Numero 22
Ottobre
2016

ArsTEXnica

Rivista italiana di TEX e LATEX

GUIT

http://www.guitex.org/arstexnica/

TEXnica Ars

GUIT – Gruppo Utilizzatori Italiani di TEX

ArsTEXnica è la pubblicazione ufficiale del GUIT

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

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

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

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

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

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

arstexnica@guitex.org.

Nota sul Copyright

Associarsi a GUIT

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

Indirizzi

Gruppo Utilizzatori Italiani di TEX
c/o Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Industriale
Via Claudio 21
80125 Napoli – Italia
`http://www.guitex.org`
guit@sssup.it

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

Codice ISSN 1828-2369

Stampata in Italia
Napoli: 15 Ottobre 2016

GUIT meeting 2016

TREDICESIMO CONVEGNO NAZIONALE SU TEX, LATEX E TIPOGRAFIA DIGITALE

29 ottobre 2016

Università Cattolica del Sacro Cuore,
via Trieste 17, Brescia

Programma del convegno

9:30 Benvenuto. Inizio dei lavori.

9:45 *LaTeX tra i banchi. Possibili applicazioni in ambito scolastico di LaTeX*. Rossana Giacopini.
10:15 *Il LaTeX come soluzione al problema dell'accesso a testi con formule da parte di disabili visivi*. Massimo Borsero, Nadir Murru e Alice Ruighi.
10:45 *Importanza di LaTeX per utenti con deficit visivo*. Michela Montrasio.

11:15 — Pausa caffè (30 min).

11:45 *LuaTeX* + `pdfliteral`. Roberto Giacomelli.
12:15 *Tavole di Ishihara*. Renato Battistin.

12:45 — Pausa pranzo (1 h 45 min).

14:30 *Grafica 3D con Geogebra e TikZ*. Luciano Battaia.
15:00 *Forindex: computer-aided indexing*. Guido Milanese.
15:30 *Liste, cicli, LaTeX3*. Enrico Gregorio.

16:00 — Pausa caffè (30 min).

16:30 *Towards a New Bibliography Format*. Jean-Michel Hufflen.
17:00 *Il formato PDF archiviabile con gli aggiornamenti di TeX Live 2016*. Claudio Beccari.

17:30 Discussione sulle prospettive del gruppo.

18:30 Chiusura dei lavori. Arrivederci.

La partecipazione è libera e gratuita, previa registrazione entro il 26 ottobre.
Registrazione online: `www.guitex.org/home/meeting`

GUIT Gruppo Utilizzatori Italiani di TEX

`www.guitex.org`

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 22, Ottobre 2016

Gruppo Utilizzatori Italiani di TEX

Editoriale

Claudio Beccari

Eccoci arrivati al GUIT Meeting del 2016 nella prestigiosa sede del campus bresciano dell'Università Cattolica del Sacro Cuore. Ringrazio il prof. Guido Milanese per essersi prodigato nell'organizzare questo meeting, per il quale abbiamo ricevuto il buon numero di contributi scientifici che sono raccolti in questo numero di ArsTEXnica.

Quest'anno c'è una novità fra gli argomenti, cioè l'uso di LaTeX nelle scuole sia come mezzo per iniziare gli allievi all'informatica, sia per fornire un aiuto agli allievi che necessitano di assistenza speciale da parte della scuola.

Un articolo di Rossana Giacopini ci parla dei numerosi vantaggi che derivano dall'uso di LaTeX nella scuola; viene usato sia per il lavoro individuale sia per quello di gruppo degli allievi; risulta molto utile per superare alcuni difetti individuali, come la disgrafia e la dislessia, può essere usato per i disabili visivi. La prof. Giacopini sottolinea anche il fatto che, essendo il sistema TeX un software sostanzialmente libero, certamente gratuito, è possibile insegnare agli allievi il rispetto della legalità.

Massimo Borsero, Nadir Murru, e Alice Ruighi presentano una bella panoramica di programmi, sistema TeX compreso, usabili dai portatori di minorazioni visive, dalla visione ridotta alla cecità totale. Esaminano i pro e i contro dei vari programmi e della loro effettiva usabilità su diverse piattaforme dotate di diversi sistemi operativi e focalizzano la loro attenzione su LaTeX come il migliore programma, anche se non ancora completamente perfetto, per il supporto agli allievi delle scuole e dell'università quando sono affetti dalle minorazioni suddette.

Michela Montrasio racconta la sua esperienza di persona cieca lungo il suo curriculum di studi dal liceo al corso di laurea magistrale in matematica che sta seguendo ora; descrive dal punto di vista dell'utente finale i dispositivi sia hardware sia software disponibili per aiutare le persone cieche o con minorazioni visive. La sua analisi è interessantissima e la sua conclusione è che LaTeX, affiancato da software adeguato per la lettura a voce dei testi, è assolutamente l'unico software usabile quando si passa agli studi superiori; lei stessa non potrebbe seguire i suoi corsi di laurea magistrale in matematica se non disponesse di LaTeX.

Roberto Giacomelli sta diventando esperto con LuaTeX e sta sperimentando alcune funzionalità poco o per niente documentate nei manuali che riguardano questo motore di composizione. Si tratta del disegno programmato, sia ricorrendo a LuaTeX per inserire nel file di uscita i comandi grafici interpretabili dai visualizzatori di file PDF, sia le funzionalità che derivano dalla libreria `luamplib` creata per usare il linguaggio METAPOST per fare sì che LuaTeX produca direttamente alcuni tipi di disegni. L'articolo riporta numerosi esempi sia con il comando `\pdfliteral` sia con il linguaggio METAPOST integrato in LuaTeX.

Renato Battistin ci presenta della grafica a colori: le immagini di Ishihara che vengono usate par la rilevazione e la classificazione dele anomalie visive. Si tratta di disegni formati da tanti cerchietti di vari diametri colorati con sfumature di un dato colore collocati su uno sfondo formato ancora da cerchietti di vari diametri ma colorati con sfumature di un altro colore. A seconda dei colori e delle sfumature dei disegni e dello sfondo, le persone affette da certi disturbi visivi riescono o non riescono a distinguere l'immagine. Una raccolta di tavole con immagini, disegni e colori diversi permette appunto di rilevare le anomalie visive e di classificarle a seconda del tipo. LaTeX con il suo disegno programmato mediante Ti*k*Z risulta prezioso per questo lavoro diagnostico.

L'articolo di Luciano Battaia si rivolge completamente alla geometria dei solidi e alla loro rappresentazione "tridimensionale" mediante Ti*k*Z; come editor usa Geogebra, che è in grado di esportare codice Ti*k*Z che, a sua volta può essere ulteriormente modificato per necessità speciali. Ma i disegni dei solidi platonici, nonché quello dei solidi speciali, come il "pallone da calcio", o alcuni solidi stellati, come la Stella di Urbino, o icosaedro stellato, con tutte le sue 60 facce, sono esempi interessantissimi di che cosa si può fare con LaTeX.

Guido Milanese ci presenta un suo nuovo software, *Forindex*, che, raggiunta una certa stabilità, è ora disponibile su CTAN per il beneficio specialmente dei linguisti, ma non solo, per creare indici analitici in un modo molto particolare, che ha come caratteristica importante quella di non richiedere praticamente nessun lavoro manuale di marcatura nel file sorgente. Ridotto all'essenziale sarebbe come a dire: Forindex, prendi questo elenco di parole, cercatele nel file sorgente che ti indico, e componi l'indice analitico. Ovviamente c'è molto di più di ciò che è concentrato in quella frase. L'interfaccia grafica è particolarmente utile per personalizzare l'indice secondo le più svariate esigenze.

Enrico Gregorio scrive un approfondito tutorial sulla gestione delle liste; queste sono frequentissime come elementi per ripetere certe operazioni su una

lista di argomenti che possono essere numeri o parole. Mostra come fare sia usando il linguaggio di Plain TEX, sia quello di LATEX, sia usando il nuovo linguaggio L3, di cui è uno dei massimi esperti. Bisogna ammettere che il Linguaggio L3 si sta rivelando un serbatoio di sorprese incredibile, e grazie all'autore via via tutti lo avviciniamo con diversi livelli di abilità.

Jean-Michel Hufflen si dedica alla bibliografia, ma questa volta si concentra sul formato del file `.bib` che non consente, specialmente con il programma di elaborazione *BibTEX*, di rispettare le situazioni insolite che si presentano nel comporre voci bibliografiche particolarmente delicate. Il formato che egli suggerisce è parzialmente incompatibile con i programmi di elaborazione che si affidano al 'vecchio' formato `.bib`, ma può essere gestito correttamente dal programma *MlBibTEX*. Gli esempi che porta interessano sia gli studiosi di scienze umanistiche sia quelli di scienze sperimentali.

Da molti anni mi interesso dell'archiviabilità a lungo termine dei documenti; ho già scritto diversi articoli sull'argomento, e questo è un ulteriore articolo che descrive i notevoli miglioramenti che si possono ottenere con i programmi e i pacchetti distribuiti nel 2016, almeno con TeXLive e MacTeX. Il problema dell'archiviazione dei documenti in forma non cartacea è una cosa molto importante anche in ambito universitario dove ogni anno si devono archiviare decine di migliaia di tesi (in tutta Italia); molte università già chiedono il deposito di file PDF archiviabili invece delle copie cartacee delle tesi. Ma ci sono moltissimi altri ambiti, specialmente amministrativi e giudiziari, in cui l'archiviazione elettronica è diventata una necessità assoluta.

Ringrazio moltissimo i membri del comitato di redazione, il consiglio scientifico che ha esaminato e valutato gli articoli qui pubblicati, i revisori editoriali e i correttori che sono intervenuti sugli articoli in italiano e in inglese per migliorarne il testo. Grazie agli autori e grazie allo staff di ArsTEXnica la rivista si mantiene all'alto livello che è sotto gli occhi dei lettori.

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

LaTeX tra i banchi – Possibili applicazioni in ambito scolastico di LaTeX

Rossana Giacopini

Sommario

Negli ultimi anni, la scuola italiana si è avvicinata sempre di più alle nuove tecnologie, avviando un processo di digitalizzazione. LaTeX, grazie alla sua flessibilità, è uno strumento che può essere utile aiutando insegnanti, alunni e studenti con bisogni educativi speciali, inoltre promuove valori trasmessi ogni giorno negli istituti scolastici.

Abstract

In recent years, Italian education system has progressively introduced computers in teaching, starting a digitalisation process. LaTeX, due to its flexibility, is an instrument that may be useful for teachers, pupils and students with special education needs.

Resumen

Durante los últimos años, el sistema escolar italiano se ha acercado cada vez más a las nuevas tecnologías, empezando así un proceso de digitalización. LaTeX, gracias a su flexibilidad, es un instrumento que puede ayudar a los profesores y a los alumnos, incluso con necesidades educativas especiales.

1 Normativa scolastica vigente

1.1 Normativa europea

La normativa che definisce i contenuti, le abilità e competenze che gli studenti devono acquisire a scuola sono sia nazionali sia europei. Il documento chiave emanato dall'Unione Europea sono le *Competenze chiave per l'apprendimento permanente* (2006); il documento elenca le competenze indispensabili perché ogni cittadino europeo sia in grado di adattarsi al mondo globalizzato in continuo mutamento, tenendo conto anche delle fasce di popolazione in difficoltà.

La prima competenza indicata è la *comunicazione nella madrelingua*, che comprende non solo la conoscenza del lessico e della grammatica, ma anche una solida padronanza dei registri e delle tipologie testuali; anche per quanto riguarda la *competenza nella lingua straniera* sono elencate le abilità indicate precedentemente.

L'Unione Europea ritiene essenziale anche la *competenza digitale*, cioè non solo essere in grado di utilizzare i *software* per svolgere attività di base, i *social network* e un minimo di alfabetizzazione informatica, ma anche di essere consapevoli dei benefici e dei rischi correlati alle tecnologie, di usarle in maniera responsabile per quanto riguarda l'aspetto etico, giuridico, dei diritti d'autore. Si contempla anche l'uso degli strumenti informatici nella vita professionale e sociale, auspicando che i cittadini europei sfruttino le opportunità date da tali strumenti al fine di migliorare il loro lavoro, interagire con gli altri sia per fini personali che lavorativi anche in gruppi.

1.2 Normativa italiana

Lo Stato italiano, recependo le direttive comunitarie, ha redatto i seguenti documenti che regolano l'ordinamento scolastico nazionale:

- le *Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione* (2012),
- le *Indicazioni nazionali per i licei* (2010),
- il *Regolamento per gli istituti tecnici* (2010),
- il *Regolamento per gli istituti professionali* (2010).

La prima normativa, che stabilisce i piani di studio per il primo ciclo (scuola dell'infanzia, primaria e secondaria di I grado), stabilisce che fin dai primi anni di scolarizzazione i bambini dovrebbero gradualmente avvicinarsi al computer vedendolo utilizzare dall'insegnante; è dalla scuola primaria in avanti che viene introdotto sistematicamente lo studio dell'informatica all'interno della programmazione di *Tecnologia*: dopo una breve introduzione alle parti fondamentali del computer, si prosegue con l'uso di applicativi per scopi didattici o comunemente diffusi, per giungere alla elaborazione di brevi ricerche individuali o di gruppo con programmi di videoscrittura, di diapositive, con informazioni e risorse reperite attraverso Internet.

Durante il secondo ciclo d'istruzione (dalla scuola secondaria di II grado in poi), seppur con doverose distinzioni legate sia al tipo di istituto scelto sia agli indirizzi di specializzazione, si approfondisce l'uso di *software* e mezzi di comunicazione di uso corrente nella futura vita professionale: per la progettazione, per l'animazione e la grafica, per la contabilità, per la gestione delle attività commerciali, per la pubblicità.

Anche la riforma scolastica del 2015 nota come *Buona scuola* e il più recente *Piano Nazionale*

Scuola Digitale definiscono in maniera sempre più decisa la svolta da una scuola analogica fatta di libri, lavagne di ardesia e quaderni ad una scuola digitale che prevede l'uso delle "Lavagne interattive multimediali", di piattaforme e registri scolastici elettronici e di un uso continuato delle tecnologie non solo in classe ma anche a casa. A tale scopo è stata delineata la figura chiave dell'*animatore digitale*, un docente che si occupa della progressiva digitalizzazione della scuola di appartenenza: si forma e forma a sua volta i colleghi, fa da tramite e punto di riferimento in questo lungo processo.

2 Perché LaTeX in classe

Alla luce della normativa e del processo di modernizzazione e informatizzazione in atto nel mondo della scuola, vediamo come LaTeX possa essere impiegato in classe, tenendo conto anche delle esigenze di alunni con "Bisogni Educativi Speciali".

2.1 Uso in ambito didattico

Questo strumento, rispetto ad altri solitamente diffusi nelle scuole, ha delle caratteristiche che lo rendono adatto anche in ambito didattico, in quanto si rivela una risorsa estremamente flessibile da vari punti di vista.

Si tratta di uno strumento *multipiattaforma*, cioè completamente indipendente dal sistema operativo, quindi utilizzabile da tutti gli utenti che possiedano un computer senza che o il software utilizzato o la versione dello stesso installata influisca sulla lettura o sulla struttura del prodotto, alterandolo rispetto alla stesura originale; inoltre i file di *output* sono nella quasi totalità dei casi apribili con i programmi normalmente presenti in un computer, accedendo facilmente e da più dispositivi al documento.

Altro aspetto interessante è la totale *disponibilità* e *gratuità* di LaTeX: ciascun utente può cioè installare autonomamente e senza alcun esborso economico (a parte gli eventuali costi di connessione) l'intera collezione di pacchetti a prescindere dal dispositivo e aggiornare il sistema quando lo desidera, avendo così a disposizione uno strumento rinnovato e pronto per creare nuovi prodotti. I primi quattro aspetti fanno sì che gli alunni, se opportunamente formati, abbiano a disposizione uno strumento *comune*, che non altera il lavoro svolto sia autonomamente che in gruppo perché ad ogni apertura e aggiunta non viene modificato involontariamente ciò che è stato preparato in precedenza, il che riduce errori e imprevisti all'atto della presentazione del risultato finale.

La flessibilità di LaTeX non si riduce ai meri aspetti informatici, ma anche alle opportunità date dai singoli pacchetti che compongono l'intera *suite*: fin dall'inizio si può scegliere la *tipologia testuale* dello scritto o ad esempio decidere di comporre una presentazione con delle diapositive o una prova scritta, scegliere lo stile (e quindi l'aspetto grafico) senza preoccuparsi dell'impostazione grafica e dell'impaginazione perché tramite i pacchetti viene gestita già dal sistema stesso. Questo fa sì che l'utente si concentri principalmente sul contenuto e non sull'estetica dello stesso soffermandosi sulla struttura dell'elaborato, aiutato dai comandi previsti dalla classe del documento: l'inserimento di un comando errato dà segnale di errore all'atto della compilazione, per cui chi scrive è guidato e controllato al tempo stesso dal sistema, inoltre l'impaginazione automatica del documento è regolata da formule che assicurano un esito visivamente impeccabile.

La gestione delle *lingue* grazie ad appositi pacchetti agevola la sillabazione e la gestione delle parole per andare a capo, la titolazione automatica nella lingua principale del testo e delle bibliografie secondo i principali stili impiegati permettendo di citare fonti in più lingue: ciò consente in particolar modo agli insegnanti dell'ambito linguistico e ai loro studenti di evitare errori di composizione e di preparare velocemente e senza dover rivedere più volte elaborati da sottoporre agli altri.

Non meno saliente, il fattore *tempo* assume un duplice aspetto quando si stende un testo utilizzando LaTeX: da un lato il lavoro di redazione e impaginazione si riduce grazie alla presenza delle *classi* dei documenti che guidano l'utente nella redazione e organizzano la pagina, dall'altro implica una produzione di testi in due tempi, la redazione del sorgente e la successiva compilazione al fine di ottenere il file effettivo. La produzione in due tempi può disorientare l'utilizzatore abituato a fruire di strumenti di tipo sincrono in quanto non vede il risultato del suo lavoro mentre sta scrivendo, ma si può ovviare a tale inconveniente compilando spesso il sorgente o avvalendosi di *editor* che consentono la visione in contemporanea di sorgente ed *output*. In un'ottica più ampia e di conservazione e gestione dei dati, LaTeX consente di ottenere *file* più duraturi in quanto rimane sempre a disposizione il sorgente originale che può essere ricompilato alla luce degli aggiornamenti disponibili, oltre al fatto che i file *output* sono elaborati in formati altamente leggibili come ad esempio il pdf o possono essere rielaborati in formati nuovi semplicemente compilando il sorgente con un comando diverso in base al *file* d'arrivo.

2.2 Uso in presenza di Bisogni Educativi Speciali

In ambito scolastico, per alunni con bisogni educativi speciali si intendono categorie di studenti che presentano delle difficoltà a causa di disabilità, disturbi specifici dell'apprendimento o situazioni di disagio; la scuola italiana da tempo si preoccupa di venire incontro a ciascun studente affinché possa apprendere nella maniera più efficace possibile anche in presenza di tali fattori di svantaggio.

I discenti che presentano dei disturbi specifici dell'apprendimento necessitano di materiali strutturati in maniera chiara sia dal punto di vista contenutistico che visivo: in questo caso LaTeX viene in aiuto agli insegnanti grazie alla sua rigida struttura da un lato per organizzare in maniera logica le informazioni, dall'altro per disporre in maniera esteticamente più gradevole e quindi più facilmente leggibile il contenuto.

In casi di dislessia o di apprendenti ipovedenti si possono sostituire il *font* utilizzato e/o la dimensione dei caratteri in modo da impostare un documento scritto più grande mantenendo al tempo stesso le proporzioni tra le varie parti: l'uso di caratteri tipo Verdana o Arial risultano infatti più leggibili di quelli con grazie, di conseguenza l'alunno può riconoscere più facilmente le singole lettere e quindi comprendere con minor difficoltà il contenuto. In generale, uno scritto esteticamente più accattivante può fungere da motivazione nella lettura, a prescindere dal fatto che i ragazzi siano o meno in situazione di difficoltà.

I problemi legati all'ortografia (anche in alunni sordi con scarsa padronanza della lingua orale) possono essere agevolmente aggirati dagli apprendenti stessi in autonomia digitando il testo per mezzo di un apposito *editor* corredato da un correttore automatico che comprenda le lingue utilizzate; non solo: le difficoltà di sillabazione o nell'andare a capo sono automaticamente risolte dal sistema LaTeX stesso, che provvede a spezzare in maniera corretta la parola solo quando necessario.

Anche gli alunni che presentano difficoltà nell'aspetto motorio della scrittura (disgrafia) o nell'uso dello spazio (disprassia) possono trarre benefici dall'uso di questo strumento: i primi non sono costretti a scrivere con strumenti tradizionali (carta e penna) ma digitano al computer un testo leggibile per se stessi e per gli altri indipendentemente dalla propria manualità, i secondi non devono preoccuparsi dell'impaginazione perché è il sistema a gestire lo spazio entro il quale scrivere.

Al fine di venire ulteriormente incontro alle difficoltà degli alunni più deboli, si può valutare un percorso facilitato in cui non viene installato un semplice *editor*, ma un software dotato di una interfaccia utente più accessibile e comprensibile che li agevoli nel lavoro quotidiano sia a scuola che a casa, come ad esempio LyX.

2.3 Uno strumento portatore di valori

Considerando LaTeX non solo come strumento informatico ma nella sua totalità, si nota che si tratta di un prodotto che promuove dei valori, i quali sono a loro volta portati avanti ogni giorno nelle scuole:

- solidarietà,
- cooperazione e collaborazione,
- condivisione di obiettivi e risultati,
- legalità.

Gli sviluppatori e gli utilizzatori, siano essi riuniti in gruppi formalizzati o meno o singolarmente, producono in continuazione dei pacchetti che soddisfano delle esigenze proprie o altrui. Può capitare che un utente non esperto o non in grado di risolvere problemi inerenti la compilazione dei propri documenti chieda aiuto attraverso i *forum* e che altri rispondano in suo soccorso o che grazie all'apporto di più persone si giunga ad una soluzione soddisfacente; in altri casi sono i programmatori stessi a inventare nuovi pacchetti e a metterli a disposizione di chi ne ha bisogno corredandoli con una guida che ne descriva il funzionamento. Tutti questi esempi sono segno di un profondo senso di *collaborazione*, *collaborazione* e *solidarietà* che è strettamente correlata ad un senso di comunità estesa a tutti gli utenti nel mondo, fatta di persone che spesso non si conoscono ma che instaurano relazioni di reciproco aiuto attraverso la Rete.

Anche la questione della *legalità* assume una certa rilevanza: ciò che viene messo a disposizione è legale e viene legalmente distribuito grazie ad Internet, in particolar modo l'archivio CTAN e le associazioni nazionali degli utenti; trattandosi di materiale facilmente reperibile e gratuito il ricorso a risorse di tipo illegale a costo zero non ha ragion d'essere.

3 Il lavoro sul campo

Le esperienze sul campo sono tutt'oggi limitate e sono legate principalmente alle materie scientifiche.

Il primo esempio è un articolo apparso nel 1991 sulla rivista TUGboat dal significativo titolo *TEX in Schools: Just Say No*. Partendo dall'assunto che non sia opportuno introdurre TeX in ambito scolastico, l'autore illustra le motivazioni a favore della propria tesi analizzando un eventuale approccio su disciplina specifica o tramite integrazione all'interno delle diverse discipline (Informatica, materie linguistiche, Arte) e la scarsa praticità d'uso da parte degli insegnanti durante la preparazione dei materiali per le loro lezioni.

Sull'impiego nelle lezioni di informatica, Neuwirth evidenzia quanto risulti macchinoso usufruire di questa risorsa quando già è disponibile un altro linguaggio (LOGO) più adatto agli alunni. In campo linguistico invece egli teme che i docenti, a suo parere abituati ad insegnare ai loro alunni come esprimere le loro idee su carta, insieme a loro si preoccupino eccessivamente dell'impostazione della pagina tralasciando il cuore del loro operato, cioè il contenuto; il suo consiglio è quindi di affidarsi ad un comune *word processor* per questo tipo di attività. Nel settore artistico, l'autore nota una scarsa o nulla preparazione degli studenti sulla tipografia o su come comporre un documento esteticamente apprezzabile, così come lo vede inutile

persino in matematica per rappresentare concetti o grafici.

All'articolo sopraccitato risponde con una lettera inviata alla rivista Al Cuoco, all'epoca membro del Dipartimento di Matematica di una *high school* statunitense (corrispondente ad una scuola secondaria di II grado italiana). Il suo parere è diametralmente opposto: è convinto che niente sia eccessivamente complicato per i ragazzi, che una delle problematiche della scuola americana sia proprio la scarsa stima delle capacità degli alunni, che loro sono in grado di apprendere ad usare il computer più rapidamente degli adulti e dimostra che si può utilizzare TEX a scuola raccontando la propria esperienza personale di insegnante.

Spiega che da cinque anni sia i professori del dipartimento sia gli alunni stanno utilizzando questo strumento con successo: i primi hanno iniziato ad usarlo grazie a degli appositi corsi e sono riusciti a convertire gli appunti e le dispense scritte a mano in testi facilmente leggibili da distribuire agli apprendenti, i secondi ne hanno imparato il funzionamento durante le lezioni di matematica e hanno pubblicato una sorta di giornalino contenente gli elaborati di tutta la classe.

I progetti del professore americano non sono circoscritti all'ambito scientifico: l'orizzonte si amplia fino al testo e decide di coinvolgere anche i discenti che sono più interessati ad altre materie: ha proposto TEX anche agli allievi di economia, i quali hanno reagito in modo molto positivo e sono stati in grado di redigere dei testi esteticamente piacevoli che comprendono anche formule matematiche.

L'ultimo esempio invece è frutto di un'esperienza italiana in cui LATEX è stato usato in un Liceo scientifico di Pordenone: un insegnante di matematica ha insegnato ai suoi alunni della classe terza ad usare il pacchetto *PSTricks* per disegnare figure geometriche. Dapprima illustra come disegnare dei punti e dei triangoli sul piano cartesiano, in seguito fa aggiungere le notazioni scientifiche (denominazione dei vertici, ampiezza e marcatura degli angoli, misure dei lati), poi passa alla costruzione di figure non collocate sul piano cartesiano, giungendo alla progettazione di inversioni circolari.

I risultati del suo operato sono visibili e concreti: i ragazzi con l'aiuto del docente e di alcune macro ad opera dello stesso riescono a gestire disegni complessi anche quando prodotti su carta servendosi del computer e di LATEX.

Il Prof. Battaia, nell'articolo apparso nel 2007, conclude il suo intervento nella speranza di poter collaborare coi colleghi di lettere e lingue straniere: alcuni di loro sono soddisfatti della resa grafica dei loro elaborati, ma sono rimasti impressionati positivamente dalla facilità d'uso di LATEX e dall'impatto estetico; inoltre si mostra convinto che nella scuola superiore (oggi scuola secondaria di II grado) sia davvero possibile insegnare agli studenti come funziona LATEX.

4 Conclusioni

Il processo di progressiva digitalizzazione della scuola sta portando ad un uso sempre più frequente di strumenti informatici, questo comporta la scelta di risorse adatte alle esigenze di tutti gli attori del settore. LATEX, essendo per sua natura componibile, può rispondere alle richieste in ambito didattico tenendo conto di alunni e insegnanti, delle necessità pratiche in classe e a casa, degli aspetti economici e legali e di conservazione dei documenti. Se opportunamente preparati e accompagnati, anche gli studenti possono fruire dei vantaggi della sua flessibilità in elaborati individuali o di gruppo, aggirando gli eventuali ostacoli dovuti alle difficoltà di apprendimento.

Attualmente le proposte in ambito didattico sono legate alle materie scientifiche, invece sembra inspiegabilmente debole il fronte delle materie umanistiche: se è pur vero che TEX è stato inventato per creare una soluzione ai problemi tipografici legati soprattutto alle formule matematiche, LATEX invece è strettamente connesso con le tipologie testuali e si è arricchito man mano di svariati pacchetti pensati proprio per i linguisti.

Si auspica che gli studi approfondiscano questo settore tradizionalmente lontano dal mondo dell'informatica al fine di costruire un ponte tra queste due realtà spesso considerate inconciliabili avvicinando dapprima i docenti che si occupano del settore mostrando loro le risorse a loro disposizione grazie a LATEX per poi poterle presentare anche agli alunni durante le loro lezioni.

Riferimenti bibliografici

(2006). «Raccomandazione del Parlamento Europeo e del Consiglio relativa a competenze chiave per l'apprendimento permanente». URL http://eur-lex.europa.eu/legal-content/IT/TXT/?uri=URISERV%3Ac11090.

(2006). «Recommendation of the European Parliament and of the Council on Key Competences for Lifelong Learning». URL http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=URISERV:c11090&from=IT.

(2015). *BES a scuola: I 7 punti chiave per una didattica inclusiva*. Erickson. URL https://books.google.it/books?id=KmDoDAAAQBAJ.

Al Cuoco (1991). «TEX in Schools: Why Not?» *TUGboat*. URL https://www.tug.org/TUGboat/tb12-2/tb32letters.pdf.

Cesare Cornoldi, D. L. e. a. (2013). *Dislessia e altri DSA a scuola. Strategie efficaci per*

gli insegnanti. Le guide Erickson. Centro Studi Erickson. URL https://books.google.it/books?id=wBXxnh-mjHUC.

Konrad Neuwirth (1991). «TEX in Schools: Just Say No». *TUGboat*. URL https://www.tug.org/TUGboat/tb12-1/tb31kneuwirth.pdf.

Luciano Battaia (2007). «LATEX nella Scuola Media Superiore: applicazioni didattiche con PSTricks». *ArsTEXnica*, (4), pp. 45–50. URL http://www.guitex.org/home/numero-4.

Ministero dell'Istruzione, dell'Università e della Ricerca (2010a). «Indicazioni nazionali degli obiettivi specifici di apprendimento per i licei». URL http://www.indire.it/lucabas/lkmw_file/licei2010/indicazioni_nuovo_impaginato/_decreto_indicazioni_nazionali.pdf.

— (2010b). «Regolamento per gli istituti professionali». URL http://www.indire.it/lucabas/lkmw_file/nuovi_professionali/linee_guida/_LINEE%20GUIDA%20ISTITUTI%20%20PROFESSIONALI_.pdf.

— (2010c). «Regolamento per gli istituti tecnici». URL http://www.indire.it/lucabas/lkmw_file/nuovi_tecnici/INDIC/_LINEE_GUIDA_TECNICI_.pdf.

— (2012). «Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione». URL http://www.indicazioninazionali.it/documenti_Indicazioni_nazionali/indicazioni_nazionali_infanzia_primo_ciclo.pdf.

— (2015a). «Legge 13 luglio 2015, n. 107 Riforma del sistema nazionale di istruzione e formazione e delega per il riordino delle disposizioni legislative vigenti». URL http://www.gazzettaufficiale.it/eli/id/2015/07/15/15G00122/sg.

— (2015b). «Piano Nazionale Scuola Digitale». URL http://www.istruzione.it/scuola_digitale/allegati/Materiali/pnsd-layout-30.10-WEB.pdf.

▷ Rossana Giacopini
Università Cattolica del Sacro Cuore, sede di Brescia
Abilitata TFA II ciclo ex classi A345-346 Inglese
Attestato superamento corso LIS di II livello Ente Nazionale Sordi
rossana dot giacopini at gmail dot com

Il LaTeX come soluzione al problema dell'accesso a testi con formule da parte di disabili visivi

Massimo Borsero, Nadir Murru, Alice Ruighi

Sommario

In questo articolo presentiamo il problema dell'accessibilità a testi con formule da parte di disabili visivi. Forniremo una breve panoramica sulle tecnologie assistive usate e su alcune attuali soluzioni al problema, evidenziandone limiti e difetti. Analizzeremo quindi l'uso del LaTeX sia per la lettura che per la scrittura di testi con formule da parte di disabili visivi, osservando che appare come la soluzione più efficace e promettente. In particolare, vedremo quali sono le caratteristiche che lo rendono molto adatto a questo contesto, il suo insegnamento nelle scuole e il problema di generare documenti PDF accessibili a partire da questo linguaggio (problema ancora aperto).

Sommario

In this paper we present the problem of accessibility of texts with formulae by visually impaired people. We provide a short overview about assistive technologies and about some current solutions of this problem highlighting limits and lacks. We analyze the use of LaTeX regarding both the reading and the writing of texts with formulae by visually impaired people, remarking that it seems to be the more efficient and promising solution. Specifically, we analyze the features that make it very suitable in this context, its teaching in the schools and the problem of creating accessible PDF documents with this language (problem still open).

1 Introduzione

Negli ultimi anni le tecnologie assistive hanno visto incrementare notevolmente il loro sviluppo e la loro diffusione. In particolare, nel campo della disabilità visiva, tecnologie come screen reader, barre Braille e ingranditori (nel caso di ipovisione) consentono ormai un pieno utilizzo del computer in maniera del tutto autonoma. Uno screen reader (letteralmente "lettore di schermo") è un software dotato di sintesi vocale che legge ciò che compare nel monitor del computer e ne consente l'uso tramite comandi da tastiera. Una barra Braille è un hardware che scrive in Braille ciò che viene letto dallo screen reader.

Per mezzo di tali tecnologie, persone con disabilità visiva possono quindi fruire di materiale digitale incrementando notevolmente la possibilità, ad esempio, di intraprendere percorsi di studi, anche a livello universitario. Tuttavia, tali tecnologie mostrano alcuni limiti quando devono maneggiare testi contenenti formule. Per una panoramica si vedano, e.g., Archambault *et al.* (2007) e Armano *et al.* (2014). Questi problemi sono dovuti principalmente al fatto che tali tecnologie sono adatte a interpretare e maneggiare strutture "unidimensionali", ovvero testo in linea, e trovano quindi difficoltà nel caso di elementi non in linea come le formule.

La soluzione consiste nel realizzare testi con formule mediante linguaggi di marcatura (come ad esempio i linguaggi HTML e MathML). In questo modo, persone con disabilità visiva possono essere messe in condizione di leggere testi con contenuti matematici, anche se, come vedremo, a seconda del linguaggio di marcatura usato possono presentarsi comunque alcune problematiche. Un'altra questione molto importante riguarda la scrittura autonoma di testi con formule da parte di persone con disabilità visiva. Linguaggi come il MathML risultano troppo complessi per questo scopo e infatti sono stati sviluppati alcuni linguaggi di marcatura appositamente dedicati a disabili visivi, come il linguaggio LAMBDA (si veda, e.g., Bernareggi (2010)). Utilizzando però un linguaggio appositamente nato e sviluppato per disabili visivi si perde di vista il concetto di inclusività.

