Linkers & Loaders*. Morgan Kaufmann Publisher.

LUATEX DEVELOPMENT TEAM (2009). *LuaTEX Reference*. `luatexref-t.pdf`. Available in manual folder of `luatex-snapshot-0.42.0.tar.bz2`.

LUTZ, M. (2009). *Learning Python*. O'Reilly, 4ª edizione.

Metagraph (2009). «METAGRAPH: (un)directed graphs with METAPOST». URL <http://vigna.dsi.unimi.it/metagraph>.

NIEMEYER, G. (2009). «Lunatic Python». URL <http://labix.org/lunatic-python>.

NLTK (2009). «Natural Language Toolkit». URL <http://www.nltk.org/>.

SAGE (2009). «SAGE». URL <http://www.sagemath.org/>.

SCARSO, L. (2009). «User:Luigi.scarso/luatex lunatic». URL http://wiki.contextgarden.net/User:Luigi.scarso/luatex_lunatic.

SciPy (2009). «SciPy». URL <http://www.scipy.org/>.

THE ROOT TEAM (2009). «ROOT: A Data Analysis Framework». URL <http://root.cern.ch>.

WALMSLEY, P. (2007). *XQuery*. O'Reilly.

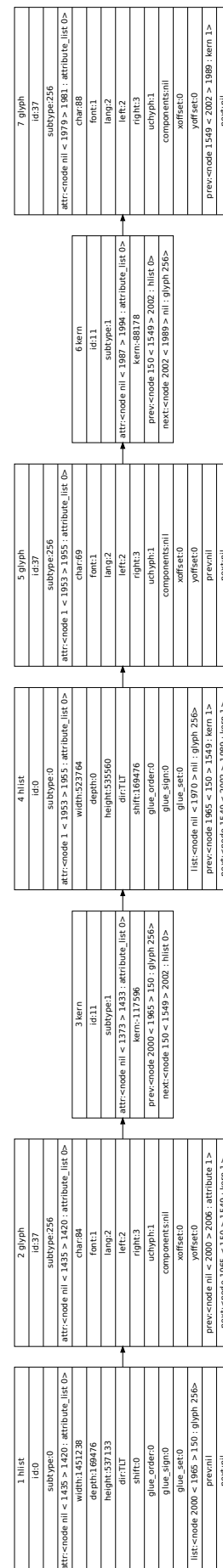
wikimedia downloads (2009). «Wikimedia Downloads». URL <http://download.wikimedia.org>.

WIKIPEDIA (2009). «ROOT». URL <http://en.wikipedia.org/wiki/ROOT>.

Wikiversity (2008). «Getting stats out of Wikiversity XML dumps». URL http://en.wikiversity.org/wiki/Getting_stats_out_of_Wikiversity_XML_dumps.

▷ Luigi Scarso
Logo S.r.l.

TEX



graphviz — nodelist

Il respawn di Infomedia ($\text{\LaTeX}$ -based)

Emmanuele Somma

Sommario

Infomedia, noto editore italiano di riviste di programmazione, rinasce anche grazie al software libero. Per comporre le sue riviste si affida a $\text{\LaTeX}$ e ad altri strumenti discussi nell'articolo.

Abstract

Infomedia, famous Italian publisher of programming magazines, has born again thanks to free software. To typeset its magazines it relies on $\text{\LaTeX}$ and some other tools discussed in the paper.

1 Infomedia e l'editoria per l'informatica

Per un breve periodo della mia vita, tra la laurea e il primo concorso come programmatore al Ministero della Giustizia, ho lavorato in una piccola azienda, con sede a Ponsacco vicino Pisa.

Era un'azienda un po' strana: piccola, pochi dipendenti, giovane. Il più anziano poteva avere più o meno la mia età, circa 25 anni. E il cui fondatore era più giovane di me.

Pubblicavano riviste. Le uniche riviste italiane di programmazione, al tempo, nate quando i magazine del settore decisero di mollare gli spazi in cui pubblicavano listati e programmi per buttarsi nel più ricco mercato dell'informatica aziendale, prove hardware, marchette commerciali e altra paccottiglia del genere. Mainstream ma noioso, a vederla con gli occhi di un programmatore.

Sì può certamente dire che intere generazioni di professionisti italiani trovavano in Computer Programming, DEV, Login e Visual Basic Journal (VBj) un'occasione privilegiata per imparare veramente cosa fosse un computer, conoscere come usarlo e mantenersi aggiornati.

Credo di averci lavorato non più di sei mesi. Con un piede dentro il processo editoriale e uno dentro il nascente settore dei servizi Internet. Quella Internet fatta ancora di modem, provider in teleselezione, dove non era raro trovarci ancora Gopher e Veronica.

In particolare mi occupai di un progetto, dopo la mia partenza (e di quella del fondatore, quasi contemporanee), mai più realizzato: la realizzazione di una 'community' degli sviluppatori italiani. Al tempo quasi non esisteva nemmeno la parola community!

Sei mesi che hanno cambiato la mia vita.

Ho continuato a collaborare con loro anche dopo il mio trasferimento a L'Aquila, e poi a Roma. Per un certo tempo mi occupai dell'impaginazione delle riviste online. Nel frattempo tante cose erano cambiate nel mondo informatico: erano cambiate per loro, erano cambiate per me. Rimase una grande amicizia ma poi, quando mi è capitato di proporre una nuova iniziativa editoriale su Linux, ho dovuto cercare nella concorrenza un terreno più fertile per le mie idee. È però rimasta l'amicizia di un tempo.

Per qualche anno ho portato avanti la mia rivista con l'altro editore, tra alti e bassi, talvolta discutendone più con gli amici di Ponsacco, sicuramente più focalizzati sulle prospettive professionali, che non con il mio editore, con una visione più generalista.

Poi un nuovo concorso, maggiori impegni, qualche dissapore con l'editore e... sì, è bello sapere che le iniziative che contribuiscono a realizzare, in un certo modo, sopravvivono anche senza di te. Che non hai perduto vecchi amici, ma te ne sei fatti di nuovi.

Ancora una volta il mondo informatico però era cambiato, ed era cambiato molto il modo di fare editoria tecnica. E io uscivo da un'iniziativa con un po' di esperienza in più in questo campo ma sinceramente pensavo di aver chiuso un capitolo. E per celebrarne questa soddisfacente chiusura spedii un lungo messaggio ai miei 'pari', cioè alcuni editor-in-chief di riviste italiane, che avevo avuto il piacere di conoscere nel tempo e che, come me, avevano vissuto relazioni non sempre edificanti con i loro editori.

Senza scendere nei dettagli il messaggio trattava della qualità delle relazioni interne alle redazioni, alla riduzione dei budget per le collaborazioni esterne e quindi alla nascita, mai vista prima, dei redattori interni 'generalisti' buoni a scrivere articoli su tutto e per tutto: i riempitivi. Lamentavo della nascita delle riviste-riempitivo per colpa della riduzione dei budget. Agli editor-in-chief miei amici scrissi invece dell'esperienza molto soddisfacente, e assolutamente anticipata dovuta al tema proprio della mia rivista, di ottenere articoli 'liberi', basati cioè su licenze di distribuzione 'open source' dalla 'Comunità'. Io stesso, fin dal 1998, avevo scritto in GNU FDL, dopo lunga discussione con l'editore, e poi compiuto una transizione alle neonate e poco diffuse Creative Commons.

Un passaggio di questa lettera trattava anche del rapporto con le fasi produttive, impaginazione e stampa, che da sempre è stata una delle croci che gli editor hanno dovuto accollarsi e che in generale

contribuiva a far decadere la qualità dell'intero processo editoriale per l'assoluta rigidità, e talvolta incompatibilità con le proprie stesse norme, degli impaginatori professionisti capaci di combinare qualsiasi disastro in articoli altrimenti perfetti. Sostenevo si potesse fare una rivista senza ricorrere agli impaginatori.

Nel frattempo l'anima commerciale delle riviste cresceva a causa della riduzione delle disponibilità aziendali, per la necessità di mantenere alti gli utili d'impresa e per il malinteso confronto con Internet che portava gli editori a rafforzare, invece che rilassare, le misure restrittive sul copyright. I contratti di edizione, di cui feci incetta tra gli amici in quel periodo, diventavano sempre più assurdi. Era veramente difficile firmarli a cuor leggero.

Si aggiunga a ciò quell'oscuro periodo in cui, nella generale riduzione degli investimenti pubblicitari sulla carta stampata, spiccavano in positivo le elargizioni di un'unica azienda, inutile far nomi, che contribuirono a drogare, e non poco, anche il contenuto editoriale.

In buona sostanza, era facile la previsione di una crescente disaffezione dei lettori per le riviste a contenuto tecnico. Non però per l'incombenza di Internet, quanto per il vero tracollo qualitativo dei contenuti. Solo una transizione verso altri modelli di business, probabilmente con minore componente commerciale, avrebbe potuto permettere il 'salvataggio' di una presenza editoriale in campo professionale, mutuando per quanto possibile le esperienze internazionali delle associazioni professionali del settore. Gli editori però avrebbero preferito, stanno preferendo, morire asfissati che iniziare questi processi.

Quando, dopo qualche anno, i miei vecchi amici di Ponsacco tirarono fuori dal cilindro tutte quelle considerazioni per verificare quanto fosse possibile riorganizzarsi in base a quelle considerazioni ormai era troppo tardi.

Il Gruppo Editoriale aveva già cessato la pubblicazione di tre delle cinque testate, l'emorragia di dipendenti e collaboratori era già molto avanzata e non si era chiaramente più in grado di portare avanti la produzione delle altre riviste.

Negli ultimi 15 anni Computer Programming, DEV, VBJ e Login avevano contribuito a far conoscere e definire il mondo della programmazione in Italia rendendolo popolare anche nelle scuole e tra i ragazzi. Alla loro chiusura il panorama italiano dell'editoria professionale nel campo della programmazione era diventato desolante.

L'editoria tradizionale sfoggiava, sul tema programmazione, praticamente un unico concorrente con una natura fin troppo commerciale, qualche altra rivista sfociava sul sistemistico, specie nel campo Linux.

In Internet invece erano proliferate una grande quantità di iniziative lodevoli (siti, blog, aggregato-

ri), alcune di buon livello qualitativo, nessuna delle quali però sembrava in grado di diventare un vero punto di riferimento per una comunità nazionale che, d'altro canto, non pareva avere una grande 'coscienza di ruolo', se non di classe. Inoltre duravano il tempo dell'autopromozione del coordinatore che, passato a impegni più gravosi, non riusciva a mantenere continuità alla produzione o a passarla ad altri.

Una rivista periodica sembra poca cosa: d'altro canto oggi Internet non ci dà accesso a risorse infinite e tutte alla portata di mano?

Quello che c'è alle spalle di una rivista periodica è un lavoro che andrebbe molto meglio considerato. Poi capita che una volta conosciuto non si può non apprezzare.

Gli obblighi imposti dalla periodicità fanno mantenere attiva una macchina di selezione ed adattamento dell'informazione che produce con continuità un contenuto 'abbastanza buono' (good enough).

Se i processi di selezione ed adattamento sono validi, non è difficile accorgersi del valore, anche economico, di quei contenuti perché, ad esempio, permettono ad alcuni lettori di risparmiare una gran quantità di tempo e risorse per 'tenersi in pari', non tanto nei campi in cui sono già competenti (che copriranno sempre meglio di quanto potrà fare, periodicamente, una rivista) ma tutti gli altri vicini o afferenti su cui da un lato non possono permettersi di rimanere ignoranti, dall'altro non possono sprecare una grande quantità di tempo anche solo per cercarsela, l'informazione.