In questo contributo, presenteremo l'uso del linguaggio LaTeX come soluzione ai problemi di accessibilità, sia per la lettura che per la scrittura, di testi contenenti formule da parte di disabili visivi. Il linguaggio LaTeX, oltre ad essere un'ottima soluzione a queste problematiche, consente ai disabili visivi di intraprendere un percorso completamente inclusivo, in quanto il LaTeX è il linguaggio più diffuso per la stesura di testi scientifici.

Il presente contributo è strutturato come segue. Nella sezione 2 è presentato in generale il problema dell'accessibilità a testi con formule da parte di disabili visivi, fornendo una breve panoramica su alcuni strumenti per affrontare tale problema (come il linguaggio MathML e il sistema LAMBDA), evidenziandone anche i limiti. La sezione 3 è dedicata alla discussione dell'uso del linguaggio LaTeX come soluzione per l'accesso a testi con formule. In particolare, si discuteranno le caratteristiche che rendono il LaTeX molto adatto in questo contesto (sezione 3.1), il suo possibile insegnamento nelle scuole (sezione 3.2), e quali sono alcuni aspetti ancora da sviluppare che ne incrementerebbero ulteriormente l'uso e l'utilità (sezione 3.3). Infine, nella sezione 4 si trovano le conclusioni.

```
<mrow>

        <mi>a</mi>
        <msup>
            <mi>x</mi>
            <mn>3</mn>
        </msup>
        <mo>+</mo>
        <mi>b</mi>
        <msup>
            <mi>x</mi>
            <mn>2</mn>
        </msup>
        <mo>+</mo>
        <mi>c</mi>
        <mi>x</mi>
        <mo>+</mo>
        <mi>d</mi>
        <mo>=</mo>
        <mn>0</mn>

</mrow>
```

Figura 1: Scrittura in MathML della formula $ax^3 + bx^2 + cx + d = 0$

2 Accessibilità di formule da parte di disabili visivi

Oggigiorno, persone con disabilità visiva beneficiano di screen reader (software che leggono il contenuto del monitor di un computer e ne consentono l'uso tramite comandi da tastiera) potendo così usare in maniera autonoma il computer e accedere a materiale digitale. Gli screen reader più diffusi sono NVDA (gratuito) e JAWS in ambiente Windows, VoiceOver in ambiente Mac e ORCA in ambiente Linux. Tali tecnologie assistive sono adatte a maneggiare solamente strutture in linea e quindi non hanno alcuna difficoltà a leggere il normale testo. Invece, in presenza di strutture non in linea, come immagini e formule, le prestazioni sono ancora non soddisfacenti. Per poter rendere leggibili le formule mediante screen reader è necessario che esse siano scritte mediante un linguaggio di marcatura, che provvede a una scrittura linearizzata della formula e quindi potenzialmente maneggiabile dagli screen reader.

Un linguaggio di marcatura utile in questo senso è il linguaggio MathML (Mathematical Markup Language) che permette di scrivere formule su pagine web. La scrittura di una formula mediante questo linguaggio risulta piuttosto lunga e complessa. Ad esempio una semplice equazione di terzo grado $ax^3 + bx^2 + cx + d = 0$ è realizzata in MathML mediante la serie di comandi nella figura 1. Con questa scrittura però la formula è stata linearizzata, poiché essa corrisponde ora a una serie di righe di solo testo che possono essere lette dagli screen reader. Chiaramente, la lettura delle righe nella figura 1 non rende per nulla comprensibile la formula, in quanto le istruzioni lette sono troppo lunghe e complesse. Tuttavia alcuni screen reader, quando processano una serie di righe corrispondenti a istruzioni MathML, forniscono una traduzione "al volo" di queste righe in un linguaggio più naturale. Ad esempio, nel caso della formula in esame verrà fornita la lettura "a x al cubo più b x al quadrato più c x più d uguale a zero". Tuttavia, con l'uso del linguaggio MathML si aprono molte problematiche. Infatti, le prestazioni degli screen reader nella lettura di formule così scritte sono pesantemente influenzate da una serie di variabili: tipo di screen reader usato, tipo di browser, sistema operativo. In ambiente Mac, la lettura avviene in maniera corretta solo usando lo screen reader VoiceOver e il browser Safari. In ambiente Windows, la lettura avviene in maniera corretta solo usando lo screen reader NVDA, il browser Mozilla Firefox e con il supporto del software gratuito MathPlayer 4. In ambiente Linux, non vi è possibilità di fruire di formule scritte in MathML. Inoltre, è opportuno sottolineare che, vista la complessità di scrittura del linguaggio MathML, difficilmente un autore scriverà un testo web con formule usando direttamente tale linguaggio. Solitamente viene usato un javascript chiamato MathJax che permette di scrivere le formule in maniera più comoda (anche usando il linguaggio LaTeX) e converte poi tali formule in MathML. Tuttavia, l'uso di questo javascript compromette l'accessibilità della formula finale. In questo modo, infatti, essa risulta accessibile solo su ambiente Windows usando lo screen reader JAWS e il browser Internet Explorer o lo screen reader NVDA e i browser Internet Explorer e Mozilla Firefox (ma in questo caso l'accesso è più difficoltoso, in quanto occorre effettuare dei passaggi preliminari non immediatamente agevoli).

Un'altra problematica sull'accessibilità di formule in MathML riguarda la comodità della loro lettura. Abbiamo infatti visto che una semplice equazione di terzo grado (che ha una scrittura molto breve e compatta) viene letta dagli screen reader come "a x al cubo più b x al quadrato più c x più d uguale a zero". La lettura di formule più complesse risulta quindi ancora più lunga e difficoltosa da interpretare. Per ovviare a questo problema esiste la possibilità, per gli utilizzatori del Braille, di usare il Braille matematico, ovvero simboli Braille che corrispondono a simboli matematici, rendendo più rapida e concisa la lettura di una formula. Tuttavia, usando il linguaggio MathML, solo per mezzo dello screen reader NVDA e del browser Mozilla Firefox è possibile ottenere la scrittura della formula in Braille matematico sull'apposita barra. Infine, osserviamo che il linguaggio MathML per la sua complessità non può essere certamente utilizzato da disabili visivi per scrivere testi con formule. Per una panoramica sull'accessibilità di formule su web si veda Bernareggi e Archambault (2007).

```
// x + 1 ⌀ x - 1 \\
```

Figura 2: Scrittura in LAMBDA della formula $\frac{x+1}{x-1}$

Figura 3: Scrittura in Braille della formula $\frac{1+\sqrt{5}}{2}$

Come visto, quindi, l'uso del linguaggio MathML, pur fornendo una risposta ai problemi di accessibilità delle formule, non è ottimale per molti motivi. Sono quindi state sviluppate delle soluzioni ad hoc, come il sistema LAMBDA (Linear Access to Mathematics for Braille Device and Audio–synthesis), Bernareggi (2010). LAMBDA è composto da un linguaggio di marcatura appositamente sviluppato, un editor dedicato e un convertitore MathML. Il linguaggio matematico in LAMBDA è stato sviluppato in modo che ogni simbolo sia tradotto "al volo" in parole. Questa traduzione è implementata nell'editor in modo che possa essere usato da disabili visivi sia per la lettura che per la scrittura della matematica. In LAMBDA, ad esempio, la formula $\frac{x+1}{x-1}$ viene realizzata mediante la riga nella figura 2. I segnalatori grafici di inizio frazione, linea di frazione e fine frazione osservabili nella figura 2 possono essere inseriti mediante scorciatoie da tastiera e vengono tradotti automaticamente a parole o in Braille matematico da screen reader e barre Braille, permettendo così di poter fruire in maniera completa delle formule scritte con il sistema LAMBDA. Ad esempio, per iniziare una frazione è sufficiente premere Ctrl+Shift+Q e per terminarla Ctrl+K. La linea di frazione viene inserita mediante il comando Ctrl+I. Tali comandi possono essere inseriti anche usando il menù dell'editor che è completamente accessibile mediante screen reader. Inoltre, consente di svolgere esercizi di matematica in maniera accessibile. Osserviamo però che anche il sistema LAMBDA ha grossi limiti. Prima di tutto è disponibile solo su ambiente Windows. Poi, essendo un sistema nato e sviluppato per disabili visivi, non è per nulla inclusivo, in quanto gli unici conoscitori di tale linguaggio sono gli stessi utenti disabili. Quindi non è possibile per uno studente, ad esempio, trovare testi che siano stati già scritti in tale linguaggio, ma occorre effettuare delle conversioni spesso difficoltose (se non impossibili) o con risultati insoddisfacenti. Infine, non è adatto per affrontare studi di matematica a livello universitario, poiché la lista dei simboli matematici a disposizione non è completa.

Tutte le problematiche viste sull'accessibilità di formule e possibili soluzioni spingono a valutare e studiare nuove opportunità. Il linguaggio LaTeX appare essere la migliore soluzione possibile, come vedremo meglio nella prossima sezione.

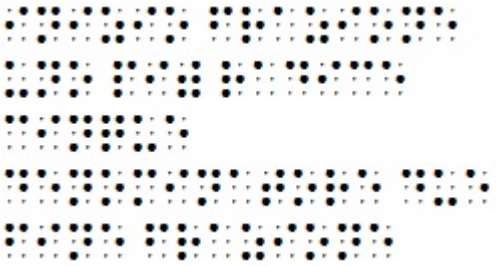


Figura 4: Scrittura in Braille della frase "inizio frazione uno più radice cinque denominatore due fine frazione"

3 Il LaTeX come soluzione per l'accessibilità e l'inclusività

3.1 Le potenzialità del LaTeX

Il LaTeX, essendo un linguaggio di marcatura, è adatto ad essere maneggiato e letto dalle tecnologie assistive. Gli screen reader possono infatti leggere direttamente le formule scritte in LaTeX, dal momento che esse sono linearizzate come nel caso del linguaggio MathML. Nel caso del LaTeX, però, la scrittura di una formula è molto più breve, compatta e comprensibile. Anche persone che non sono disabili visivi che conoscono il LaTeX sono in grado di leggere e capire una formula direttamente dalla sua scrittura in LaTeX. Inoltre, può essere senz'altro usato da persone con disabilità visiva per scrivere testi con formule.

Esistono poi software che ne facilitano l'utilizzo e ne incrementano l'accessibilità. Il più interessante è il software BlindMath (gratuito). Esso è un editor facilitato per il LaTeX, in cui le formule possono essere inserite sia scrivendo direttamente in LaTeX sia usando un menù che consente di inserire le formule senza conoscere i comandi LaTeX. Tale menù è completamente accessibile tramite screen reader e scorciatoie da tastiera. Oltre a fornire un metodo semplificato e accessibile di inserimento delle formule, BlindMath consente anche di avere una lettura semplificata. L'utente può scegliere se leggere direttamente i comandi LaTeX corrispondenti alle formule oppure avere una lettura più "naturale" della formula. Ad esempio la formula $\frac{1+\sqrt{5}}{2}$, può essere letta come "\frac{1+\sqrt{5}}{2}" oppure come "inizio frazione uno più radice cinque denominatore due fine frazione". Un'altra funzione molto utile di BlindMath è la traduzione in Braille matematico delle formule. Ad esempio, la formula precedente viene tradotta in Braille matematico come nella figura 3, che risulta un'espressione molto più compatta rispetto alla traduzione in Braille della frase "inizio frazione uno più radice cinque denominatore due fine frazione", tradotta in Braille nella figura 4.

Utenti disabili visivi possono anche utilizzare editor classici per leggere e scrivere testi in LaTeX. Tra questi, quello caratterizzato dalla migliore accessi-

bilità è TEXnicCenter. L'editor risulta pienamente utilizzabile mediante tecnologie assistive, ad eccezione di alcuni menù, come il menù "Build", dove le finestre sono accessibili ma con alcune difficoltà. Inoltre, anche la finestra "Build Output", dove in fase di compilazione vengono segnalati eventuali errori, è raggiungibile mediante screen reader con difficoltà, ma una volta raggiunta risulta accessibile. Si segnala l'esistenza del software gratuito "latex-access" che consente di attivare (a discrezione dell'utente) la lettura delle formule in linguaggio naturale direttamente nell'editor TEXnicCenter.

Gli ulteriori vantaggi dell'uso del LATEX rispetto a soluzioni come il MathML e LAMBDA riguardano la sua diffusione e la sua inclusività. Infatti, il LATEX è il linguaggio più usato per scrivere documenti scientifici, quindi è semplice reperire o convertire materiale in questo formato. Infine, il LATEX può essere usato e fruito in maniera accessibile su qualsiasi sistema operativo.

3.2 Insegnamento del LATEX nelle scuole

Prima di affrontare brevemente la tematica dell'insegnamento del LATEX nelle scuole, è opportuno un rapido richiamo alle direttrici culturali e legislative entro cui questo insegnamento si inserisce.

1. In primo luogo la cosiddetta *competenza digitale*, che nella Raccomandazione del Parlamento europeo EU0 (2006) è definita come segue: "la competenza digitale consiste nel saper utilizzare con dimestichezza e spirito critico le tecnologie della società dell'informazione (TSI) per il lavoro, il tempo libero e la comunicazione. Essa è supportata da abilità di base nelle TIC: l'uso del computer per reperire, valutare, conservare, produrre, presentare e scambiare informazioni nonché per comunicare e partecipare a reti collaborative tramite Internet".

   Tale competenza è esplicitamente richiamata nel profilo finale delle competenze al termine del primo ciclo di istruzione MIUR (2012) (ovvero a 14 anni), dove si afferma che "lo studente ha buone competenze digitali, usa con consapevolezza le tecnologie della comunicazione per ricercare ed analizzare dati e informazioni, per distinguere informazioni attendibili da quelle che necessitano di approfondimento, di controllo e di verifica e per interagire con soggetti diversi nel mondo". Infine, la recente legge 107/15 "Buona Scuola", di cui è possibile trovare un'analisi critica per esempio in Bruschi (2015), pone tra gli obiettivi primari in carico alle istituzioni scolastiche (comma 7) lo "sviluppo delle competenze digitali degli studenti, con particolare riguardo al pensiero computazionale, all'utilizzo critico e consapevole dei social network e dei media nonché alla produzione e ai legami con il mondo del lavoro". Inoltre (comma 56) "Al fine di sviluppare e di migliorare le competenze digitali degli studenti e di rendere la tecnologia digitale uno strumento didattico di costruzione delle competenze in generale, il Ministero dell'istruzione, dell'università e della ricerca adotta il Piano nazionale per la scuola digitale, in sinergia con la programmazione europea e regionale e con il Progetto strategico nazionale per la banda ultralarga". Il piano ha tra i suoi fini (comma 58) la "realizzazione di attività volte allo sviluppo delle competenze digitali degli studenti, anche attraverso la collaborazione con università, associazioni, organismi del terzo settore e imprese", ed è stato emanato alla fine del 2015 MIUR (2015) comportando un notevole incremento delle risorse dedicate all'educazione digitale.

2. In secondo luogo, il concetto di *inclusione scolastica*. Si definisce "inclusiva una scuola che permette a tutti gli alunni, tenendo conto delle loro diverse caratteristiche sociali, biologiche e culturali, non solo di sentirsi parte attiva del gruppo di appartenenza, ma anche di raggiungere il massimo livello possibile in fatto di apprendimento", adattamento da Booth *et al.* (2002). Il concetto di inclusione è già presente nella celebre legge 104/92 GU1 (1992), ma è posto come fulcro del Sistema Nazionale di Istruzione e Formazione dalla legge 170/10 GU2 (2010) e successive Note applicative, dove si afferma esplicitamente che "va quindi potenziata la cultura dell'inclusione, e ciò anche mediante un approfondimento delle relative competenze degli insegnamenti curricolari, finalizzata ad una più stretta interazione tra tutte le componenti della comunità educante".

Fatte queste debite premesse, l'insegnamento del LATEX si inserisce perfettamente nel vasto dibattito scientifico e culturale che ha portato all'elaborazione delle leggi citate in premessa. Inoltre, a differenza di altri interventi (anche molto costosi) che in passato sono stati spinti più da mode didattiche estemporanee e che, alla prova dei fatti, hanno dimostrato scarsa efficacia (si veda, a riguardo, Calvani (2013)), l'insegnamento del LATEX può essere una risposta concreta (e gratuita) a numerosi problemi di inclusione scolastica di alunni con disabilità visiva e/o con Disturbi Specifici di Apprendimento (DSA).

1. Innanzitutto, si tratta di una strumento intrinsecamente inclusivo *tra studenti*. Infatti, a differenza del Braille o del sistema LAMBDA, conosciuti e compresi solo dai disabili visivi (e pochi educatori specializzati), può essere insegnato con efficacia *all'intera classe*. La sua compatibilità con gli screen reader, con le

modalità esposte in precedenza in questo articolo, lo rende adatto all'uso degli studenti con difficoltà visive (ma anche a diverse categoria di Bisogni Educativi Speciali, come studenti dislessici o con deficit di attenzione), ma al contempo non differenzia l'uso del mezzo tecnologico tra i vari studenti della classe. Questo fatto è un grande aiuto nel consolidamento del gruppo classe, perché lo studente senza problemi di vista realizza che lo studente con disabilità visiva è in grado di apprendere e svolgere compiti con le sue stesse modalità.

2. Poi, è anche uno strumento inclusivo *tra docenti*. Non c'è nulla di peggio, per il successo educativo, di docenti dello stesso consiglio di classe che utilizzano metodologie diverse e/o confliggenti nei riguardi dello stesso studente. Invece il LaTeX, per la sua semplicità di utilizzo (almeno ad un livello elementare) può essere adoperato con profitto non solo da docenti di discipline scientifiche e tecnologiche, ma anche da tutti i docenti del consiglio di classe, e, segnatamente, dai docenti di sostegno.

3. Infine, l'insegnamento del LaTeX si colloca naturalmente nell'alveo dello sviluppo delle competenze digitali, ed è un tipo di tecnologia che, una volta appresa, resta nel bagaglio di conoscenze ed abilità dello studente, indipendentemente dalla sua particolare situazione. Il fatto poi che sia gratuito e liberamente disponibile consente a ogni studente di approfondire e diversificare quanto appreso secondo le necessità (anche lavorative) future senza vincoli economici.

Come notazione finale, si suggerisce di approfittare del particolare momento storico, in cui sono disponibili maggiori fondi per l'educazione digitale e la formazione dei docenti, per organizzare corsi ed attività di diffusione (eventualmente "formando dei formatori") della cultura e delle potenzialità del LaTeX anche, ma non solo, lungo le direttrici sopra esposte.

3.3 Prospettive: pacchetti per la realizzazione di PDF accessibili

Nonostante il linguaggio LaTeX sia largamente diffuso e sia ormai il linguaggio maggiormente usato per la stesura di testi scientifici, la creazione di PDF accessibili mediante il LaTeX risulta ancora un problema aperto. Sottolineiamo che si tratta di una questione molto importante, in quanto la sua soluzione permetterebbe di diffondere documenti PDF con formule accessibili, mentre ad ora la piena accessibilità del documento può avvenire solo fruendo del sorgente LaTeX.

Un generico file PDF creato a partire da file LaTeX, infatti, viene letto dagli screen reader, in modo non del tutto soddisfacente, riscontrando alcuni problemi legati alla sua accessibilità. La nostra analisi si concentra in particolare su tre problemi: il riconoscimento dei livelli di intestazione, la lettura di una lingua diversa da quella principale usata nel testo e la lettura di formule matematiche. Si noti che mentre l'ultimo problema è proprio dei testi di natura scientifica, i primi due sono problemi di carattere più generale.

In un documento PDF le intestazioni sono essenzialmente dei *tag* usati per marcare titoli di sezioni, sottosezioni, ecc. Essi sono particolarmente utili per realizzare documenti accessibili, poiché permettono di navigare in maniera efficiente e comoda un documento. Infatti, se sono presenti i livelli di intestazione, mediante uno screen reader (usando gli appositi comandi da tastiera) il lettore può passare in maniera rapida da una sezione o sottosezione all'altra del documento. Quando viene realizzato un file PDF dal LaTeX, malgrado nel file PDF venga generato un indice delle sezioni e sottosezioni del documento, non vengono create intestazioni riconoscibili dagli screen reader.

Un'altra questione importante per l'accessibilità di un documento riguarda la possibilità da parte degli screen reader di riconoscere la lingua del testo che sta elaborando. Infatti, se gli screen reader riconoscono la lingua in cui è scritta una frase, essa risulterà pronunciata in modo corretto, altrimenti la sua lettura risulta pesantemente compromessa. Quando viene realizzato un file PDF dal LaTeX, risulta riconoscibile da parte degli screen reader solamente la lingua principale del testo. Se nel testo sono presenti frasi in lingua differente dalla principale, esse non vengono pronunciate in maniera corretta, ma lette come se fossero scritte nella lingua principale del documento.

Infine, le formule nel file PDF realizzato a partire da un file LaTeX risultano completamente inaccessibili, ovvero non vengono lette dagli screen reader.

Alcune di queste informazioni si possono trovare per esempio in Archambault e Carpio (2016).

Ad oggi non paiono esistere soluzioni ottimali a queste problematiche, ma abbiamo individuato alcuni risultati soddisfacenti che si servono di alcuni pacchetti LaTeX congiuntamente ad Adobe Acrobat Pro DC (software a pagamento) e che potrebbero essere un punto di partenza per sviluppi futuri.

Un primo pacchetto interessante è accessibility.sty scritto da Babett Schalitz (con documentazione solo in tedesco). Un secondo pacchetto che abbiamo considerato è stato pdfcomment.sty, che serve principalmente per scrivere commenti che possono essere letti sul file PDF. Entrambi i pacchetti sono stati migliorati e implementati da Andy Clifton e Ross Moore, che hanno rispettivamente creato i pacchetti accessibility-meta.sty e mathsem.sty. Tuttavia, entrambi i lavori sono ancora in via di sviluppo, non sono presenti su CTAN e i file messi a disposizione dagli autori creano dei problemi

nella fase di compilazione. Per questi motivi non li abbiamo utilizzati nella nostra analisi, preferendo invece concentrarci sulle potenzialità di accessibility.sty e pdfcomment.sty.

Abbiamo analizzato e testato tali pacchetti mediante un semplice file LaTeX di prova (con lingua principale italiana) composto da alcune sezioni e sottosezioni, una frase in lingua inglese e alcune formule. A partire da questo file di base, abbiamo creato quattro file PDF sfruttando varie combinazioni tra i due pacchetti in esame. Sono stati utilizzati Adobe Acrobat Pro DC e lo screen reader NVDA per analizzare e leggere i file PDF generati. Poiché l'accessibilità di un file è legata alle intestazioni presenti nel documento, ci siamo serviti dello strumento "tag-check" presente in Adobe Acrobat Pro DC. Esso verifica se esistono dei tag nel documento e pertanto indica se il file PDF è dotato di una struttura.

Il primo documento PDF è stato realizzato senza l'utilizzo dei due pacchetti in esame. Da una prima analisi con tag-check esso è risultato privo di intestazioni e tag per la lingua ("PDF con tag: no"). Infatti, il testo del file PDF viene letto dallo screen reader NVDA (ad eccezione delle formule) senza individuare livelli di intestazione e risultando così non navigabile. Utilizzando il menù "Accessibilità" di Acrobat è possibile aggiungere dei tag al file (in modo che tag-check fornisca un risultato positivo, ovvero "PDF con tag: sì"), ma non ottenendo il risultato sperato dal punto di vista dell'accessibilità. Infatti, usando NVDA il documento risulta ancora non navigabile, la frase in inglese viene letta con pronuncia italiana e le formule non accessibili.

Il secondo file PDF è stato creato usando il solo pacchetto accessibility.sty. In questo caso, tag-check di Acrobat fornisce un risultato positivo ("PDF con tag: sì"). Tuttavia, testando il file PDF con NVDA, si ottengono gli stessi risultati precedenti. Ora però, aggiungendo i tag con il menù "Accessibilità" di Acrobat, si è ottenuto un documento con intestazioni riconosciute da NVDA, con cui risulta quindi possibile navigare il documento (malgrado alcune imperfezioni). La frase in inglese e la formula continuano però ad avere gli stessi problemi descritti per il file precedente.

Il terzo file PDF è stato creato usando il solo pacchetto pdfcomment.sty. Come nel caso del primo file, da un'analisi con tag-check risulta privo di intestazioni e l'aggiunta dei tag con Acrobat non fornisce alcun risultato dal punto di vista dell'accessibilità. Tuttavia, utilizzando i comandi

```
\pdfmarkupcomment[opzioni]{testo}
  {commento}
```

e

```
\pdftooltip{testo}{commento}
```

è possibile commentare il contenuto matematico. Se si usa il comando `\pdfmarkupcomment`, sul file PDF apparirà l'icona di un fumetto a fianco del testo commentato, se invece si usa il comando `\pdftooltip` il commento è meno invasivo poiché non ci sono icone ad indicare dov'è il commento ed esso rimane nascosto. Inserendo tali commenti le formule risultano accessibili, in quanto gli screen reader leggono i commenti presenti nel file PDF. Questa soluzione non è però certamente ottimale. Prima di tutto NVDA, prima di leggere il contenuto del commento, quando incontra la formula la legge in maniera errata creando un po' di confusione. In secondo luogo, trattandosi di un commento (ovvero di normale testo), non è possibile in alcun modo attivare i simboli Braille per la matematica. Inoltre, nel commento sono ammesse solo normali lettere, rendendo la lettura di una formula molto lunga (non è possibile ad esempio scrivere come commento il comando LaTeX corrispondente alla formula). Infine, tali commenti devono essere inseriti manualmente dall'autore del testo, rendendo la procedura molto lunga e tediosa.

Il quarto file PDF è stato creato usando entrambi i pacchetti, ottenendo i risultati più soddisfacenti per quanto riguarda il riconoscimento della struttura del documento e delle formule matematiche. Infatti, in questo caso (con l'aiuto di Acrobat come visto in precedenza) si ottiene un file PDF con intestazioni e con formule accessibili (anche se non in maniera del tutto soddisfacente come appena visto).

Per risolvere il problema legato agli errori di lettura dovuti alla lingua inglese, è possibile solamente agire manualmente sui tag del file PDF mediante Acrobat. Infatti, una volta espansa tutta la struttura, si può lavorare sulle preferenze dei singoli tag per modificare la lingua e il tipo di elemento marcato. Si noti che per modificare la lingua di una parte del testo è necessario "spezzare" il capoverso creandone uno nuovo con il testo da modificare. Ciò è necessario perché all'interno di ogni capoverso è possibile impostare una sola lingua di lettura.

In conclusione di questa analisi possiamo affermare che ad oggi esistono alcune parziali soluzioni ai problemi legati all'accessibilità dei documenti PDF generati con LaTeX, ma queste sono ben lontane dall'essere ottimali. Innanzi tutto, come notato precedentemente, non sono totalmente perfette nei risultati. In secondo luogo necessitano l'uso di Acrobat Reader Pro DC, software a pagamento e pertanto non fruibile da tutti. Infine, si tratta di soluzioni che richiedono un grande lavoro a posteriori. Pertanto l'obiettivo auspicato è quello di sviluppare dei pacchetti per l'accessibilità che non richiedano l'utilizzo di strumenti esterni e che realizzino un file PDF totalmente accessibile direttamente dopo la compilazione del file LaTeX.

4 Conclusione

Nell'articolo abbiamo esaminato il problema dell'accessibilità di testi con formule da parte di disabili visivi. Abbiamo visto che alcune attuali soluzioni

come il linguaggio MathML e il sistema LAMBDA non sono completamente soddisfacenti per vari motivi: il MathML è un linguaggio molto complesso e la sua accessibilità è soggetta a molte variabili e non sempre è ottimale; il sistema LAMBDA non è inclusivo, può essere usato solo su sistemi Windows, è poco diffuso ed è limitante per affrontare matematica ad un livello universitario. Il LaTeX invece appare essere un linguaggio completamente accessibile (sia per leggere che per scrivere testi con formule) e inclusivo (anche in vista del suo insegnamento nelle scuole), utilizzabile su tutti i sistemi operativi ed esistono software (gratuiti) che ne facilitano l'uso e ne incrementano l'accessibilità. Il suo insegnamento nelle scuole appare un passaggio ormai realizzabile nell'ottica dell'insegnamento di linguaggi di marcatura (come il linguaggio HTML) e per via, come detto, della sua inclusività. Inoltre, sottolineiamo che tra i nuovi insegnanti di matematica il linguaggio LaTeX è molto conosciuto. Si segnala a questo riguardo la tesi P.A.S. (Percorsi Abilitanti Speciali, Bracco (2015)), in cui è stato sperimentato con successo l'insegnamento del LaTeX a una classe di prima liceo, per facilitare gli studi di una studentessa con disabilità visiva presente nella classe. Infine, mediante il LaTeX è anche possibile generare documenti PDF parzialmente accessibili. Risulta un problema ancora aperto la generazione di PDF completamente accessibili.

5 Ringraziamenti

Il presente lavoro è svolto nell'ambito di una convenzione tra l'I.Ri.Fo.R./UICI (Istituto per la Ricerca, la Formazione e la Riabilitazione/Unione Italiana Ciechi e Ipovedenti) e l'Università degli Studi di Torino.

Si ringraziano la dott. Armano e la prof. Capietto per il supporto e il prof. Andretta per averci stimolato nel presentare questo contributo.

Riferimenti bibliografici

(1992). «Legge-quadro per l'assistenza, l'integrazione sociale e i diritti delle persone handicappate (104/92)». *Gazzetta ufficiale della Repubblica italiana*, **39**.

(2006). «Raccomandazione del parlamento europeo e del consiglio del 18 dicembre 2006 relativa a competenze chiave per l'apprendimento permanente (2006/962/ce)». *Gazzetta ufficiale dell'Unione europea*, **30** (2006), pp. 10–18.

(2010). «Nuove norme in materia di disturbi specifici di apprendimento in ambito scolastico (170/10)». *Gazzetta ufficiale della Repubblica italiana*, **244**.

Archambault, D. e Carpio, M. (2016). «Developing a culture of accessibility in a university context». In *Conference on Digitization and E-Inclusion in Mathematics and Science*. Kanagawa, Japan.

Archambault, D., Stoger, B., Fitzpatrick, D. e Miesenberger, K. (2007). «Access to scientific content by visually impaired people». *Upgrade*, (2), p. 14.

Armano, T., Capietto, A., Illengo, M., Murru, N. e R., R. (2014). «An overview on ict for the accessibility of scientific texts by visually impaired students». In *Conference SIREM-SIE-L*. Perugia, Italy.

Bernareggi, C. (2010). «Non–sequential mathematical notations in lambda system». *Computers Helping People with Special Needs, Lecture Notes in Computer Science*, **6180**, pp. 389–395.

Bernareggi, C. e Archambault, D. (2007). «Mathematics on the web: emerging opportunities for visually impaired people». *Proc. of the 2007 International Cross–Disciplinary Conference on web Accessibility (W4A)*, pp. 108–111.

Booth, T., Ainscow, M., Black-Hawkins, K., Vaughan, M. e Shaw, L. (2002). «Index for inclusion». *Developing learning and participation in schools*.

Bracco, M. (2015). «Apprendimento della matematica e disabilità visiva: ostacoli ed opportunità. Un caso di studio». *Tesi Finale P.A.S.*

Bruschi, M. (2015). *La legge n.107 commentata e leggi della scuola a confronto*. Edises.

Calvani, A. (2013). «Le tic nella scuola: dieci raccomandazioni per i policy maker». *Formare*, **13** (4), p. 30.

MIUR (2012). «Indicazioni nazionali per il curricolo per la scuola dell'infanzia e del primo ciclo di istruzione».

— (2015). «Decreto n. 851 del 27 ottobre 2015».

▷ Massimo Borsero
Dipartimento di Matematica "G. Peano", Università degli Studi di Torino
massimo dot borsero at unito dot it

▷ Nadir Murru
Dipartimento di Matematica "G. Peano", Università degli Studi di Torino
nadir dot murru at unito dot it

▷ Alice Ruighi
Dipartimento di Matematica "G. Peano", Università degli Studi di Torino
alice dot ruighi at unito dot it

Importanza di LaTeX per utenti con deficit visivo

Michela Montrasio

Sommario

L'obbiettivo di questo articolo è quello di spiegare in che modo il linguaggio LaTeX, usato come sistema di videoscrittura, possa essere di grande aiuto a persone con deficit visivo che intendano occuparsi di matematica.

Per questo motivo, partendo dalla mia esperienza di studentessa di matematica non vedente, racconterò come ho utilizzato LaTeX nel mio percorso di studi soffermandomi anche sulle difficoltà incontrate; descriverò poi alcune strategie adottate per velocizzare la lettura e la scrittura con questo sistema; infine cercherò di mettere in luce le potenzialità di questo linguaggio rispetto ad altri sistemi di scrittura della matematica accessibili a utenti non vedenti.

Abstract

The aim of this article is to explain in which way LaTeX, used as system to write Mathematics, can be useful to visually impaired people who are going to deal with this subject.

For this reason, starting from my own experience as a blind Math student, I am going to relate how I have been using LaTeX during my studies, focusing on the difficulties I met, too; then, I am going to describe some strategies I have adopted, in order to quicken my reading and writing with this system; finally, I am going to highlight the power of LaTeX when compared to other systems to write Mathematics, accessible to blind users.

1 Introduzione

Da studentessa ventiduenne non vedente iscritta alla Laurea Magistrale in Matematica presso l'Università degli Studi di Milano-Bicocca, ho sentito parlare per la prima volta di LaTeX al momento della mia immatricolazione, in quanto mi è stato proposto come soluzione dai miei docenti nel momento in cui ho presentato loro le problematiche legate alla scrittura di formule matematiche in una forma a me leggibile. In questo articolo spiegherò in che modo ho utilizzato il linguaggio LaTeX durante i miei studi universitari.

2 Strumenti informatici utilizzati

Per rendere più comprensibile il resto di questo articolo, comincio con un'introduzione riguardante gli strumenti che permettono a una persona non vedente di utilizzare un PC.

L'ausilio principale è un dispositivo chiamato *barra braille* (figura 1), di forma rettangolare, costituito da una sequenza di celle, che viene collegato al PC: su quest'ultima, grazie ad appositi software detti *screen reader*, viene visualizzato in codici braille, riga per riga, il contenuto (testuale) dello schermo.

Premesso ciò, nello studio della matematica, non solo a livello universitario, i non vedenti incontrano immediatamente una difficoltà pratica, che riguarda la rappresentazione in forma scritta di formule ed espressioni. Infatti, il linguaggio braille ha una struttura di tipo sequenziale e qualsiasi informazione deve essere rappresentata in maniera lineare. Questo non è un problema nel caso di informazioni testuali, ma costituisce un grosso ostacolo nella scrittura di formule matematiche, in cui spesso si utilizzano strutture che richiedono una rappresentazione su più livelli (ad esempio frazioni, matrici...).

3 LaTeX come soluzione

Durante i miei studi universitari LaTeX è stato per me un'ottima soluzione al problema esposto sopra: infatti, con questo linguaggio, le formule e i simboli grafici in esse contenuti vengono rappresentati attraverso stringhe di simboli alfanumerici, perfettamente leggibili con la barra braille.

LaTeX mi è quindi risultato comodo per vari motivi: in primo luogo, poiché la maggior parte dei docenti universitari lo utilizza per redigere il proprio materiale, mi è stato possibile accedere a dispense, esercizi preparati dai miei professori, attraverso la lettura dei file sorgenti (il file pdf prodotto con la compilazione non è accessibile con la barra braille per i motivi esposti sopra). In secondo luogo, ho potuto scrivere a mia volta appunti, esercizi ed esami in questo linguaggio, e, compilando i miei file in pdf, il mio materiale risulta facilmente leggibile da chiunque. Per dare un esempio, riporto in figura 2 una schermata del mio PC durante l'editazione di appunti, che mostra una parte del relativo file sorgente in TeX (preciso che mi è stato possibile prendere appunti durante le lezioni chiedendo ai miei professori di dettare tutto ciò che scrivevano alla lavagna).

Come emerge da quanto detto sopra, ho usato LaTeX per uno scopo un po' diverso da quello per cui questo sistema è nato, utilizzandolo semplicemente come sistema di videoscrittura.

Oltre a permettermi la scrittura della matematica, ho trovato LaTeX molto comodo anche per

FIGURA 1: Barra braille TouchMe 5

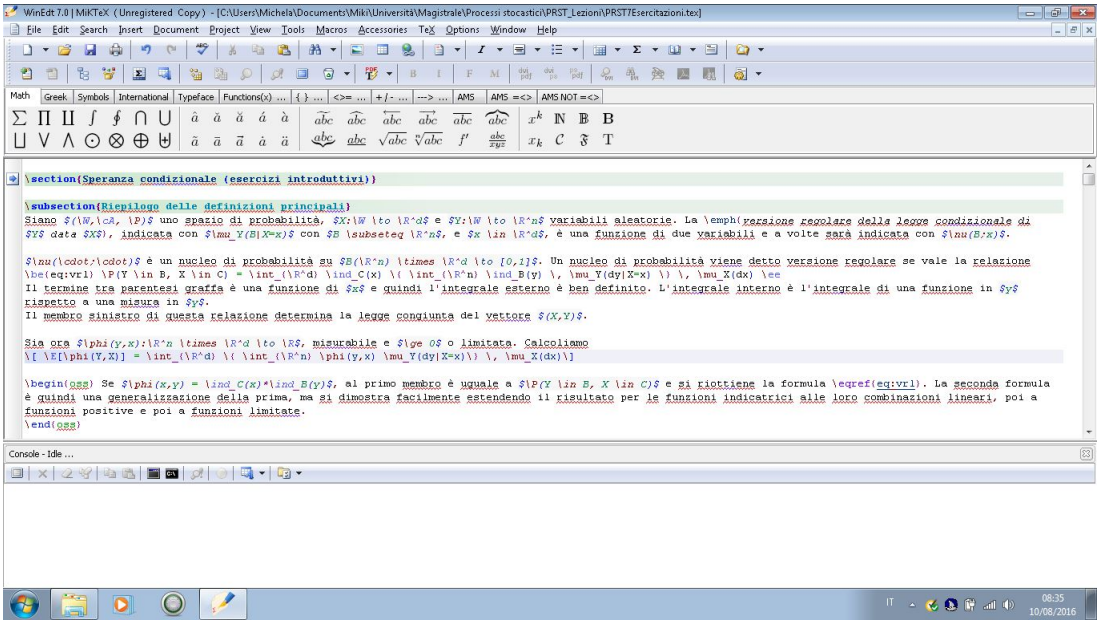


FIGURA 2: Schermata tratta dagli appunti del corso di Processi Stocastici

gestire l'impaginazione dei documenti, e per la creazione di tabelle e schemi di vario tipo, che risultano invece laboriosi se creati ad esempio con Microsoft Word o con altri programmi.

4 Rappresentazione di grafici

C'è un altro aspetto che ancora non ho discusso e che è rilevante nello studio di alcuni campi della matematica: lo studio di grafici e figure. Come si può immaginare, la barra braille non permette la lettura di immagini di nessun tipo, e neanche l'uso di LATEX risolve tale questione, ad eccezione di diagrammi semplici.

Tuttavia, ho a disposizione alcuni strumenti per poter disegnare grafici in rilievo: quello che ho utilizzato maggiormente è un piano di gomma (figura 3), su cui si possono appoggiare dei fogli lucidi; tracciando linee su tali fogli, esse rimangono in rilievo.

FIGURA 3: Piano di gomma con fogli lucidi per realizzare disegni in rilievo

5 Strategie per velocizzare la scrittura

Per cercare di portarmi avanti, ho imparato le funzioni base di LATEX poco prima di iniziare l'università, leggendo qualche guida e provando a scrivere qualche esercizio, ma i veri passi avanti con questo linguaggio li ho fatti solo quando mi sono trovata direttamente ad affrontare situazioni come prendere appunti durante le lezioni, leggere le dispense scritte dai docenti e così via; in tali momenti infatti ho iniziato ad elaborare qualche strategia per risolvere le difficoltà che mi si presentavano di volta in volta.

Un primo problema è stato quello di acquisire velocità nella scrittura, in modo da poter prendere rapidamente appunti durante le lezioni e non essere troppo lenta nello svolgimento degli esami; a tale proposito naturalmente la possibilità offerta da LATEX di creare abbreviazioni per i vari comandi (attraverso le stringhe `\newcommand` e `\renewcommand`) mi ha aiutato molto, anche perché, utilizzando comandi abbreviati, anche la lettura dei miei file sorgente, su cui devo studiare, mi risulta più scorrevole.

Un'altra soluzione efficace mi è stata offerta dall'editor di testo *WinEdt*, che ho utilizzato per scrivere i miei documenti in LATEX: infatti, tale editor consente anche di associare ad una combinazione di tasti scelta dall'utente una stringa di caratteri; in tal modo con un unico comando da tastiera è possibile digitare in un colpo solo anche comandi molto lunghi e contenenti caratteri speciali come parentesi graffe e virgolette. Per fare un esempio, sul mio PC ho fatto in modo che, premendo CTRL + SHIFT + f venga digitato il comando `\frac{}{}`, e che il cursore mi si posizioni dopo la prima parentesi graffa, così da poter digitare subito il numeratore.

WinEdt contiene inoltre alcuni menù attraverso cui è possibile inserire le strutture tipiche del linguaggio LATEX come matrici, elenchi puntati e ambienti per l'inserimento di formule matematiche.

6 Lettura dei documenti

Un'altra questione riguarda la lettura dei file sorgenti del materiale preparato da altre persone: come tutti sanno, in ogni file scritto in TEX ci sono molti comandi che non hanno nessuna rilevanza dal punto di vista della sostanza, legati alla spaziatura, all'impaginazione e alla formattazione del testo. La lettura del sorgente può quindi diventare dispersiva perché si è costretti a fare un'operazione di "filtraggio", separando ciò che riguarda la sostanza da ciò che riguarda la forma finale del documento e che può essere tranquillamente ignorato ai fini della comprensione. Inoltre, ogni persona usa abbreviazioni personalizzate per i vari comandi nei propri file sorgente, e quindi, ogni volta che ricevo del materiale scritto da una persona nuova, devo abituarmi a nuovi comandi e a un nuovo modo di impostare il documento.

6.1 Conversione da LATEX a html

All'inizio, per risolvere questo problema, alcuni docenti della mia università hanno ideato un modo per convertire una dispensa scritta in LATEX in un archivio html; il programma utilizzato per questo scopo è stato creato combinando LATEX con i software MathJax e PlasTEX.

Tale software, partendo da una dispensa scritta in TEX, fornisce come output un suo archivio html, leggibile con i browser più comuni e soddisfacente le regole di accessibilità delle pagine web: in quest'ultimo le formule matematiche vengono conservate inalterate, scritte in LATEX, oppure convertite in linguaggio mathml (vedi paragrafo 7.3), secondo le preferenze dell'utente; nel caso in cui si scelga di mantenerle in TEX, le abbreviazioni dei comandi vengono risostituite con i comandi interi, inoltre i comandi di formattazione vengono eliminati (ad esempio, l'ambiente itemize viene omesso,

e al posto dei comandi `\item` viene anteposto un asterisco alle varie voci dell'elenco). In aggiunta, all'inizio del file html compare un indice di link cliccabili che, oltre a permettermi di capire in quanti capitoli e paragrafi è suddiviso il documento, mi consente di raggiungere la sezione del documento desiderata cliccando sul suo titolo. Per dare un'idea dell'output del convertitore, riporto in figura 4 una schermata degli appunti di geometria 1 forniti dal mio professore.

Purtroppo questo software non funziona con tutti i file LaTeX, non è facile da utilizzare e la conversione di una dispensa richiede un lavoro molto lungo. Io non ho mai imparato a usarlo autonomamente (anche perché non funziona in ambiente Windows), ma nei primi anni di università i miei professori hanno convertito per me da LaTeX ad html dispense e libri di testo con questo software.

6.2 Strategie di lettura usate ora

Il sistema descritto sopra mi è stato molto d'aiuto nei primi tempi, ma ora che ho un po' più di esperienza e che ho acquisito più familiarità con il linguaggio LaTeX ho finalmente superato i vari problemi di lettura, e convertire gli archivi LaTeX in html per me non è più fondamentale; ora mi risulta più immediato capire quali parti del documento TeX ignorare e a quali prestare attenzione, e, con il tempo, ho capito come muovermi in un modo agevole all'interno del preambolo dei vari documenti e come risalire al comando associato ad una certa abbreviazione; bastano poche indicazioni da chi mi manda il materiale per capire in che modo è strutturato l'archivio, e in quanti capitoli è suddiviso. Infine, per quanto riguarda la navigazione all'interno dei file sorgenti mi risultano molto comodi i comandi messi a disposizione dall'editor WinEdt, che permettono di cercare una determinata stringa di testo all'interno di un file, o di posizionarmi su una certa riga del documento digitandone il numero.

C'è però un altro motivo per cui il convertitore da LaTeX a html è risultato utile, e riguarda i libri di testo. Alcuni autori giustamente non si fidano a fornire agli utenti non vedenti i sorgenti dei loro libri di testo per motivi di sicurezza (un utente potrebbe farne un uso scorretto, modificare il sorgente a proprio piacimento e rimettere in circolazione il libro di testo in rete). In una situazione di questo tipo un autore ha preferito usare il convertitore e passarmi il suo libro sotto forma di archivio html non modificabile.

7 Confronto con altri sistemi di videoscrittura

LaTeX non è la modalità più utilizzata dai non vedenti per scrivere la matematica, e ora analizzerò alcuni software utilizzati prima dei miei studi universitari confrontandoli con LaTeX.

7.1 Codice personalizzato

Fino alla scuola media per scrivere la matematica per me è stato sufficiente utilizzare un qualsiasi editor di testo, e concordare alcune convenzioni con i miei professori: ad esempio, per scrivere una frazione, racchiudevo numeratore e denominatore in blocchi distinti di parentesi, separati da uno slash; sostituivo inoltre i vari simboli matematici con caratteri alfanumerici scelti da me o con un'abbreviazione del loro nome (la figura 5 mostra alcuni esercizi scritti in questo modo).

Questa modalità di lavorare consisteva, in un certo senso, in una creazione di un linguaggio di videoscrittura personalizzato, e che non è troppo distante dal modo in cui ho poi utilizzato LaTeX (anche se, ovviamente, il materiale che producevo era comprensibile solo alle persone con cui avevo stabilito le convenzioni, e la sua lettura non era scorrevole per una persona abituata alla matematica nella forma grafica usuale).

Talvolta è stato comodo utilizzare questo sistema anche durante l'università perché consente una scrittura più rapida rispetto al LaTeX: in particolare ho stabilito una serie di convenzioni con alcune persone, che le hanno utilizzate per rendermi accessibili le formule contenute in alcuni libri di testo.

7.2 LAMBDA

Successivamente sono stati creati programmi con una doppia funzionalità:

1. descrivere formule anche complesse attraverso appositi simboli braille, in modo che anch'esse vengano rappresentate su un unico livello e lette dall'utente attraverso la barra braille;

2. in un secondo momento, visualizzare sullo schermo, nella forma grafica usuale, le formule digitate dall'utente.

Pertanto, durante la scuola superiore, ho utilizzato anch'io un programma di questo tipo, chiamato *LAMBDA* (il nome sta per Linear Access to Mathematics for Braille Device and Audio- synthesis). Esso è un sistema di scrittura matematica per computer progettato espressamente per l'uso con display Braille e sintesi vocale. Il primo prototipo è stato sviluppato all'interno di un progetto di ricerca europeo al quale partecipano 15 partner di 8 paesi diversi ed è basato su un sistema di scrittura lineare: le formule matematiche sono scritte in modo testuale con una sequenza regolare di caratteri della stessa dimensione.

In particolare, LAMBDA è formato da un linguaggio di mark-up, da un editor e da un convertitore, e permette di compiere le seguenti operazioni:

- **digitare nell'editor**, utilizzando la tastiera, **formule matematiche**, che vengono visua-

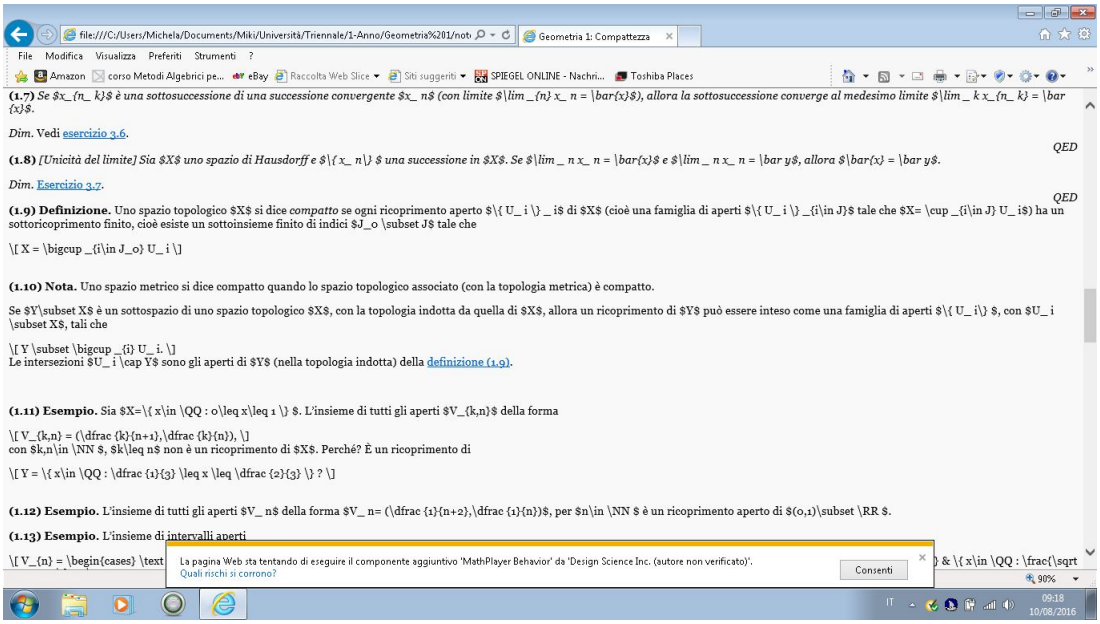


FIGURA 4: Parte degli appunti di topologia relativi alla compattezza

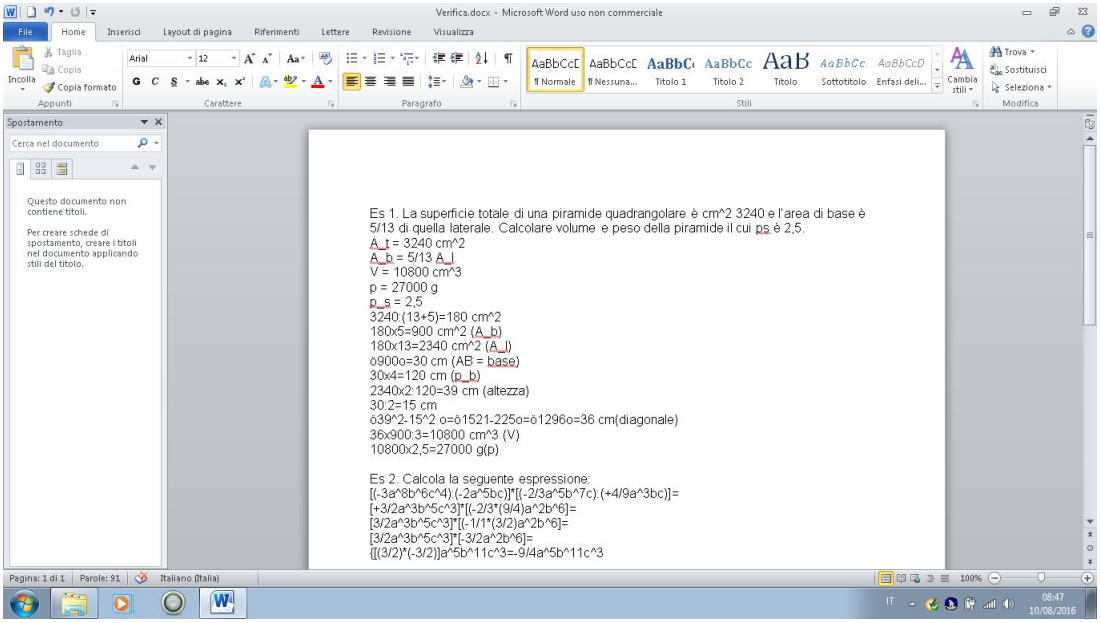


FIGURA 5: Due esercizi tratti da una verifica di terza media. Ho scelto di racchiudere l'argomento della radice quadrata all'interno di *ö . . . o* perché i caratteri braille corrispondenti richiamano il carattere usato nel braille matematico per la radice.

lizzate sulla barra braille attraverso il braille matematico a 8 punti; la figura 6 mostra parte di un esercizio scritto con l'editor di LAMBDA.

- **visualizzare istantaneamente le formule in formato grafico**; la figura 7 mostra l'esercizio dell'immagine precedente tradotto nella scrittura matematica usuale.
- **importare nell'editor formule scritte con equation editor** da utenti vedenti (in genere dai professori).
- **esportare le formule digitate in MathML** (vedi paragrafo 7.3).

Da un lato LAMBDA ha una sintassi molto più immediata rispetto a quella di LaTeX (ogni simbolo matematico richiede al più due caratteri braille), e la lettura di un documento scritto con LAMBDA è molto più scorrevole per una persona non vedente; dall'altro lato però le strutture matematiche offerte da LAMBDA diventano insufficienti per affrontare lo studio della matematica a livello universitario.

Ciò che mi ha portato a preferire LaTeX rispetto a LAMBDA è il fatto che quest'ultimo è un software progettato esclusivamente per utenti non vedenti, e quindi trovare documentazione scritta in questo formato è molto difficile, a meno che non venga preparata appositamente. LaTeX invece è ampiamente utilizzato a livello mondiale e quindi, nella maggior parte dei casi, la documentazione in questo formato è già esistente e non dev'essere preparata apposta per me.

7.3 Linguaggio MathML e descrizioni fornite da MathPlayer

L'ultimo sistema di videoscrittura che presento è il linguaggio di mark-up *MathML*, a cui ho accennato precedentemente.

MathML (acronimo di Mathematical Markup Language) è un linguaggio derivato dall'XML usato per rappresentare simboli e formule matematiche, in modo che possano essere presentati in una forma chiara nei documenti web.

Se una pagina web contiene formule scritte in MathML, un plugin detto *MathPlayer* le rende leggibili anche con screen reader e barra braille: infatti, mentre sullo schermo tali equazioni compaiono nella forma grafica usuale, Mathplayer fa in modo che sulla barra braille, al loro posto compaia una loro descrizione a parole, in inglese.

Io credo che avere della documentazione con formule scritte in MathML sia un'ottima cosa per un non vedente, anche se la sintassi delle formule fornita da MathPlayer è molto più prolissa rispetto a quella di LaTeX.

Per fare un esempio, la formula

$$\sqrt{\frac{x}{x-1}} \ge 0 \qquad \forall x \ge 0$$

che nel linguaggio LaTeX si legge come

```
\[ \sqrt{\frac{x}{x-1}} \ge 0\qquad
\forall x \ge 0\]
```

viene tradotta da Mathplayer in un modo simile al seguente:

```
square root of, xi, over, xi minus 1, end
square root, greater than, 0, for all,
xi, greater than, 0
```

Inoltre, produrre file in linguaggio MathML è molto meno immediato che con LaTeX.

Purtroppo il linguaggio MathML non è utilizzato in tutti i siti internet con contenuti matematici, in molti di essi le formule vengono inserite semplicemente come immagini e quindi ovviamente risultano inaccessibili con screen reader; nella maggior parte dei casi poi, appunti ed esercizi di matematica sono caricati sulla rete in file pdf.

In alcune occasioni è invece stato possibile accedere al materiale caricato sulla rete grazie a LaTeX, infatti alcuni siti lo utilizzano per redigere formule. Ad esempio su wikipedia le pagine di contenuto matematico vengono scritte attraverso un linguaggio detto *wikitesto* che è identico a LaTeX per quanto riguarda i comandi di matematica e ne differisce per i comandi di formattazione. Inoltre, alcuni siti contenenti articoli di matematica di alto livello, come `arxive.org`, rendono scaricabile, oltre al file pdf, anche il file sorgente.

8 Conclusione

Concludo questo articolo con alcune osservazioni e precisazioni.

La prima riguarda un altro sistema che alcuni non vedenti scelgono per poter usare il PC, in sostituzione di (o congiuntamente a) display Braille: la *sintesi vocale*, a cui lo screen reader invia informazioni, in modo che legga il contenuto dello schermo. Nell'articolo non ho parlato di questo sistema perché personalmente non lo trovo molto comodo, soprattutto per studiare la matematica: in un documento LaTeX, per esempio, il sintetizzatore legge ogni singolo carattere senza comprendere il significato dei comandi, perciò la lettura di una formula ad alta voce diventa molto lunga e fa perdere la visione complessiva, che invece si ottiene rapidamente leggendo autonomamente. Sono comunque in corso ricerche per creare sintetizzatori sempre più sofisticati che danno un'intonazione alle formule lette.

Sottolineo che in questo percorso di studi, computer e barra braille sono stati di fondamentale importanza: io credo che senza l'aiuto di un PC, per me non sarebbe stato possibile nemmeno studiare la matematica ad un livello liceale, o comunque sarebbe stato *molto* più faticoso, e quindi sono stata molto fortunata ad avere l'opportunità di sfruttare questi strumenti potenti!

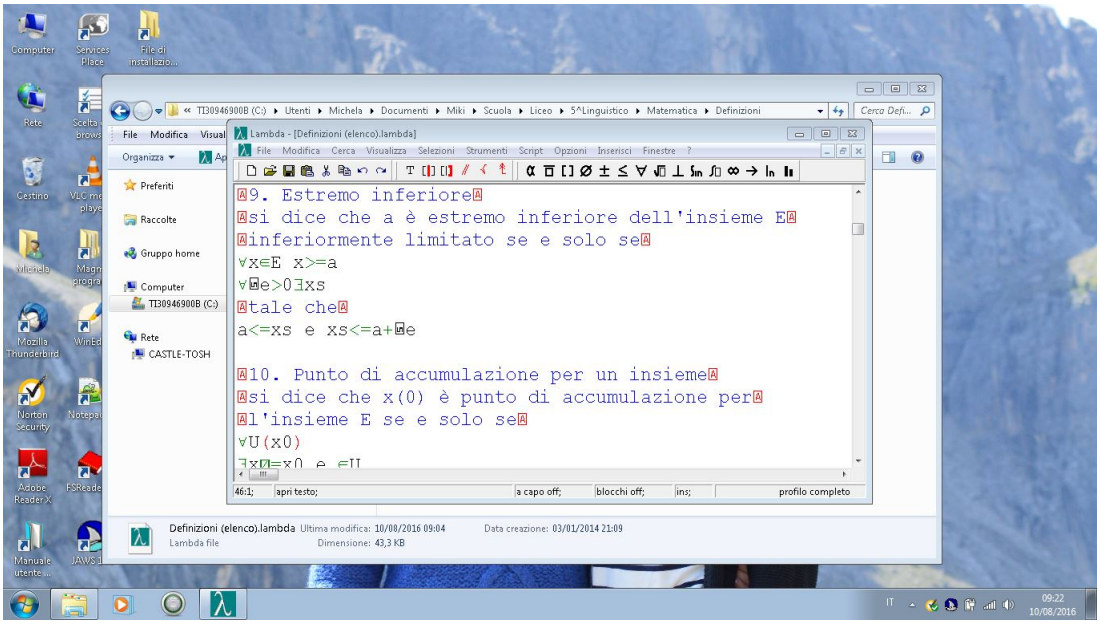


FIGURA 6: Appunti di analisi: alcune definizioni (editor)

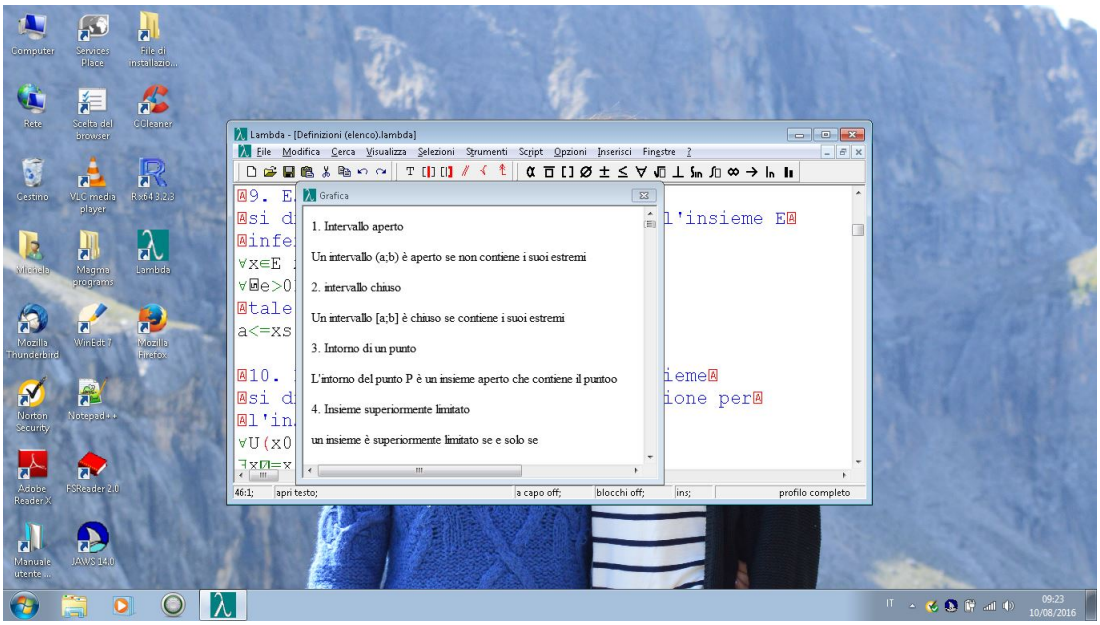


FIGURA 7: Appunti di analisi: alcune definizioni (modalità grafica)

Concludo dicendo che il motivo per cui ho apprezzato di più LaTeX è il seguente: esso ha favorito la mia comunicazione in ambito universitario con docenti e altri studenti, permettendoci di scambiarci materiale facilmente utilizzabile da entrambi.

Il fatto di trovarmi in un ambiente in cui LaTeX viene ampiamente usato, e di aver incontrato docenti che, con grandissima disponibilità, mi hanno dato tanto materiale in TeX e proposto metodi per migliorare, mi ha motivato e incoraggiato ad imparare al più presto questo linguaggio, nonostante gli sforzi iniziali, perché la fatica che, soprattutto nei primi tempi, ho incontrato nell'abituarmi a LaTeX è stata sicuramente ricompensata!

Riferimenti bibliografici

AA.VV. «Lambda». www.veia.it/it/prodotto_lambda.

Archambault, D. «Non visual access to mathematical contents: State of the art and prospective». http://inftyreader.org/inftyreader-kb/Non%20Visual%20Access%20to%20Mathematical%20Contents%20-%20State%20of%20the%20Art%20and%20Prospective.pdf, Università Pierre e Marie Curie di Parigi.

Armano, T. e Capietto, A. «Per una matematica accessibile e inclusiva». http://www.integr-abile.unito.it/documenti/posterbologna.pdf, Dipartimento di Matematica G. Peano dell'Università di Torino.

Artico, G. «Il TeX e i non vedenti». http://www.artico.name/access/, Dipartimento di Matematica dell'Università di Padova.

Ferrario, D. L. «xhtmlate: a latex-to-xhtml+mathjax conversion tool.» http://www.matapp.unimib.it/~ferrario/var/x.html. Dipartimento di Matematica e Applicazioni dell'Università degli Studi di Milano-Bicocca.

▷ Michela Montrasio
Università Bicocca Milano
michela underscore montrasio
at fastwebnet dot it

LuaTEX + pdfliteral

Roberto Giacomelli

Sommario

\pdfliteral è un comando ben conosciuto del motore di composizione pdftex con quasi lo stesso significato del comando \special di TEX. Quello che si ottiene con queste primitive è l'inclusione nel documento finale di materiale grezzo che poi il driver potrà interpretare.

In questo articolo verrà esaminato da un punto di vista pratico come lo sviluppatore LuaTEX programmando esclusivamente in Lua la libreria node, può efficacemente inserire materiale pdfliteral per produrre semplice grafica.

Abstract

\pdfliteral is a well known command of the typesetting engine pdftex with nearly the same meaning of TEX's \special. What these commands do is to insert raw material in the final document that then the device drivers can later interpret.

In this paper it will be examined with a practice point of view how the LuaTEX developer can insert pdfliteral material programming exclusively in Lua by means of the node library to produce simple graphic figures.

1 Motivazione

In queste settimane sto lavorando a un nuovo pacchetto che per il momento terrò segreto. Una delle necessità funzionali di questo lavoro mi ha portato a interessarmi delle primitive del formato PDF per produrre con la massima efficienza possibile semplice grafica come linee o rettangoli.

Oltre a questo, per alcune semplici considerazioni, ho deciso di sviluppare il pacchetto in LuaTEX e perciò è nata la seguente domanda: programmando solamente in Lua esiste un modo di disegnare usando direttamente i codici nativi PDF?

La risposta è si e nelle prossime sezioni ne troverete una conferma. Nella prima parte dell'articolo illustrerò il modo in cui è possibile inserire grafica nei documenti TEX nonostante il compositore non ne sia in grado, sfruttando funzionalità esterne a esso. Nella seconda parte esaminerò da vicino il tipo di nodo pdf_literal di LuaTEX per creare dei semplici disegni con le specifiche del formato PDF. Nella terza e ultima parte applicativa creerò disegni al tratto programmando solamente in Lua all'interno di LuaTEX.

2 Risorse e riferimenti

In questa sezione indicherò i principali riferimenti per la comprensione del testo e alcune notizie per la compilazione degli esempi di codice riportati nel testo.

2.1 Requisiti per la comprensione

Per la comprensione del testo occorre un minimo di conoscenza del linguaggio Lua, che comprende la sintassi dei costrutti di base e i concetti principali per adoperare le tabelle, l'unico tipo di dato strutturato predefinito. Questa conoscenza si acquisisce in poche ore ma si avvisa il lettore che per essere produttivi, anche con Lua occorre apprendere il linguaggio in profondità, capirne le diverse componenti semantiche come le closure, le funzioni di prima classe, gli iteratori e le metatabelle, nonché conoscere almeno nei fondamenti il funzionamento interno dell'interprete — che è scritto in C — e le conseguenze della presenza del Garbage Collector.

Per fortuna la conoscenza si può acquisire per gradi e a seconda delle necessità. La migliore guida per Lua è il libro dell'Autore principale stesso del linguaggio, Roberto Ierusalimschy, denominato PIL acronimo di *Programming in Lua* (Ierusalimschy, 2013). Ormai questo testo sta diventando obbligatorio nella libreria di un buon sviluppatore TEX accanto al TEX book (Knuth, 1984) o al LATEX Companion (Mittelbach *et al.*, 2004). In effetti, visto che parleremo di grafica è più opportuno consultare il LATEX Graphics Companion (Goossens *et al.*, 2007).

Un'altra risorsa per documentarsi su Lua è il mio articolo che presentai al GUIT meeting 2012 a Napoli (Giacomelli, 2012). Nella prima parte di questo lavoro sono spiegati il tipo tabella e i princìpi della programmazione a oggetti in Lua.

Dal lato TEX, è necessario conoscere la differenza tra motore di composizione e formato, ovvero la differenza per esempio tra LuaTEX e LuaLATEX essendo gli esempi scritti in plain. Molto utile è poi conoscere i fondamenti dei registri box. Per questi importanti elementi è utile studiare gli Appunti di programmazione in LATEX e TEX di Enrico Gregorio (Gregorio, 2009) scaricando il documento dalla rete, oppure consultare il libro TEX by Topic (Eijkhout, 2014) che qualche tempo fa il GUIT inviò a tutti i propri soci.

2.2 Esecuzione del codice

Fondamentale, e quindi altamente consigliato, è compilare gli esempi riportati con la propria distri-

buzione TEX — preferibilmente riscrivendo i sorgenti per rendersi meglio conto dei concetti espressi dalla semantica del codice — e sperimentare gli effetti di modifiche o estensioni.

Tutti gli esempi di codice sono stati compilati con le versioni dei programmi e dei pacchetti inclusi nella TeX Live 2016 con il sistema operativo Ubuntu 14.04. Quest'ultimo dato dovrebbe essere ininfluente perché il codice TEX è multi-piattaforma.

Nei sorgenti ho inserito la *riga magica* per facilitare la compilazione con gli shell editor che offrono questa comoda funzionalità. Per esempio, nei listati per predisporre l'editor alla compilazione con LuaTEX si potrà trovare specificato:

```
% !TeX program = LuaTeX
```

Durante la compilazione è possibile incorrere in errori dovuti alle recenti modifiche dei nomi delle primitive in LuaTEX introdotte dalla versione 0.85 per rendere più razionale la separazione dei moduli interni. Il pacchetto luatex85 risolve queste incompatibilità almeno fino a quando non saranno aggiornate le varie librerie, come per esempio il pacchetto TikZ che evidentemente ricorre al suo interno alle primitive rinominate.

2.3 Gestione della memoria

Questo lavoro è dedicato alla tecnologia dei nodi di LuaTEX per realizzare grafica vettoriale in modo diretto con il linguaggio PDF. Trascurerò aspetti essenziali per le applicazioni reali come la gestione della memoria.