Una rivista equilibrata e aggiornata permette inoltre sempre di tenere sott'occhio una possibile 'next big thing' senza la necessità di ingurgitarsi la lettura di migliaia potenziali fonti d'informazione, e su cui ogni volta fare la tara.

I contenuti 'good-enough', poi, hanno un incredibile valore per chi vuole avvicinarsi ad un mondo e diventano risorse preziose in un'ottica d'istruzione e di mentoring. Istruzione e mentoring che dovrebbero essere l'obiettivo proprio di una comunità professionale *seria*.

Non mi spingo oltre in queste considerazioni; sarò felice di discuterne eventualmente nell'ed-board di Infomedia.

La perdita di uno spazio di selezione e adattamento, come erano state le riviste Infomedia, è stata quindi significativa.

Erano venuti così meno gli unici luoghi italiani che negli anni avevano mostrato indipendenza e professionalità nel 'rappresentare' questo ambiente in modo aperto e popolare, senza dover ricorrere ai gradi e ai galloni accademici che non sempre, in Italia, hanno selezionato la qualità, ma soprattutto è venuta meno una costante attenzione nel trattare il non semplice tema della programmazione e della sistemistica senza essere trascinati troppo da



Sei capitato sul sito del Turing Club, la prima (e unica) [associazione](#) italiana dei programmatori professionisti, e appassionati di programmazione in Italia.

Ci ispiriamo ad [Alan Mathison Turing](#), perché tra i padri della computer science è stato il primo a capire che la nascita del calcolatore digitale avrebbe fatto sorgere una nuova leva di professionisti e che si sarebbe sviluppata una loro cultura e professionalità specifica. Turing realizzò il primo linguaggio e fu quindi senza dubbio il primo programmatore in senso moderno.

Poiché troviamo che la figura di Turing, in Italia, sia ricordata solo legata alla vasta aneddotica sulla sua vita tormentata, o un po' per i suoi importanti contributi scientifici, ma troppo poco rispetto al suo ruolo di innovatore delle strutture sociali e culturali, noi programmatori e sistemisti vogliamo segnare, con questa attribuzione, il ringraziamento imperituro a questa fondamentale figura anche della nostra cultura, e per la quotidianità della nostra vita.

Il Turing Club tiene alla promozione della cultura informatica, specie tra le giovani generazioni e per questo si impegna. L'informatica ha infatti cambiato la vita della nostra Società, ma troppo spesso si immagina che alle spalle di questa innovazione ci sia una sorta di incomprensibile magia. Questa "magia" non è che il frutto di giorni e notti di programmazione appassionata il cui valore sociale è ben superiore al guadagno che se ne ricava.

In una società come quella italiana poi, per ragioni anagrafiche e generazionali, per l'assenza di un sistema di credito avanzato e per la miopia della classe direttiva e politica, il talento, la competenza e l'innovatività dei programmatori piuttosto che essere premiata, viene giorno dopo giorno disconosciuta e marginalizzata, con il pratico risultato di spingere l'Italia nel Medioevo tecnologico.

Chiunque tu sia, quali che siano i tuoi strumenti di lavoro, il tuo linguaggio di programmazione, la tua piattaforma o framework, il tuo sito o il tuo programma... [questo](#) è il Turing Club



FIGURA 1: Messaggio "autopromozionale" del Turing Club

logiche commerciali legate ad altre attività business dell'editore. Non c'era più nessuno "sul pezzo" della programmazione.

Il Gruppo Editoriale Infomedia era riuscito fino all'ultimo a rimanere un 'editore puro', completamente focalizzato sul tema. E se da un lato gli si poteva imputare una certa inerzia nel prendere coscienza delle mutate condizioni del contesto e una certa inazione nei confronti del fenomeno Internet, dall'altra non si poteva non riconoscergli che era rimasta comunque fedele al proprio pubblico, mentre gli altri correvano dietro alla completa de-specializzazione, per rincorrere più alte vendite e sopportare le bizzarrie di un sistema di distribuzione in edicola che è un vero sconcio italiano. Ma soprattutto aveva negli anni prodotto una mole incredibile di informazione, dando voce ad oltre 800 autori, e realizzando molte migliaia di articoli tecnici, presentazioni di prodotti, tecnologie, protocolli, strumenti di lavoro, tecniche di sviluppo e semplici trucchi e trabocchetti, con oltre 6 milioni di copie distribuite e trentamila pagine stampate avevano fatto di questa impresa la più grande ed influente realtà dell'editoria specializzata nel campo della programmazione e della sistemistica.

Lo stesso rapporto con l'advertising, argomento a cui io personalmente ero molto sensibile essendo in effetti stato causa del raffreddamento dei rapporti con il mio precedente editore, era rimasto di gran lunga più sano che non in altri casi.

D'altronde l'impossibilità di mantenere le proprie attività era avvenuta non già per una riduzione degli abbonati, per la maggior parte clienti istituzionali, università e aziende, grandi e piccole, rimasti sostanzialmente costanti nei precedenti dieci anni, ma per i soliti problemi di liquidità in cui vengono a trovarsi piccole aziende del genere

strozzate dal sistema di credito, dovuti peraltro ad attività collaterali alla produzione delle riviste vere e proprie.

Alla perdita di questa realtà molti programmatori non hanno saputo e voluto adattarsi.

Già dai primi momenti, anche grazie alla disponibilità dei soci del precedente Gruppo Editoriale, che pure apparivano poco interessati a continuare con l'impresa, si è ragionato sul come permettere alla Comunità dei programmatori di non disperdere questo patrimonio. Promotore di questo tentativo un gruppo di programmatori e sistemisti, che in modo informale e talvolta conviviale e un po' goliardico, aveva al tempo già fatto proprie considerazioni circa l'importanza di rendere più popolare il 'pensiero informatico' nella società italiana: il Turing Club. Nella figura 1 è riportata la pagina web del manifesto del gruppo.

Non era però ragionevole pensare ad infondere nuova linfa nella vecchia azienda costruita su un modello alquanto tradizionale di editoria.

Il tentativo di una nuova Infomedia è stato quindi non tanto quello di trovare il modo di "continuare" tal quale, quanto piuttosto di "innovare" anche i metodi editoriali stessi con lo stesso talento che solitamente i programmatori infondono nelle loro iniziative.

2 Una nuova Infomedia

Molti pensano che inesorabilmente le riviste cartacee e addirittura il modello di produzione delle riviste periodiche sia tramontato per sempre di fronte all'innegabile successo di Internet.

Chi si è ritrovato attorno alla nuova Infomedia non la pensa così e ritiene che non solo esista ancora uno spazio, ma che la nuova Infomedia rappresenti un'occasione unica per mettere in atto modelli di

produzione mutuati dalle migliori esperienze della Rete.

La nuova Infomedia si è quindi organizzata attorno ad una idea di Impresa Sociale di Comunità, partecipata da programmatori e sistemisti, separando le attività di gestione dell'informazione realizzate da un board comunitario professionale e quelle di produzione gestite da una impresa strumentale (Infomedia Editori Coop) anch'essa non a fini di lucro.

Dall'inizio di quest'anno si è riuscito a riprendere le pubblicazioni di:

- CP Computer Programming (bimestrale)
- DEV developing software (bimestrale)

Coloro che si sono lanciati in quest'impresa, come volontari e per puro spirito di servizio, hanno ancora molto da imparare e i risultati sono certamente migliorabili, ma senza dubbio è già possibile vedere quanto sia importante il lavoro svolto fin qui.

È opinione comune dei lettori che hanno continuato a leggere le riviste che il livello qualitativo medio dei numeri fin qui prodotti sia cresciuto notevolmente. Per non parlare delle collaborazioni internazionali che abbiamo avuto il piacere di ricevere: Peter Norvig, Paul Graham, Richard P. Gabriel, Neal Gafter, Walter Bright, Joel Spolsky.

Tanto per citare le collaborazioni degli ultimi sei mesi.

Va notato il coinvolgimento che si sta iniziando a creare attorno all'iniziativa. Dal primo numero, gestito da un risicato manipolo e tre mesi di lavoro per la pubblicazione, agli ultimi, che stanno mantenendo le scadenze mensili alternate di due riviste. Tutto grazie alla contribuzione di una comunità composta da quasi cinquanta persone come editor, autori, revisori, sviluppatori, ma soprattutto come membri di un 'editorial-board' aperto e pubblico, che in un certo senso sta iniziando a guidare l'evoluzione dell'iniziativa (la figura 2 mostra il sito della nuova Infomedia).

3 La sfida

La sfida più grande di questo 'sistema' è rappresentata dal modello di open-publishing che si sta realizzando. Un modello che riesca a coniugare la contribuzione libera e aperta, secondo uno schema mutuato dalle migliori esperienze delle riviste scientifico-professionali, specie nel mondo anglosassone, e il livello qualitativo e le garanzie di continuità delle riviste commerciali. Con la prospettiva realistica di ritornare ad una distribuzione cartacea e con la velleità di superare, anche, la

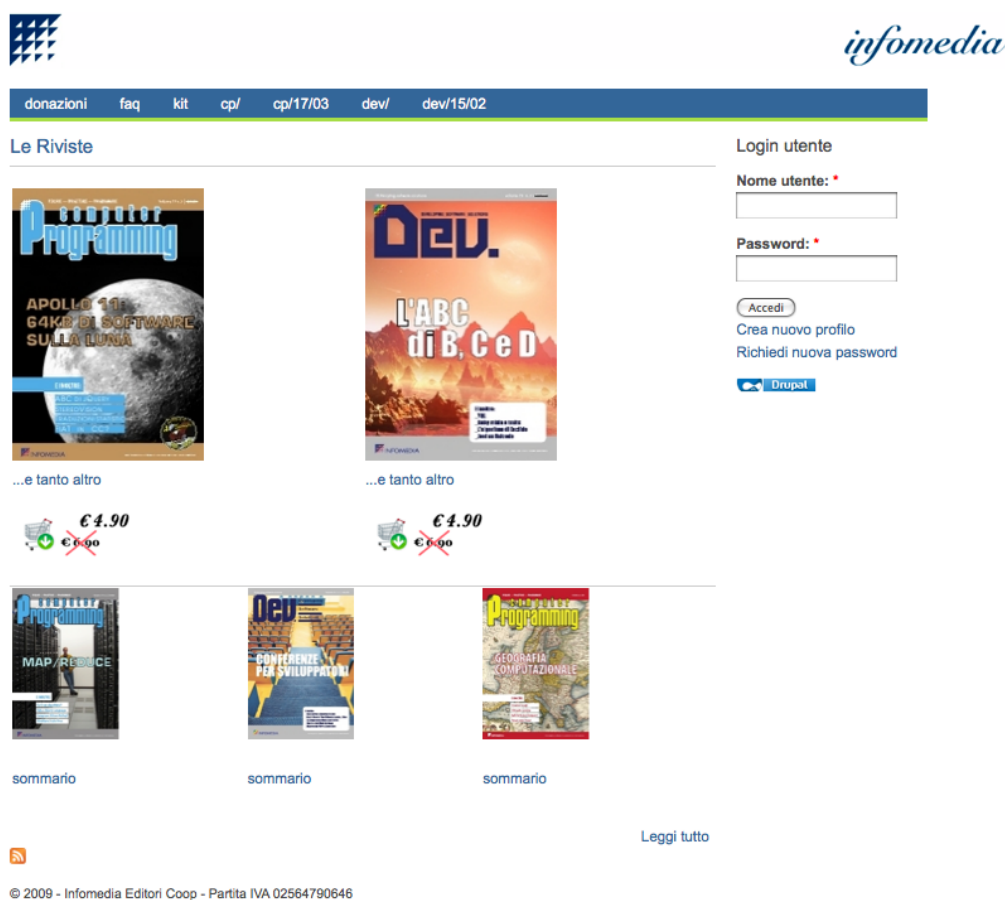


FIGURA 2: Home page del sito della nuova Infomedia

diffusione precedente, proprio grazie all'interazione con la comunità.