In Lua gli oggetti in memoria vengono automaticamente distrutti dal Garbage Collector quando non più necessari. Si tratta di un componente software che a runtime, ovvero durante l'esecuzione del programma, interviene sull'informazione in memoria non più indirizzabile.

Non tutti i tipi di dato sono soggetti alla gestione automatica della memoria. Tra questi ci sono proprio i nodi di LuaTEX, implementati come oggetti *userdata*. Se si perde il riferimento ai nodi la memoria che essi occupano diventa non raggiungibile causando un memory leak (mancata deallocazione di registri di memoria). Se si eliminano i nodi prematuramente si causerà la perdita di validità dei puntatori causando un segmentation fault (accesso alla memoria non valido).

In generale questi effetti disastrosi si evitano verificando molto attentamente il codice. D'altra parte si tratta di un argomento tutt'altro che secondario di cui soffre la programmazione in linguaggi di sistema come il C. Aggiungo per completezza che è possibile effettuare questi controlli automaticamente, per esempio il linguaggio Rust introduce alcuni concetti semantici che portano il compilatore a intercettare le situazioni di errore sulla memoria che possono condurre anche a problemi di sicurezza.

3 La grafica in TEX

In questa sezione presenterò diverse modalità con cui è possibile creare grafica — la possibilità di creare disegni specificando forme e coordinate attraverso un linguaggio — in TEX. Le entità geometriche vettoriali non possono essere elaborate da TEX perché esso non conosce alcuna primitiva grafica ma dà la possibilità di inserire nel documento qualsiasi codice si voglia con il comando `\special`.

All'apertura del documento composto il programma di lettura, che chiamerò *driver*, interpreterà queste porzioni di codice e ne produrrà il risultato a video o in stampa. Nel caso in cui il codice *speciale* non sia corretto potremo ricevere in apertura un errore di corruzione del file oppure effetti imprevedibili che renderebbero inutilizzabile il documento.

Per realizzare un disegno occorre perciò studiare attentamente il formato del documento in uscita, normalmente corrispondente al PDF, per iniettare il codice speciale conforme senza che TEX possa avvertirci di eventuali errori.

Ma è necessario fare in modo che il disegno non si sovrapponga agli altri elementi della pagina perché TEX della figura non conosce nulla perciò nemmeno le sue dimensioni. Queste informazioni, note come *bounding box*, devono essere calcolate e poi comunicate al motore di composizione perché lasci lo spazio corrispondente, per esempio creando una scatola orizzontale `hbox`.

Da notare che poiché si è legati alle funzionalità grafiche del driver esterno, il disegno ne avrà le stesse limitazioni. Per esempio, le specifiche PDF non comprendono il disegno di curve di secondo grado come i cerchi, così è necessario approssimare queste funzioni con curve di Bézier del terzo ordine, come spiega Claudio Beccari nel suo articolo uscito per il precedente GUITmeeting di Trento (Beccari, 2015), dal titolo "I calcoli matematici in `pdftex`".

3.1 Le primitive grafiche del formato PDF

Ma qual è il codice speciale secondo il formato PDF per disegnare una linea? Studiando il PDF Reference ufficiale (Adobe Systems Incorporated, 2006) che si può scaricare dal sito di Adobe, si scopre che la descrizione di una linea si basa sul concetto di *percorso*, una sequenza di punti nel piano definiti attraverso il sistema di coordinate (x, y).

Per disegnare una linea si da inizio a un percorso *muovendoci* nel primo punto con l'operazione `moveto`, proseguendo nel secondo punto con l'operazione `lineto` e infine disegnando il percorso appena costruito con l'operatore `stroke`.

La sintassi di questi operatori ricorda la notazione inversa polacca: prima si inseriscono gli argomenti e poi il nome dell'operatore tanto che

Tabella 1: Sintassi in notazione postfissa di alcuni operatori per la costruzione di grafica vettoriale nei documenti nel formato PDF.

Sintassi	Categoria/Descrizione
	Categoria stato grafico speciale:
`q`	Salva lo stato grafico corrente nello stack
`Q`	Ripristina lo stato grafico precedente nello stack
⟨*a*⟩ ⟨*b*⟩ ⟨*c*⟩ ⟨*d*⟩ ⟨*e*⟩ ⟨*f*⟩ `cm`	Modifica la matrice di trasformazione corrente concatenando quella specificata dai sei numeri degli operandi
	Categoria stato grafico generale:
⟨*dim*⟩ `w`	Imposta lo spessore di lineea nello stato grafico
⟨*code*⟩ `j`	Imposta lo stile con cui unire le linee del percorso nello stato grafico
⟨*code*⟩ `J`	Imposta lo stile con cui disegnare gli estremi delle linee di un percorso nello stato grafico
	Categoria costuzione percorsi:
⟨*x*⟩ ⟨*y*⟩ `m`	Inizia un nuovo tratto di percorso (un percorso è l'unione di più percorsi continui detti *sub-path*)
⟨*x*⟩ ⟨*y*⟩ `l`	Aggiunge al percorso una linea dal punto corrente al punto specificato che diviene poi il nuovo punto corrente del percorso
⟨*x*⟩ ⟨*y*⟩ ⟨*width*⟩ ⟨*height*⟩ `re`	Aggiunge al percorso un rettangolo come completo sub-path
`h`	Chiude il percorso corrente aggiungendo un segmento dal punto corrente al punto iniziale del sub-path
	Categoria tracciamento percorsi:
`S`	Disegna il percorso corrente

per essa il termine usato nel reference di Adobe è *notazione postfissa*. Il disegno di una linea inclinata dall'origine del sistema di riferimento al punto di coordinate $(32, 16)$ ha come codice speciale il seguente:

```
0   0 m
32 16 l
S
```

Non una singola istruzione ma ben tre in cui tutte le misure sono in punti grandi `bp`[1], e le operazioni descritte sono identificate da un codice letterale, i cui significati sono riportati nella tabella 1.

Queste codifiche derivano dal formato Postscript, un vero e proprio linguaggio basato sulla struttura dati dello stack in notazione postfissa. Un bell'articolo per chi volesse approfondire questo linguaggio si trova nel numero 7 di ArsTEXnica (Nisi, 2009) a opera di Riccardo Nisi.

3.2 Un esempio in `pdftex`

Prima di realizzare il disegno della linea appena descritta con il classico compositore `pdftex`, è utile precisare che in questo motore è disponibile il comando `\pdfliteral` da preferire rispetto a `\special` per la sintassi leggermente più semplice e per il nome più significativo. Possiamo leggerne la definizione nel manuale di `pdftex` (Thành, Hàn Thế́ *et al.*, 2016). Il comando accetta un argomento tra graffe che prima verrà espanso e poi inserito nel documento PDF inalterato.

Ecco il codice dell'esempio:

```
% !TeX program = pdftex
\pdfcompresslevel=0
\nopagenumbers
\pdfliteral {
0 0 m
32 16 l
S}
\bye
```

Poiché abbiamo richiesto che il file PDF non venga compresso dando un opportuno valore al comando `\pdfcompresslevel`, possiamo leggerlo con un editor di testi per rintracciare il nostro codice speciale per il disegno della linea, ecco il frammento d'interesse:

```
stream
1 0 0 1 72 769.89 cm
 0 0 m 32 16 l S

endstream
```

La riga speciale compare immutata al di sotto di una riga inserita dal programma che definisce la *matrice di trasformazione*, come evidenzia il codice `cm` a cui accennerò nella prossima sezione.

L'insieme delle proprietà come spessore di linea e colore fanno parte dello stato grafico. Non solo è possibile modificarne i valori ma se ne può creare una copia salvandola nello *stack* per poi ripristinarla successivamente.

Nel momento del ripristino, quei valori tornano al valore precedente. Si attiva così un ambiente locale all'interno del quale le modifiche ai parametri permangono convenientemente solo fino al suo termine, esattamente come succede per i gruppi in TEX.

1. Il bp è l'unità di misura *punto grande* del formato Postscript e vale 1/72 di pollice perciò circa 0,3528 mm mentre il pt punto tipografico vale 1/72,27 pollici.

Lo stato grafico si copia nello stack corrente con l'operatore `q` e può essere ripristinato con l'operatore `Q`. Modifichiamo il codice per il disegno della linea per lavorare in locale. Aggiungiamo anche l'impostazione dello spessore del tratto a 4 bp con l'operatore `w` e lo stile di chiusura sulla forma circolare con l'operatore `J` (proprietà definita *round cap* individuata con il valore 1):

```
% !TeX program = pdftex
\pdfcompresslevel=0
\nopagenumbers
\pdfliteral {
q
4 w
1 J
0 0 m
32 16 l
S
Q
}
\bye
```

3.3 La matrice di trasformazione

Per completezza di esposizione, accenno agli effetti della *matrice di trasformazione*. Si tratta di un operatore geometrico che applicato alle entità che seguono la definizione della matrice, ne modifica scala, posizione e rotazione.

La chiave dell'operatore di modifica della matrice è `cm`. Esso ha l'effetto di concatenare alla matrice di trasformazione dello stato grafico quella nuova definita da ben 6 parametri numerici. L'ordine di applicazione influisce sulla geometria del disegno finale.

I sei parametri indipendenti dati in ordine producono quattro diversi tipi di trasformazioni elementari:

- *traslazione* del sistema di riferimento specificata dai parametri ($1 \quad 0 \quad 0 \quad 1 \quad t_x \quad t_y$), dove t_x e t_y sono le coordinate di traslazione dell'origine;
- *scalamento* del sistema di riferimento con i parametri ($s_x \quad 0 \quad 0 \quad s_x \quad 0 \quad 0$), dove s_x e s_y sono i fattori di scala secondo i rispettivi assi;
- *rotazione* del sistema di riferimento con i parametri ($\cos\theta \quad \sin\theta \quad -\sin\theta \quad \cos\theta \quad 0 \quad 0$) dove l'angolo θ è l'angolo di rotazione;
- *distorsione* del sistema di riferimento con i parametri ($1 \quad \tan\alpha \quad \tan\beta \quad 1 \quad 0 \quad 0$) dove l'angolo α è la distorsione rispetto all'asse x e l'angolo β è la distorsione rispetto all'asse y.

Se osserviamo la riga dello stream mostrato nell'esempio precedente, si riconosce facilmente che il comando `\pdfliteral` ha inserito l'operatore `cm` con una matrice di trasformazione che trasla il disegno di 1 pollice in direzione x e di 271,60 mm in direzione dell'asse y, che è la distanza che si ottiene sottraendo dall'altezza della pagina di 297 mm il valore di 1 pollice.

Il disegno è stato inserito sulla pagina nel punto attivo in quel momento che, essendo il documento vuoto se non per il codice speciale, è il punto più in alto a sinistra della gabbia del testo, definita per default con margini di 1 pollice. Come già ricordato, il motore nella composizione dell'oggetto speciale non tiene in alcuna considerazione le dimensioni della figura trattandola come un'entità puntiforme.

Da notare che la traslazione è controllata anche dall'opzione del comando `\pdfliteral` che può essere `direct` o `page`. Nel caso venga specificata una di queste due opzioni infatti, non verrà inserita la matrice di trasformazione. Per i dettagli, si confronti ancora il manuale del compositore `pdftex` compilando qualche esempio di codice per valutare il comportamento del comando.

3.4 La libreria `mplib` di LuaTEX

La grafica non è nativa di TEX e nemmeno in LuaTEX, ma in quest'ultimo motore di composizione è disponibile la libreria `mplib` che altro non è che un interprete Metapost.

Con Metapost si ha a disposizione un potente motore per la grafica in grado di risolvere nativamente equazioni lineari, eseguire funzioni ricorsive, memorizzare figure, e molto altro ancora. Per un'introduzione si veda l'articolo di Klaus Höppner apparso su questa stessa rivista (Höppner, 2008) qualche anno fa.

Il funzionamento della libreria è semplice: si chiama il metodo `execute()` su un'istanza di `mplib`, fornendogli il codice Metapost. Se non ci sono errori otterremo una tabella Lua contenente la descrizione geometrica degli oggetti grafici (si consulti il manuale di LuaTEX per i dettagli (The LuaTEX development team, 2016)) e alcuni metodi per ricavarne la rappresentazione nei formati Postscript o SVG. L'output PDF invece può essere estratto per esempio con l'ausilio del pacchetto luamplib (Hagen *et al.*, 2016) che scorre le entità grafiche e le traduce in primitive PDF.

Con questo pacchetto disegnamo la solita linea inclinata di spessore 4 bp compilando con LuaTEX il seguente sorgente:

```
% !TeX program = LuaTeX
% LuaTeX >= 0.85
\pdfvariable compresslevel=0
\input luamplib.sty

\mplibcode
beginfig(0);
draw (0,0) -- (32,16)
     withpen pencircle scaled 4;
endfig;
\endmplibcode
\bye
```

Il pacchetto luamplib definisce l'ambiente in stile plain `\mplibcode` e `\endmplibcode`, e non solo traduce le entità grafiche nel formato PDF, ma nasconde tutti i dettagli delle chiamate in Lua e cura l'inserimento della figura nel documento con una scatola orizzontale dalle opportune dimensioni.

Con la stessa tecnica precedente, andiamo ora a leggere il frammento di stream contenuto nel file PDF non compresso con un editor di testo. La parte d'interesse è la seguente:

```
stream
1 0 0 1 74 751.89 cm
q
0.000 0.000 0.000 rg 0.000 0.000 0.000 RG
10.000000 M
1 j
1 J
4.000000 w
0.000000 0.000000 m
32.000000 16.000000 l
S
Q

endstream
```

Rispetto al codice manuale presentato nella sezione precedente abbiamo delle nuove linee di codice che agiscono localmente: le chiavi `rg` e `RG` impostano lo spazio colore e il colore stesso, la chiave `M` imposta la dimensione massima della larghezza di giunzione delle linee potenzialmente infinita quando l'angolo tra i segmenti vale 180°, e infine la chiave `j` seleziona il codice 1 che corrisponde allo stile di unione delle linee al modo *round join*. Nel nostro esempio, essendoci una sola linea lo stile di join non influisce sul disegno.

La matrice di trasformazione opera ancora una traslazione semplice nel punto corrente della pagina ma con uno spostamento orizzontale verso destra di 2 bp e uno spostamento verso il basso si 4 bp, dovuti allo stile circolare del disegno degli estremi e allo spessore della linea di 4 bp. Il punto d'inserimento dell'oggetto speciale puntiforme è quindi l'origine del suo sistema di riferimento.

La cosa importante è che anche l'uso del motore grafico interno `mplib` si traduce nelle stesse righe di stream secondo le specifiche PDF. Diverso è solamente il modo in cui queste linee di codice vengono prodotte, cioè attraverso il potente linguaggio grafico di Metapost.

3.5 Il pacchetto TikZ

Nel formato plain la stessa linea si può disegnare anche con il pacchetto PGF/TikZ con il seguente codice:

```
% !TeX program = pdftex
\pdfcompresslevel=0
\nopagenumbers
\input tikz.tex

\noindent\tikzpicture
\draw[line width=4bp, line cap=round]
    (0,0) -- (32bp,16bp);
\endtikzpicture
\bye
```

Esaminando ancora una volta la sequenza di streaming contenuta nel file PDF ottenuto compilando con `pdftex` il sorgente precedente, si ottiene un risultato equivalente:

```
stream
1 0 0 1 74 751.89 cm
q
0 0 0 RG
0 0 0 rg
0.3985 w
q
q
4.00005 w
1 J
0.0 0.0 m
32.0004 16.0002 l
S
Q
Q
n
Q

endstream
```

Si noti come la precisione delle coordinate numeriche di `pgf` è leggermente inferiore rispetto a quella della libreria interna `mplib` accoppiata con il pacchetto luamplib per la conversione della geometria nello stream PDF.

Tuttavia le linee di codice speciale sono praticamente le stesse. In più compare solo l'operatore `n` che prescrive che i percorsi siano terminati senza che siano disegnati, ma il punto in cui compare si trova al di fuori del disegno, come mostrano gli operatori di copia/ripristino dello stato grafico. Interessante notare come `q` e `Q` siano annidati tre volte, probabilmente perché si tratta di uno schema fisso effettivamente utile solo nel momento in cui l'utente richiede funzionalità di disegno particolari.

4 LuaTEX e i nodi `pdfliteral`

I *nodi* programmati in Lua sono elementi tipografici completi che possono essere immessi nelle liste principali orizzontale, verticale o matematica durante la composizione del documento per essere posizionati sulla pagina.

Essi possono essere concatenati in liste a rappresentare un intero capoverso oppure possono essere impacchettati in scatole orizzontali o verticali.

Si ottiene lo stesso risultato tipografico che con le usuali primitive del motore di composizione, ma la differenza sostanziale la fa il fatto che i nodi sono programmati in Lua e ciò amplia notevolmente le possibilità applicative del sistema.

4.1 Un esempio di codice

Per mostrare l'equivalenza della tecnologia dei nodi con i comandi TEX classici, e come utile riferimento, scegliamo un primo esempio semplice di una scatola orizzontale di larghezza 64 pt contenente tre lettere distanziate in modo uguale. Con TEX si realizza questa struttura con il codice:

```
\hbox to 64pt{A\hfil B\hfil C}
```

con il risultato seguente:

A B C

La stessa costruzione può essere ottenuta programmando esclusivamente in Lua. Una volta attivato il *modo Lua* con la primitiva `\directlua`, si procederà in ordine con il:

- creare i tre nodi di tipo `glyph` per le lettere A, B e C;
- creare i due nodi di tipo `glue` per gli spazi allungabili;
- concatenare tutti i nodi in una lista nella corretta sequenza;
- impacchettare la lista in una scatola di tipo orizzontale dall'esatta larghezza;
- inviare la scatola alla lista principale del compositore.

I passi da compiere sono implementati nel seguente listato completo e pronto per essere compilato da LuaTEX, che riprodurrà l'esempio delle tre lettere:

```
% !TeX program = LuaTeX
\directlua{
function glyph(c)
    local n = node.new("glyph")
    n.char = string.byte(c)
    n.font = font.current()
    return n
end

function glue()
    local n = node.new("glue")
    n.stretch = 32*65536
    return n
end
}

\leavevmode\directlua{
local a = glyph "A" --[=[ creaz nodi --]=]
local b = glyph "B"
local c = glyph "C"
local g1, g2 = glue(), glue()

--[=[ concatenazione lista --]=]
local head = node.insert_after(a, a, g1)
head = node.insert_after(head, g1, b)
head = node.insert_after(head, b, g2)
head = node.insert_after(head, g2, c)

--[=[ impacchettamento lista nel box --]=]
local hbox = node.hpack(
    head, tex.sp "64pt", "exactly"
)

--[=[ invio box alla main list --]=]
node.write(hbox)
}
\bye
```

In Lua con la tecnologia dei nodi, i processi di lettura dei token, d'espansione, di esecuzione, non ci sono. Si passa subito all'ultima fase del processo visuale — si tratta della modalità in quattro tempi con cui opera TEX — perché i nodi elementari sono assemblati in strutture tipografiche già pronte per il posizionamento sulla pagina.

Con piccole modifiche al codice, è possibile memorizzare la scatola in un registro che poi verrà inserito nel documento con gli usuali comandi `\box` per usare e svuotare il registro oppure `\copy` per usare ma mantenere inalterato il contenuto del registro:

```
% !TeX program = LuaTeX

% a new box register
\newbox\mybox

\directlua{
--[==[ as before ... --]==]

--[==[ packing the list in a box --]==]
local hbox = node.hpack(
    head, tex.sp "64pt", "exactly"
)

--[==[ saving list in a box register --]==]
tex.box["mybox"] = hbox
}

\box\mybox
\bye
```

Nei listati precedenti vi sono molti dettagli che non spiegerò. Rimando il lettore al mio precedente articolo introduttivo sulla tecnologia dei nodi (Giacomelli, 2016).

4.2 Nodi grafici `pdf_literal`

Il nodo `pdf_literal` è un sottotipo del nodo `whatsits` che raggruppa molte funzionalità dirette verso il formato PDF. Esso accetta solamente tre parametri: una lista di attributi, il numero che imposta il `mode` che corrisponde alle opzioni possibili per la primitiva `\pdfliteral` e il campo stringa `data` a cui assegnare lo stream.

Questo è tutto quello che è necessario sapere per disegnare la linea inclinata presa come semplice prototipo:

```
% !TeX program = LuaTeX
% LuaTeX >= 0.85
\pdfvariable compresslevel=0
```

FIGURA 1: Risultato del codice LuaTEX riportato nel testo in cui il codice Lua per l'inserimento diretto di un nodo di tipo `pdf_literal` è inserito tra due parole. Per maggiore chiarezza sono state aggiunte due linee tratteggiate sugli assi coordinati e l'ingrandimento rispetto alle dimensioni originali.

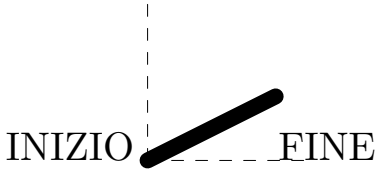


FIGURA 2: Risultato del codice LuaTEX riportato nel testo in cui la figura è inserita all'interno di una scatola orizzontale delle dimensioni corrispondenti a quelle del bounding box del disegno. Per maggiore chiarezza sono state aggiunte due linee tratteggiate sugli assi coordinati e un ingrandimento rispetto alle dimensioni originali.

```
\nopagenumbers

\directlua{
local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = [[q 4.0 w 1 J 0 0 m 32 16 l S Q]]
node.write(n)
}
\bye
```

Se si compila con LuaTEX l'esempio minimo, tutto funziona correttamente: la scrittura del codice non ha riservato sorprese: si ottiene una pagina vuota tranne che per la linea posizionata in alto a sinistra. Caricando il file PDF in un editor di testi, possiamo verificare che effettivamente lo stream compare immutato nel documento finale e che al solito è stata inserita automaticamente la matrice di trasformazione per traslare la figura nel punto d'inserimento più alto e a sinistra della gabbia del testo.

Inserendo del testo prima e dopo il comando `\directlua` nel codice precedente, per esempio le parole INIZIO e FINE, il risultato è quello riportato nella figura 1. L'oggetto grafico è stato posizionato nel punto d'inserimento sulla pagina subito dopo la lettera O della parola INIZIO sulla linea di base e con dimensioni nulle. La linea appare sovrapposta al testo che non si accorge minimamente della presenza dell'oggetto assumendo la stessa esatta posizione che avrebbe assunto in assenza del nodo `pdfliteral`.

4.3 Creare spazio per la figura

Una possibile soluzione per far si che il compositore mantenga sulla pagina lo spazio necessario alla figura, consiste nell'inserire una scatola orizzontale vuota delle dimensioni uguali a quelle della figura con il codice:

```
\raise <ydim> \hbox to <xdim> {}
```

Nel caso della linea d'esempio, considerando la chiusura di forma tonda delle estremità di raggio 2 bp, le dimensioni della figura sono $x_{\text{dim}} = 36\,\text{bp}$ e $y_{\text{dim}} = 20\,\text{bp}$, dunque:

```
% as before ...
INIZIO%
\raise 20bp \hbox to 36bp {}%
\directlua{
local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = [[q 4.0 w 1 J 0 0 m 32 16 l S Q]]
node.write(n)
}FINE
\bye
```

Ovviamente questa soluzione non funziona. La scatola non si sovrappone alla figura, semplicemente perché il punto d'inserimento dopo l'`hbox` si sarà spostato a destra sulla linea base di 36 bp ed è in quel punto che viene inserito il nodo `n`. Utile esercizio per il lettore è verificare l'effetto dell'aggiunta di uno spazio negativo con `\hskip -36bp`.

Non ci rimane che inserire la figura all'interno della scatola nell'angolo in basso a sinistra con il seguente codice:

```
% as before ...
INIZIO%
\hbox to 36bp {%
\vbox to 20 bp {}
\directlua{
local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = [[q 4.0 w 1 J 0 0 m 32 16 l S Q]]
node.write(n)
}%
\hfill}FINE
```

Il risultato di questo secondo tentativo riportato nella figura 2 sembra sia corretto ma osservando attentamente con l'aiuto dei tratteggi aggiunti al disegno, si nota che la linea risulta di poco più bassa rispetto alla linea base. Il motivo è che il nodo viene inserito rispetto all'origine del sistema di riferimento della figura. Nel nostro caso la linea ha uno spessore di 4 bp con estremità arrotondate e perciò *riempie* un'area al di sotto dell'asse delle ascisse con un punto più basso che scende di 2 bp.

La soluzione adottatata dal pacchetto `luamplib` è quella di creare una scatola orizzontale in quattro successivi passaggi in TEX. Semplificando il codice del pacchetto si legge:

```
\newbox\mpbox
\hbox{
  % set some dimensions
  ...
  \setbox\mpbox\vbox{% passaggio 1
```

```
    \noindent
    % insert here graphic material
    ...
    }
    \setbox\mpbox\hbox % passaggio 2
      {\hskip-\MPllx bp%
       \raise-\MPlly bp%
       \box\mpbox
    }%
    % passaggio 3
    \setbox\mpbox\vbox to \MPheight {
       \vfill
       \hsize\MPwidth
       \wd\mpbox0pt%
       \ht\mpbox0pt%
       \dp\mpbox0pt%
       \box\mpbox
    }%
    % passaggio 4
    \wd\mpbox\MPwidth
    \ht\mpbox\MPheight
    \box\mpbox
}
```

Nel codice riportato, al passaggio 2, si riconosce facilmente la correzione per tener conto di un bounding box che può essere al di sopra o al di sotto dell'asse x oppure a sinistra o a destra dell'asse y: le dimensioni `\MPllx` e `\MPlly` sono infatti le coordinate con segno dell'angolo in basso a sinistra del bounding box. Le distanze `\MPwidth` e `\MPheight` sono invece le dimensioni assolute del rettangolo che circoscrive la figura.

La costruzione di una scatola orizzontale per il corretto posizionamento della figura non è così semplice essendo necessario spostare l'origine del sistema di riferimento del disegno per far coincidere il bounding box ai bordi della scatola stessa.

4.4 Bounding box con i nodi

Quello che nella sezione precedente ho individuato come passaggio 2 si può realizzare anche con i nodi. Illustrerò tra un momento due diverse soluzioni che prevedono entrambe una scatola dentro l'altra, diversamente dal codice di `luamplib` dove a ogni passaggio la scatola viene per così dire *espansa* in se stessa.

In generale, se (P_0, P_1) sono i punti nel piano che definiscono i vertici rispettivamente inferiore/sinistro e superiore/destro del rettangolo del bounding box della figura, la traslazione rispetto all'angolo della scatola orizzontale dovrà essere di $-x_0$ e $-y_0$, mentre la sua larghezza dovrà essere $w = x_1 - x_0$ e la sua altezza $h = y_1 - y_0$.

4.4.1 Prima soluzione

L'idea è di effettuare prima lo spostamento verticale dell'oggetto speciale puntiforme all'interno di una scatola `vbox` usando una lunghezza rigida, e poi fare la stessa cosa ma in una scatola orizzontale a cui infine sono imposte le dimensioni uguali a quelle del bounding box.

Per mostrare solamente il codice Lua, che compie queste operazioni, scriverò una funzione che chiamerò `pack()`. Gli argomenti necessari sono il nodo `n` con il materiale speciale, e le quattro coordinate del bounding box rispetto al sistema di riferimento della figura:

```
function pack(n, x0, y0, x1, y1)
    local vskip = node.new("glue")
    vskip.width = -y0
    local vhead = node.insert_after(
        n, n, vskip
    )
    local vbox = node.vpack(vhead)

    local hskip = node.new("glue")
    hskip.width = -x0
    local hhead = node.insert_after(
        hskip, hskip, vbox
    )
    local hbox = node.hpack(hhead)

    hbox.width = x1 - x0
    hbox.heigth = y1 - y0
    return hbox
end
```

4.4.2 Seconda soluzione

La seconda soluzione si basa sul campo `shift` dei nodi `hlist`. Si effettua per prima cosa lo spostamento orizzontale con la solita lunghezza rigida e poi si impongono alla scatola orizzontale lo shift verticale e le dimensioni:

```
function pack(n, x0, y0, x1, y1)
    local hskip = node.new("glue")
    hskip.width = -x0
    local hhead = node.insert_after(
        hskip, hskip, n
    )
    local hbox = node.hpack(hhead)
        hbox.shift = y0

    local hbox2 = node.hpack(hbox)
    hbox2.width = x1 - x0
    hbox2.height = y1 - y0
    return hbox2
end
```

Rispetto alla precedente, questa soluzione ha il vantaggio di creare una sola grandezza elastica. Da notare che il campo `shift` non avrebbe effetto se la scatola non fosse inserita in una ulteriore scatola. Questo comportamento non è proprio del tutto coerente, e le future versioni di LuaTEX potrebbero apportare modifiche.

4.4.3 Test finale

Il risultato della prova finale è mostrato in figura 3. È prodotto con il seguente sorgente in cui va aggiunto il corpo della funzione `pack()` con una delle due precedenti soluzioni:

```
% !TeX program = LuaTeX
% LuaTeX >= 0.85
```

FIGURA 3: Il risultato del test finale del metodo per impacchettare all'interno di una scatola orizzontale dalle corrette dimensioni, il semplice disegno di una linea inclinata a cui è stato aggiunto per utile riferimento un rettangolo dal bordo sottile e tratteggiato. La figura è ingrandita con la stessa scala di quelle anologhe precedenti.

```
\pdfvariable compresslevel=0
\nopagenumbers
\newbox\mybox

\directlua{
function pack(n, x0, y0, x1, y1)
--[==[ as before --]==]
end
}

\directlua{
local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = [[
q
4.0 w 1 J 0 0 m 32 16 l S
0.125 w
[3] 0 d
-2 -2 36 20 re
S
Q
]]

local x0, y0 = tex.sp "-2bp", tex.sp "-2bp"
local x1, y1 = tex.sp "34bp", tex.sp "18bp"
tex.box["mybox"] = pack(n, x0, y0, x1, y1)
}
INIZIO\box\mybox FINE
\bye
```

Il fatto sorprendente è che i due file PDF ottenuti con le due diverse soluzioni della funzione `pack()` risultano essere *identici*.

Un altro fatto interessante, è che se affianchiamo il disegno più di una volta come nel codice:

```
IZ\copy\mybox\copy\mybox\box\mybox FN
```

lo stream che otterremo nel file PDF sarà il seguente[2]:

```
stream
BT
/F1 9.9 Tf 1 0 0 1 91.9 749.8 Tm [(IZ)]TJ
ET
1 0 0 1 126.027 751.89 cm
 q 4.0 w 1 J 0 0 m 32 16 l S Q
1 0 0 1 36 0 cm
 q 4.0 w 1 J 0 0 m 32 16 l S Q
1 0 0 1 36 0 cm
 q 4.0 w 1 J 0 0 m 32 16 l S Q
1 0 0 1 -198.027 -751.89 cm
BT
/F1 9.9 Tf 1 0 0 1 232.0 749.8 Tm [(FN)]TJ
ET

endstream
```

2. Le righe sono state accorciate solo un po' senza alterarne il significato per rimanere nella giustezza della colonna eliminando qualche decimale e il disegno del rettangolo tratteggiato di contorno della linea.

dove compare dopo ogni figura la matrice di trasformazione che rappresenta la traslazione verso destra di 36 bp, la larghezza del disegno, che ogni volta si somma alla precedente.

Nella prossima sezione mi dedicherò ad alcuni esempi applicativi programmando i disegni solo in Lua per produrre finalmente qualcosa di più interessante rispetto alla linea inclinata utilizzata come esempio per tutta questa sezione.

4.5 Disegno misto con TikZ

Una volta che il disegno realizzato in Lua con lo stream nel formato PDF si trova all'interno di un registro `box`, è possibile inserirlo in una precisa posizione all'interno di una figura in TikZ?

Traducendo in codice plain, la domanda appena posta è la seguente:

```
% !TeX program = LuaTeX
\input tikz.tex

% define \mybox as before...

\tikzpicture
% draw a control grid
\draw[step=2bp,help lines]
    (0,0) grid (50bp,42bp);
% draw the control red rectangle
\draw[red, line width= 1.08pt]
    (12bp, 20bp) rectangle (48bp, 40bp);
% draw the 'external' box
\node at (30bp, 30bp) {\box\mybox};
\endtikzpicture
```

Compilando si ottiene l'esatto posizionamento all'interno del rettangolo rosso della linea inclinata che lo ricordo ha dimensioni esterne di 36 bp per 20 bp. Essa viene posizionata nel nodo con punto d'ancoraggio il centro esattamente come previsto — se lo si desidera si possono usare diversi punti d'ancoraggio usando l'opzione `anchor`.

Questa tecnica funziona e rende possibile fare grafica mista con parti di figura realizzate in TikZ e altre parti realizzate in Lua.

5 Disegni in libertà

Ora ho davanti come un foglio bianco avendo a fianco la matita. Sfoglio qualche libro e trovo quasi subito qualcosa da disegnare...

5.1 Cornici 1

Il primo disegno è riportato in figura 4. Si tratta di una serie di quadrati concentrici di spessore degradante verso l'esterno. Per realizzarla ho scritto

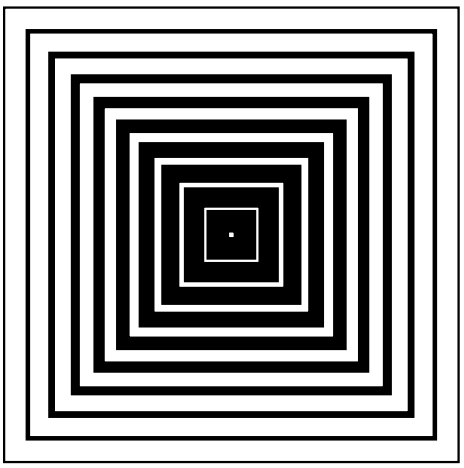


FIGURA 4: Una serie di quadrati concentrici di diverso spessore e misura del lato ottenuto con la tecnologia dei nodi con le specifiche grafiche del formato PDF. Lo spessore del tratto degrada dal centro verso l'esterno mentre al contrario per la distanza tra i quadrati.

alcune funzioni Lua di servizio che operano tutte su una tabella/array sempre passata come primo argomento. Ogni funzione aggiunge una stringa di testo in sequenza che contiene le istruzioni speciali `pdfliteral`. Per esempio la funzione per disegnare un rettangolo è la seguente:

```
plib.rect = function(fig, x0, y0, x1, y1)
    local t = fig.data
    local fmt = "%0.6f %0.6f %0.6f %0.6f re"
    t[#t+1] = string.format(fmt,
                x0, y0, x1-x0, y1-y0
    )
    local w = fig.width/2
    if not fig.bb then
        fig.bb = {x0-w, y0-w, x1+w, y1+w}
    else
        local xmin = fig.bb[1]
        local ymin = fig.bb[2]
        local xmax = fig.bb[3]
        local ymax = fig.bb[4]
        if x0 - w < xmin then
            fig.bb[1] = x0-w
        end
        if y0 - w < ymin then
            fig.bb[2] = y0-w
        end
        if x1 + w > xmax then
            fig.bb[3] = x1+w
        end
        if y1 + w > ymax then
            fig.bb[4] = y1+w
        end
    end
end
```

La maggior parte del codice è impegnata ad aggiornare il bounding box della figura, indispensabile per il posizionamento sulla pagina, secondo la logica seguente: ogni volta che si disegna un elemento, poiché ne conosco la geometria, posso verificarne i limiti d'ingombro. A scapito della generalità ma a vantaggio delle prestazioni, si potrebbe invece impostare il bounding box secondo la logica: se conosci la figura che stai disegnando allora ne conosci anche le dimesioni.

Questa seconda logica è quella adottata dal pacchetto pstricks (ZANDT, 2003) che richiede all'utente le coordinate dei confini della figura come argomento obbligatorio dell'ambiente pspicture. Il primo approccio invece, è quello adottato dal pacchetto TikZ.

Riporto il codice di una seconda funzione, quella che restituisce la stringa da asssegnare al campo `data` del nodo, perché non utilizza la concatenazione di stringhe ma la funzione `table.concat()` della libreria standard di Lua, molto più efficiente nell'assemblare un insieme di stringhe:

```
plib.stream = function (fig)
    local t = fig.data
    return table.concat(t, "\n")
end
```

Con le funzioni ausiliarie raggruppate nella tabella chiamata `plib`, la parte più significativa del codice che produce la figura dei quadrati è la seguente:

```
\directlua{
local fig = plib.newfig()

local i = 1
local s = 10
local k = 1

while s > 0 do
    plib.penwidth(fig, s)
    local m = i + s/2
    plib.rect(fig, -m, -m, m, m)
    plib.stroke(fig)
    i = i + s + k
    s = s - 1
    k = k + 1
end
plib.closefig(fig)

local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = plib.stream(fig)
tex.box["mybox"] = pack(n, plib.bb(fig))
}
```

Dal lato tecnico delle specifiche PDF, è importante osservare *quando* nel codice avviene la chiamata alla funzione `plib.stroke()` — che non fa altro che aggiungere la `S`. Siccome lo spessore del tratto cambia per ogni quadrato, è necessario inserire l'operatore `stroke` dopo ogni inserimento dell'operatore `re` per il disegno del rettangolo. Se non fosse così, per esempio se inserissimo una `S` solamente alla fine, i percorsi verrebbero si disegnati ma tutti con l'ultimo spessore di linea inserito.

Questo significa che l'operatore `w` per lo spessore di linea non fa altro che scrivere quel parame-

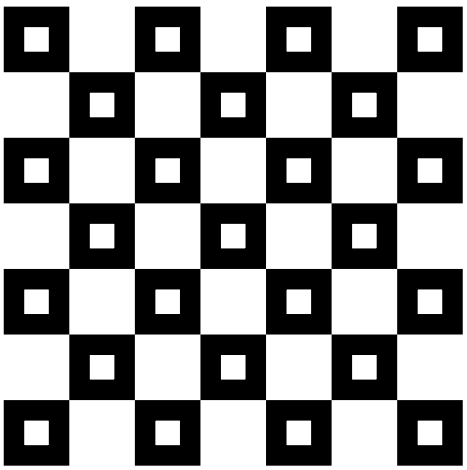


FIGURA 5: Secondo esempio applicativo, disegno di una scacchiera di rettangoli.

tro nello stack corrente sovrascrivendo il valore precedente.

5.2 Cornice 2

La parte più interessante del disegno di figura 5 è realizzare l'alternanza tra celle piene e celle vuote. Una possibilità è usare una variabile booleana che cambia di stato a ogni iterazione all'interno di un doppio ciclo `for`, uno esterno per le righe e uno interno per le colonne. Questo metodo funziona solamente se la dimensione della scacchiera è un numero dispari:

```
\directlua{
local row = 6
local isfull = true

local s = 2.5
local h = 16

local fig = plib.newfig()
plib.penwidth(fig, 2*s)

for i=0,row do
    local y0 = i * h + s
    local y1 = (i+1) * h - s
    for j=0,row do
        if isfull then
            local x0 = j * h + s
            local x1 = (j+1) * h - s
            plib.rect(fig, x0, y0, x1, y1)
        end
        isfull = not isfull
     end
end

plib.stroke(fig)
plib.closefig(fig)

local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = plib.stream(fig)
tex.box["mybox"] = pack(n, plib.bb(fig))
}
```

Ma le cose si possono vedere da un diverso punto di vista, quello che considera il disegno non come una scacchiera ma come la somma di griglie traslate una rispetto all'altra. Per cominciare scriverò la funzione `grid()` che disegna oggetti su una griglia regolare ricevendo tra gli argomenti una funzione che per condizione accetta le coordinate di un punto per utilizzarlo come centro di una figura.

In Lua le funzioni sono oggetti di prima classe, possono cioé essere memorizzate in variabili, una particolarità molto potente, ma non esiste il concetto d'*interfaccia* ovvero un modo per imporre che la funzione abbia una ben precisa firma[3]. Del resto Lua non è un linguaggio statico dove il codice è compilato in un file eseguibile e i tipi devono essere stabiliti in quella fase, mentre non è vietato passare a una funzione un numero e subito dopo richiamare la stessa funzione passando però una stringa.

Assumiamo che la griglia sia quadrata, ovvero che la distanza tra le righe sia la stessa che quella tra le colonne e valga `step`, che il numero delle righe sia `nx` e quello delle colonne sia `ny`, e che il primo nodo in basso a sinistra abbia coordinate `x0`, `y0`:

```
local
function grid(drawfun, nx, ny, step, x0, y0)
    for r = 0, nx-1 do
        local x = x0 + r*step
        for c = 0, ny-1 do
            local y = y0 + c*step
            drawfun(x, y)
        end
    end
end
```

Per semplicità ho eliminato il parametro `fig` intendendo che l'oggetto sarà già stato creato nel momento in cui verranno chiamate le funzioni che risolveranno il simbolo con la tecnologia delle *closure*. Chiamerò la funzione di disegno del nodo `square(x, y)`. Essa dovrà rispettare le regole d'interfaccia per le funzioni da passare a `grid()`:

```
local fig = plib.newfig()
local s = 3.65
local h = 16
local function square(x, y)
    local a = (h - s)/2
    local x0, y0 = x - a, y - a
    local x1, y1 = x + a, y + a
    plib.rect(fig, x0, y0, x1, y1)
end
```

In definitiva, il codice per il disegno della scacchiera con una struttuta del tutto generale, può anche essere il seguente:

```
\directlua{
local
```

3. La firma di una funzione non è altro che l'informazione di quali argomenti essa riceve e di quali valori essa restituisce.

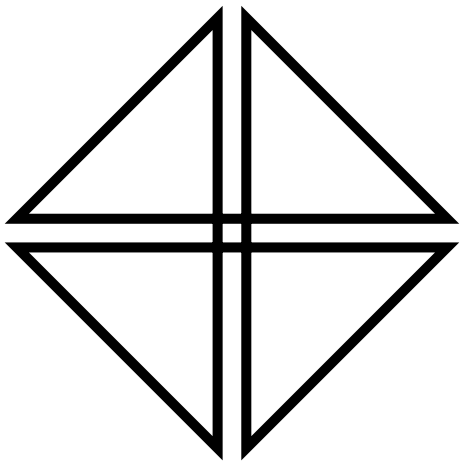


FIGURA 6: Disegno tracciabile senza mai staccare la penna dal foglio e quindi contenente un'unica linea che ritorna al punto di partenza è perciò chiusa, realizzato con il codice Lua riportato nel testo.

```
function grid(drawfun, nx, ny, step, x0, y0)
    --[==[ as before ]==]
end

local fig = plib.newfig()
local s = 5
local h = 16
local function square(x, y)
    --[==[ as before ]==]
end

plib.penwidth(fig, s)
grid(square, 4, 4, 2*h, 0, 0)
grid(square, 3, 3, 2*h, h, h)

plib.stroke(fig)
plib.closefig(fig)

local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = plib.stream(fig)
tex.box["mybox"] = pack(n, plib.bb(fig))
}
```

5.3 Linee 1

Passando ai disegni al tratto, ovvero contenenti solamente linee, ho riprodotto la spezzata di figura 6. Un buon esercizio perché per prima cosa ho imparato che non è sempre così facile determinare il bounding box. Se osserviamo la giunzione delle linee inclinate, a causa dello stile che prolunga i bordi dei tratti dotati di spessore w, i limiti si estendono oltre il punto d'intersezione degli assi di:

$$\frac{1}{2\tan\pi/8}w = 1{,}207w$$

Seconda osservazione: la figura può essere disegnata con l'idea di muovere la penna sul foglio, ovvero dando le coordinate relative rispetto al punto precedente. La funzione di servizio in Lua può essere la seguente: essa fa uso dei campi `x` e `y` della tabella che immagazzina riga per riga lo stream PDF, campi utili anche per il calcolo (semplificato) dei limiti della figura:

```
plib.lineto = function(fig, dx, dy)
    local t = fig.data
    local x, y = dx + fig.x, dy + fig.y
    fig.x, fig.y = x, y
    t[#t+1] = string.format(
        "%0.6f %0.6f l", x, y
    )
    -- bb updating
    local x0 = math.min(x, fig.x)
    local y0 = math.min(y, fig.y)
    local x1 = math.max(x, fig.x)
    local y1 = math.max(y, fig.y)

    if not fig.bb then
        fig.bb = {x0, y0, x1, y1}
    else
        if x0 < fig.bb[1] then
            fig.bb[1] = x0
        end
        if y0 < fig.bb[2] then
            fig.bb[2] = y0
        end
        if x1 > fig.bb[3] then
            fig.bb[3] = x1
        end
        if y1 > fig.bb[4] then
            fig.bb[4] = y1
        end
    end
end
```

Il frammento principale del codice che costruisce la figura inserendola in un box orizzontale è il seguente dove si comincia a disegnare dallo spigolo più alto a sinistra:

```
\directlua{
local l = 120
local d = 8
local w = 2.8

local h = (l - d)/2
local fig = plib.newfig()
plib.penwidth(fig, w)

--[==[ stile logo --]==]

plib.moveto(fig,-d/2, l/2)
plib.lineto(fig,  0, -l)
plib.lineto(fig, -h,  h)
plib.lineto(fig,  l,  0)
plib.lineto(fig, -h, -h)
plib.lineto(fig,  0,  l)
plib.lineto(fig,  h, -h)
plib.lineto(fig, -l,  0)
plib.cycle(fig)

plib.stroke(fig)
plib.closefig(fig)

local n = node.new("whatsit", "pdf_literal")
n.mode = 0
```

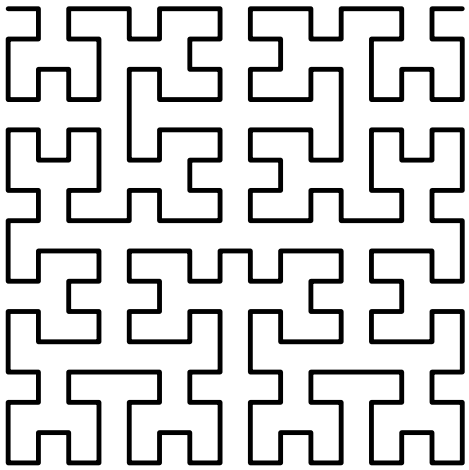


FIGURA 7: Disegno della figura ricorsiva nota come curva di Hilbert al quarto livello d'iterazione, generato con il codice Lua mostrato nel testo che fa uso dello stack delle chiamate ricorsive di funzione per memorizzare tutte le coordinate dei punti.

```
n.data = plib.stream(fig)
tex.box["mybox"] = pack(n, plib.bb(fig))
}
```

5.4 Linee 2

Il secondo e ultimo esempio di disegno a linee è la curva di Hilbert, un unico percorso ricorsivo che tende a riempire un quadrato, iterazione dopo iterazione. David Hilbert elaborò la curva nel 1891 come variante della teoria di Giuseppe Peano sulle curve che riempiono il piano, che a sua volta fu ispirato da George Cantor con i suoi primi risultati sulla cardinalità di insiemi infiniti.

La curva, rappresentata in figura 7, ha come elemento base il quadrato disegnato senza uno dei lati. Dal nostro punto di vista si tratta di un percorso che connette tutti i quadrati privi di uno dei lati disposti su una griglia regolare di un dato livello d'iterazione.

I segmenti crescono con ragione esponenziale con le iterazioni tanto che se alla prima iterazione sono 3 alla quinta sono già 832. Disegnarli con Lua e i nodi `pdf_literal` non è agevole come lo è in Metapost, dove abbiamo a disposizione le operazioni sui vettori con il tipo `pair` e quelle sul tipo `path`. Anni fa scrissi il codice in Metapost che riporto di seguito adattato alla compilazione con LuaTEX[4]:

```
% !TeX program = LuaTeX

\nopagenumbers
\pdfvariable compresslevel=0
\input luamplib.sty
\mplibcode
% Recursive Hilbert function
vardef hilbertCurve(expr c, x, level)=
   % the four basepoint
   save hilpath;
   path hilpath;
   save ax, bx, a, b;
   pair ax, bx, a, b;

   ax := c  rotatedabout(x, 90);
   bx := c  rotatedabout(x,-90);
   a  := ax rotatedabout(c ,-90);
   b  := bx rotatedabout(c , 90);

   if level = 1:
      hilpath := a--ax--bx--b;
   else:
      save atmp, btmp, axtmp, bxtmp;
      pair atmp, btmp, axtmp, bxtmp;
      atmp  := 0.25[a,b];
      btmp  := 0.25[b,a];
      axtmp := 1.25[a,ax];
      bxtmp := 1.25[b,bx];

      hilpath :=
      reverse
      hilbertCurve( a,  atmp, level-1)--
      hilbertCurve(ax, axtmp, level-1)--
      hilbertCurve(bx, bxtmp, level-1)--
      reverse
      hilbertCurve( b,  btmp, level-1);
   fi
   hilpath
enddef;

beginfig(1)
   pickup pencircle scaled 5;
   draw hilbertCurve(origin,(0,-100), 4);
endfig;
end
\endmplibcode
\bye
```

Grazie a Metapost, l'eleganza del codice è palese, e lo è anche la sequenza di streaming che riporta esattamente la definizione del percorso senza una sbavatura:

```
stream
1 0 0 1 262 579.89 cm
q
1.000 0.000 0.000 rg 1.000 0.000 0.000 RG
10.000000 M
1 j
1 J
5.000000 w
-187.500000 187.500000 m
-162.500000 187.500000 l
-162.500000 162.500000 l
-187.500000 162.500000 l
-187.500000 137.500000 l
-187.500000 112.500000 l
-162.500000 112.500000 l
-162.500000 137.500000 l
-137.500000 137.500000 l
...
endstream
```

4. La spiegazione del codice si trova nel post che pubblicai nel mio blog all'indirizzo https://robitex.wordpress.com/2009/12/31/la-curva-di-hilbert-in-metapost/.

In Lua possiamo costruirci gli stessi operatori vettoriali di Metapost così da poter semplicemente tradurre il codice ricorsivo del listato precedente. Altra via è esaminare attentamente la geometria dell'oggetto ricorsivo e ideare qualche tipo di rappresentazione. Scelgo questa seconda strada, naturalmente.

Il primo problema che si pone è questo: nella funzione ricorsiva non possiamo sapere quale sia il primo punto della curva, infatti sappiamo solo che a un certo punto avremo raggiunto il livello d'iterazione desiderato e dovremo disegnare il quadrato con il lato aperto anziché generare altri quattro vertici e proseguire la ricorsione con la misura del lato dimezzato.

Si potrebbe per questo aggiungere alla funzione ricorsiva un argomento di tipo booleano che sia vero se dobbiamo eseguire l'operazione `moveto` e falso se `lineto`, con lo svantaggio di dover controllare dal corpo della funzione ogni volta questo valore.

Potremo allora, invece di procedere subito al disegno, memorizzare la sequenza di punti in un array. Terminata la ricorsione l'array contiene tutto il percorso perciò si esegue l'operazione `moveto` per il primo punto e dal secondo punto in poi l'operazione `lineto`.

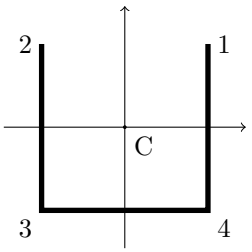
Del resto questo succede anche nella soluzione in Metapost precedente, dove una variabile di tipo `path` memorizza tutto il percorso che poi viene disegnato una volta completato dalla ricorsione. Tuttavia, con le funzioni di servizio memorizziamo già lo stream in un array perciò avremo duplicato le informazioni, prima in un array con le coppie di coordinate e poi in uno con le stringhe speciali PDF.

Penso invece che il codice si semplifichi se lasciamo che sia una funzione di servizio che chiameremo `pathto()` a scegliere l'operazione da eseguire sfruttando il fatto che nella tabella dello stream memorizziamo le coordinate del punto precedente nei campi `x` e `y` per calcolare senza considerare lo spessore delle linee, il bounding box. Se il percorso non è ancora iniziato questi campi sono nulli e quest'informazione discrimina tra il caso primo punto e il caso punto successivo:

```
plib.pathto = function(fig, x, y)
    local t = fig.data
    if fig.x then
        t[#t+1] = string.format(
            "%0.6f %0.6f l", x, y
        )
        -- bb updating
        local x0 = math.min(x, fig.x)
        local y0 = math.min(y, fig.y)
        local x1 = math.max(x, fig.x)
        local y1 = math.max(y, fig.y)
        if x0 < fig.bb[1] then
            fig.bb[1] = x0
        end
        if y0 < fig.bb[2] then
            fig.bb[2] = y0
        end
        if x1 > fig.bb[3] then
            fig.bb[3] = x1
        end
        if y1 > fig.bb[4] then
            fig.bb[4] = y1
        end
        fig.x, fig.y = x, y
    else
        t[#t+1] = string.format(
            "%0.6f %0.6f m", x, y
        )
        fig.x, fig.y = x, y
        fig.bb = {x, y, x, y}
    end
end
```

Inizio ora a illustrare la soluzione che ho elaborato per rappresentare due informazioni essenziali per il disegno ricorsivo della curva: quale lato del quadrato non deve essere disegnato e con quale orientamento si deve costruire il percorso aperto dei tre lati rimanenti: da un estremo all'altro o viceversa?

Osserviamo la seguente figura: si tratta della curva di Hilbert di livello 1, un quadrato con il lato aperto verso l'alto, una scelta che non influisce se non sulla rotazione del disegno:



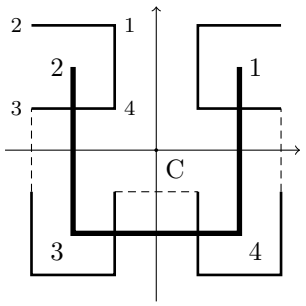
Il centro del quadrato ha coordinate $C = (x_c, y_c)$ e i suoi vertici un numero che è quello del quadrante di appartenenza nel sistema di riferimento locale. Note le coordinate del centro e la misura del lato del quadrato si ricavano immediatamente le coordinate dei vertici.

In Lua, potremo creare una tabella con i segni degli assi dei quadranti per calcolare con solo due espressioni le coordinate di un punto conoscendone il quadrante:

```
quadr = {
    { 1,  1}, -- quadrante 1
    {-1,  1}, -- quadrante 2
    {-1, -1}, -- quadrante 3
    { 1, -1}, -- quadrante 4
}

function point(xc, yc, l, q)
    local x = xc + quadr[q][1]*l/2
    local y = yc + quadr[q][2]*l/2
    return x, y
end
```

Il livello 2 si costruisce disegnando quadrati aperti secondo la posizione, sui quattro vertici del livello 1 dimezzando la misura del lato, come mostra la figura sottostante:



In questo schema ho evidenziato con il tratteggio le linee che completano la curva di secondo livello, di unione tra i quadrati. Se costruiamo il percorso inserendo i vertici nell'ordine corretto, nel disegno troveremo anche questi tratti di collegamento, ma non eviteremo di inserire due linee anche quando queste sono allineate.

I livelli successivi si ottengono continuando a costruire su tutti i vertici del livello precedente i tre lati dei quadrati di lunghezza dimezzata.

Definita la geometria si può affrontare il nocciolo del problema: trovare un modo per conoscere per ogni quadrato quale lato non deve essere disegnato e con quale verso il percorso deve essere ordinato. Questi due problemi sono risolti nella funzione precedente in Metapost con le operazioni sui vettori e applicando la funzione `reverse` sui percorsi parziali.

La soluzione che ho trovato per definire completamente il percorso di disegno per ogni singolo quadrato, si basa su un numero intero con segno: il valore assoluto del numero individua il primo vertice secondo il quadrante e il segno indica se si deve percorrere i lati in senso antiorario (segno positivo) o in senso orario (segno negativo).

Per esempio il disegno del quadrato di livello 2 in alto a sinistra della figura è definito semplicemente dal numero -2. Infatti, seguendo la regola, si parte dal nodo 2 ma per via del segno negativo si segue il senso orario trovando la sequenza di nodi 2, 1, 4, e 3. Lo stesso quadrato ma con percorso invertito si disegna a partire dal numero 3, che produce la sequenza dei quattro nodi 3, 4, 1, 2.

Empiricamente ho poi trovato il numero con segno da assegnare a ciascuno dei quattro quadrati da generare ricorsivamente per ogni vertice del livello precedente. Quest'informazione è memorizzata nella tabella che assume il ruolo di invariante per l'iteratore, alla chiave chiamata `nodes`.

Senza entrare troppo nei dettagli del codice, l'iteratore — tecnicamente si tratta di un iteratore senza stato — con cui possono essere creati i vertici del quadrato in un nodo nella sequenza corretta dal numero del nodo di partenza è il seguente:

```
quadr = {
    { 1,  1}, -- quadrante 1
    {-1,  1}, -- quadrante 2
    {-1, -1}, -- quadrante 3
    { 1, -1}, -- quadrante 4
}

function itervertex(t, i)
    i = i + 1
    if i < 4 then
        local n = math.abs(t.n)
        if t.rev then
            n = n - i
            if n < 1 then n = n + 4 end
        else
            n = n + i
            if n > 4 then n = n - 4 end
        end
        local x = t.xc + quadr[n][1]*t.l/2
        local y = t.yc + quadr[n][2]*t.l/2
        return i, t.nodes[i+1], x, y
    end
end

function vertex(xc, yc, l, n)
    local rev = n < 0 and true or false
    if rev then
        last = -(n - 2)
        if last > 4 then last=last - 4 end
    else
        last = -(n + 2)
        if last < -4 then last=last + 4 end
    end
    local inv_state = {
        xc=xc, yc=yc, l=l, rev=rev, n=n,
        nodes = {-n, n, n, last},
     }
    return itervertex, inv_state, -1
end
```

La funzione ricorsiva controlla il livello: se uguale a 1 chiama la funzione `pathto` aggiungendo linee a partire dal punto precedente così la curva risulterà un'unica spezzata, altrimenti chiama se stessa su ognuno dei quattro vertici (da notare che l'iteratore è impiegato in entrambi i rami del costrutto `if`):

```
function hilbertCurve(fig, xc, yc, l, n, lv)
  if lv == 1 then
    for _,_, x, y in vertex(xc, yc, l, n) do
      plib.pathto(fig, x, y)
    end
  else
    for _,v, x, y in vertex(xc, yc, l, n) do
      hilbertCurve(fig, x, y, l/2, v, lv-1)
    end
  end
end
```

Il codice LuaTEX per creare il disegno non dovrà far altro che impostare lo spessore della linea, gli stili di chiusura del tratto, e poi chiamare la funzione ricorsiva con le coordinate del centro della curva, la dimensione del lato del quadrato di livello 1, il codice nodo che definisce sia il lato aperto e il

verso del disegno, e per ultimo il numero dei livelli da raggiungere.

Specificando il codice nodo +2 avremo il lato aperto uguale a quello superiore e l'ordine di tracciamento antiorario, perciò il percorso della curva comincerà in alto a sinistra e finirà in alto a destra:

```
\newbox\mybox

\directlua{
local dim = 200
local w = 5

local fig = plib.newfig()
plib.penwidth(fig, w)
plib.linecap(fig, 1)
plib.linejoin(fig, 1)

hilbertCurve(fig, 0, 0, dim, 2, 4)

plib.stroke(fig)
plib.closefig(fig)

local n = node.new("whatsit", "pdf_literal")
n.mode = 0
n.data = plib.stream(fig)
tex.box["mybox"] = pack(n, plib.bb(fig))
}
\leavevmode\box\mybox
```

La parte di codice che risolve la curva più complesso e meno leggibile è quello dell'iteratore. Lascio al lettore come esercizio, il compito di riscrivere la funzione ricorsiva generando il percorso della curva di Hilbert a partire dal primo nodo in alto a sinistra della griglia di vertici distanti $l/2^{\text{level}}$ e calcolando la posizione del nodo successivo tra le quattro possibilità conoscendo la direzione della mossa precedente.

Se agli spostamenti secondo la direzione degli assi attribuiamo i numeri $+x = 1$, $-x = 3$, $+y = 2$, $-y = 4$, cominciando dal punto più in alto a sinistra, la costruzione del livello 2 è conseguenza della successione delle 13 mosse:

```
1, 4, 3,
4,
4, 1, 2,
1,
4, 1, 2,
2,
3, 2, 1
```

Qual è la sequenza di mosse per il disegno della curva di Hilbert di livello n? Si tratta ancora di un problema ricorsivo?

6 Conclusioni

Con questo lavoro ho esplorato con pieno successo di risultati la tecnologia dei nodi di LuaTEX per creare grafica con le primitive PDF. Molte altre cose, come gli oggetti riferimento, attendono di entrare nel nostro laboratorio.

Le funzionalità a disposizione per il formato d'uscita PDF dei documenti composti con LuaTEX sono davvero molte. Abbiamo appena cominciato a sperimentarle e se si considera che possono essere programmate in Lua all'interno del motore di composizione, si giunge alla conclusione che le potenzialità applicative del nuovo motore sono notevolissime.

La stretta integrazione tra le librerie Lua e gli oggetti del compositore TEX consente di ottenere risultati tipografici probabilmente unici tra tutti i programmi utilizzati per la produzione di documenti digitali.

LuaTEX è ormai prossimo alla versione 1.0, la prima a essere stabile, un momento spero importante per il futuro del sistema TEX.

7 Ringraziamenti

Come al solito ringrazio la mia famiglia che mi concede (parecchio) tempo, specie nelle giornate estive, e in particolare mio figlio Matteo che mi ha disegnato appositamente la figura 6 a linea continua, molto interessante nella sua semplicità e perfetta per gli scopi dell'articolo, e mia moglie Silvia che ha letto la bozza con appassionata pazienza alla ricerca di errori, refusi, incoerenze. Grazie famiglia!

Luigi Scarso mi ha aiutato a capire un passaggio che riguarda il campo `shift` dei nodi di LuaTEX e non ha esitato a migliorare i sorgenti con osservazioni e precisazioni tecniche. Grazie mille.

Ringrazio infine tutta la redazione di ArsTEXnica e chi ha lavorato con impegno per la buona riuscita del meeting annuale dell'associazione.

8 L'editor Atom

Una nota riguardo alle modalità con cui è stato prodotto questo articolo, composto con il compositore `pdflatex`, riguarda l'editor utilizzato per i sorgenti. Normalmente per questo compito si utilizza uno shell editor, ovvero un programma per il testo che offre funzioni per la compilazione TEX svolgendo appunto un ruolo d'interfaccia.

Questa volta ho usato Atom, disponibile sul sito `https://atom.io/`, un editor multi-piattaforma per la scrittura di codice molto interessante anche per i sorgenti TEX.

Il cursore multiplo intelligente, la gestione automatica della tabulazione, i riquadri laterali dell'albero del progetto e della miniatura del contenuto, lo spostamento delle righe verso l'alto o verso il basso con una semplice combinazione di tasti, sono solo alcune delle sue abilità per comporre il testo.

Le funzionalità integrate di Atom per la gestione del sistema di controllo delle revisioni `git`, lo rende ancor di più adatto per produrre con il sistema TEX documenti come libri o tesi di laurea anche molto complessi e con più autori.

Riferimenti bibliografici

Adobe Systems Incorporated (2006). *The PDF Reference 1.7*. Addison-Wesley, 6ª edizione. URL http://www.adobe.com/devnet/pdf/pdf_reference_archive.html.

Beccari, C. (2015). «I calcoli matematici in pdftex». *ArsTEXnica*, (20), pp. 7–15. URL http://www.guitex.org/home/it/arstexnica.

Eijkhout, V. (2014). *TEX by Topic*. Lehmanns media.

Giacomelli, R. (2012). «Grafica ad oggetti con LuaTEX». *ArsTEXnica*, (14), pp. 53–71. URL http://www.guitex.org/home/numero-14.

— (2016). «Primi esperimenti con node di LuaTEX». *ArsTEXnica*, (21). URL http://www.guitex.org/home/numero-21.

Goossens, M., Mittelbach, F., Roegel, D. e Voss, H. (2007). *The LATEXGraphics Companion (2nd Edition)*. Addison-Wesley Professional, 2ª edizione.

Gregorio, E. (2009). «Appunti di programmazione in LATEX e TEX». https://profs.sci.univr.it/~gregorio/introtex.pdf. Giugno 2009. Seconda edizione.

Hagen, H., Hoekwater, T., Roux, E., Gesang, P. e Dohyun, K. (2016). «The luamplib package». $TEXMFDIST/doc/luatex/luamplib/luamplib.pdf. Version 2016/03/31 v2.11.3.

Höppner, K. (2008). «Introduction to MetaPost». *ArsTEXnica*, (6), pp. 5–9. URL http://www.guitex.org/home/it/arstexnica.

Ierusalimschy, R. (2013). *Programming in Lua*. Lua.org, 3ª edizione.

Knuth, D. E. (1984). *The TeXbook*. Addison-Wesley Professional, 1ª edizione.

Mittelbach, F., Goossens, M., Braams, J. e Carlisle, D. (2004). *The LATEXCompanion (Tools and Techniques for Computer Typesetting)*. Addison-Wesley Professional, 2ª edizione.

Nisi, R. (2009). «Il PostScript in LATEX». *ArsTEXnica*, (7), pp. 32–46. URL http://www.guitex.org/home/numero-7.

Thành, Hàn Thế, Rahtz, S., Hagen, H., Henkel, H., Jackowski, P. e Schröder, M. (2016). «The luamplib package». $TEXMFDIST/doc/pdftex/manual/pdftex-a.pdf. Version 2016/05/09.

The LuaTEX development team (2016). *LuaTEX Reference Manual*, 0.95 edizione.

Zandt, T. V. (2003). «User's guide». $TEXMFDIST/doc/generic/pstricks/pstricks-doc.pdf. Version 97.

▷ Roberto Giacomelli
Carrara
giaconet dot mailbox at gmail dot com

Tavole di Ishihara

Renato Battistin

Sommario

Le tavole di Ishihara vengono impiegate per la rilevazione e la classificazione di anomalie visive cromatiche. Dopo una descrizione dei difetti visivi cromatici e delle proprietà cromatiche delle tavole atte ad individuarli, viene descritto un algoritmo per la loro riproduzione mediante comandi PSTricks generati da un modulo Python.

Abstract

The Ishihara plates are used for the detection and classification of chromatic visual abnormalities. After a description of the color vision defects and color properties of the plates suitable to locate them, an algorithm is described for their reproduction by PSTricks commands generated by a Python module.

1 Introduzione

La percezione del colore è un campo di indagine molto vasto che coinvolge aspetti metrologici, fisiologici e psicologici. Il parziale o mancato riconoscimento del colore può essere rilevante sia in ambito professionale che sociale; basti pensare ai segnali di avviso o di pericolo quali semafori, cartelli stradali, segnalatori di uscite di sicurezza, ecc.

L'occhio umano è attrezzato generalmente per la visione tricromatica grazie alla presenza nella retina di tre tipi di fotorecettori in forma di cellule nervose dette *coni*. Tali cellule hanno la proprietà di essere sensibili in maniera differente tra loro alla radiazione elettromagnetica nella regione spettrale del visibile, che copre l'intervallo compreso tra 400 e 760 nm circa.

La mancanza di uno dei fotorecettori, oppure una loro differente o minore risposta allo stimolo visivo, induce una variazione del riconoscimento cromatico rispetto ad un presupposto riconoscimento cromatico normale. Le anomalie vengono classificate a seconda del tipo di fotorecettore coinvolto. Tenendo presente che i fotorecettori vengono ordinati per lunghezza d'onda decrescente relativamente alle loro regioni di sensibilità spettrale, si parla di protanomalia, deuteranomalia e tritanomalia, a seconda che l'anomalia ricettiva coinvolga la regione spettrale rispettivamente del rosso, del verde e del blu. I soggetti aventi una risposta cromatica minore sono detti protanomali, deuteranomali e tritanomali; la loro visione permane in generale tricromatica ma con una percezione dei colori differente da quella della maggioranza delle persone. Se la funzionalità di uno dei fotorecettori è nulla, i soggetti sono classificati rispettivamente come protanopi, deuteranopi e tritanopi e la loro visione è bicromatica.

La classificazione cromatica della visione di un soggetto può essere eseguita con varie tecniche sperimentali, tuttavia il meccanismo di fondo consiste nel mostrare al soggetto delle immagini di combinazioni di varie graduazioni e tinte di colore che sono distinguibili dalla maggioranza dei soggetti. Una metodo storicamente noto e tuttora in uso è quello delle tavole pseudo-isocromatiche di Ishihara (Ishihara, 1972), comunemente note come *tavole di Ishihara*[1].

Le tavole sono tutte composte da un fondo uniforme generalmente chiaro e da un insieme di dischi di diametro e di colore variabile a seconda dell'eventuale tipo di disabilità cromatica che si vuole individuare sul soggetto a cui viene sottoposta la tavola. Ogni tavola presenta l'insieme dei dischi disposti entro un cerchio, sebbene la valenza di questa raffigurazione sia essenzialmente estetica. La maggioranza delle tavole ha i dischi disposti e scelti cromaticamente in modo tale da raffigurare il profilo di un numero quando osservati ad un'opportuna distanza e sotto un'opportuna illuminazione. Il riconoscimento dei profili può variare a seconda del tipo di anomalia visiva cromatica del soggetto.

In alcune tavole non sono raffigurati dei numeri ma dei percorsi più o meno tortuosi, delimitati da due estremi sulla circonferenza del cerchio; il soggetto esaminato in questo caso è inviato a tracciare il percorso.