A ciò andrebbe aggiunta una seria riflessione circa la prospettiva della distribuzione dei contenuti in una modalità completamente open-access, attraverso licenze Creative Commons.

Ciò che però rende esclusiva questa esperienza, a parte le prospettive numeriche dei sottoscrittori e la diffusione nella pur piccola realtà italiana, sono gli strumenti adottati per realizzare la promessa di open-publishing, e quindi abbattere le barriere all'ingresso del lavoro editoriale e semplificare notevolmente sia la contribuzione di articoli sia il percorso di generazione del 'prodotto finale', ovvero il PDF pronto per la stampa cartacea.

In questo il sistema L^AT_EX gioca un ruolo molto rilevante. E vista la sua centralità gli deve essere tributato un plauso speciale proprio come tecnologia abilitante.

In effetti la produzione della nuova Infomedia non fa che seguire l'esperimento condotto dalla testata americana Free Software Magazine (FSM), costruita con una class L^AT_EX realizzata specializzando le classi `journal` e `paper` da Gianluca Pignalberi. L'articolo di Gianluca (PIGNALBERI, 2005b) spiega fin troppo bene i problemi a cui va incontro chi volesse impaginare una rivista con L^AT_EX.

Va subito detto che, pur nella sua veste internazionale, anche l'iniziativa FSM è in un certo senso figlia della 'vecchia' Infomedia. Infatti il direttore, Tony Mobily, è stato per un lunghissimo periodo collaboratore, socio e responsabile di alcune testate Infomedia. Oltre che, per me, amico fraterno.

Se si può parlare di una 'scuola Infomedia', sia io che Tony, e lui molto più di me, ci siamo sicuramente diplomati lì.

La nuova Infomedia deve quindi moltissimo a Tony Mobily e Gianluca Pignalberi: non solo tutta quella familiarità e amicizia che una comune passione può generare, ma proprio l'aver messo a nostra disposizione tutto il codice necessario e l'aiuto per comprenderlo ed imparare ad usarlo.

Sebbene nel frattempo la stessa FSM abbia abbandonato la produzione cartacea e anche l'impaginazione L^AT_EX, la nuova Infomedia ha invece raccolto il testimone e lo ha integrato in un sistema più completo, per affrontare il problema della produzione distribuita della rivista.

Sono tre le differenze principali introdotte nel sistema Infomedia (chiamato dagli editor 'making' poiché la directory principale di lavoro è così denominata):

- l'uso di un file di input in formato solo testo con markup semplificati (tipo Wiki o Textile) al posto del L^AT_EX stesso chiamato file `.tchl.txt`, e quindi di uno strumento di traduzione da `.tchl.txt` ad un apposito sottoinsieme di L^AT_EX realizzato in Emacs Lisp;

- l'adozione di un programma d'interazione command-line per l'esecuzione di tutti i comandi necessari alla composizione e, in parte, al publishing della rivista, realizzato in Python;
- l'uso di SVN, non solo nello sviluppo dei sorgenti del software realizzato e delle macro L^AT_EX, ma per la gestione complessiva del flusso editoriale, articoli, figure, elementi grafici e file di configurazione.

Il sistema realizzato ha ormai raggiunto un buon livello di maturità, sono stati realizzati già sei numeri delle riviste, per un totale di più di 350 pagine lavorate completamente con questo sistema.

4 I markup nel file `.tchl.txt`

Uno degli ambiti di miglioramento del sistema proposto da Free Software Magazine era senza dubbio il loro modello di redazione dei singoli articoli.

Per contribuire a FSM bisognava in definitiva costruire un file L^AT_EX, contenente un sottoinsieme di comandi molto limitato.

Tutti questi singoli file venivano poi collezionati in una sottodirectory comune e poi richiamati dal file principale con delle macro d'inclusione.

La composizione finale della rivista prevedeva quindi la realizzazione di un documento principale L^AT_EX, di classe `fsmjournal`, in cui fare l'input degli articoli ad uno ad uno attraverso specifiche macro. Era responsabilità dell'editor costruire tale file introducendovi gli articoli e tutti gli altri componenti necessari (editoriali, sommario, pagine pubblicitarie, ecc. ecc.).

Questo lavoro, quindi, non poteva che essere delegato ad un T_EXnico accorto, che intervenisse sugli articoli, sul file principale e talvolta sulle macro per risolvere 'creativamente' gli inevitabili problemi di impaginazione.

L'innovazione introdotta da TCHL è stato l'utilizzo di un formato testuale molto semplice per la redazione degli articoli. Non un L^AT_EX ridotto quanto piuttosto un formato testuale wiki agli stereotipi.

Il progetto BHL (<http://www.nongnu.org/bhl/>) di Bastien Guerry, già dal 2002 mirava a fornire all'utente Emacs un semplice sistema di redazione di documenti basato su un markup implicito e minimale, simile a quello usato nella redazione di un Wiki, che poi poteva essere automaticamente trasposto in formato HTML, SGML (Linuxdoc), Texinfo o L^AT_EX.

Avevo utilizzato una forma leggermente adattata di BHL per l'editing di Linux Magazine, accordandomi con il settore impaginazione a cui fornivo gli articoli in formato con un markup HTML minimale per il corpo dell'articolo, e un file testuale a parte contenente tutte le informazioni identificative dell'articolo (titolo, sottotitolo, autore, didasca-

lie, ecc.) da copia-incollare linea per linea nelle gabbie d'impaginazione. Una pazzia che solo un Cercopithecus-Impaginator poteva immaginare.

Il passaggio da questo alla compilazione diretta del file L^AT_EX preteso da fsmjournal fu fatto, ovviamente, in uno snap.

L'idea era quella di riuscire a 'nascondere' alla vista degli editor la presenza del L^AT_EX, non tanto per evitar loro di dover acquisire competenze in questo sistema, quanto per impedirgli di pasticciare con ambienti e comandi poco consoni alla composizione, vanificando il lavoro anche degli altri editor, potenzialmente al lavoro sulla stessa rivista in contemporanea.

Il vantaggio principale stava quindi nel cablare nel codice Lisp, una volta per tutte, le traduzioni testo→L^AT_EX, in particolar modo in relazione a quegli ambienti classici di una rivista del genere (figure, listati, box).

Una volta tradotto il markup file .tchl.txt in uno .tex, era poi possibile attraverso la linea di comando richiedere la composizione dell'articolo o, calato l'articolo in una struttura di rivista, ricompilare l'intera rivista.

Non c'è bisogno di scendere nei dettagli del markup perché, in realtà, quello adottato poco aggiunge a sistemi simili. Essendo stato realizzato prima di tanti altri è, ovviamente, incompatibile e pure un po' specializzato. Allo stato attuale lo consideriamo già un po' legacy e ci piacerebbe poterlo uniformare a qualcosa di più comune, in modo anche da poter riusare strumenti di traduzione testo→L^AT_EX differenti (purtroppo il Lisp gode di poco seguito nella community, per ora).

5 L'interfaccia a linea di comando

Il programma d'interfaccia (oggi chiamato tc2) ha evoluto gli script shell già presenti nel sistema di impaginazione FSM, automatizzando i compiti comuni che un editor compie: creare una gerarchia di directory per la rivista, creare degli stub per i singoli articoli, comporre un articolo o un intero numero, ma anche interagire con i server online per estrarre gli articoli depositati nel magazzino degli articoli disponibili (chiamato fantasiosamente Inbound di redazione), per sincronizzare la copia di lavoro con il sistema di controllo di revisione dei file o per pubblicare gli articoli o la rivista nel sistema di divulgazione attualmente utilizzato (SCRIBD).

È quest'interfaccia che costruisce il file principale L^AT_EX, che l'editor quindi non gestisce più direttamente, in dipendenza delle opzioni che vengono passate sulla linea di comando o in appositi file di configurazione in formato INI.

Ancora una volta l'obiettivo dello sviluppo software (poco più di 800 righe di Python) è stato quello di costruire attorno al L^AT_EX uno 'schermo' per permettere ad un editor, potenzialmente

incompetente su questa tecnologia, di portare a termine l'intera operazione di composizione.

La scelta si è rivelata vincente. Agli editor attuali, che pure hanno qualche conoscenza di L^AT_EX, molto raramente viene chiesto di accedere a quel livello per risolvere i problemi di composizione. Nell'ottica di distribuire ancor di più il ruolo dell'editor questo era un passo determinante.

A differenza dei primi tempi, oggi raramente il processo d'impaginazione produce errori nell'esecuzione dell'impaginazione L^AT_EX, ma quando ciò accade non sempre è facile per un editor che non lo padroneggi determinare le cause dell'errore.

6 I server di condivisione

Un'altra introduzione significativa nel processo di publishing, proprio per tener fede all'idea di openness che si voleva infondere al progetto è stata la condivisione del 'codice' delle riviste (non degli strumenti di programmazione, ma proprio degli articoli) attraverso un server SVN. In prospettiva anche il deposito degli articoli potrebbe essere realizzato in questa modalità.

Gli editor hanno quindi accesso alle riviste, sottodirectory di una repository SVN, e possono lavorare in autonomia sulle sezioni di una stessa rivista per poi 'committare' il loro risultato agli altri editor.

Questa non è una grande novità per i programmatori e, tra tutte, questa è stata l'innovazione sicuramente più 'amata' dagli editor, che sono essi stessi programmatori.

Essendo i file di produzione (sia gli articoli che le configurazioni) semplici file di testo, l'utilizzo di un sistema di controllo delle revisioni è perfettamente adeguato e svolge il proprio compito efficacemente ed efficientemente.

Simpaticamente anche nella realizzazione dei contenuti delle riviste si ritrovano gli stessi problemi che solitamente si hanno con il codice sorgente dei linguaggi di programmazione (es. la rottura delle 'build' o i conflitti di redazione, con i soliti conseguenti strascichi polemici).

7 Le macro L^AT_EX

I primi numeri delle riviste sono stati composti con macro L^AT_EX solo leggermente riviste rispetto a quelle di FSM. Si è preferito rimandare il lavoro di fine-tuning e di 'pulizia' ad un secondo momento.

In un primo tempo questo lavoro è stato condotto adattando il codice presente alla nuova struttura, senza neppure molto comprenderne il contenuto. Non sono un T_EXnico così bravo da poter comprendere un codice così complesso senza uno studio approfondito, e con le mille cose da fare per rimettere in sesto l'editore questo momento si allontanava sempre di più.

Poi quest'estate, in una di quelle notti da hacker in cui qualsiasi cosa sembra essere possibile, ho trovato la forza d'animo di compattare il codice dei pacchetti *journal* e *paper* (di cui parla più approfonditamente Gianluca in PIGNALBERI (2005a) e PIGNALBERI (2005b)), cancellando più del 30% di codice superfluo e riordinando una infinità di piccole cose impiastricciate alla men peggio precedentemente.

È nata così la classe *imediaj.cls* e il package *imediacy.sty* per la generazione delle copertine.

Rispetto alla versione di FSM poche differenze minori, e alcune veramente sostanziali.