Lo scopo del presente lavoro è mostrare una procedura per simulare graficamente le tavole di Ishihara con gli strumenti LaTeX. La simulazione non ha tanto lo scopo di riprodurre esattamente le forme e le tonalità cromatiche delle tavole Ishihara, essendo questo più un aspetto storico-tipografico che pratico, ma di definire degli algoritmi e delle tecniche tipografiche per riprodurre quelle caratteristiche cromatiche e di forma che tanto utili le rendono per l'esame dei difetti cromatici della visione. Una volta individuate tali tecniche, sarà possibile impiegarle oltre che per simulare le tavole di Ishihara anche per crearne di nuove, utili per ricercare particolari difetti cromatici oppure per specifici ambiti di indagine (ad esempio quello di sensibilità cromatica).

Il primo passo è un'analisi colorimetrica delle tavole di Ishihara per ricavarne i parametri

1. Per una panoramica storica si veda http://www.eyemagazine.com/feature/article/ishihara.

cromatici dei dischi e le loro disposizioni nella tavola. Quindi verrà descritto un algoritmo per simulare la disposizione dei dischi delle tavole di Ishihara. Il passo successivo sarà quello di colorare opportunamente i dischi in modo da ottenere una tavola che contenga sia l'informazione cromatica utile all'esaminatore per classificare il tipo di visione cromatica del soggetto esaminato, sia le forme geometriche delle regioni cromatiche rilevanti per la prova al fine di consentire al soggetto esaminato di comunicarle all'esaminatore.

2 Tinta, saturazione e brillanza

Le caratteristiche cromatiche di un colore possono essere quantificate in termini di coordinate cromatiche definibili sulla base delle proprietà di risposta dei fotorecettori agli stimoli luminosi. I fotorecettori presentano una risposta sostanzialmente lineare e tra loro indipendente allo stimolo luminoso; questo comportamento permette di operare con le coordinate cromatiche assimilandole a componenti di vettori di uno spazio vettoriale. Tra le conseguenze notevoli di questa proprietà vettoriale ci sono l'additività degli stimoli cromatici (somma vettoriale), l'indipendenza della riposta cromatica dall'intensità dello stimolo (prodotto di un vettore per uno scalare) e la possibilità di uguagliare la somma di due stimoli alla somma di altri due stimoli di cui uno cromaticamente neutro (luce bianca) e l'altro spettralmente puro (luce composta da radiazione monocromatica). Quest'ultima proprietà è alla base dei uno dei vari sistemi cromatici esistenti, quello che definisce lo stimolo cromatico in termini di tinta, saturazione e brillanza[2], HSB (*hue, saturation, brightness*).

3 Analisi di alcune tavole

L'analisi cromatica delle tavole di Ishihara[3] può essere eseguita strumentalmente, ad esempio con un colorimetro direttamente su una copia cartacea delle tavole, oppure "digitalmente" partendo da una riproduzione digitale delle tavole e impiegando un qualsiasi programma in grado di rilevare le proprietà cromatiche dei pixel dell'immagine. Nel presente caso l'indagine è stata condotta digitalmente.

Ovviamente il rilievo cromatico su una versione digitalizzata delle tavole di Ishihara comporta una perdita di accuratezza nelle proprietà cromatiche rilevate. Tuttavia, essendo le tavole concepite per evidenziare differenze nella percezione cromatica,

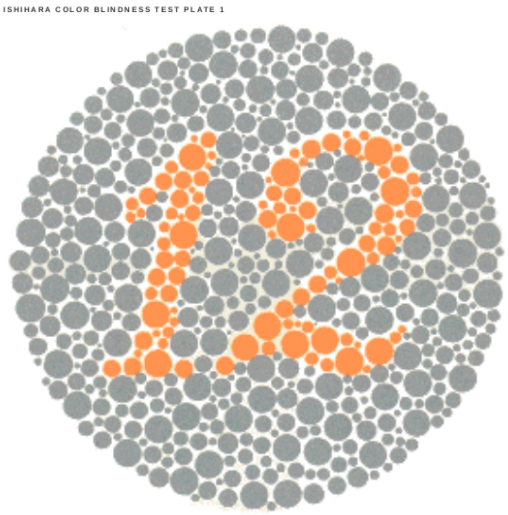


Figura 1: La prima delle tavole di Ishihara, visibile a tutti i soggetti con deficienze visive.

è ragionevole supporre che tali differenze siano meno influenzate dalla mancanza di accuratezza. Le tavole di Ishihara sono numerate in ordine progressivo[4] ed il rilievo delle coordinate cromatiche viene eseguito mediante il programma Gpick[5].

Di una tavola vengono rilevate con Gpick le coordinate cromatiche HSB di un numero statisticamente rilevante di dischi della stessa tinta, così da ottenere una terna di intervalli di valori di tinta, saturazione e brillanza. Vengono in questo modo caratterizzati cromaticamente i colori delle varie regioni di una tavola, in particolare dei dischi costituenti il fondo e la figura rappresentata nella tavola. La tavola 1 (Fig. 1) raffigura il numero '12' con dischi tutti di colore identico, mentre il fondo è composto di dischi tutti sostanzialmente di colore grigio (Tab. 1). Le unità di misura sono unitarie su una scala da 0 a 360 per la tinta, H, e percentuali da 0 a 100 per la saturazione, S, e la brillanza, B. L'elevato valore di saturazione della regione numerale ha lo scopo di facilitare il riconoscimento del numero '12' alle persone soggette a cecità parziale ai colori rosso (*protanomali*) e verde (*deuteranomali*). Il valore di brillanza sostanzialmente identico per tutti i dischi della tavola avrebbe lo scopo di rendere irriconoscibile il numero ai soggetti com-

Tabella 1: Analisi della tavola 1

Regione	H	S (%)	B (%)
Numero '12'	15-19	80-90	51-56
Fondo	115	4-5	52-55

2. Per stimoli luminosi generati da superfici illuminate invece che direttamente da sorgenti luminose, è più corretto parlare di *chiarezza* al posto di brillanza e di *croma* invece che di saturazione (*cfr.* Claudio Oleari (1998), par. 3.5.3); per non appesantire la terminologia continueremo con l'abuso di linguaggio usando i termini relativi all'osservazione diretta delle sorgenti.

3. Esistono più versioni delle tavole con un numero variabile di tavole e, a volte, una loro differente numerazione. Faremo riferimento alla versione a 24 tavole già citata.

4. Non si confonda il numero progressivo di una tavola con l'eventuale figura numerica in essa riprodotta.

5. Gpick - Advanced color picker - v. 0.2.5, Copyrights © 2009-2013, Albertas Vyšniauskas and Gpick development team, http://code.google.com/p/gpick/.

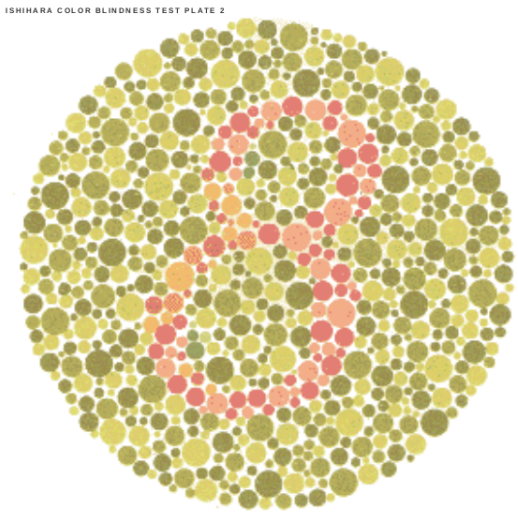


Figura 2: Una tavola di Ishihara impiegata per individuare i soggetti protanomali o deuteranomali.

pletamente ciechi ai colori (*acromatopi*); tuttavia anche questi ultimi sono in grado, seppure con qualche difficoltà, di riconoscere la sagoma del numero '12' grazie ad una sagace disposizione dei dischi.

La tavola 2 (Fig. 2) raffigura il numero '8' per soggetti normali (Tab. 2) e può raffigurare il numero '3' per soggetti protanomali o deuteranomali. Questa tavola presenta dischi oltre che di più tinte, a sostanziale parità di tinta, anche di distinti valori di saturazione. Il dato rilevante è la leggera differenza di valore di tinta dei dischi nella parte complementare alla figura del numero '3' che completa la figura del numero '8'; tale differenza è associata a valori di saturazione maggiori e ad una maggiore variabilità della brillanza. Tali differenze cromatiche nella regione numerale dell'intero numero '8' non impediscono un riconoscimento di quest'ultimo da parte di un soggetto normale, ma possono indurre ad un riconoscimento della sola regione grafica del '3' in presenza di protanomalie o deuteranomalie.

4 La simulazione delle tavole

La simulazione grafica delle tavole di Ishihara richiede la risoluzione di almeno due problemi:

- creazione di un insieme di dischi di diametro variabile, non sovrapposti e raggruppati in una forma circolare

Tabella 2: Analisi della tavola 2

Regione	H	S (%)	B (%)
Numero '3'	0-20	50-60	60-65
Complemento di '3' a '8'	19-24	60-80	50-65
Fondo	45-47	45-55	40-60

- colorazione di sottoinsiemi dei dischi creati in modo da generare delle figure riconoscibili

Il primo problema può essere affrontato implementando algoritmi che possono essere distinti grosso modo in due classi: a) algoritmi stocastici; b) algoritmi deterministici. Trattandosi di due concetti in antitesi, sembra strano che i due algoritmi possano condurre al medesimo risultato, ossia produrre una disposizione apparentemente casuale di dischi di dimensione variabile e per di più raggruppati per formare un cerchio.

In effetti il metodo stocastico risulta intuitivamente il più diretto e soprattutto il più collaudato in altri ambiti, ad esempio nella simulazione dinamica di insiemi di particelle sottoposte a forze interattive di Van der Walls. In sostanza si tratterebbe di collocare entro una buca di potenziale bidimensionale di forma circolare sufficientemente ampia un certo numero di particelle, nel nostro caso dischi, debolmente interagenti tra di loro con un potenziale di Lennard-Jones[6]. Il raggio della buca verrebbe quindi ridotto progressivamente lasciando evolvere dinamicamente il sistema fino ad ottenere una disposizione casuale, compatta e di forma circolare di dischi non sovrapposti. In questo articolo, tuttavia, risolveremo il problema utilizzando un algoritmo principalmente deterministico.

Il secondo problema, quello delle colorazione dei dischi, invece verrà risolto graficamente impiegando PSTricks.

5 L'algoritmo di collocazione dei dischi

Questo algoritmo nella sua forma più intuitiva è facilmente riproducibile. Bastano un foglio, una penna ed un compasso:

- traccia un punto Q in centro ad un foglio;
- punta il compasso sul punto Q e disegna un cerchio C di raggio R sufficientemente ampio da contenere un numero elevato di dischi 'c' aventi un raggio 'r' almeno un ordine di grandezza inferiore ad R;
- traccia in posizione casuale il primo disco 'c' di raggio 'r' ma in modo da includere il punto Q;
- scegli una semiretta di direzione casuale e con origine nel punto Q, traccia su di essa con il compasso un disco 'c' il più possibile vicino al centro del cerchio C ma senza sovrapporre alcuno dei dischi 'c' già tracciati e rimanendo entro il cerchio C;

6. Sono potenziali di interazione tra coppie di particelle della forma $V(r_{ij}) = 4\epsilon\left(\left(\frac{\sigma}{r_{ij}}\right)^6 - \left(\frac{\sigma}{r_{ij}}\right)^{12}\right)$

- ripeti il punto precedente fino a riempimento del cerchio C.

L'algoritmo in generale però non riesce a riprodurre accuratamente le tavole. Per poterlo fare, infatti, sono necessarie alcune precisazioni.

La scelta di collocare il centro dei dischi lungo semirette di direzione casuale può sembrare a priori una scelta adeguata vista la collocazione dei dischi nelle tavole. In pratica, il numero di semirette impiegabili in maniera efficiente per collocare i dischi vicini al centro del cerchio principale è fissato grosso modo dal quoziente tra l'angolo giro e l'angolo al centro del cerchio sotteso da un disco di dimensione media e posto ad una distanza media. Va inoltre aggiunto che, quando in numero non sufficientemente elevato, le semirette pseudocasuali ottenibili da un generatore omonimo non sono affatto distribuite uniformemente lungo l'angolo giro e le prime semirette generate tendono a 'catalizzare' la posizione dei dischi raggruppandoli[7]. Una soluzione è quella di generare semirette in maniera sistematica, ad esempio ogni due gradi per un totale di 180 semirette, aggiungendo una leggera casualità nella direzione. L'insieme degli angoli delle semirette è rappresentabile in Python da una *lista*[8]:

```
for i in range(0,nAngles):
    angleList = angleList +
    [(i + random.random() -
    0.5) * 2 * piGreek / nAngles]
```

dove `angleList` è la lista degli angoli generati; `random.random()` è una funzione pseudo-casuale che genera numeri con distribuzione uniforme nell'intervallo (0, 1); l'operazione di somma in questo caso indica a Python di aggiungere un nuovo angolo i-esimo alla lista attuale, il cui valore è un numero pseudo-casuale compreso nell'intervallo

$$\left(\frac{2\pi}{nAngles}(i-0.5), \frac{2\pi}{nAngles}(i+0.5)\right) \qquad (1)$$

Le tavole originali di Ishihara sono composte da dischi con un elevato grado di impacchettamento non ottenibile con dischi tutti del medesimo raggio. Come parametro di raffronto possiamo usare la densità di impacchettamento che è definibile nel nostro caso semplicemente come il quoziente tra l'area della superficie ricoperta dai dischi e l'area del cerchio di raggio uguale alla distanza dal centro del cerchio principale del centro del disco più distante. Gauss dimostrò che un reticolo regolare di dischi identici raggiunge la massima densità di impacchettamento pari a $\frac{\pi}{6}\sqrt{3} \simeq 0.907$ quando i loro centri vengono disposti su un reticolo esagonale; tuttavia l'impacchettamento di dischi entro un cerchio non è così efficiente[9].

Nelle tavole di Ishihara l'impacchettamento ha una connotazione casuale per cui il grado di ricoprimento sarebbe notevolmente inferiore se non ricorressimo a dischi di raggi 'r' differenti i quali hanno il vantaggio, se opportunamente selezionati, di permettere la raffigurazione anche di piccoli dettagli. La scelta dei raggi, del loro numero e della quantità di dischi corrispondenti richiede un po' di pratica; tuttavia, se il desiderio è riprodurre il più fedelmente possibile le tavole originali di Ishihara, allora è sufficiente fare un censimento dei raggi ivi utilizzati. In ogni caso è naturale scegliere una lista di raggi in modo tale che sia possibile collocare i dischi più piccoli tra gli spazi lasciati dai dischi maggiori.

Una volta scelte la direzione lungo cui collocare il centro del disco, la lista dei raggi e la relativa proporzione nella quantità di dischi, è necessario determinare per ciascun disco la posizione più vicina al centro del cerchio principale senza generare sovrapposizioni con i dischi già presenti. A tale scopo è necessario definire un criterio di non sovrapposizione tra il disco da collocare 'c' e ciascun disco i-esimo 'c_i' già collocato. Il disco c_i ha raggio r_i e centro (x_i, y_i). Invece, quando espresso in coordinate polari con origine nel centro del cerchio principale, il disco 'c' ha raggio 'r' e centro $(\rho\cos\theta, \rho\sin\theta)$.

Si può dimostrare la seguente condizione di non sovrapposizione tra i due dischi:

$$\big((\rho < \rho_-) \vee (\rho > \rho_+)\big) \wedge \ (\rho > 0) \wedge (\Delta > 0) \qquad (2)$$

dove ρ_- e ρ_+ sono le due soluzioni generate dall'eventuale sovrapposizione dei due dischi 'c_i' e 'c':

$$\rho_\pm = x_i\cos\theta + y_i\sin\theta \pm \sqrt{\Delta} \qquad (3)$$

mentre il radicando è

$$\Delta = (ax_i + by_i)^2 + (R_i + R)^2 - x_i^2 - y_i^2 \qquad (4)$$

Supponiamo che il cerchio principale contenga già al suo interno in prossimità del suo centro un certo numero di dischi non sovrapposti ed impacchettati. La collocazione del nuovo disco all'interno del cerchio principale avviene come segue.

L'insieme degli angoli disposti pseudouniformente lungo l'intervallo $(0, 2\pi)$ è rappresentato da una lista A[].

L'insieme dei dischi già collocati entro il cerchio principale è associato a tre liste P[], X[], Y[] rappresentanti rispettivamente la lista dei raggi, quella delle ascisse e quella delle ordinate dei centri dei dischi.

Sia R[] la lista dei raggi dei dischi ancora da collocare entro il cerchio principale.

Consideriamo il primo elemento della lista A[] supponendo che abbia valore θ_1.

7. In maniera analoga a quanto accade per i tetramini nel gioco di tassellatura Tetris.

8. Una lista è una successione indicizzata di elementi eterogenei.

9. http://mathworld.wolfram.com/CirclePacking.html.

Per ogni disco c_i già collocato all'interno del cerchio principale vengono determinati gli insiemi $I(\theta_1, c_i)$ dei valori ρ per i quali sussiste eventualmente l'equazione (2). Ogni insieme $I(\theta_1, c_i)$ è eventualmente l'unione di due intervalli $\big(0, \rho^-(\theta_1, c_i)\big)$ e $\big(\rho^+(\theta_1, c_i), \infty\big)$[10]. Viene calcolata l'intersezione tra tutti gli $I(\theta_1, c_i)$ al variare di 'i' ottenendo un insieme di valori di ρ, $I(\theta_1)$,

$$I(\theta_1) = \bigcap_{i=1}^{N} I(\theta_1, c_i) \tag{5}$$

eventualmente unione di più intervalli disgiunti

$$I(\theta_1) = \bigcup_{j=1}^{T-1} \Big(\rho_j(\theta_1), \rho_{j+1}(\theta_1) \Big) \tag{6}$$

dove $\rho_1(\theta_1) < \rho_2(\theta_1) < \cdots < \rho_T(\theta_1)$ da cui viene ricavato il valore inferiore $\rho_1(\theta_1)$ che può essere un numero positivo o nullo (es. per il primo disco ad essere posto entro il cerchio principale) che viene aggiunto ad una lista temporanea $\rho[\,]$.

L'angolo θ_2 successivo a θ_1 viene preso dalla lista A[ ] e viene calcolato $\rho_1(\theta_2)$, incrementando quindi la lista $\rho[\,]$ con un secondo valore. L'operazione viene ripetuta per tutti gli angoli entro la lista A[ ] ed al termine viene calcolato il valore minimo tra tutti gli angoli:

$$\rho_{\min} = \min_i \rho_1(\theta_i) \tag{7}$$

A questo punto siamo in grado di collocare il disco 'c' e quindi di aggiornare le liste P[ ], X[ ], Y[ ] rispettivamente con i tre valori R[1], $\rho_{\min}\cos\theta_{\min}$, $\rho_{\min}\sin\theta_{\min}$ dove $\theta_{\min}$ è l'angolo della direzione dove la distanza minima è $\rho_{\min}$.

La lista R[ ] viene accorciata elidendo l'elemento R[1] e facendo in tal modo assumere alla lista un nuovo valore R[1] da cui ripartire per la determinazione delle coordinate del disco successivo. L'algoritmo viene arrestato quando la lista R[ ] è vuota.

6 La generazione delle tavole

Le figure che compaiono nelle tavole originali di Ishihara sono principalmente numeri, ma ci sono anche percorsi curvilinei. Una volta ottenuta la figura circolare con i dischi di varie dimensioni applicando l'algoritmo descritto in precedenza, il passo successivo è colorare i singoli dischi in modo da riprodurre le tavole di Ishihara o comunque delle tavole con delle figure che abbiano la medesima funzione.

Abbiamo visto all'inizio alcuni esempi di analisi delle tavole di Ishihara. In generale le tavole presentano delle regioni colorimetricamente distinte, ma ciascuna regione ha una certa variabilità cromatica che si manifesta in una leggera differenza di tinta, di saturazione oppure di brillanza tra i dischi. Riprodurre tale variabilità è semplice grazie alle funzioni di generazioni di numeri pseudocasuali presenti nella grande maggioranza dei linguaggi di programmazione.

La figura di tipo numerico, o di altra natura, invece richiede un algoritmo apposito per essere riprodotta. Una soluzione ovvia e semplice è quella di impostare un sistema di riferimento cartesiano ad esempio centrato sulla figura circolare della tavola. La figura da riprodurre viene quindi descritta entro il sistema di riferimento con una poligonale. Se ci sono più figure (ad esempio nel caso della tavola composta dalla combinazione del numero '3' e del suo tratto 'complementare' per formare il numero '8') l'operazione viene semplicemente ripetuta. Figure del piano non semplicemente connesse, come ad esempio la lettera 'R', devono prima essere rese semplicemente connesse usando come loro approssimazione un'opportuna poligonale (Fig. 6).

10. In pratica, ogni insieme $I(\theta_1, c_i)$ dipendente dall'angolo θ_1 e dal disco c_i contiene tutti i valori di ρ per cui non vi è sovrapposizione tra i due dischi 'c' e 'c_i'.

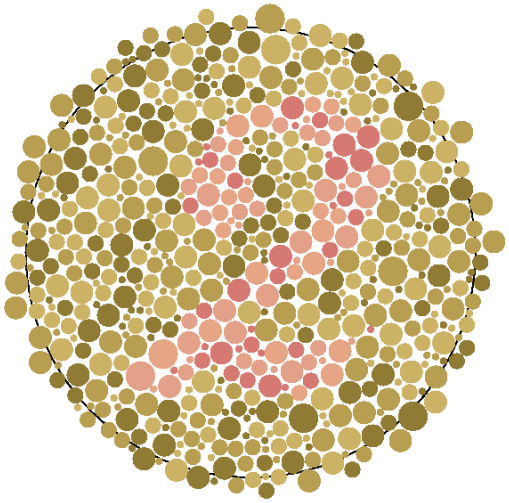


FIGURA 3: Il numero '2' ottenuto con 600 dischi, 270 semirette, la densità è pari a 0.63.

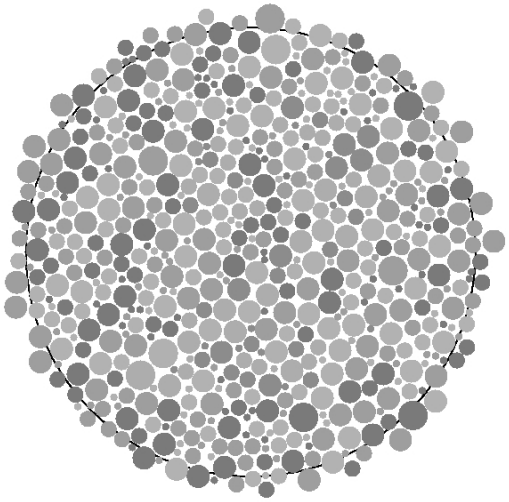


FIGURA 4: Il numero '2' della Fig. 3 visto da un soggetto acromatopo.

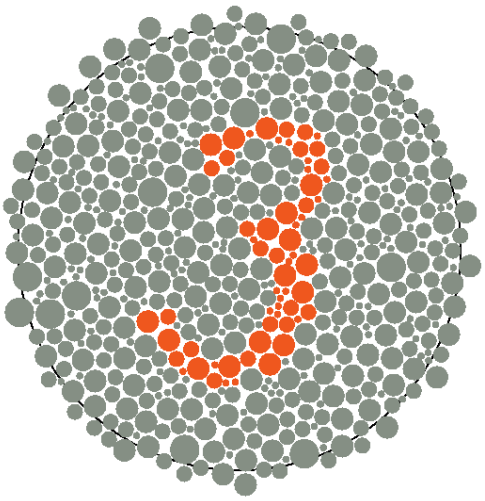


Figura 5: Il numero '3'. Invisibile ad un acromatopo a differenza del numero '12' della tavola 1 di Ishihara.

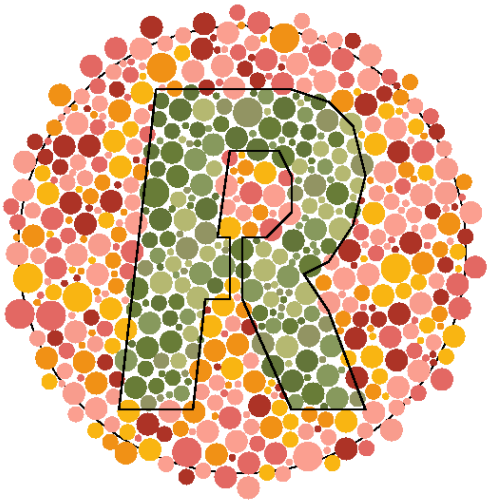


Figura 6: La lettera 'R' resa con una regione poligonale semplicemente connessa.

Una volta definita la poligonale, ad esempio mediante una lista ordinata delle coordinate dei suoi vertici, rimane da definire l'algoritmo per discriminare i dischi interni al suo interno da quelli esterni. Per determinare se un disco è situato all'interno o all'esterno della poligonale è sufficiente calcolare l'angolo sotteso dalla poligonale rispetto ad esso[11]:

$$\sum_{i=0}^{N-1} \arcsin\left(\frac{(v_{i \bmod N} \times v_{(i+1) \bmod N})_z}{|v_{i \bmod N}| \cdot |v_{(i+1) \bmod N}|}\right) \qquad (8)$$

dove 'N' è il numero di vertici della poligonale, v_i è il vettore con centro nel disco e vertice nel punto i-esimo della poligonale; il numeratore dell'argomento della funzione arcoseno è la componente lungo l'asse 'z' del prodotto vettore tra i due vettori v_i e v_{i+1} assumendo che entrambi giacciano nel piano 'x, y' di un sistema cartesiano 'x, y, z'. L'angolo sotteso assume valore '0', se il centro del disco è esterno alla poligonale, oppure '2π', se interno.

Il modulo Python genera un file LaTeX contenente il codice PSTricks incluso in un'ambiente `picture`. Dimensione, unità di misura, posizione della tavola sono impostati come parametri nel codice. Le variabili sono il numero di dischi, il numero di semirette, ossia di angoli, le liste delle coordinate x, y dei punti delle poligonali, le liste dei colori dei dischi interni ed esterni alle poligonali. Per quanto riguarda le liste delle coordinate dei centri dei dischi e quella dei loro raggi, il modulo ha sia la possibilità di generarli ex-novo oppure di caricare il file dati corrispondente. Quest'ultima possibilità permette di salvare tempo macchina, talvolta minuti, e riduce il tempo di esecuzione ad una frazione di secondo. L'algoritmo di generazione dei dischi non è ottimizzato, dato che una volta ottenuta una disposizione soddisfacente dei dischi, questa può essere riutilizzata in quanto non fornisce alcun suggerimento al soggetto esaminato. Ovviamente Ishihara non poteva ai suoi tempi riutilizzare una disposizione dei dischi per un'altra combinazione di colori o di figure; tutte le sue tavole, quindi, sono uniche da questo punto di vista.

7 Conclusioni

Le tavole di Ishihara vengono ancora impiegate dopo un secolo dalla loro prima compilazione. La loro riproduzione con gli strumenti della tipografia digitale richiede la definizione di algoritmi che risolvano problemi quali la disposizione pseudocasuale ravvicinata di dischi senza sovrapposizione e la loro partizione in regioni con colorazioni atte all'individuazione di soggetti con anomalie visive cromatiche. Tale riproduzione può essere fatta generando del codice PSTricks mediante un modulo Python. La riproduzione delle tavole mediante esecuzione di codice Python ha permesso di chiarire il ruolo di alcuni parametri grafici e colorimetrici insiti nelle tavole di Ishihara, quali il numero di dischi, la distribuzione delle loro dimensioni e delle loro coordinate colorimetriche.

Riferimenti bibliografici

Claudio Oleari, a. c. d. (1998). *Misurare il colore*. Hoepli.

Ishihara, S. (1972). *Tests for colour-blindness*. Kanehara Shuppan Co., Ltd. - Tokyo - Japan.

▷ Renato Battistin
Via G. Matteotti 1
I-31041 CORNUDA TV
rbattistin at apf dot it

11. Ricordiamo che nel piano euclideo l'*indice di avvolgimento*, $\nu(\gamma, P)$ di una curva γ rispetto ad un punto P è, detto in parole semplici, il numero di volte che la curva si avvolge attorno al punto. Rigorosamente, nel piano complesso: $\nu(\gamma\, P) = \frac{1}{2\pi i} \oint_\gamma \frac{\mathrm{d}z}{z-P}$.

Grafica 3D con Geogebra e Ti*k*Z

Luciano Battaia

Sommario

Questo articolo esamina la possibilità di utilizzare un software di geometria dinamica come Geogebra per costruire immagini geometriche tridimensionali con successiva proiezione parallela in 2D, in modo da consentire una agevole esportazione in codice PGF/Ti*k*Z.

L'utilizzo di un software come Geogebra rende possibile costruire immagini decisamente complesse senza la necessità di programmare direttamente in codice PGF/Ti*k*Z e sfruttando le potenzialità di una programmazione sostanzialmente basata sull'uso del mouse.

Sono discussi i dettagli della proiezione parallela e proposti numerosi esempi dei risultati grafici che si possono ottenere, in alcuni casi anche fornendo i codici sorgenti.

Abstract

This article considers the opportunity of using a software of dynamical geometry such as Geogebra to produce tridimensional geometric pictures with subsequent 2D parallel projection, in order to allow an easy export in PGF/Ti*k*Z code.

The use of a software such as Geogebra makes the production of very complex pictures easy enough, without any programming in PGF/Ti*k*Z code and taking advantage of a programming substantially mouse-driven.

Details of the parallel projection are discussed and numerous examples of the pictures produced are proposed, in some cases also including source codes.

1 Grafica 3D in LaTeX

La produzione di grafica vettoriale di alta qualità da inserire in un documento prodotto con LaTeX[1], in particolare annotata nello stesso stile del documento, è sempre stata una necessità essenziale per tutti gli utenti, soprattutto per la produzione di testi di carattere scientifico.

Per la grafica 2D non ci sono particolari problemi e i due strumenti più diffusi PSTricks e PGF/Ti*k*Z[2] con i loro pacchetti derivati risolvono egregiamente (quasi) ogni problema. È altresì da segnalare che molte situazioni possono essere gestite direttamente da LaTeX con l'ambiente *picture*, senza l'uso di pacchetti esterni: si veda l'articolo di Claudio Beccari (Beccari, 2011) per una estesa trattazione delle possibilità offerte da questo quasi sconosciuto ambiente.

I sorgenti contenenti codice PSTricks hanno, in LaTeX, necessariamente bisogno di una doppia compilazione per poter produrre output in pdf, ma con le tecniche oggi disponibili[3] tutto avviene in maniera automatica in background, senza bisogno di particolari interventi da parte dell'autore. I sorgenti contenenti codice Ti*k*Z, al contrario, sono direttamente compilabili in pdfLaTeX. Anche se le potenzialità teoriche di PSTricks sono superiori a quelle di Ti*k*Z, in realtà (quasi) tutto quello che si produce con PSTricks si può produrre con Ti*k*Z e la scelta tra uno dei due pacchetti è più che altro questione di gusto personale. L'autore di questo articolo ha lavorato per anni, con grande soddisfazione, solo con PSTricks[4], per poi passare progressivamente a Ti*k*Z, che ora viene usato in maniera quasi esclusiva nonostante permanga la convinzione che la scrittura di un sorgente in PSTricks sia decisamente più semplice ed intuitiva che non in Ti*k*Z. Il seguito di questo articolo conterrà esclusivamente riferimenti ed esempi in Ti*k*Z.

Una gran parte del materiale compilato con LaTeX e contenente grafica 3D utilizza software esterni di modellazione tridimensionale per produrre immagini in formato accettabile dal compilatore pdfLaTeX, immagini che poi vengono annotate con codice LaTeX extra che consente di collocare le scritte nei posti giusti: si veda a questo proposito l'articolo di Agostino de Marco (De Marco, 2008).

Altre strategie utilizzano sistemi esterni programmabili con appositi linguaggi che producono in uscita frammenti di codice anche in Ti*k*Z. L'uso di *Sketch*[5], per esempio, è discusso nell'articolo di Agostino de Marco (De Marco, 2007) e, con qualche variante peraltro molto significativa che presenteremo più avanti, è sostanzialmente una strada di questo tipo che intendiamo proporre in questo articolo. Anche TeXgraph di Patrick Fradin[6] usa questa strategia per produrre grafici molto sofisticati e complessi sia di funzioni di più variabili che di tipo più propriamente geometrico, con la possibilità di produrre frammenti di codice Ti*k*Z e, cosa molto interessante, di esportare in formato

1. In questo articolo faremo riferimento solo al processore LaTeX e non a LuaTeX, XeTeX, o altro.
2. In seguito richiamato solo con Ti*k*Z.
3. Per esempio usando il pacchetto auto-pst-pdf.
4. Si veda per esempio Battaia (2007).
5. La cui ultima versione è però del 2012.
6. http://texgraph.tuxfamily.org/

POV-Ray con cui si ottengono immagini veramente spettacolari[7] Se qualcuno ha voglia di sobbarcarsi la fatica di studiare un (complesso) linguaggio di programmazione grafica, questa potrebbe essere la soluzione giusta.

Un ulteriore approccio è quello fornito da *Asymptote*, il cui codice può essere inserito direttamente in un sorgente LaTeX mediante il pacchetto asymptote. Una ampia introduzione a questa tecnica si trova in De Marco (2009).

Esistono ancora altri approcci (*Ipe*[8] e *Inkscape*[9] per esempio) e naturalmente numerosi software commerciali: tutto questo testimonia l'importanza del problema della creazione di grafica vettoriale di alta qualità (naturalmente non solo per l'inserimento in documenti LaTeX).

Per la gestione dei grafici di funzioni di due variabili e di superfici tridimensionali si può utilizzare il pacchetto pgfplots che si appoggia a TikZ e del quale si può leggere una concisa ma efficace introduzione nell'articolo di Agostino de Marco e Roberto Giacomelli (De Marco e Giacomelli, 2011). Qui però noi siamo interessati alla creazione di grafici più propriamente "geometrici": solidi di vario tipo, in particolare poliedri, quadriche e loro intersezioni, ecc. È teoricamente anche possibile utilizzare TikZ con un po' di programmazione, come mostra l'esempio della figura 1, realizzata da Tomasz M. Trzeciak e che si può reperire su http://texample.net/tikz/examples/map-projections/.

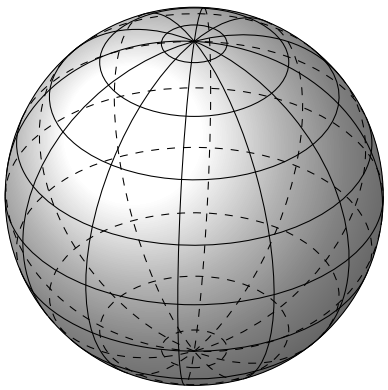


Figura 1: Sfera con meridiani e paralleli creata da Tomasz M. Trzeciak con TikZ.