Tra quelle minori essenzialmente nuovi ambienti per listati, figure, zoom, pubblicità, riquadri di varia natura sia floating che in posizione di rigore, implementati in L^AT_EX ma richiamati da appositi markup nel file *.tchl.txt*, nonché un insieme di nuovi comandi come *tchlarticle*, *thcleditorial*, *tchltoc*, ecc., che agiscono da interfaccia per il programma principale L^AT_EX, e di cui parleremo dopo. È stata anche resa più flessibile l'introduzione delle testate grafiche per gli articoli e migliorata la leggibilità dell'indice, nonché gestita la numerazione a due componenti (volume, fascicolo) in luogo di quella basata sul solo numero d'uscita.

Quelle sostanziali riguardano invece:

- la 'configurazione' delle riviste;
- la gestione dei colori;
- la gestione dei font;
- la realizzazione della copertina.

7.1 Creazione e configurazione di una rivista

Ogni uscita della rivista è contenuta in una directory apposita con una propria gerarchia e denominata con un'etichetta relativa alla testata (CP, DEV... seguita da due cifre relative al volume e due relative al fascicolo nel volume (ad es. CP1703).

Nella directory principale del fascicolo è presente un file principale di configurazione (CP1703.conf), tre directory (*a/* contiene gli articoli, *c/* contiene le sezioni d'inclusione T_EX previste dall'impaginazione ma dipendenti dall'uscita, e *g/* le immagini e i grafici).

All'interno della directory *a/* è presente una sottodirectory per ogni articolo denominato con la label del fascicolo e una label univoca per l'articolo, ad es. CP1703_apollo11.

La configurazione della rivista è ottenuta con un file in formato INI contenente una 'stanza' dedicata alla rivista, e un insieme di chiavi, obbligatorie, per indicare le informazioni variabili. Ad es:

```
[CP-1703]
volume=17
number=03
issue=1703
month=Luglio-Agosto
```

```
year=2009
```

Questi valori, ed altri letti da un file di configurazione generale, saranno riportati all'interno del file L^AT_EX generato automaticamente dal programma d'interfaccia *tc2* come variabili L^AT_EX, ad esempio:

```
%%% BASE VARS
\magazine={CP}
\headstrip={}
\headingcolor={}
\issuevolume=17
\issuenum={3}
\issuenumber={1703}
\issuemonth={Luglio-Agosto}
\issueyear=2009
\issnnumber={}
%%% MAGAZINE VARS
\newtoks\varmagmagazine
\varmagmagazine={Computer Programming}
\newtoks\varmagheading
\varmagheading={CP}
%%% ISSUE VARS
\newtoks\varissuenumber
\varissuenumber={03}
\newtoks\varissuemonth
\varissuemonth={Luglio-Agosto}
\newtoks\varissueyear
\varissueyear={2009}
\newtoks\varissuevolume
\varissuevolume={17}
\newtoks\varissueissue
\varissueissue={1703}
```

Per indicare la successione degli articoli sono utilizzate le chiavi *prog<nn>*, come in:

```
prog01=cover
prog02=ad,i_0901_smartcard.png
prog03=edit
prog04=toc
prog11=focus,apollo11
prog12=focus,apollo11vs
prog41=article,abcjquery
prog42=article,stereovision
prog43=article,statmachineintro,
      i_0901_programmatore_mb.png
prog80=column,kmcm
prog88=column,autori,0904_redsign_mb.png
prog89=intoc,logo.png
prog98=ad,i_0901_sinibaldi.jpg
prog99=ad,i_0901_CPweb.png
```

Queste chiavi hanno una o più componenti separate dalle virgole. La prima componente indica la "classe" dell'elemento, la seconda l'etichetta relativa.

Le chiavi principali sono quelle relative agli articoli, specificamente *focus*, per gli articoli della sezione speciale che tratta dell'argomento relativo allo strillo di copertina, *column* per le rubriche e *article* per tutti gli altri.



FIGURA 3: Le prime due pagine di un articolo mostrano il miglior abbinamento dei colori del titolo e del suo sfondo rispetto a quanto fatto in Free Software Magazine

Il secondo componente relativo alle chiavi utilizzate per gli articoli è l'etichetta che si troverà nella denominazione della directory. Un eventuale terzo componente indica la presenza di una mezza pagina pubblicitaria a sviluppo orizzontale o verticale a seconda che il nome dell'immagine termini con `_mb` (mezza bassa) o `_mv` (mezza verticale).

Altre chiavi sono presenti per l'impaginazione della copertina (`cover`), dell'editoriale (`edit`), del sommario (`toc`), e di pagine pubblicitarie (`ad`) o contenenti immagini a piena pagina (`img`).

Nel sorgente L^AT_EX le chiavi relative alla successione degli articoli saranno riportate come appositi comandi, ad esempio:

```
\tchlcover
\tchlad{i_0901_smartcard.png}{}
\tchledit{}{}
\tchltoctoc{}{}
\tchlfocus{apollo11}{}
\tchlfocus{apollo11vs}{}
\tchlarticle{abcjquery}{}
\tchlarticle{stereovision}{}
\tchlarticle{statmachineintro}
    {i_0901_programmatore_mb.png}
\tchlcolumn{kmcm}{}
\tchlcolumn{autori}{0904_redsign_mb.png}
\tchltoctoc{logo.png}{}
\tchlad{i_0901_sinibaldi.jpg}{}
\tchlad{i_0901_CPweb.png}{}

```

Viene così completato il file L^AT_EX che compilato un paio di volte realizza il PDF completo della rivista.

7.2 La gestione dei colori e dei font

Nel sistema FSM la definizione dei colori delle differenti componenti della rivista era contenuta all'interno delle classi di documento (`journal` e `paper`) in modo rigido, ed anche un po' disordinato. Questo innanzitutto imponeva una certa quantità di acrobazie direttamente all'interno dei file L^AT_EX degli articoli quando, per ovvie ragioni, la composizione cromatica dei titoli mal si addiceva agli sfondi delle testate. Questo vale tanto di più nell'utilizzo fatto da Infomedia dove, a differenza di FSM, le testate sono più varie e talvolta legate esclusivamente all'articolo piuttosto che alla sezione della rivista.

Era quindi necessario passare ad uno schema in cui si potessero selezionare cromatismi differenti non solo per le differenti testate, ma da fascicolo a fascicolo della stessa testata, e addirittura da articolo ad articolo nello stesso fascicolo. Ovvero doveva essere possibile definire cromatismi differenti per qualsiasi elemento dell'impaginazione, a livello di singolo articolo (come in figura 3).

Si è risolto questo problema prima definendo un insieme completo di nomi di colori relativi alle differenti parti della rivista (sono oltre 100). E successivamente accludendo ad ogni fascicolo (nella directory `c/`) un file (ad es. `CP1703_tcccolor.tex`) che definisse prima l'insieme dei colori e poi sezioni alternative in base ad una etichetta definibile a livello di variabili d'articolo.

Seppure molto semplificato, si può vedere l'esempio successivo:

```
\newcommand{\colorscheme}[1]{

```

```
% Standard colors
\definecolor{normal}
  {rgb}{ 0, 0, 0}
\definecolor{background}
  {rgb}{ 1, 1, 1}

\IfSubStringInString\{whiteonblack}
  {#1}{%
  \definecolor{normal}
    {rgb}{ 1, 1, 1}
  \definecolor{background}
    {rgb}{ 0, 0, 0}
}{}

\IfSubStringInString\{redonblue}
  {#1}{%
  \definecolor{normal}
    {rgb}{ 1, 0, 0}
  \definecolor{background}
    {rgb}{ 0, 0, 1}
}{}
}
```

In questo modo richiamando nel file $\text{\LaTeX}$ dell'articolo il comando

```
\colorscheme{whiteonblack}
```

si ottiene la modifica dei colori.

Questo mi dà modo anche di introdurre una altra peculiarità della nuova struttura. Come è stato detto in precedenza, l'editor cui spetterebbe la definizione dei colori non dovrebbe scendere al livello del $\text{\LaTeX}$ per queste definizioni.

In realtà il sistema di traduzione $\text{\texttt{.tchl.txt}} \rightarrow \text{\TeX}$ permette la definizione di *variabili* al livello dell'articolo con il markup testuale TCHL che sono poi trasposte in variabili $\text{\LaTeX}$.

Ad esempio nell'intestazione del file $\text{\texttt{.tchl.txt}}$ di un articolo è possibile rinvenire le seguenti variabili

```
#!title!Atterraggio lunare
#!author!Emmanuele Somma
#!journalpart!Cover Story
#!subtitle!Come conquistare la luna
  in soli 4 kbyte di RAM
#!colorscheme!white
```

che saranno trasformate in $\text{\LaTeX}$ dal programma ELisp come:

```
\title{Atterraggio lunare}
\author{Emmanuele Somma}
\journalpart{Cover Story}
\subtitle{Come conquistare la luna
  in soli 4 kbyte di RAM}
\colorscheme{white}
```

In questo modo è possibile introdurre l'indicazione del colorscheme direttamente nel file TCHL.

Per ciò che riguarda la gestione dei font lo schema adottato è sostanzialmente simile.

8 La copertina

Per la generazione della copertina fino al penultimo fascicolo ci siamo affidati, sulle orme di FSM, all'introduzione di una pagina contenente un'immagine realizzata da un grafico. Ma in definitiva non volevamo rinunciare alla possibilità di far introdurre gli strilli dagli editor invece che dai grafici. Già dall'ultimo numero di CP — Computer Programming — è stata realizzata l'integrazione direttamente in $\text{\LaTeX}$ della copertina completa tranne gli strilli e gli strilli generati dal $\text{\LaTeX}$ (figura 4), in verità letti anch'essi dal file esterno di configurazione del fascicolo come:

```
[CP-1703]
...
coverlineA=ABC di jQuery
coverlineB=StereoVision
coverlineC=Traduzioni statistiche
coverlineD=FIAT in CC?
covercolor=cCOVwhite
...
```

Per rendere ancora più semplice questa composizione, e per migliorare la veste grafica è stato deciso di adottare nel prossimo futuro una linea grafica più minimalistica studiando una completa rivoluzione della copertina.

Dal punto di vista $\text{\LaTeX}$ è stato realizzato un package apposito per la copertina (*imediacy.sty*) che, allo stato attuale non fa che comporre l'immagine di sfondo e gli strilli in posizione obbligata, ma nella nuova versione grafica permetterà l'introduzione di tutte le componenti necessarie, come overlay in un certo ordine ben definito.

A dire la verità uno dei motivi per giungere a questo risultato è anche dovuto ad un problema 'laterale' non indifferente: in questo momento non si è fatta una differenziazione della composizione in relazione alla modalità di distribuzione separando, ad esempio, la composizione per l'online da quella della stampa (che invece FSM faceva), il file PDF prodotto dalla composizione è decisamente molto pesante (30 megabyte circa), di questo il buon 30% è dovuto alla copertina realizzata dal grafico in risoluzione per la stampa. L'adozione di una grafica minimale riduce questo 'costo' ma soprattutto la ricerca dell'immagine di copertina che raramente risulta adeguata al contenuto.

9 Conclusioni

Il tentativo di fare un respawn di Infomedia, variando le sue caratteristiche fondamentali come struttura editoriale e modello di business, è una sfida che ci sta appassionando ogni giorno di più.



FIGURA 4: La copertina di Computer Programming Volume 17 n° 3 ha gli strilli composti con LATEX

I risultati di questi mesi sono molto evidenti, ma è ancora ben poca cosa rispetto a quello che potrebbe essere fatto da una comunità più forte e attiva.

LATEX gioca una parte determinante in questo progetto vista la forte determinazione di continuare la produzione cartacea della rivista.

In effetti lo spazio per migliorare questa componente è molto ampio viste le limitazioni e le rigidità che la contraddistinguono adesso (pur facendo un lavoro assolutamente apprezzabile ed adeguato nel frattempo).