Riportiamo, per un utile confronto con quanto diremo successivamente, il codice di questa figura, suddiviso in due parti: la prima da inserire nel preambolo del documento e costituita da alcuni comandi appositamente creati, la seconda da inserire nel corpo del documento e contenente le istruzioni vere e proprie per costruire la figura.

7. Esempi si trovano nel già citato sito di TeXgraph mentre un'immagine, collegata a quanto diremo e che riteniamo eccezionale, si può trovare in http://www.les-mathematiques.net/phorum/read.php?10,661628,661774.

8. http://ipe.otfried.org/

9. https://inkscape.org/it/

```
%Inserire nel preambolo del documento
\newcommand\pgfmathsinandcos[3]{
  \pgfmathsetmacro#1{sin(#3)}
  \pgfmathsetmacro#2{cos(#3)}}
\newcommand\LongitudePlane[3][current plane]{
  \pgfmathsinandcos\sinEl\cosEl{#2}
  \pgfmathsinandcos\sint\cost{#3}
  \tikzset{#1/.style={cm={\cost,%
    \sint*\sinEl,0,\cosEl,(0,0)}}}}
\newcommand\LatitudePlane[3][current plane]{
  \pgfmathsinandcos\sinEl\cosEl{#2}
  \pgfmathsinandcos\sint\cost{#3}
  \pgfmathsetmacro\yshift{\cosEl*\sint}
  \tikzset{#1/.style={cm={\cost,0,0,%
    \cost*\sinEl,(0,\yshift)}}}}
\newcommand\DrawLongitudeCircle[2][1]{
  \LongitudePlane{\angEl}{#2}
  \tikzset{current plane/.prefix %
    style={scale=#1}}
  \pgfmathsetmacro\angVis{atan(sin(#2)*%
    cos(\angEl)/sin(\angEl))}
  \draw[current plane] (\angVis:1) %
    arc (\angVis:\angVis+180:1);
  \draw[current plane,dashed](\angVis-180:1)%
    arc (\angVis-180:\angVis:1);}
\newcommand\DrawLatitudeCircle[2][1]{
  \LatitudePlane{\angEl}{#2}
  \tikzset{current plane/.prefix %
    style={scale=#1}}
  \pgfmathsetmacro\sinVis{sin(#2)/cos(#2)*%
    sin(\angEl)/cos(\angEl)}
  \pgfmathsetmacro\angVis{asin(min(1,%
    max(\sinVis,-1)))}
  \draw[current plane] (\angVis:1) %
    arc (\angVis:-\angVis-180:1);
  \draw[current plane,dashed](180-\angVis:1)%
    arc (180-\angVis:\angVis:1);}

%Inserire nel corpo del documento
\begin{tikzpicture}
\def\R{2.5}
\def\angEl{35}
\filldraw[ball color=white](0,0)circle(\R);
\foreach \t in {-80,-60,...,80} &
  { \DrawLatitudeCircle[\R]{\t} }
\foreach \t in {-5,-35,...,-175} %
{ \DrawLongitudeCircle[\R]{\t} }
\end{tikzpicture}
```

È proprio l'esame del codice di questa figura che ci ha stimolato ad approfondire lo studio di TikZ per verificare la possibilità di creare le figure di cui avevamo bisogno. Purtroppo il manuale di TikZ di Till Tantau, è estremamente voluminoso e la curva di apprendimento, dopo una pendenza iniziale abbastanza lieve, si impenna rapidamente. Abbiamo perciò deciso di tentare altre strade.

2 L'intervento di Geogebra

Per motivi didattici abbiamo da sempre estesamente usato diversi software di geometria dinamica, principalmente *Cabri* e, successivamente, *Geogebra*, soprattutto per il fatto che il

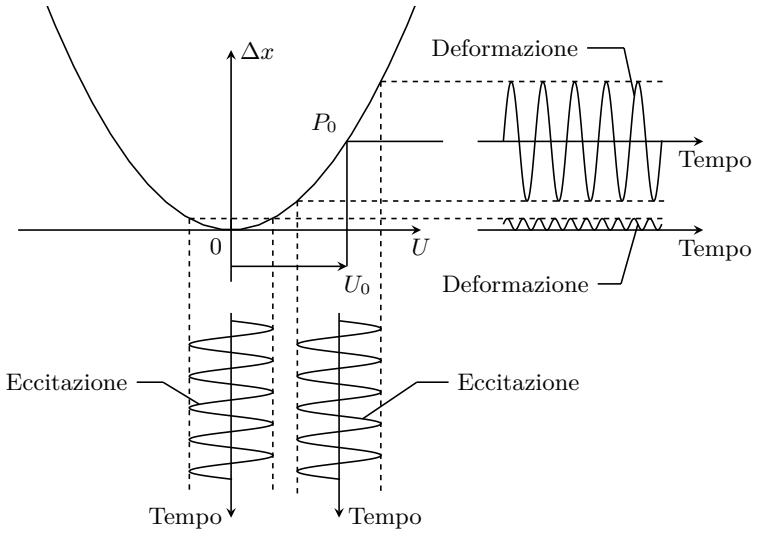


FIGURA 2: Una figura prodotta con Geogebra ed esportata in TikZ, realizzata per una tesi di laurea in fisica.

suo uso è gratuito per scopi non commerciali. L'utilizzo basico di questo software è talmente semplice che può essere adatto anche a bambini di scuola elementare: si veda per esempio lo splendido sito *Splashragazzi*, all'indirizzo `http://splashscuola.altervista.org/esercizi/geogebra/geogebra_elementare.shtml`.

La cosa interessante per quanto ci riguarda è la possibilità di costruire figure complesse bidimensionali ed esportarle producendo codici[10] TikZ perfettamente funzionanti, anzi se non molto "puliti": alcuni semplici adattamenti, in particolare riguardo la posizione delle etichette, possono soddisfare comunque molte esigenze dei produttori di documenti contenenti grafica 2D di alta qualità, senza conoscere quasi per nulla i complicati meandri della programmazione diretta in TikZ. Il vantaggio nell'uso di Geogebra è principalmente legato al fatto che la maggior parte delle costruzioni può avvenire con il mouse, senza ricorrere all'uso di alcun codice. Solamente per richieste abbastanza complesse è necessaria l'immissione di alcune istruzioni in una apposita riga di comando. Anche se la filosofia di questa produzione è piuttosto del tipo WYSIWYG, quindi a priori lontana da quella di un utente LaTeX puro, riteniamo che per la produzione di immagini sia decisamente da preferire. La quasi totalità delle immagini bidimensionali che abbiamo prodotto negli anni per i testi di appunti di argomento matematico per il triennio terminale del Liceo Scientifico e i primi anni di università (corsi di laurea in Ingegneria, Economia, Tecnologie multimediali) sono stati prodotti con questo sistema, e si tratta di migliaia di figure. L'avvento di questa possibilità ha reso per noi obsoleto l'uso di pacchetti per altro molto ben fatti come per esempio tkz-euclide, o il suo predecessore della serie PSTricks, pst-eucl.

Con un po' di conoscenze più approfondite di Geogebra si possono realizzare con pochi click del mouse e alcuni comandi molto semplici anche figure molto complesse. A titolo d'esempio si veda la figura 2, realizzata per una tesi di laurea specialistica in fisica.

Da qualche tempo è uscita la versione 5 di Geogebra che consente anche la costruzione di figure tridimensionali e la trattazione di funzioni in due variabili e di superfici dello spazio. Purtroppo per la versione 3D non è prevista l'esportazione in codice TikZ (e nemmeno in altri formati) e non crediamo che la cosa sarà possibile nemmeno in futuro, perlomeno in un futuro non troppo lontano.

A questo punto è venuto spontaneo chiedersi: se non è possibile l'esportazione diretta delle figure 3D, perché non realizzare direttamente in Geogebra una proiezione di qualche tipo da 3D a 2D e poi esportare quest'ultima? In fondo ogni figura 3D non è altro che una opportuna proiezione 2D di un oggetto tridimensionale.

Nella successiva ricerca in internet per valutare l'opportunità e la fattibilità di una idea del genere ci siamo imbattuti nell'interessante articolo di Keith Wolcott (WOLCOTT, 2012), in cui sono discussi numerosi problemi della grafica 3D con TikZ, con la proposta di alcune soluzioni. Il lavoro di Wolcott prende le mosse dalla necessità di costruire un grafico che evidenzi la circonferenza intersezione di due sfere. Viene estesamente usato il codice di Tomasz M. Trzeciak, di cui abbiamo già parlato, per creare l'immagine di una sfera con meridiani e paralleli e l'articolo si conclude con l'immagine desiderata, che riproduciamo nella figura 3.

Purtroppo, come lo stesso Wolcott riconosce, la figura non è corretta, in quanto non sono gestite accuratamente le parti delle due sfere che si sovrap-

10. Geogebra è anche in grado di produrre codici PSTricks e *Asymptote*, ma non ci occupiamo qui di questa capacità.

pongono. Questo fatto rende evidente le difficoltà della gestione di questo tipo di problemi con il solo codice TikZ (gestione che riteniamo comunque possibile), e dunque la necessità di cercare altre soluzioni.

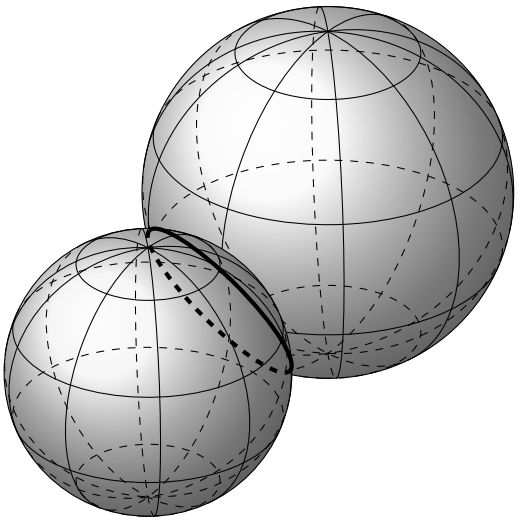


FIGURA 3: Intersezione di due sfere in TikZ, come presentata nell'articolo di Keith Wolcott.

La prima cosa che abbiamo cercato di realizzare è la riproduzione della sfera di Tomasz M. Trzeciak, utilizzando Geogebra e limitando all'essenziale la programmazione in codice TikZ. Il risultato ottenuto è proposto nella figura 4: in seguito saranno proposti i dettagli su come ottenerla.

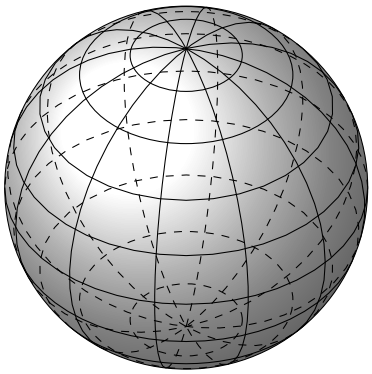


FIGURA 4: Sfera con meridiani e paralleli, costruita esportando il codice da Geogebra.

Anche di questa figura proponiamo il codice sorgente, nella figura 6. Nel seguito non proporremo altri codici: la cosa sarebbe poco significativa, in quanto essi vengono prodotti in automatico da Geogebra.

L'intervento manuale sul codice prodotto da Geogebra è limitato all'introduzione della riga contenente il comando `\shadedraw`, che serve a produrre l'ombreggiatura. È immediato constatare che il codice di Trzeciak, sopra riportato, è decisamente più elegante e conciso, ma non bisogna dimenticare che il codice di Geogebra è, appunto, prodotto in automatico da Geogebra, quasi senza interventi da parte dell'utente. Questo codice è meno pulito anche perché usa solo comandi elementari di TikZ, in particolare quasi esclusivamente il comando `\draw`. C'è anche un'altra osservazione da fare: in un documento in cui ci sono decine di figure il preambolo del documento, dove vanno inseriti i nuovi comandi, diventa decisamente complesso (rendendo quindi conveniente avere un file separato che li raggruppi) e la gestione di eventuali modifiche potrebbe diventare problematica.

Dal punto di vista del risultato finale ci pare che non sia percepibile alcuna differenza tra le sfere delle figure 1 e 4.

Prima di passare ai dettagli della tecnica che intendiamo proporre è doveroso citare il pacchetto tikz-3dplot di Jeff Hein che in sostanza implementa alcune macro in linguaggio TikZ al fine di disegnare, in maniera relativamente semplice, sistemi di coordinate tridimensionali e semplici figure tridimensionali. Il pacchetto è stato sviluppato dall'autore per la sua esigenza di produrre accurate immagini tridimensionali di vettori e, in questo campo, raggiunge pienamente lo scopo. Il suo utilizzo per situazioni più generali, pur se in teoria possibile, è nella pratica decisamente complesso. Riportiamo, nella figura 5, un esempio d'uso prodotto da uno studente di fisica e il cui codice si può reperire in `http://tex.stackexchange.com/questions/39577/`.

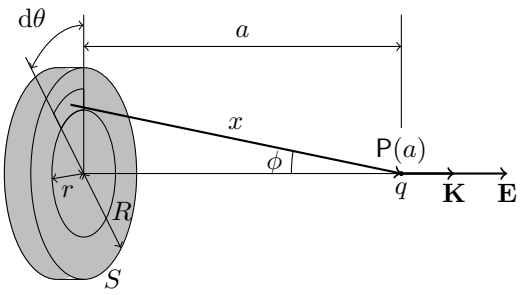


FIGURA 5: Una figura 3D prodotta con il pacchetto tikz-3dplot.

Anche con questo pacchetto, tuttavia, vi è la necessità di programmazione (anche abbastanza complessa) in codice TikZ e, inoltre, le figure a cui siamo interessati sono ancora difficilmente trattabili.

3 Un po' di matematica dietro le quinte

Geogebra è un software di geometria dinamica[11] molto articolato e con numerose possibilità. In so-

11. Riportiamo esplicitamente la Licenza d'uso: "You are free to copy, distribute and transmit GeoGebra free of charge for non-commercial purposes". Dettagli e condizioni si trovano nella finestra di guida del programma.

```
\begin{tikzpicture}[x=0.8cm,y=0.8cm]
\clip(-3.,-1.5) rectangle (5.,5.5);
\shadedraw[ball color = white](1.,2.) circle (3.);
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.87*3.*cos(\t r)+%
0.493*2.16*sin(\t r)},{-0.493*3.*cos(\t r)+-0.87*2.16*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.87*3.*cos(\t r)+0.493*2.16*sin(\t r)},{-0.493*3.*cos(\t r)+-0.87*2.16*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.475*3.*cos(\t r)+0.88*1.4772116295183126*sin(\t r)},{-0.88*3.*cos(\t r)+-%
0.475*1.477*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.475*3.*cos(\t r)+%
0.88*1.477*sin(\t r)},{-0.88*3.*cos(\t r)+-0.475*1.477*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.113*3.*cos(\t r)+0.994*0.3990666646784674*sin(\t r)},{-0.994*3.*cos(\t r)+-%
0.113*0.399*sin(\t r)});
\draw [shift={(1.,2.)}]  plot[domain=-3.142:0.,variable=\t]({1.*3.*cos(\t r)+%
0.*1.928*sin(\t r)},{0.*3.*cos(\t r)+1.*1.928*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.113*3.*cos(\t r)+%
0.994*0.399*sin(\t r)},{-0.994*3.*cos(\t r)+-0.113*0.399*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.228*3.*cos(\t r)+-0.974*0.786*sin(\t r)},{0.974*3.*cos(\t r)+-0.228*0.786*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.228*3.*cos(\t r)+-%
0.974*0.786*sin(\t r)},{0.974*3.*cos(\t r)+-0.228*0.786*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.608*3.*cos(\t r)+-0.794*1.76*sin(\t r)},{0.794*3.*cos(\t r)+-0.608*1.76*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.608*3.*cos(\t r)+-%
0.794*1.76*sin(\t r)},{0.794*3.*cos(\t r)+-0.608*1.76*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.964*3.*cos(\t r)+-0.265*2.263*sin(\t r)},{0.265*3.*cos(\t r)+-0.964*2.263*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.964*3.*cos(\t r)+-%
0.265*2.263*sin(\t r)},{0.265*3.*cos(\t r)+-0.964*2.263*sin(\t r)});
\draw [rotate around={0.:(1.,4.186)}] (1.,4.186) ellipse (0.927 and 0.596);
\draw [rotate around={0.:(1.,3.86)}] (1.,3.86) ellipse (1.763 and 1.133);
\draw [rotate around={0.:(1.,0.141)},dash pattern=on 3pt off 3pt] (1.,0.141) %
ellipse (1.763 and 1.133);
\draw [rotate around={0.:(1.,-0.186)},dash pattern=on 3pt off 3pt] (1.,-0.186) &
ellipse (0.927 and 0.596);
\draw [shift={(1.,3.35)}] plot[domain=-3.797:0.656,variable=\t]({1.*2.427*cos(\t r)+%
0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\draw [shift={(1.,3.35)},dash pattern=on 3pt off 3pt]  plot[domain=0.656:2.486,variable=%
\t]({1.*2.427*cos(\t r)+0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\draw [shift={(1.,2.71)}] plot[domain=-3.418:0.276,variable=\t]({1.*2.853*cos(\t r)+%
0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,2.71)},dash pattern=on 3pt off 3pt]  plot[domain=0.276:2.865,variable=%
\t]({1.*2.853*cos(\t r)+0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({1.*3.*cos(\t r)+0.*1.928*sin(\t r)},{0.*3.*cos(\t r)+1.*1.928*sin(\t r)});
\draw [shift={(1.,1.29)}] plot[domain=3.418:6.007,variable=\t]({1.*2.853*cos(\t r)+%
0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,1.29)},dash pattern=on 3pt off 3pt]  plot[domain=-0.276:3.418,variable=%
\t]({1.*2.853*cos(\t r)+0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,0.649)}] plot[domain=3.797:5.628,variable=\t]({1.*2.427*cos(\t r)+%
0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\draw [shift={(1.,0.649)},dash pattern=on 3pt off 3pt]  plot[domain=-0.656:3.797,variable=%
\t]({1.*2.427*cos(\t r)+0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\end{tikzpicture}
```

FIGURA 6: Listato del codice per la sfera della figura 4, codice quasi integralmente generato da Geogebra.

stanza esso prevede[12] due finestre per grafici 2D, una finestra per grafici 3D, una finestra contenente un foglio di calcolo abbastanza potente, una finestra CAS per calcoli numerici e algebrici (anche simbolici), un calcolatore di probabilità, una finestra di algebra dove sono proposte le coordinate dei punti, le equazioni delle curve, le lunghezze dei

12. Almeno nella versione 5.0.268 attualmente disponibile: il software è in rapido sviluppo e non sono escluse novità anche nel breve periodo.

segmenti, ecc., rappresentati nelle finestre grafiche. Tutte le finestre possono interagire tra di loro: per quanto ci interessa, per esempio, tutto quanto ottenuto nella finestra 3D può essere opportunamente trasferito alla finestra 2D. Le opzioni non finiscono qui: c'è, per esempio, la possibilità di una programmazione Javascript, ma non intendiamo occuparci qui in dettaglio di questo. Infine è prevista una riga di comando per tutto quanto non si può definire direttamente con il mouse. Segnaliamo infine che eventuali annotazioni testuali nelle varie finestre grafiche si inseriscono con codice LaTeX.

L'idea di base per le costruzioni che vogliamo realizzare è, come già più sopra accennato, di eseguire una proiezione parallela da 3D a 2D direttamente in Geogebra per poi esportarla, usando l'apposita funzione di Geogebra, in codice TikZ. Supposto dunque di avere un sistema cartesiano ortogonale nello spazio tridimensionale (che Geogebra visualizza nella finestra 3D) con l'asse z verticale ascendente, indichiamo con α una rotazione attorno all'asse verticale e con β una rotazione attorno ad un asse orizzontale. Si può allora costruire nel piano 2D il triedro proiezione, i cui vettori di base indichiamo con $\vec{i}$, $\vec{j}$, $\vec{k}$, con le seguenti formule:

$$\begin{aligned} \vec{i} &= \big(-\cos(\alpha), -\sin(\alpha)\sin(\beta)\big) \\ \vec{j} &= \big(\sin(\alpha), -\cos(\alpha)\sin(\beta)\big) \\ \vec{k} &= \big(0, \cos(\beta)\big) \end{aligned} \quad .$$

Queste formule possono essere scritte in diverse (limitate) varianti: abbiamo scelto quella che ci era più famigliare in quanto è quella usata da Herbert Voss (Voss, 2014) per il suo pacchetto (molto interessante) pst-3dplot, pacchetto che abbiamo estesamente usato prima di scoprire le possibilità offerte da Geogebra.

Se si fissa l'origine in $\mathsf{O} = (0, 0)$, come conviene, in Geogebra si devono creare due "slider" angolari con i nomi α e β, dopodiché i vettori indicati si costruiscono con i seguenti comandi, dove abbiamo tenuto conto del fatto che Geogebra non usa particolari formattazioni per la scrittura dei vettori,

$$\begin{aligned} \mathrm{i} &= \text{Vettore}[\mathsf{O}, (-\cos(\alpha), -\sin(\alpha)\sin(\beta))] \\ \mathrm{j} &= \text{Vettore}[\mathsf{O}, (\sin(\alpha), -\cos(\alpha)\sin(\beta))] \\ \mathrm{k} &= \text{Vettore}[\mathsf{O}, (0, \cos(\beta))] \end{aligned}$$

Fatto questo, se si ha un punto $\mathsf{P} = (x_\mathsf{P}, y_\mathsf{P}, z_\mathsf{P})$ nella finestra 3D, il suo corrispondente nella proiezione sarà

$$\mathsf{P}' = x_\mathsf{P}\,\vec{i} + y_\mathsf{P}\,\vec{j} + z_\mathsf{P}\,\vec{k},$$

ovvero, nel linguaggio di Geogebra,

$$\mathsf{P}' = x(\mathsf{P})\,\mathrm{i} + y(\mathsf{P})\,\mathrm{j} + z(\mathsf{P})\,\mathrm{k}.$$

Se invece si ha una curva C di equazioni parametriche[13] $(f(t), g(t), h(t))$, con il parametro t opportunamente variabile tra due estremi, la sua proiezione 2D avrà, sempre nel linguaggio di Geogebra, equazioni parametriche

$$\begin{aligned} x' &= f(t)\,x(\mathrm{i}) + g(t)\,x(\mathrm{j}) + h(t)\,x(\mathrm{k}) \\ y' &= f(t)\,y(\mathrm{i}) + g(t)\,y(\mathrm{j}) + h(t)\,y(\mathrm{k}) \end{aligned} \quad .$$

In sostanza usando queste formule si può costruire, come vedremo in dettaglio su qualche esempio, la proiezione 2D di un qualunque grafico realizzato in Geogebra 3D.

A questo punto, lavorando nella finestra 2D di Geogebra, si può scegliere la migliore vista per la figura in esame, variando opportunamente gli angoli α e β: è anche questa una delle opportunità offerte dall'uso di Geogebra, in quanto è molto difficile in generale decidere a priori qual è l'angolo di visuale più opportuno per una data immagine, e solo sperimentando dinamicamente si riesce a trovare la soluzione adatta.

Naturalmente nemmeno Geogebra banalizza il problema della grafica 3D. In effetti, è chiaro che chi ha bisogno di immagini di questo tipo deve avere una adeguata preparazione matematica, in particolare deve poter maneggiare con disinvoltura vettori ed equazioni di curve e superfici dello spazio: nulla si ottiene gratuitamente!

Alla luce di queste formule vediamo come si può ottenere l'immagine della sfera proposta nella figura 4. Costruita, nella finestra 3D, la sfera di centro l'origine e raggio r, se ne costruisce la proiezione 2D, che è semplicemente la circonferenza di centro l'origine e raggio r. Si procede poi con i paralleli e i meridiani. Supponiamo di voler costruire 5 paralleli. Essi si troveranno alle seguenti latitudini: $-60°$, $-30°$, $0°$, $30°$, $60°$. I piani che li contengono avranno equazioni, rispettivamente,

$$\begin{aligned} z &= r\sin(-60°) \;;\; z = -r\sqrt{3}/2 \\ z &= r\sin(-30°) \;;\; z = -r/2 \\ z &= r\sin(0°) \;;\; z = 0 \\ z &= r\sin(30°) \;;\; z = r/2 \\ z &= r\sin(60°) \;;\; z = r\sqrt{3}/2 \end{aligned} \quad ,$$

piani che si possono tracciare nella finestra 3D semplicemente scrivendone l'equazione nella barra di inserimento.

Per ciascuno di essi si fa costruire a Geogebra l'intersezione sfera-piano e, nella circonferenza di intersezione, si scelgono con il mouse 5 punti a caso; determinate, con le formule sopra riportate, le proiezioni di questi punti nella finestra 2D, si costruisce, con l'apposito comando di Geogebra, la conica per essi: si avrà la proiezione 2D del parallelo. Analogo discorso per i meridiani.

13. Geogebra è in grado di gestire curve in equazione parametrica anche molto complesse.

Una volta tracciati tutti i meridiani e i paralleli voluti si sceglie dinamicamente qual è l'angolo di visuale desiderato. Successivamente, per ciascuno di essi si stabilisce qual è la parte visibile e quale quella invisibile scegliendo, per quest'ultima, se visualizzarla in tratteggio e magari con uno spessore di linea ridotto, o non visualizzarla affatto. A questo punto basta solo esportare il tutto con l'apposito comando di Geogebra in formato TikZ, scegliendo quale parte della finestra esportare: il codice prodotto è direttamente funzionante e vi si possono apportare le modifiche desiderate. Generalmente le eventuali etichette non sono piazzate nella maniera migliore, ma è molto facile adeguare il piazzamento ai propri gusti.

In realtà con un po' di esperienza nell'uso di Geogebra il processo che abbiamo descritto può essere automatizzato in gran parte. Per esempio è possibile costruire in modo ricorsivo i meridiani e i paralleli, utilizzando il comando Successione di Geogebra, ma abbiamo voluto qui proporre una tecnica assolutamente di base.

4 Intersezione di due sfere

Ritorniamo all'immagine dell'intersezione di due sfere proposta da Keith Wolcott.

In questo caso si tratta di disegnare due sfere con i loro meridiani e paralleli come sopra descritto, tenendo conto che una delle due sfere può avere centro l'origine, l'altra è meglio che abbia un centro variabile parametricamente[14] in modo da poter poi scegliere la visuale più adatta ad evidenziare la circonferenza intersezione. È opportuno che anche il raggio delle sfere (o almeno di una della due) sia parametrizzato, per la stessa ragione di scelta della visuale.

Fatta disegnare da Geogebra la circonferenza intersezione, si scelgono su di essa cinque punti e, con la tecnica già descritta sopra, si trova la sua ellisse immagine in 2D: a questo punto non resta che scegliere la visuale più opportuna e, successivamente, per ciascun meridiano, parallelo e per l'ellisse intersezione la parte visibile e quella invisibile, decidendo come visualizzarla. Anche se quest'ultima parte è un po' noiosa (e richiede una pazienza certosina!), non ha bisogno di alcuna tecnica particolare: si tratta di un'operazione di base in Geogebra.

Il risultato che abbiamo ottenuto è mostrato nella figura 7.

La figura 7 mostra la corretta gestione delle parti di sfera sovrapposte e, a nostro modo di vedere, presenta un angolo di visuale particolarmente adatto al caso. Abbiamo scelto di non disegnare affatto, nemmeno con un tratteggio speciale o con una linea sottile, le parti di sfera sovrapposte, e questo per una migliore leggibilità della figura. Non costerebbe alcuna fatica ulteriore visualizzare anche qualcosa di queste parti nascoste, particolarmente se si volesse produrre una figura di dimensioni molto più grandi, ed è anche questa, ancora una volta, una delle opportunità offerte facilmente dal tipo di approccio che stiamo proponendo.

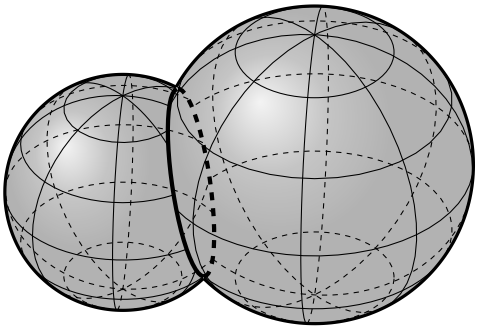


FIGURA 7: Intersezione di due sfere ottenuta mediante codice esportato da Geogebra.

5 Spirale su una sfera

L'esempio che proponiamo ora richiede un minimo di matematica in più, ma nessuna ulteriore fatica dal punto di vista della costruzione del codice. Si tratta di disegnare una linea complessa, in questo caso una spirale con infinite spire, su una sfera. In situazioni come queste è indispensabile utilizzare le equazioni parametriche della curva in questione, equazioni che si possono comunque trovare facilmente in rete, per esempio sul famoso *MathWorld* di Wolfram[15]. Se la sfera ha centro l'origine, raggio r e a è una costante, le equazioni sono le seguenti

$$\begin{aligned} x(t) &= \frac{\cos t}{\sqrt{1+a^2t^2}} \\ y(t) &= \frac{\sin t}{\sqrt{1+a^2t^2}} \\ z(t) &= \frac{-at}{\sqrt{1+a^2t^2}} \end{aligned} .$$

Nell'immagine che proponiamo abbiamo scelto $r = 1.8$ e $a = 0.12$. Al solito se queste quantità sono impostate parametricamente, si può sperimentare con l'immagine in Geogebra al fine di scegliere i valori più opportuni e solo alla fine esportare il risultato. Il tracciamento della curva, sia in 3D (cosa di poco interesse in questo caso) sia in 2D non richiede alcuna particolare tecnica: basta usare le formule per la proiezione 2D di una curva parametrica 3D, formule che abbiamo già introdotto. Al solito la cosa che resta da fare è valutare quali parti sono visibili e quali sono invisibili, e come renderle nell'immagine da inserire nel documento LaTeX.

Il risultato è mostrato nella figura 8.

14. Operazione di routine in Geogebra!

15. http://mathworld.wolfram.com/

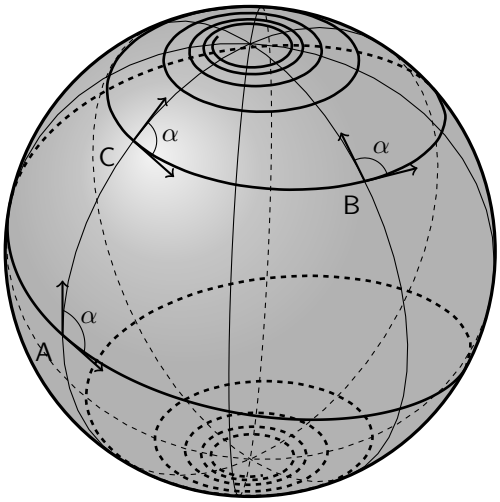


Figura 8: Spirale tracciata su una sfera. Codice ottenuto per esportazione da Geogebra.

Se il parametro t nelle equazioni della spirale tende all'infinito, la spirale stessa tende al polo Sud e al polo Nord della sfera.

È d'obbligo una ulteriore precisazione su questa figura, come su analoghe che coinvolgono il tracciamento di curve complesse, ma questo è un problema collegato alla struttura di TikZ[16], e non ha niente a che fare con le difficoltà di programmazione nel linguaggio del pacchetto. Il problema è che curve complesse non sono tracciabili con i comandi di TikZ e bisogna ricorrere a programmi esterni che generino una matrice di punti per il tracciamento. La scelta più semplice è quella di usare GNUPLOT. Per fare questo occorre avere installato il programma, che è distribuito con una sua propria licenza di software free. Occorre inoltre avere abilitato il compilatore scelto per LaTeX all'uso di programmi esterni, per esempio con pdflatex –shell-escape <nomefile>.tex. Tuttavia, a parte l'abilitazione del compilatore che si deve fare una volta per tutte e che serve anche per altre cose, Geogebra gestisce automaticamente nell'esportazione in TikZ anche questo fatto!

La figura 8 evidenzia la proprietà fondamentale di questo tipo di spirale, e cioè il fatto che in ogni punto l'angolo tra la spirale stessa e il meridiano passante per quel punto è costante. Si noti anche che dalla stessa figura si evince che la proiezione parallela non mantiene gli angoli, come è ovvio, ma mantiene il parallelismo, come si può notare se si osservano nei due punti A e C due dei quattro vettori che sono paralleli nello spazio e che rimangono paralleli nella proiezione.

16. Anzi, ancora più correttamente, allo stesso motore matematico di LaTeX.

6 Poliedri

Nell'esperienza di produzione di grafica 3D che abbiamo fin qui accumulato, uno dei campi in cui l'intervento di Geogebra è stato davvero provvidenziale è quello dei poliedri e dei loro sviluppi.

Geogebra implementa infatti delle routine per la costruzione di poliedri e in particolare dei solidi platonici con la visualizzazione dinamica dei loro sviluppi. Una volta costruito un poliedro regolare e il suo sviluppo (ad un qualunque stadio) nella finestra 3D, è sufficiente proiettare i vertici uno ad uno nel piano 2D e poi congiungerli con segmenti (operazione standard in Geogebra) per ottenere la figura 2D richiesta. Non rimane, come già più volte ricordato, che scegliere il miglior angolo di visuale possibile, valutare quali sono le parti visibili e quali quelle invisibili, decidendo come rendere queste ultime ed esportare il tutto in formato TikZ. Qui bisogna poi intervenire sul codice per inserire le ombreggiature, tenendo conto del punto da cui si vuol fare provenire la luce.

Proponiamo nella figura 9 l'immagine del dodecaedro ottenuta con la tecnica descritta. Non ci si lasci ingannare dall'apparente semplicità della figura: un calcolo "manuale" della posizione[17] dei vertici del dodecaedro non è per niente semplice, meno che mai il relativo calcolo se fatto eseguire direttamente dal codice TikZ.

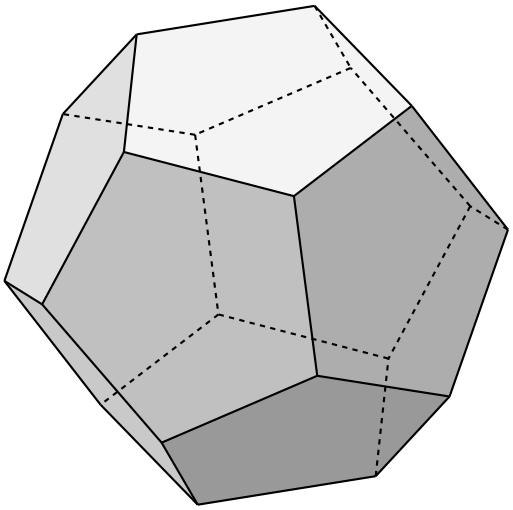


Figura 9: Il dodecaedro regolare. Figura ottenuta mediante esportazione di codice da Geogebra.