Per avere informazioni:

- Il sito del Turing Club: <http://the.programmers.net>
- Il sito contenente il progetto TCHL, la linea di comando d'interazione `tc2`, il programma di traduzione Lisp e le classi e i package LATEX necessari: <https://www.assembla.com/wiki/show/turing>

- Il wiki con il materiale e le idee sulla produzione: <http://wiki.infomedia.it>
- Il blog con le comunicazioni più importanti: <http://blog.infomedia.it>
- Il nuovo sito Infomedia per la distribuzione delle riviste (accesso limitato ai sostenitori): <http://www.infomedia.it>

Riferimenti bibliografici

- PIGNALBERI, G. (2005a). «Creating Free Software Magazine». *Free Software Magazine*, (7).
- (2005b). «Della produzione di una rivista in LATEX». In *Atti del GJTrmeeting2005*. Pisa.

▷ Emmanuele Somma
Infomedia

Processing “Computed” Texts

Jean-Michel Hufferlen

Sommario

This article is a comparison among methods that may be used to derive texts to be typeset by a word processor. By ‘derive’, we mean that such texts are extracted from a larger structure. The present standard for such a structure uses XML-like format, and we give an overview of the available tools for this derivation task.

Keywords Typesetting computed texts, $\text{\TeX}$, $\text{\LaTeX}$, $\text{\ConTeXt}$, $\text{\XeTeX}$, $\text{\LuaTeX}$, XML, XSLT, character maps, XQuery, XSL-FO.

Sommario

Questo articolo è una comparazione dei metodi che possono essere usati per derivare, tramite un *word processor*, testi da comporre. Per ‘derivare’ intendiamo che questi testi saranno estratti da una struttura più grande. Lo standard corrente per tale struttura usa un formato simile a XML, e daremo una panoramica degli strumenti disponibili per la derivazione.

Parole chiave Composizione di testi elaborati, $\text{\TeX}$, $\text{\LaTeX}$, $\text{\ConTeXt}$, $\text{\XeTeX}$, $\text{\LuaTeX}$, XML, XSLT, mappe di caratteri, XQuery, XSL-FO.

1 Introduction

Formats based on the $\text{\TeX}$ typesetting engine—e.g., *Plain $\text{\TeX}$* (Knuth, 1984), or $\text{\LaTeX}$ (Lamport, 1994), or $\text{\ConTeXt}$ (Hagen, 2001)—are known as wonderful tools to get high-quality print outputs. Of course, they have been designed to typeset texts directly written by end-users, at first. But other texts may be *computed*, in the sense that they result from computation applied to some data, in particular, items belonging to data bases. A very simple example is given by a bill computed by means of a spreadsheet program like Microsoft Excel: the master file is an .xls file—that is, the whole information about data is centralised into this .xls file—but we may wish such a bill to be typeset nicely using a word processor comparable with $\text{\LaTeX}$. We personally experienced a more significant example: at the University of Franche-Comté, we manage the projects proposed to Computer Science students in several degrees located at Besançon, in the East of France. That is, we collect the proposals of projects, control the assignment of student groups to projects. Then, at the semester’s end, we organise the projects’ oral defences, and rate students from information transmitted by juries. During projects, information is transmitted

to students and projects’ supervisors either on the Web, or by means of printed documents. Managing only a list of project specifications and enriching it progressively is insufficient: it is better for oral defences’ announcement to be shown w.r.t. defences’ chronological order, and this order is unknown when projects are proposed. Likewise, we may wish to give the grades got by students according to the decreasing order of these grades, as well as students’ alphabetical order. In these cases, we have to perform a sort operation before typesetting the result. These examples are not limitative: other operations may be needed if we are only interested in a subset of the information concerning projects.

It is well-known that $\text{\TeX}$ ’s language is not very suitable for tasks directly related to programming. Readers interested in putting a sort procedure into action using this language can refer to (Roegel, 1998): this *modus operandi* may be viewed as a worthwhile exercise, but is unusable in practice, especially when it is not immediate to get sort keys from items to be sorted. A better idea is to use a format suitable for information management, with interface tools serving several purposes. Presently, the indisputable standard to model such formats is XML¹, providing a rich toolbox for this kind of task. But these tools have advantages and drawbacks: we are going thoroughly into these points in this article. Reading it requires only basic knowledge about $\text{\LaTeX}$ and the tools related to XML.

2 A simple example

The examples given in the introduction are ‘real’ applications, but for sake of simplicity, we consider an easier example², pictured in Figure 1. This XML text describes some items of a series of stories—*Doc Savage*—first published as pulps in the 1930’s, then republished as pocket books in the 1960’s³. As it can be noticed in the given example, the original publication order—for pulps—was not followed by the series of pocket books. In addition, some stories were unpublished as pulps (e.g., *The Red Spider*) or retitled when published as pocket books (e.g., *The Deadly Dwarf*, previously titled *Repel*), in which

1. eXtensible Markup Language. Readers interested in a general introductory book to this formalism can refer to (Ray, 2001).

2. Concerning the first example we have mentioned—the example related to a spreadsheet program—readers interested in it can refer to (Boulanger, 2001).

3. All the source texts mentioned throughout this article can be downloaded *in extenso* from <http://lifc.univ-fcomte.fr/home/~jmhufferlen/texts/>.

```
<story-list>
  <story>
    <title>The Deadly Dwarf</title>
    <pulp nb="56">Repel</pulp>
    <pocket-book nb="28"/>
  </story>
  <story>
    <title>The Land of Terror</title>
    <pulp nb="2"/>
    <pocket-book nb="8"/>
  </story>
  <story>
    <title>The Lost Oasis</title>
    <pulp nb="7"/>
    <pocket-book nb="6"/>
  </story>
  <story>
    <title>The Man of Bronze</title>
    <pulp nb="1"/>
    <pocket-book nb="1"/>
  </story>
  <story>
    <title>The Red Spider</title>
    <pocket-book nb="95"/>
  </story>
  <story>
    <title>World's Fair Goblin</title>
    <pulp nb="74"/>
    <pocket-book nb="39"/>
  </story>
</story-list>
```

FIGURE 1: Master file using XML-like syntax.

case, the pulp’s title is given as the contents of the `pulp` element⁴. More precisely, if the contents of a `pulp` element is empty, that means that the pulp’s title was the same as the pocket book’s.

Now we propose to search the information given in Figure 1, extract the items published as pulps, sorts them according to the publication order⁵. The title given is the pulp’s; if the corresponding pocket book has been retitled, a footnote must give the ‘new’ title. Of course, we wish the result to be typeset nicely, as (LA)TEX is able to do. That is, a good solution should look like the source text, processable by *Plain TEX*, given in Figure 2. As above-mentioned, a TEX-based solution could use TEX commands for the elements `story-list`,

4. In this case, such a story is more commonly known under the pocket book’s title, because pocket books are easier to get than pulps, very rare. That is why our `title` elements always refer to pocket books’, the contents of `pocket-book` elements being always empty. The Web page mentioned above includes specifications of the taxonomy of these texts.

5. This example seems to us to be pertinent, because there is no order ‘better’ than others: the original order is based on pulps, but—as above-mentioned—some stories are unpublished as pulps, whereas sorting stories according to pocket books’ order allows us to sort all the stories, but this is not really chronological.

```
\item{1} The Man of Bronze
\item{2} The Land of Terror
\item{7} The Lost Oasis
\item{56} Repel\footnote*{Book’s title of
  pulp no.~56: The Deadly Dwarf}
\item{74} World’s Fair Goblin

\end
```

FIGURE 2: Stories’ titles sorted by pulp numbers.

`story`, `title`, `pulp`, and `pocket-book`, but would lead to complicated programming.

3 Using tools related to xml

3.1 xslt producing TEX sources

XSLT⁶ (W3C, 2007b) is the language commonly used for transformations of XML texts. The text of a stylesheet that may be used to get Figure 2 from Figure 1 is given *in extenso* in Figure 3. This stylesheet takes as much advantage as possible of the features introduced by XPath’s⁷ and XSLT’s new version⁸ (2.0): for example, we have made precise the types of the used variables, by means of `as` attributes—these types being provided by the library of XML Schema⁹ (W3C, 2008)—such information allows an XSLT processor to perform type-checking. Likewise, we have made precise the type of templates’s results, as far as possible.

Let us remark that this stylesheet implicitly uses the fact that the strings of our input file—the elements’ contents—do not contain any special character of TEX: for example, a ‘#’ character must be replaced by ‘\#’ in order to be processed correctly by TEX. If we are interested in building (LA)TEX texts, *character maps* (W3C, 2007b, § 20.1) allow the replacement of a single character by a string, as shown in Figure 4 where we show how to process most of *Plain TEX*’s special characters. Another solution may be the generation of Unicode texts (The UNICODE CONSORTIUM, 2006) and the use of a Unicode-compliant TEX-like engine¹⁰, e.g. XELATEX

6. eXtensible Stylesheet Language Transformations. Introductions to this language have been given in BachoTEX conferences: (Hufflen, 2005, 2006, 2008a).

7. XPath (W3C, 2007a) is the language used to address parts of an XML text.

8. A study of these new features can be found in Hufflen (2008a).

9. Schemas allow users to define *document types*, such a document type can be viewed as taxonomy common to some XML texts. There exist several schema languages, but only XML Schema is interfaced with XSLT 2.0. The prefix ‘`xsd:`’ gets access to XML Schema’s library, whereas the prefix ‘`xs1:`’ characterises XSLT constructs. Readers interested in more details about XML namespaces can consult (Ray, 2001, pp. 41–45). An example using XML Schema is given at the Web page mentioned above.

10. Engines are not formats: a *format* is a set of pre-loaded definitions based on primitives of a TEX-like engine, whereas an *engine* is TEX or is derived from TEX by adding

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="2.0" id="pulp" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" extension-element-prefixes="xsd">
  <xsl:output method="text" encoding="ISO-8859-1"/>
  <xsl:strip-space elements="*" />
  <!-- (The best way to specify an End-Of-Line character:) -->
  <xsl:variable name="eol" select="'&#xA;'" as="xsd:string"/>
  <xsl:template match="story-list">
    <xsl:apply-templates select="story[pulp]">
      <xsl:sort select="xsd:integer(pulp/@nb)" />      <!-- Numerical sort. -->
    </xsl:apply-templates>
    <xsl:value-of select="$eol,'end',$eol" separator=""/>
  </xsl:template>
  <xsl:template match="story">
    <xsl:variable name="pulp-0" select="pulp" as="element(pulp)" />
    <xsl:variable name="pulp-nb-0" select="$pulp-0/@nb" as="xsd:integer" />
    <xsl:variable name="pulp-title-0" select="data(pulp-0)" as="xsd:string" />
    <xsl:variable name="title-processed" as="xsd:string">
      <xsl:apply-templates select="title" />
    </xsl:variable>
    <xsl:value-of
      select="'\item{',$pulp-nb-0,}' ',
        if ($pulp-title-0) then
          $pulp-title-0,'\footnote*{Book&apos;s title of pulp no.~',$pulp-nb-0,': ',
            $title-processed,}'" else
          $title-processed,
        $eol'
      separator=""/>
    </xsl:template>
    <xsl:template match="title" as="xsd:string"><xsl:apply-templates/></xsl:template>
  </xsl:stylesheet>

```

FIGURE 3: Getting a source text for *Plain T_EX* by means of an XSLT stylesheet.

(Kew, 2007) or Lua_{T_EX} (Hagen, 2008a).

XSLT is not limited to the generation of texts, the `method` attribute may also be set to `html` or `xhtml`¹¹ (W3C, 2007b, § 20), in which case it allows Web pages to be generated. Likewise, this `method` attribute set to `xml` means that XML texts are to be generated. Using these output methods provides a great advantage: since any XSLT stylesheet is an XML text, an XSLT processor checks that the final document is well-formed, in particular, opening and closing tags must be balanced. The generation of (L^A)_{T_EX} texts lacks analogous check: an XSLT processor cannot ensure that opening and closing braces are balanced, as in _{T_EX}; likewise, when L^A_{T_EX} texts are generated, it is impossible to ensure that environments—e.g., `\begin{document}... \end{document}`—are bal-

anced. Since such errors are not detected statically, they just appear when generated texts are typeset by a word processor. Let us notice that such check would be more difficult about the texts generated for the Con_{T_EX}t format, because the opening and closing commands for Con_{T_EX}t's environments are `'\start...'` and `'\stop...'`, e.g., the equivalent formulation for `\begin{document}... \end{document}` in L^A_{T_EX} is `\starttext... \stoptext` in Con_{T_EX}t.

3.2 xslt producing xsl-fo texts

Since deriving XML texts by means of XSLT provides a better check level, an alternative idea is to get XSL-FO¹² (W3C, 2006) texts. Such texts use an XML-like syntax that aim to describe high-quality print outputs. As shown in (Huffle, 2007), there are some similarities between L^A_{T_EX} and XSL-FO, the latter providing, of course, more systematic markup. This language is verbose, but it is not devoted to direct use: XSL-FO texts usually result

or redefining some primitives. *Plain T_EX* and L^A_{T_EX} are formats, X_L_{T_EX} and Lua_{T_EX} are engines.

11. (eXtensible) HyperText Markup Language. XHTML is a reformulation of HTML—the original language of *Web* pages—using XML conventions. (Musciano and Kennedy, 2002) is a good introduction to these languages.

12. eXtensible Stylesheet Language—Formatting Objects.

```
<xsl:character-map name="some-special-characters">
  <xsl:output-character character="#" string="\#"/>
  <xsl:output-character character="%" string="\%"/>
  <xsl:output-character character="$" string="\$"/>
  <xsl:output-character character="&" string="\&"/>      <!-- &  -->
  <xsl:output-character character="\ " string="\$backslash$"/>
  <xsl:output-character character="{ " string="\{$"/>      <!-- In Plain TEX, the commands  -->
  <xsl:output-character character="}" string="\}$"/>      <!-- \{ ' and \}' are only usable in  -->
  <!-- in math mode (cf. (Knuth, 1984, Exercise 16. 12)).  -->
  <xsl:output-character character="~" string="\char'7E"/> <!-- Using hexadecimal code.  -->
</xsl:character-map>
```

FIGURE 4: Using a character map in XSLT 2.0.

from applying an XSLT stylesheet¹³.

The advantage of this approach is clear: generated texts in XSL-FO are well-formed. However, XSL-FO lacks *document classes*, as in L^AT_EX. Some elements allow the description of *page models*, but end-users are entirely responsible for this definition. XSL-FO provides much expressive power about placement of *blocks*¹⁴, but is very basic on other points. For example, let us consider footnotes, end-users are responsible of choosing footnote marks. Figure 5 gives the result of applying an XSLT stylesheet providing an XSL-FO text for our example. The first child of the `fo:footnote` element gives the footnote reference, the second child is the actual footnote’s contents (W3C, 2006, § 6.12.3). This `fo:footnote` element seems to be low-level in comparison with L^AT_EX’s `\footnote` command¹⁵. Of course, if you want footnotes to be numbered automatically, XSLT addresses this problem. However, another point seems to us to be more debatable: end-users are responsible for putting a leader separating footnotes from the text’s body¹⁶ (we show how to proceed in Figure 5). So some footnotes may be preceded by a leader, some may not. This point may seem anecdotal, but for L^AT_EX users, some features of XSL-FO can be viewed as low-level and be difficult to handle.

Last but not least, most present XSL-FO processors do not implement the whole of this language, even if they can successfully process most of XSL-FO texts used in practice¹⁷. So some features may be unusable, whereas an equivalent construct will work in (L^A)T_EX. XSL-FO has been designed to deal with the whole of Unicode, so it shows how the

Unicode bidirectional algorithm (UNICODE CONSORTIUM, 2008) is put into action (Hufflen, 2009), but this point may also be observed with X_FT_EX.

3.3 XQuery producing T_EX sources

XQuery (W3C, 2007c) is a query language for data stored in XML form, as SQL¹⁸ does for relational data bases¹⁹. XQuery can be used to search documents and arrange the result, as an XML structure or a simple text (possibly suitable for a T_EX-like engine). An XQuery program processing our example in order to get Figure 2’s text is given in Figure 6. Like XSLT 2.0, XQuery uses XPath 2.0 expressions and the datatype library provided by XML Schema²⁰. Such programs, using FLWOR²¹ expressions, are more compact than equivalent ones in XSLT. However, XQuery is suitable only for generating simple texts: advanced features like character maps in XSLT are provided by some XQuery processors, but are not portable. A simple example of an implementation-dependent feature is given in Figure 6: we declare that the result is not an XML text by a non-portable option, `saxon:output`²². Of course, if XQuery is used to generate XML texts, they are well-formed, but no analogous check can be done about texts generated for a T_EX-like engine. In other words, XQuery has the same drawback than XSLT.

3.4 A curiosity: dsssl

DSSSL²³ (dsssl, 1996) was initially designed as the stylesheet language for displaying SGML²⁴ texts.

13. The use of several prefixes for namespaces—usually ‘xsl:’ and ‘fo:’—clearly distinguish elements belonging to XSLT and XSL-FO.

14. Roughly speaking a *block* in XSL-FO is analogous to a *minipage* in L^AT_EX (Hufflen, 2007, § 1.2).

15. In fact, XSL-FO’s footnotes can be compared with Plain T_EX’s `\footnote` command, as shown in Figure 2.

16. That is done in standard L^AT_EX class, but not universal: as an example, the *arstexnica* class, used for articles of the *ArsT_EXnica* journal, does not put it.

17. We personally use the Apache FOP (Formatting Objects Processor) (apache-fop).

18. Structured Query Language. A good introductory book about it is (Melton and Simon, 1993).

19. A short introduction to XQuery is given in (?).

20. ... although the type declarations using the `as` keyword are optional. We put them in Figure 6 for sake of clarity and for taking as much advantage as possible of XQuery’s type-checker.

21. ‘For, Let, Where, Order by, Return’, the keywords used throughout such expressions.

22. The XQuery processor we have used for this example is Saxon (Kay, 2008).

23. Document Style Semantics Specification Language.

24. Standard Generalised Markup Language. Now it is only of a historical interest. Readers interested in this metalanguage can refer to (Bradley, 1997).

```

<fo:list-block ...>
  ...
  <fo:list-item>
    <fo:list-item-label ...><fo:block>56</fo:block></fo:list-item-label>
    <fo:list-item-body ...>
      <fo:block>
        Repel<fo:footnote>
          <fo:inline font-size="x-small" vertical-align="super">*</fo:inline>
          <fo:footnote-body>
            <fo:block text-align-last="justify"><fo:leader leader-pattern="rule"/></fo:block>
            <fo:block font-size="'xx-small'">
              <fo:inline font-size="xx-small" vertical-align="super">*</fo:inline>
              Book's title for pulp no. 56: The Deadly Dwarf
            </fo:block>
          </fo:footnote-body>
        </fo:footnote>
      </fo:block>
    </fo:list-item-body>
  </fo:list-item>
  ...
</fo:list-block>

```

FIGURE 5: How to put a footnote in XSL-FO (see the equivalent *Plain T_EX* source text in Figure 2).

DSSSL includes a core expression language that is a side-effect free subset of the **Scheme** programming language (Kelsey et al., 1998). XML being a subset of SGML, stylesheets written using DSSSL can be applied to XML texts. DSSSL is rarely used now, that is why we just cite it²⁵, an example of using it for a bibliography style can be found in (Hufflen, 2008b, § 3.5). End-users do not put down (L^A)T_EX commands when they develop a DSSSL stylesheet, but the **jade**²⁶ program, as shown in (Goossens et al., 1999, § 7.5.2), can generate T_EX-like texts, the typesetting engine usable to process such results being JadeT_EX.

4 Enriched T_EX engines

If we go back to programs based on T_EX-like typesetting engines, there are two other possible methods, based on T_EX-engines ‘enriched’ by using a ‘more classical’ programming language. In both cases, XML texts are pre-processed by procedures belonging to a programming language, and the result is sent to a T_EX-like typesetting engine. Of course, such *modus operandi* is suitable only if we want to generate (L^A)T_EX texts, it would be of little interest to get XML texts or pages written using (X)HTML²⁷.

PyT_EX (Fine, 2005) is written using **Python** (Martelli, 2006) and uses T_EX as a daemon. Getting the components of a ‘computed’ text is left to **Python**—more precisely, to the **Python** functions

25. A DSSSL stylesheet generating a text equivalent to Figure 2’s is given at the Web page mentioned above.

26. James Clark’s **Awesome DSSSL Engine**.

27. Unless a converter from (L^A)T_EX to (X)HTML is used, of course.

dealing with XML texts—and successive results are sent to T_EX, in turn.

LuaT_EX (Hagen, 2008a) is a T_EX-like typesetting engine that is able to call functions written using **Lua** (Ierusalimsky, 2006). On another point, this engine can process texts using XML-like syntax, as shown in (Hagen, 2008b). So **Lua** can be used to extract and sort the right information from an XML text, and this information is typeset by the engine. Figure 7 gives an implementation of our example in ConT_EXt MkIV: it uses **Lua** to define an interface with sorting functions, the other functionalities being put into action using T_EX-like commands. As in ConT_EXt, the layout is controlled by set-up commands:

```

\start...setup
...
\stop...setup

```

—for example, **title** elements’ contents are just displayed, processing **pulp** elements yields displaying the number or the title, depending on a mode—then our XML file is processed ‘atomically’, by means of the **\xmlprocessfile** command. This approach is promising, but let us recall that LuaT_EX and ConT_EXt Mk2IV have not reached yet stable state: that is planned for the year 2010. Another important drawback: as shown by the examples using the **\xmlfilter** command in Figure 7, this command uses path expressions, very close to XPath expressions, but not identical. For example, **command**—used to connect a selected item to the set-up command that processes it—obviously does not belong to XPath. On the contrary, some XPath expressions are not recognised, even if ‘simple’ paths are processed. Some tricks may be used

```

declare namespace saxon = "http://saxon.sf.net/" ;
declare namespace xsd = "http://www.w3.org/2001/XMLSchema" ;

declare option saxon:output "omit-xml-declaration=yes" ;

declare variable $eol as xsd:string := "&#xA;" ;
declare variable $filename as xsd:string external ;

if (doc-available($filename)) then
  string-join((for $story-0 as element(story) in doc($filename)/story-list/story[pulp]
    let $pulp-nb-0 as xsd:untypedAtomic := data($story-0/pulp/@nb),
        $pulp-nb-0-int as xsd:integer := xsd:integer($pulp-nb-0),
        $pulp-nb-0-string as xsd:string := xsd:string($pulp-nb-0),
        $pulp-title-0 as xsd:string := xsd:string(data($story-0/pulp)),
        $story-title-0 as xsd:string := xsd:string(data($story-0/title))
    order by $pulp-nb-0-int
    return ("<item{", $pulp-nb-0-string, "> ",
        if ($pulp-title-0) then
          $pulp-title-0, "<footnote*{Book's title of pulp no.-",
          $pulp-nb-0-string, ": ", $story-title-0, ">" else
          $story-title-0, $eol),
        $eol, "<end", $eol),
    "")) else
  ()

```

FIGURE 6: Getting a source text for *Plain T_EX* by means of XQuery.

as workarounds, but we think that complete compatibility with XPath should be reached.

5 Conclusion

If we sum up the methods shown throughout this article, the approaches that seem suitable are XQuery for simple examples, XSLT for more ambitious ones. The use of LuaT_EX could be interesting in a near future, too.

However, we think that there are two directions that should be explored. The first would be a modern implementation of XSL-FO using a T_EX-like typesetting engine. Such an implementation has begun: *Passive T_EX* (Carlisle et al., 2000), but this project is presently stalled. We think that processing XSL-FO could re-use the experience got by (I_A)T_EX developers, even if syntaxes are very different²⁸. The second direction would be the definition and implementation of an output mode of XSLT suitable for (I_A)T_EX; some additional services could be performed: for example, checking that braces and environments are balanced. Such an output mode already exists in *nbst*²⁹, the language of bibliography styles close to XSLT and used by MIBiBT_EX³⁰ (Hufflen, 2003), but it concerns only the way to process L^AT_EX's special characters.

28. Besides, it is well-known that T_EX recognises only its own formats, what complicates cooperation between T_EX and other programs.

29. New Bibliography STyles.

30. MultiLingual BiBT_EX.

Acknowledgements

I wish to thank Hans Hagen who greatly helped me debug and improve LuaT_EX source texts. Thanks to Karl Berry, who made precise some terminology notions. Last but not least, thanks to Gianluca Pignalberi for his patience and his Italian translations of the abstract and keywords.

References

- apachefop. *Apache FOP*. <http://xmlgraphics.apache.org/fop/>, March 2009.
- Frédéric Boulanger. L^AT_EX au pays des tableurs. *Cahiers GUTenberg*, 39–40:7–16, May 2001. In *Actes du Congrès GUTenberg 2001*, Metz.
- Neil Bradley. *The Concise SGML Companion*. Addison-Wesley, 1997.
- David Carlisle, Michel Goossens, and Sebastian Rahtz. De XML à PDF avec *xm_Ltex*, XSLT et *PassiveT_EX*. *Cahiers GUTenberg*, 35–36:79–114, May 2000. In *Actes du congrès GUTenberg 2000*, Toulouse.
- Jonathan Fine. T_EX forever! In *Proc. EuroT_EX 2005*, pages 150–158, Pont-à Mousson, France, March 2005.
- Michel Goossens, Sebastian Rahtz, Eitan M. Gurari, Ross Moore, and Robert S. Sutor. *The L^AT_EX Web Companion*. Addison-Wesley Longmann, Inc., Reading, Massachusetts, May 1999.

```

\enablemode[ds:pulp]
\startluacode      % Definitions using Lua.
...
function lxml.sorters.reset(name)
...
end      -- (... and other definitions.)
\stopluacode

\def\xmlresetsorter#1{\ctxlua{lxml.sorters.reset("#1")}}
...      % (Other definitions for LuaTEX.)

\startxmlsetups xml:ds:base
  \xmlsetsetup{ds}{*}{-}
  \xmlsetsetup{ds}{story-list|story|title|pulp|pocket-book}{xml:ds:*}
\stopxmlsetups

\xmlregisterdocumentsetup{ds}{xml:ds:base}

\startxmlsetups xml:ds:story-list
  \xmlfilter{story-list}{story/pulp/../../../../command(xml:ds:stories-plus)}
\stopxmlsetups

\startxmlsetups xml:ds:story-list-plus
  \xmlresetsorter{story-list} \xmlfilter{#1}{story/command(xml:story-list:getkeys)}
  \blank sortkeys: \blank\xmlshowsorter{ds}\blank \xmlsortentries{story-list}
  \xmlflushsorter{story-list}{xml:story-list:flush}
\stopxmlsetups

\startxmlsetups xml:story-list:getkeys
  \xmladdsortentry{story}{#1}{\xmlatt{\xmlfirst{#1}{pulp}}{nb}}
\stopxmlsetups

\startxmlsetups xml:story-list:flush \startitemize \xmlflush{#1} \stopitemize \stopxmlsetups

\startxmlsetups xml:ds:story
  \sym{\xmlfirst{#1}{pulp}} \xmlifelse{#1}{pulp}{
    {\disablemode[ds:pulp] \xmlfirst{#1}{pulp}\footnote{Book's title: \xmlfirst{#1}{title}}}{
    \xmlfirst{#1}{title}}
\stopxmlsetups

\startxmlsetups xml:ds:title \xmlflush{#1} \stopxmlsetups

\startxmlsetups xml:ds:pulp
  \doifmodeelse{ds:pulp}{\xmlatt{#1}{nb}}{\xmlflush{#1}}
\stopxmlsetups

\starttext \xmlprocessfile{ds}{ds.xml} \stoptext

```

FIGURE 7: Processing a master file by means of ConT_EXt Mk IV.

-
- | | |
|---|--|
| <p>Hans Hagen. <i>ConT_EXt, the Manual</i>. http://www.pragma-ade.com/general/manuals/cont-enp.pdf, November 2001.</p> <p>Hans Hagen. The luafication of T_EX and ConT_EXt. In <i>Proc. BachoT_EX 2008 Conference</i>, pages 114–123, April 2008a.</p> <p>Hans Hagen. Dealing with XML in ConT_EXt MkIV. MAPS, 37:25–39, 2008b.</p> <p>Jean-Michel Huppen. MIBIBT_EX's version 1.3. <i>TUGboat</i>, 24(2):249–262, July 2003.</p> <p>Jean-Michel Huppen. Introduction to XSLT. <i>Biuletyn GUST</i>, 22:64, April 2005. in <i>BachoT_EX 2005 conference</i>.</p> | <p>Jean-Michel Huppen. Advanced techniques in XSLT. <i>Biuletyn GUST</i>, 23:69–75, April 2006. In <i>BachoT_EX 2006 conference</i>.</p> <p>Jean-Michel Huppen. Introducing L^AT_EX users to XSL-FO. <i>TUGboat</i>, 29(1):118–124, 2007. EuroBachoT_EX 2007 proceedings.</p> <p>Jean-Michel Huppen. XSLT 2.0 vs XSLT 1.0. In <i>Proc. BachoT_EX 2008 Conference</i>, pages 67–77, April 2008a.</p> <p>Jean-Michel Huppen. Languages for bibliography styles. <i>TUGboat</i>, 2008(3):401–412, July 2008b. TUG 2008 proceedings, Cork, Ireland.</p> <p>Jean-Michel Huppen. Multidirectional typesetting in XSL-FO. In Tomasz Przechlewski, Karl Berry,</p> |
|---|--|

- and Jerzy B. Ludwiczowski, editors, *Proc. Ba-choT_EX 2009 Conference*, pages 37–40, April 2009.
- Roberto Ierusalimsky. *Programming in Lua*. Lua.org, 2 edition, March 2006.
- dsssl. DSSSL. International Standard ISO/IEC 10179:1996(E), 1996.
- Michael H. Kay. *Saxon. The xSLT and XQuery Processor*. <http://saxon.sourceforge.net>, March 2008.
- Richard Kelsey, William D. Clinger, Jonathan A. Rees, Harold Abelson, Norman I. Adams iv, David H. Bartley, Gary Brooks, R. Kent Dyb-vig, Daniel P. Friedman, Robert Halstead, Chris Hanson, Christopher T. Haynes, Eugene Ed-mund Kohlbecker, Jr, Donald Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy Lewis Steele, Jr, Gerald Jay Sussman, and Mitchell Wand. Revised⁵ report on the algorithmic language Scheme. HOSC, 11(1):7–105, August 1998.
- Jonathan Kew. X_TT_EX in T_EX live and be-yond. *TUGboat*, 29(1):146–150, 2007. EuroBa-choT_EX 2007 proceedings.
- Donald Ervin Knuth. *Computers & Typesetting. Vol. A: The T_EXbook*. Addison-Wesley Publish-ing Company, Reading, Massachusetts, 1984.
- Leslie Lamport. *L^AT_EX: A Document Preparation System. User’s Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.
- Alex Martelli. *Python in a Nutshell*. O’Reilly, 2 edition, July 2006.
- Jim Melton and Alan R. Simon. *Understanding the new SQL*. Morgan Kaufmann, 1993.
- Chuck Musciano and Bill Kennedy. *HTML & XHTML: The Definitive Guide*. O’Reilly & Asso-ciates, Inc., 5 edition, August 2002.
- Erik T. Ray. *Learning XML*. O’Reilly & Associates, Inc., January 2001.
- Denis B. Roegel. Anatomie d’une macro. *Cahiers GUTenberg*, 31:19–27, December 1998.
- The UNICODE CONSORTIUM. *The Unicode Stan-dard Version 5.0*. Addison-Wesley, November 2006.
- Unicode Bidirectional Algorithm*. The UNICODE CONSORTIUM, <http://unicode.org/reports/tr9/>, March 2008. Unicode Standard Annex #9.
- W3C. *Extensible Stylesheet Language (XSL). Version 1.1*. <http://www.w3.org/TR/2006/REC-xsl11-20061205/>, December 2006. w3C Recommendation. Edited by Anders Berglund.
- W3C. *XML Path Language (XPath) 2.0*. <http://www.w3.org/TR/2007/WD-xpath20-20070123>, January 2007a. w3C Recommendation Draft. Edited by Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernández, Michael H. Kay, Jonathan Robie and Jérôme Siméon.
- W3C. *XSL Transformations (XSLT). Ver-sion 2.0*. <http://www.w3.org/TR/2007/WD-xslt20-20070123>, January 2007b. w3C Recommendation. Edited by Michael H. Kay.
- W3C. *XQuery 1.0 and XPath 2.0 Functions and Operators*. <http://www.w3.org/TR/2007/REC-xpath-functions-20070123>, Jan-uary 2007c. w3C Recommendation. Edited by Ashok Malhotra, Jim Melton, and Norman Walsh.
- W3C. *XML Schema*. <http://www.w3.org/XML/Schema>, December 2008.

▷ Jean-Michel Hufflen
LIFC (EA CNRS 4157)
University of Franche-Comté
16, route de Gray
25030 Besançon Cedex
France
jmhufflen at lifc dot
univ-fcomte dot fr

Use of the `\write18` feature for composing indexes

Claudio Beccari*

1 Introduction

Any $\text{\LaTeX}$ user who has composed an index knows the trouble one gets in when s/he has spent a lot of time collecting the index entries, with the correct formatting, and eventually finishes his/her paper but forgets the last three runs in order to have the correct final index typeset at the end of the paper. The three runs are in order:

1. Run $\text{\LaTeX}$ or $\text{pdf\LaTeX}$ with the command `\makeindex` in the preamble; this will collect the index entries, their formatting, and the page numbers where the entries are located in the document; this information gets written down in a file with extension `.idx` that contains all the entries with the related information in the order they were collected.
2. Run the program `makeindex`, possibly with a personalized style file and specific options in order to produce the sorted entry list with their ranking and page numbers. This information gets written into a file with extension `.ind`; this file contains the whole `\theindex` environment with each entry marked with an `\item` or `\subitem` or `\subsubitem` prefix, together with the relevant formatting instructions for both the entries and the page numbers.
3. Run again $\text{\LaTeX}$ or $\text{pdf\LaTeX}$ to process the final index file, possibly by inputting it by means of a command such as `\input{<filename>.ind}`.

These three steps can be executed in just one run if you make use of the command `\write18{<text>}`.

Modern $\text{\TeX}$ distributions contain an executable file and a format file that have been created with special switches in such a manner that the interpreter `pdftex` may produce either a `.dvi` or a `.pdf` output file; it may make use of the commands of the extended `etex` interpreter, or refrain from doing so; it may or may not “shell out” to the system and ask it to perform some actions before resuming its main typesetting task.

Here I will exploit this last feature in order to produce the actual index in one run of (pdf) $\text{\LaTeX}$.

*This is the translation of the original paper, published in this issue, titled *Uso del comando `\write18` per comporre l'indice analitico in modo sincrono*. The translation will be published in TUGboat, and appears here courtesy of both the author and TUGboat.

2 Auxiliary files

When the command `\makeindex` is executed while the preamble gets processed, a new output stream is opened in order to write the index information into the `.idx` auxiliary file; at the same time, that command activates the various `\index` commands within the document that actually write the information into the auxiliary file. The name of this output stream is assigned to the internal $\text{\LaTeX}$ kernel control sequence `\@indexfile`.¹

This file has the specified extension `.idx` and its name is the same as that of the (main) `.tex` file that is being typeset; the name of this file is kept into the $\text{\TeX}$ control sequence `\jobname`, that may be used by any suitable macro.

Since the index is typically the last thing to be typeset in a document, `\input{<filename>.ind}` is likely the last command before the `\end{document}` statement.

So, in order to sort and produce `\jobname.ind` from `\jobname.idx`, we have to do essentially two simple operations:

1. Close the output stream identified by the internal macro `\@indexfile`, and
2. execute the `makeindex` command in order to produce the new updated `.ind` file.

3 The commands

We have to execute “immediately” the following two commands:

```
\immediate\closeout\@indexfile
\immediate\write18{
  makeindex -s style.ist \jobname.idx}
```

just before the `\input{\jobname.ind}` statement or its equivalent. Of course, if no specific index style file `style.ist` is being used, the whole clause `-s style.ist` is omitted.

The `\immediate` $\text{\TeX}$ primitive instructs the interpreter not to delay the execution of the subsequent command; without it, the $\text{\TeX}$ commands concerned with input and output streams, including the actual read or write commands, are

1. This is what the $\text{\LaTeX}$ kernel and the standard classes do. Some nonstandard classes may use other control sequences to hold the information about the index output stream: for example the class `memoir` by default uses the control sequence `\jobname` extended with the suffix `@idxfile`, but, since `memoir` may build up several indices, each one has its own output stream with a different name, that gets specified with the optional argument of the command `\makeindex`.

deferred until a page is shipped out. Obviously `\closeout` closes an output stream, in our case the one specified by the subsequent macro `\@indexfile`.

The `\write18{<text>}` command writes onto the special output stream numbered “18”; this stream is equivalent to writing the `<text>` as a system command line in the command window of the Windows family of operating systems, of an xterm window of the Unix family of operating systems, or of the terminal window of the MacOSX operating system. When the system command finishes its processing, control returns to the T_EX interpreter that goes on from the very point it had suspended its processing in order to have the operating system perform the required task.

The presence of the `@` character in the output index stream name ‘`\@indexfile`’ requires that a `\makeatletter` command be issued before the first statement and a `\makeatother` be issued just after the first statement, thus changing and reverting its category code.

On the other hand, if those two statements are defined with a macro contained in a personal `.sty` file, there is no need to change the category code of `@`. This would also be a good occasion to add useful bells and whistles to the new command, for example, checking if the output stream really exists and may be closed; checking that `\jobname.idx` exists in order to convert it; provide for an optional argument to specify the index style file so as to insert the style file clause in the system call; and warning and error messages could be added so as to let the author know if anything is inconsistent. This is not the proper place to add long stretches of T_EX or L^AT_EX code, but any reader should be capable of customizing a new command suitable for his/her needs.

4 The results

By issuing those commands, or the new command defined in the personal `.sty` file, just before giving the command `\input{\jobname.ind}` (or equivalent), the index turns out to contain exactly the index entries and page numbers that were collected during that current (pdf)L^AT_EX run, and there is no need to remember to update the index file after any correction to the source file(s), since the entries and the page numbers are always up to date.

I find these simple macros extremely valuable, and I use them myself in all my (recent) documents, after I realized the possible benefits deriving from the `\write18` command. This facility is not widely used, because it makes it possible for T_EX to execute external applications exposing the typesetter to the risk of executing malicious code; therefore some T_EX distributions are distributed with this facility disabled by default.

Nevertheless there is some good news in this regard: T_EX Live 2009, and a subsequent release of MiK_T_EX, will contain a restricted version of the `\write18` command, such that `\write18` will be enabled by default, but only a restricted list of “safe” programs (listed in a configuration file) will be allowed to be executed. And, happily, `makeindex` is one of those safe programs.

I imagine there are many other possible applications of the `\write18` facility: for example, the format transformation of graphic files is already available in the package `epstopdf.sty`; it’s up to the users to discover new ones.

▷ Claudio Beccari
claudio dot beccari at gmail
dot com

Eventi e novità

TUG 2010: T_EX's 25th anniversary

L'edizione 2010 del convegno internazionale del TUG avrà luogo dal 28 al 30 Luglio, a San Francisco, California, USA.

Trattandosi del trentaduesimo compleanno di T_EX, sarà presente anche il «Grand Wizard». Oltre a Knuth, molti membri del progetto originale T_EX di Stanford (compresi David Fuchs, John Hobby, Michael Plass, Oren Patashnik, Tom Rokicki, and Luis Trabb-Pardo) hanno annunciato la loro intenzione di partecipare.

Diverse iniziative sono in fase di organizzazione allo scopo di rendere onore alla loro opera e celebrare il trentaduesimo anniversario di T_EX.

Come ogni anno, inoltre, vi saranno interventi sui vari aspetti di T_EX e un corso introduttivo parallelo su L^AT_EX il secondo giorno.

Ulteriori informazioni relative all'organizzazione e al programma sono reperibili, o verranno via via pubblicate, sulla pagina internet dedicata al convegno: <http://www.tug.org/tug2010/>.

EuroT_EX Conference 2010 e settimo G_JT_rmeeting



Il Gruppo Utilizzatori Italiani di T_EX avrà l'onore (e l'onere) di ospitare, dal 25 al 29 Agosto 2010 l'EuroT_EX Conference. Il convegno si svolgerà, come al solito, presso la Scuola Superiore Sant'Anna a Pisa, e conterrà, al suo interno, anche la settima edizione del Convegno nazionale del G_JT_r.

Per la nostra associazione questa sarà l'occasione per rinsaldare i legami con il resto della comunità T_EX internazionale ed avere comunicazioni “di prima mano” sulle novità più significative nello sviluppo dei programmi che compongono un sistema T_EX.

Il programma e altri particolari sull'organizzazione del convegno verranno comunicati in seguito sulla pagina internet <http://www.guit.sssup.it/eurotex2010/>.

T_EX Live 2009

È prossima ad essere rilasciata la versione 2009 di T_EX Live, di cui è già possibile scaricare e testare la versione di prova (<http://www.tug.org/texlive/pretest.html>). T_EX Live 2009 verrà inclusa nel DVD T_EX Collection, prodotto dai TUG e che, non appena pronto, sarà distribuito agli iscritti.

Dopo la novità rilevante introdotta nella versione 2008, ovvero un meccanismo di *package management* che permette di avere il proprio sistema costantemente aggiornato, anche quest'anno sono presenti miglioramenti di non poco conto. Ne elenchiamo alcuni, tratti dal sito segnalato qui sopra:

- Alcune delle attuali restrizioni all'uso della primitiva `\write18`, che permette di lanciare programmi esterni da T_EX, sono state tolte. L'accesso a questa funzionalità verrà sfruttato, per esempio, da pacchetti come `epstopdf` per convertire “al volo” figure EPS in figure PDF che possano essere incluse da `pdflatex`. Le impostazioni predefinite nel file di configurazione `graphics.cfg` impediscono, comunque, di sovrascrivere PDF già esistenti.
- Il programma di grafica Asymptote è stato aggiunto alla distribuzione (e ai mirror CTAN).
- L'editore di testo T_EXworks sarà presente, ma solo per sistemi operativi Windows e Mac OS X (la versione per sistemi *NIX è stata demandata al packaging delle varie distribuzioni).
- La collezione `amsfonts` è stata aggiornata alla versione 3.0, che include nuove versioni dei font Computer Modern (e CM extra), Euler, simboli matematici AMS, e Washington Cyrillic. Disegno e *hinting* dei caratteri è stato migliorato, secondo i suggerimenti di Donald Knuth e Hermann Zapf. Le nuove metriche sono compatibili con quelle vecchie.

InfomediaEditori

Dal 1991 l'informazione per i programmatori e i sistemisti italiani

computer Programming



www.infomedia.it/i/cp

Computer Programming è la più antica e di gran lunga la più conosciuta rivista di programmazione in Italia.

Impegnata a fornire informazioni di qualità su sviluppo e programmazione, dal 1991 Computer Programming è il 'salvavita' di riferimento per gli sviluppatori italiani grazie alla sua indipendenza, alla qualità degli articoli, all'acutezza di analisi, agli autori di prestigio e all'attenzione alle esigenze dei lettori.

Su Computer Programming si possono trovare interviste o articoli dei personaggi che hanno fatto l'informatica moderna come Wirth, Stroustrup, Meyer, Waldo, Gaslin, Stepanov, Brockschmidt, Richter.

DEV. **Developing Software Solutions**



www.infomedia.it/i/dev

DEV è la rivista popolare per tutti che permette di iniziare la programmazione dall'ABC.

Ai principianti permette di addentrarsi nell'affascinante mondo della programmazione con articoli introduttivi, panoramiche e corsi di programmazione.

È un mix di tecniche, trucchi, consigli e soluzioni anche per gli esperti che desiderano rimanere aggiornati e conoscere le nuove tendenze della programmazione.

Java, C/C++, .NET, Python, Perl, Linux, Windows, programmazione Web, LAMP, SQL e computer grafica sono alcuni degli argomenti trattati che rendono DEV la più appetitosa rivista italiana per gli sviluppatori e i pratici.

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.



ArTeXnica – Call for Paper

La rivista è aperta al contributo di tutti coloro che vogliano partecipare con un proprio articolo. Questo dovrà essere inviato alla redazione di ArTeXnica, per essere sottoposto alla valutazione di recensori entro e non oltre il 14 Marzo 2010. È necessario che gli autori utilizzino la classe di documento ufficiale della rivista; l'autore troverà raccomandazioni e istruzioni più dettagliate all'interno del file d'esempio (.tex).

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

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

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

ArsT_EXnica

Rivista italiana di T_EX e L^AT_EX

Numero 8, Ottobre 2009

- 5 Editoriale
Gianluca Pignalberi, Massimiliano Dominici
- 7 Simboli matematici in T_EX e L^AT_EX
Enrico Gregorio
- 25 Produrre grafica vettoriale di alta qualità programmando *Asymptote*
Agostino De Marco
- 40 T_EX as an ebook reader
Kaveh Bazargan
- 42 La composizione di tabelle con larghezza specificata
Claudio Beccari
- 48 L'arte di gestire la bibliografia con `biblatex`
Lorenzo Pantieri
- 61 L^AT_EX e CSV
Massimiliano Dominici
- 70 `combelow`: abbasso i segni diacritici di serie B
Gianluca Pignalberi
- 76 Uso del comando `\write18` per comporre l'indice analitico in modo sincrono
Claudio Beccari
- 79 Una estensione di luatex: luatex lunatic
Luigi Scarso
- 92 Il respawn di Infomedia (L^AT_EX-based)
Emmanuele Somma
- 102 Processing "Computed" Texts
Jean-Michel Hutfien
- 110 Use of the `\write18` feature for composing indexes
Claudio Beccari
- 112 Eventi e novità