La figura 10 mostra invece un primo passo nello sviluppo dello stesso dodecaedro. In questo caso la determinazione manuale o mediante codice TikZ della posizione dei vari vertici è ovviamente ancora più complessa (anche se naturalmente possibile).

17. Per chi avesse voglia di cimentarsi con questo problema suggeriamo la bellissima costruzione che Euclide propone nel Libro XIII dei suoi Elementi, alla proposizione 17.

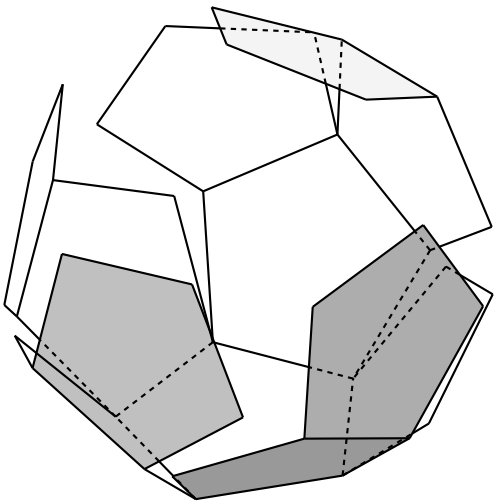


Figura 10: Il dodecaedro regolare: un primo passo nello sviluppo piano.

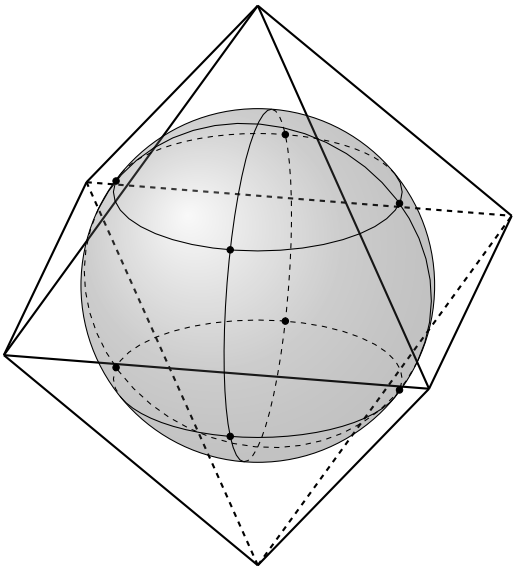


Figura 11: L'ottaedro regolare e la sfera inscritta. Sono evidenziati i meridiani e i paralleli passanti per quattro degli otto punti di tangenza, nonché i punti di tangenza stessi.

Con la tecnica basata su Geogebra, invece, non si introduce praticamente nessuna difficoltà ulteriore rispetto alla costruzione della figura originale. Anzi, se si programma correttamente la figura in Geogebra, la figura 10 si ottiene dalla 9 in pochi minuti.

Non ci sono particolari difficoltà nemmeno nel rappresentare la sfera inscritta o circoscritta a uno dei poliedri regolari: nella figura 11 proponiamo, a titolo d'esempio, il caso dell'ottaedro e della sfera inscritta. Si deve pensare che l'ottaedro sia trasparente per la sfera e i suoi meridiani e paralleli, mentre l'ottaedro e la sfera sono opachi per i loro lati o parti di meridiani e paralleli invisibili: è questa una scelta puramente personale e che può essere comunque facilmente modificata.

Nella figura 12 proponiamo la trattazione dell'ottaedro in una vista diversa da quella della figura 11 e di uno dei suoi sviluppi piani. Tra i tanti possibili sviluppi ne abbiamo scelto uno classico, ovvero quello proposto originariamente da Daniel Barbaro (Barbaro, 1769) nel Capitolo IIII, Parte Terza, pp. 48-49, col titolo "SPIEGATURA, DRITTO, ET ADOMBRATIONE del corpo detto octoedro".

La cosa interessante è che, sempre sfruttando le capacità di Geogebra, è facile anche costruire le curve descritte dai vertici del poliedro durante lo sviluppo.

Infatti, nello sviluppo di un poliedro in Geogebra è possibile introdurre un parametro variando il quale si possono vedere dinamicamente le varie fasi dello sviluppo stesso. Si può impostare Geogebra in modo da far lasciare una traccia ai vertici interessati durante il moto ed è anche possibile visualizzare questa traccia nella proiezione 2D: a questo punto è facile costruire, con una macro di Geogebra, una cubica di Bézier che approssimi, magari a tratti, la traiettoria descritta. Questa curva è direttamente esportabile in linguaggio TikZ, ottenendo così la sua rappresentazione nella figura LaTeX.

Quasi sempre sarebbe anche possibile ricavare le equazioni parametriche delle traiettorie descritte dai vertici nello sviluppo, tuttavia ciò richiederebbe conoscenze non elementari di meccanica dei corpi rigidi: per esempio, con riferimento alla figura 12, la curva descritta dal vertice A risulta dalla composizione di un moto di rotazione del triangolo ABC attorno allo spigolo BC, e da un moto di rotazione del triangolo OBC attorno al vertice fisso O. La tecnica descritta che usa le tracce lasciate dal vertice A durante il suo moto è invece elementare e non richiede alcun supplemento di calcolo, permettendo di ottenere facilmente la curva Γ.

7 Il pallone da calcio

Una volta acquisita dimestichezza con i solidi platonici si può esperimentare l'ampliamento della tecnica al caso di altri solidi, per esempio i cosiddetti solidi archimedei che si differenziano dai solidi platonici sostanzialmente[18] perché le facce, pur essendo ancora costituite tutte da poligoni regolari, non sono tutte uguali, ma sono costituite da due o tre tipi di poligoni regolari. Tuttavia essi mantengono la proprietà dei solidi platonici di avere gli spigoli tutti uguali.

18. In realtà la definizione di solido archimedeo è un po' più complessa, ma non intendiamo qui approfondire la questione. Del resto anche la nozione di poliedro regolare è tutt'altro che banale: per esempio, unendo per una faccia due tetraedri regolari si ottiene un poliedro che ha sei facce uguali e tutte costituite da poligoni (triangoli equilateri) regolari, ma non è un poliedro regolare.

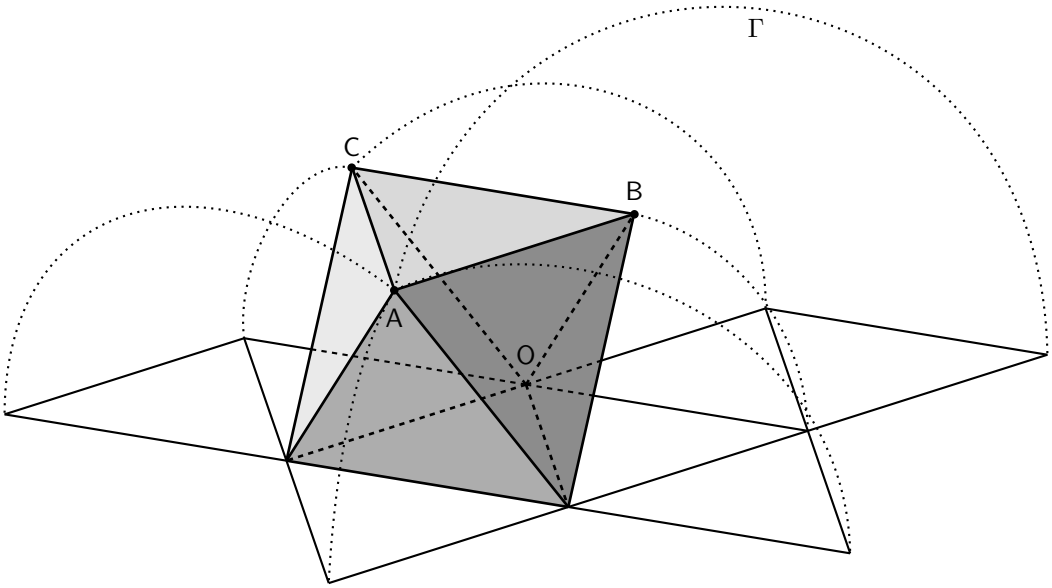


FIGURA 12: Uno dei possibili sviluppi piani dell'ottaedro: lo sviluppo completo in 3D, con evidenziazione delle curve descritte dai vertici nello sviluppo.

La tecnica più semplice per costruire alcuni di questi poliedri è quella di eseguire "troncature" a partire dai vertici dei poliedri regolari.

Per eseguire queste troncature basta considerare delle sfere di raggio variabile e centro nei vertici del poliedro: le loro intersezioni con gli spigoli del poliedro forniscono i vertici delle "facce di troncatura". Occorre una certa pazienza per ottenere i risultati voluti, ma non si tratta di tecniche complesse. Proponiamo, nella figura 13, un'immagine che mostra lo schema di questo processo nel caso in cui il poliedro di partenza sia un icosaedro. Si noti che i poligoni ottenuti dalla troncatura sono pentagoni regolari, mentre i poligoni "residui" delle facce originali dell'icosaedro sono esagoni non regolari.

Quando il raggio della sfera troncatrice raggiunge 1/3 della lunghezza degli spigoli dell'icosaedro, gli esagoni residui diventano regolari e si ottiene il cosiddetto *icosaedro troncato*: ha ovviamente 20 facce esagonali regolari (tante quante erano le facce dell'icosaedro originale) e 12 facce pentagonali regolari (tante quanti erano i vertici dell'icosaedro originale). Il risultato è proposto nella figura 14.

L'icosaedro troncato costituisce la forma base del pallone da calcio, il quale è la proiezione sulla sfera circoscritta dello stesso icosaedro troncato.

La rappresentazione grafica di un pallone da calcio richiede di poter costruire la proiezione, sulla sfera circoscritta, di tutti gli spigoli dell'icosaedro stesso, che sono 90; in realtà basta proiettare quelli visibili, perché chiaramente in una rappresentazione del pallone da calcio non ha molto interesse mostrare la struttura della "faccia nascosta".

La proiezione in questione è una proiezione centrale dal centro della sfera (che supponiamo essere l'origine per semplicità) sulla superficie della sfera

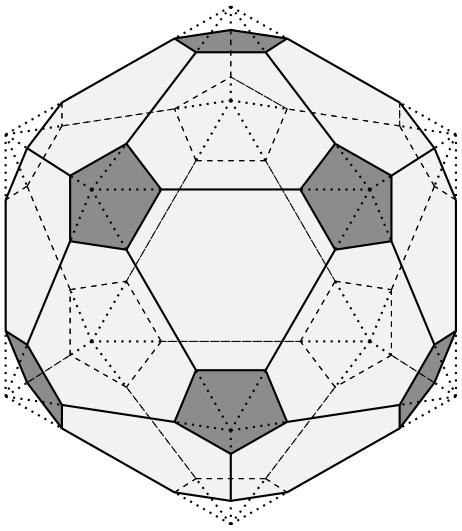


FIGURA 13: Schema della troncatura dell'icosaedro a partire dai vertici.

stessa. In linea di principio la cosa non è complessa e si articola, per ogni spigolo, nelle fasi di seguito descritte.

Supponiamo di avere un segmento $\overline{\mathsf{AB}}$ di estremi $(x_\mathsf{A}, y_\mathsf{A}, z_\mathsf{A})$ e $(x_\mathsf{B}, y_\mathsf{B}, z_\mathsf{B})$. Si comincia con il parametrizzare il segmento stesso, nel modo canonico:

$$\mathsf{P}(t)\colon \begin{cases} f(t) &= x_\mathsf{A} + (x_\mathsf{B} - x_\mathsf{A})t \\ g(t) &= y_\mathsf{A} + (y_\mathsf{B} - y_\mathsf{A})t \\ h(t) &= z_\mathsf{A} + (z_\mathsf{B} - z_\mathsf{A})t \end{cases}, \ 0 \le t \le 1.$$

Si calcola poi la norma di $P(t)$:

$$||\mathsf{P}(t)|| = \sqrt{f^2(t) + g^2(t) + h^2(t)}.$$

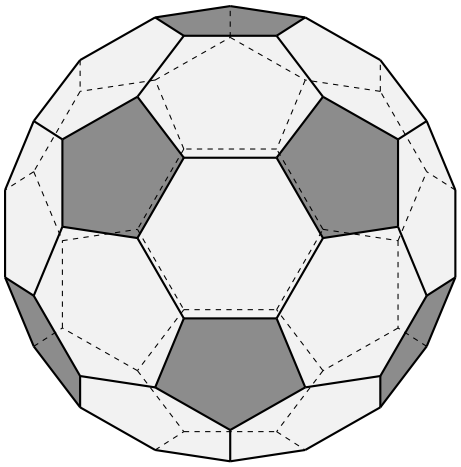


FIGURA 14: L'icosaedro troncato.

La proiezione del segmento $\overline{\mathsf{AB}}$ sulla sfera unitaria ha equazioni parametriche

$$\mathsf{Q}(t) = \frac{\mathsf{P}(t)}{||\mathsf{P}(t)||}.$$

A questo punto non resta altro da fare che usare la proiezione parallela già descritta per ottenere la stessa curva in 2D.

La figura 15 illustra il risultato di questo processo nel caso di un triangolo ABC.

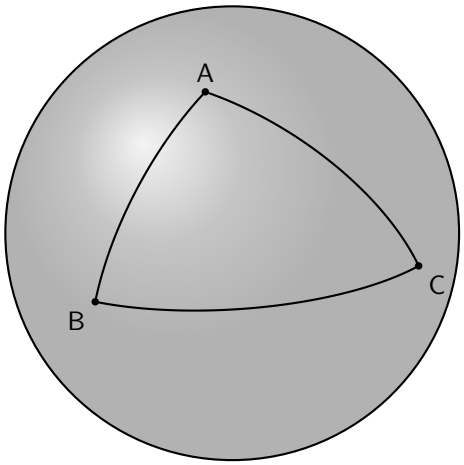


FIGURA 15: Proiezione centrale di un triangolo sulla sfera unitaria.

Rimane poi il problema del riempimento delle varie parti con il colore voluto, nel caso del pallone solo i pentagoni che sono colorati di nero.

Le formule della proiezione sono abbastanza semplici, tuttavia la loro scrittura in Geogebra richiede un bel po' di pazienza e la ricerca di strategie per velocizzare il procedimento: i codici per i vari segmenti differiscono infatti solo per i nomi dei punti, quindi una volta costruito un segmento basta fare un copia e incolla e modificare solo i nomi dei punti.

A puro titolo di curiosità riportiamo il codice Geogebra per il segmento $\overline{\mathsf{AB}}$ della figura 15: in ogni caso un copia e incolla di questo codice può essere utilizzato per qualunque segmento presente nella finestra 3D di Geogebra, con il solo cambio dei nomi degli estremi! Questo codice riproduce esattamente la combinazione delle formule di proiezione centrale e di proiezione parallela che abbiamo descritto.

```
Curva[((x(A)+(x(B)-x(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+(y(B)-y(A))t))^2+
 ((z(A)+(z(B)-z(A))t))^2))x(i)+
 ((y(A)+(y(B)-y(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))x(j)+
 ((z(A)+(z(B)-z(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))x(k),((x(A)+
 (x(B)-x(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))y(i)+
 ((y(A)+(y(B)-y(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+
 ((z(A)+(z(B)-z(A))t))^2))y(j)+
 ((z(A)+(z(B)-z(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))y(k),t,0,1]
```

Il risultato finale nel caso in esame è proposto nella figura 16

FIGURA 16: Il pallone da calcio ottenuto mediante proiezione centrale su una sfera di un icosaedro troncato.

Un ultimo suggerimento pratico, prima di chiudere l'argomento "sportivo" di questo articolo. Il codice TikZ di una figura come quella del pallone

da calcio è abbastanza complesso (per la figura proposta è costituito da ben 250 linee) e non conviene costruire un unico maxi file in Geogebra, quanto piuttosto spezzarlo in piccoli file (per esempio uno per ogni esagono e una per ogni pentagono) e poi esportare il codice un pezzo alla volta. Questo sistema rende anche decisamente più agevole la colorazione delle varie parti.

8 Sezioni coniche e sferiche

La costruzione di grafici relativi alle sezioni coniche è uno dei cavalli di battaglia di chi si occupa di grafica geometrica 3D. Proponiamo qui alcuni esempi, senza particolari commenti: la tecnica base da utilizzare è quella ormai ampiamente descritta.

Cominciamo con il proporre una figura relativa ad una situazione poco nota e che è contenuta nella Proposizione I.5 del trattato di Apollonio[19] sulle coniche. La proposizione recita più o meno così: *Ci sono due serie di sezioni coniche circolari in un cono obliquo a base circolare, una serie costituita dalle sezioni parallele alla base, l'altra costituita dalle sezioni subcontrarie alla prima serie.*

La figura 17 illustra questo fatto. Si noti che abbiamo scelto di non visualizzare i paralleli della superficie conica, inoltre delle "linee meridiane" abbiamo mostrato solo quelle visibili: tutto questo per una migliore leggibilità della figura.

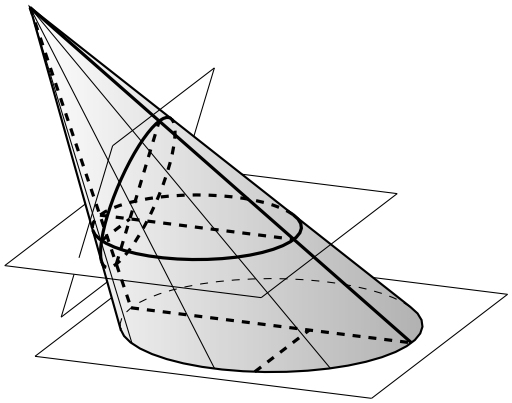


Figura 17: Le due serie di sezioni circolari in un cono obliquo a base circolare: sezioni parallele alla base e sezioni subcontrarie.

Non ci sono particolari difficoltà nella realizzazione in Geogebra 3D e successiva proiezione 2D, in quanto le linee coinvolte sono tutte coniche o segmenti. L'unico problema (di natura squisitamente matematica e non informatica) è quello di determinare l'inclinazione del piano di sezione per la sezione subcontraria: per fare questo basta seguire le indicazioni di Apollonio nella dimostrazione della già citata proposizione I.5.

19. Una versione in notazione moderna di questo celebre trattato in 8 libri, di cui solo i primi 4 ci sono pervenuti, si può trovare in Heath (1896).

La seconda immagine che proponiamo si riferisce ad una sezione conica vera e propria, precisamente l'iperbole. Tra le tante figure che si possono realizzare per questo problema abbiamo scelto la 18. Essa appare particolarmente complessa, per la separazione delle due parti del cono ottenute sezionandolo con un piano. In realtà in Geogebra basta realizzare un'unica figura del cono con l'iperbole sezione (realizzazione standard perché le linee coinvolte sono solo segmenti e coniche); successivamente si nasconde (con i comandi di Geogebra) prima una parte e poi l'altra, esportandole separatamente. I due frammenti di codice TikZ si uniscono spostando il secondo di alcuni centimetri: basta racchiudere il codice relativo come segue

```
\begin{scope}[xshift=2.4cm]
<codice della seconda parte>
\end{scope}
```

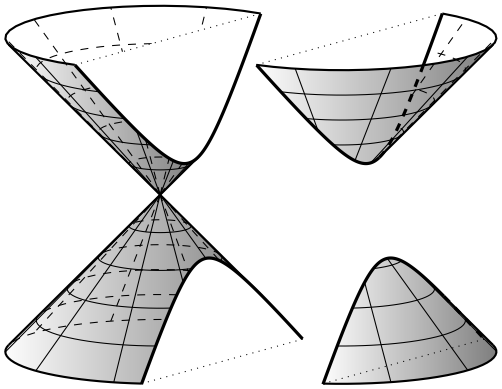


Figura 18: Sezione conica per ottenere un'iperbole.

La terza immagine che proponiamo in questa serie si riferisce all'intersezione tra un cilindro e una sfera e ci è stata suggerita durante le lezioni di un corso di Matematica per il design. Sculture galleggianti con questa forma sono state realizzate da Marta Pan[20] nel 1973 per il City Hall di Dallas.

La curva intersezione è una *finestra di Viviani* e il suo tracciamento è semplice solo se si utilizzano le relative equazioni parametriche, equazioni che comunque si possono trovare facilmente in letteratura. Il resto è ormai standard con i metodi indicati.

9 Un'ultima immagine

Chiudiamo questa trattazione sulla grafica 3D in TikZ con l'immagine di un poliedro, precisamente l'icosaedro elevato, suggeritaci da Claudio Beccari e che ha una lunga storia. Quasi certamente la sua prima pubblicazione si trova nel *De divina proportione* di Luca Pacioli, di cui sono conservate due copie (una a Ginevra e una a Milano) eseguite

20. Scultrice francese di origine ungherese, nata nel 1923 a Budapest e morta nel 2008 a Parigi.

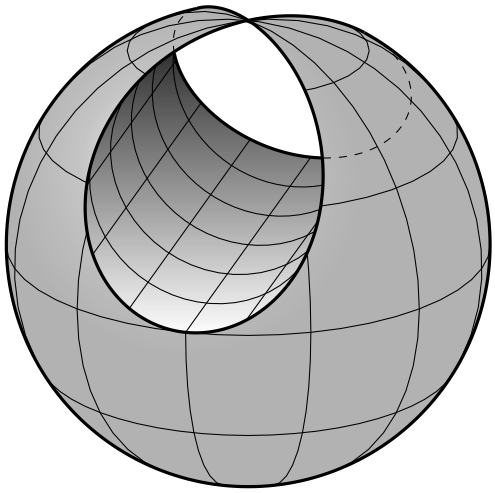


FIGURA 19: Intersezione tra un cilindro e una sfera: solido ottenuto asportando dalla sfera la parte di cilindro.

da diversi amanuensi verso la fine del 1400, oltre alle copie a stampa del 1509. I disegni sono redatti da Leonardo da Vinci, che rivela anche qui la sua mano maestra eseguendo i disegni con precisione fotografica. Ci sono due immagini di questo poliedro nel *De divina*, una del poliedro solido e una del poliedro vuoto: ne riproduciamo la prima (Icosaedron Elevatum Solidum), la seconda (Icosaedron Elevatum Vacuum) è decisamente troppo complessa (ma non è detto che non si possa fare!).

Il motivo per cui Claudio Beccari ci ha suggerito questo poliedro è il fatto che, a Urbino, è comunemente usato, nella sua forma leonardesca "vacua", per produrre lampadari in cui gli spigoli sono di ottone e le facce di vetro, con la lampadina al centro: il suo nome corrente è, appunto, *Stella di Urbino*.

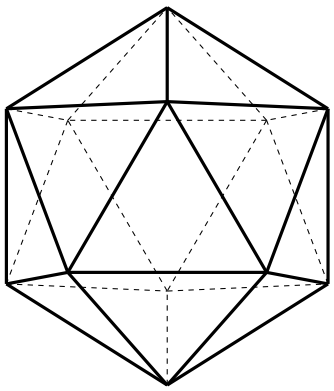


FIGURA 20: L'icosaedro.

Il solido deriva dall'icosaedro (il più complesso dei solidi platonici, con 20 facce costituite da triangoli equilateri), in cui ogni faccia è sostituita da un tetraedro regolare sporgente dall'icosaedro originale: ne proponiamo l'immagine nella figura 20.

Siccome ogni faccia viene sostituita da 3 facce (ancora triangoli equilateri) si ottiene un solido con 60 facce, tutte costituite da triangoli equilateri. Si tratta di un poliedro non convesso, anzi di un poliedro *stellato*. Esso non rientra comunque nella categoria dei poliedri stellati quasi-regolari di Keplero-Poinsot, ma è di estremo interesse avendo le facce tutte uguali (è una delle condizioni per cui un poliedro possa dirsi regolare: purtroppo mancano le altre).

La sua realizzazione in Geogebra è teoricamente standard in quanto si tratta di costruire, a partire da ogni faccia triangolare, un nuovo tetraedro che abbia quella faccia come base; in realtà è decisamente complessa in quanto occorre riuscire a gestire un elevato numero di facce, spigoli e vertici. Una volta eseguita la costruzione e la sua proiezione 2D, la cosa più delicata è valutare il migliore angolo visuale possibile, in modo da rendere efficacemente la struttura del solido. Qui non ha senso pensare di visualizzare in tratteggio o qualche altro modo gli spigoli nascosti: la figura diventerebbe illeggibile, tanto che nemmeno Leonardo ci ha provato!

È proprio nella ricerca del miglior angolo visuale possibile che si manifesta tutta la potenza del metodo grafico basato su un software di geometria dinamica: solo un genio come Leonardo ha potuto concepire la figura senza i moderni strumenti grafici (o fotografici). A parte questo fatto (molto delicato, lo ripetiamo), la costruzione non richiede tecniche particolari, in quanto si tratta di una figura costituita solo da segmenti. La figura 21 che proponiamo è sostanzialmente simile a quella di Leonardo, con un limitata variazione dell'angolo di inclinazione orizzontale.

Per una figura come questa, il colore sarebbe essenziale per una immediata comprensione della struttura completa del solido. Una strategia che può essere implementata, e che fornirebbe un sicuro aiuto nella facile lettura della figura, è quella dell'introduzione di un'opportuna ombreggiatura. In teoria non è una cosa complessa, soprattutto se si vuole una sorgente all'infinito e quindi con raggi paralleli: si tratterebbe di determinare, per ogni vertice, la sua ombra e, a partire da essa, costruire l'ombra di tutte le parti. Sicuramente la figura diventa complessa, ma sfruttando le capacità di Geogebra di visualizzare solo le parti di interesse, può essere implementata.

10 Conclusioni

Riteniamo che la tecnica proposta possa essere vantaggiosamente impiegata per la produzione di una gran parte delle figure tipo geometrico richieste in un testo di matematica. Accoppiata con pgfplots consente di produrre testi anche complessi utilizzando solo codice LaTeX, senza interventi di software esterni. Per produrre immagini molto complesse occorre naturalmente una buona conoscenza

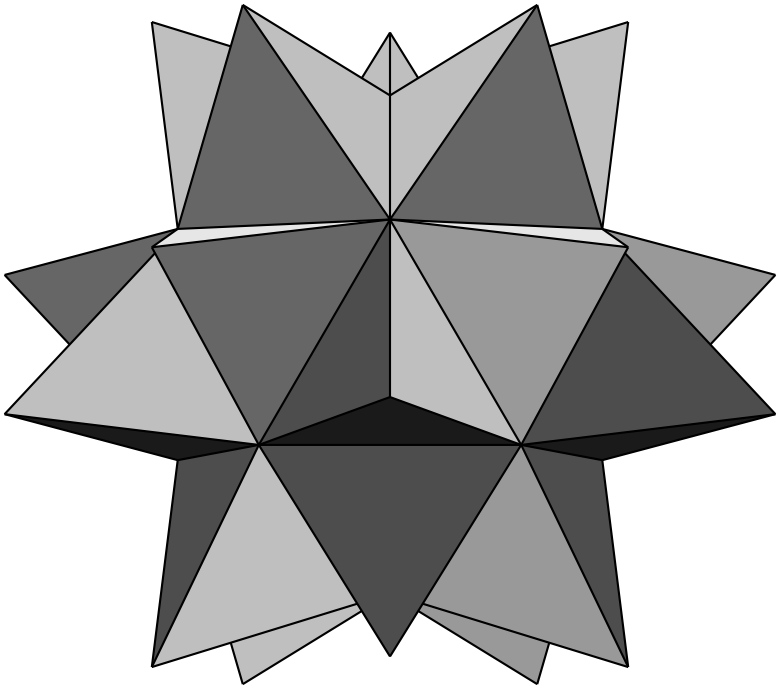


Figura 21: Icosaedro piramidato o "Stella di Urbino".

di Geogebra, ma secondo la nostra esperienza la sua curva di apprendimento è decisamente molto più piatta che non quella di TikZ. Inoltre l'uso di Geogebra presenta gli altri numerosi vantaggi che abbiamo descritto nell'articolo.

Naturalmente bisogna sempre tenere conto che non c'è rosa senza spina e non si può sperare di produrre effetti, come quelli che abbiamo mostrato, senza fatica e sperimentazione.

Chi è interessato a ulteriori approfondimenti può consultare il nostro sito http://www.batmath.it dove sono pubblicati numerosi materiali contenenti immagini costruite con il metodo qui presentato.

Riferimenti bibliografici

Barbaro, D. (1769). *La Pratica della Perspettiva*. Camillo & Rutilio Borgominieri fratelli, al Segno di S.Giorgio, Venetia.

Battaia, L. (2007). «LaTeX nella Scuola Media Superiore: applicazioni didattiche con PSTricks». *ArsTEXnica*, (4), pp. 45–50. URL http://www.guitex.org/home/numero-4.

Beccari, C. (2011). «The unknown picture environment». *ArsTEXnica*, (11), pp. 57–64. URL http://www.guitex.org/home/it/numero-11.

De Marco, A. (2007). «Illustrazioni tridimensionali con Sketch/LaTeX/PSTricks/TikZ nella didattica della Dinamica del Volo». *ArsTEXnica*, (4), pp. 51–68. URL http://www.guitex.org/home/numero-4.

— (2008). «Gestione avanzata delle figure in LaTeX». *ArsTEXnica*, (6), pp. 10–27. URL http://www.guitex.org/home/numero-6.

— (2009). «Produrre grafica vettoriale di alta qualità programmando asymptote». *ArsTEXnica*, (8), pp. 25–39. URL http://www.guitex.org/home/numero-8.

De Marco, A. e Giacomelli, R. (2011). «Creare grafici con pgfplots». *ArsTEXnica*, (12), pp. 12–38. URL http://www.guitex.org/home/it/numero-12.

Heath, T. L. (1896). *Apollonius of Perga - Treatise on Conic Sections edited in modern Notation*. Cambridge: at the University Press.

Voss, H. (2014). *3D plots: pst-3dplot*. URL https://www.ctan.org/pkg/pst-3dplot.

Wolcott, K. (2012). «Three-dimensional graphics with PGFTikZ». *TUGboat*, **33** (1), pp. 102–113.

▷ Luciano Battaia
Università Ca' Foscari
Dipartimento di Economia
luciano dot battaia at unive dot it

Forindex: computer-aided indexing

Guido Milanese

Abstract

A good index is a very important tool in writing: it improves the way readers approach books, manuals, and proceedings. Although a certain amount of data must be entered manually, some rather trivial tasks can be performed automatically, e.g. an index of geographical names. For such a purpose, `Forindex` is a standalone utility offering two basic functions: `doindex` prepares a file to be processed by `makeindex`, and `cleanindex` removes all the `\index{...}` entries in a LaTeX file, obtaining a clean file with no index tags: this can be useful if the file is to be indexed from scratch, maybe with different criteria. The program is written in Snobol4, using Zenity as GUI, which makes the code almost 100% portable. For Windows a standalone executable file is also provided.

Sommario

Un indice ben fatto costituisce uno strumento importante nel processo di scrittura: migliora la fruibilità di libri, manuali e atti di convegni. Anche se alcuni dati debbono necessariamente essere introdotti manualmente, compiti piuttosto banali possono compiersi automaticamente, per esempio un indice di nomi geografici. A questo scopo, `Forindex` è un programma standalone che offre due funzioni fondamentali: `doindex` prepara un file per essere processato da `makeindex`, mentre `cleanindex` rimuove tutte le marcature di `index{...}` in un file di LaTeX, ottenendo un file pulito senza marcatura di indicizzazione: questo può essere utile nel caso si voglia indicizzare di nuovo o con criteri diversi il documento. Il programma è scritto in Snobol4, usando Zenity come interfaccia grafica, il che lo rende portabile quasi al 100%. Per Windows si fornisce anche un file eseguibile autonomo.

1 Problem statement

Within the given scope, the aim of this program is very simple, and it is not meant to substitute human activity in this field. Complex concepts cannot be indexed by a simple program like this one, which at the moment is able to deal with single words only. However, lexical lists such as those found in humanistic manuscripts[1] can be effectively prepared using the present program. Database indexing is a different issue, of course, and retrieval and indexing is a crucial occurrence for any researcher specializing Internet philosophy and technology. Indexing of texts written in natural languages (as opposite to database, for example) is a form of text representation that can be very important for the future of a given text, e.g. its influence in research. If a text features a well structured index, potential readers will be able to locate bits of information they are interested in without being obliged to go through the whole book. To quote the words of a specialist, Marie-Francine Moens, «current text representations are often restricted to only certain terms that frequently occur in the text, or to all words from the beginning of the text, or to sentences that contain frequent terms. We assume that a representation that reflects the content in a semantically rich way will help solving the information retrieval problem in future systems» (Moens, 2002, 228). More recently, De Keyser's book on indexing features a very clear chapter defending automatic indexing (De Keyser, 2012).

Moens admits that making a suitable index often needs that some terms are manually indexed, especially when a term must be "distilled" from a concept, requiring an abstraction capability (Moens, 2002, 225), and I do agree: the procedure hereby proposed requires manual activity before and probably also after running the program. Many commercial programs on the market offer automated indexing or machine-aided indexing; the present program, obviously limited to LaTeX files, carries out a task similar to what is offered by a popular indexing program.[2]

In conclusion, although automated indexing *cannot be anathemised* as twenty years ago,[3] I still believe that computer-aided indexing, as opposite to automated indexing, is likely to be a reasonable perspective. Not only the program at the present stage of development, but also all the items in the TODO list (see p. 66) are not automated but computer-aided functions.

1. See for example Milanese (2005).

2. I refer to `Sonar Bookends`, produced by Virginia Systems (`http://www.virginiasystems.com/`) and called «popular» by Browne e Jermey (2007, 186): «its claims are not extravagant: it can produce a concordance (a page number for every word); a list of proper nouns with page numbers; or an alphabetised list of words and phrases that you supply ('go' words) with the page numbers shown». Browne's book is rather old: however, I tested the demo of the Professional edition in August, 2016, and the program, priced $395 (the non-Professional series is cheaper) does what it promises to do. For an enthusiastic description of computer-aided methodologies see Stauber (2004, 300-303): the author refers to `Macrex`, a Windows program still in the market (see `www.macrex.com`).

3. See for example the accident of a program called `Indexicon`, as narrated by Mulvany (2005, 251-252).

Insofar as LaTeX is concerned, I have found two similar projects on CTAN. Martin Thorsen Ranange's program `intex` (see `https://www.ctan.org/tex-archive/support/intex`) is a very clever Python script, from some points of view similar to the present program. `intex` requires the user to add tags (`\co{...}`) in the LaTeX file, while `forindex` prepares the basic form of an index with no manual tagging. From my perspective, this is the most important difference.

Paul Isambert's package XEInd is similar to the program hereby proposed, but it is a XeLaTeX-only project (`https://www.ctan.org/pkg/xeindex`).

Two Italian contributions are worth mentioning. Claudio Beccari's *Uso del comando `\write18` per comporre l'indice analitico in modo sincrono* (Beccari, 2009) suggests a clever way to avoid «three or four asynchronous runs of `makeindex` and `latex` or `pdflatex`» using the `\write18` command; Cevolani's article «explains how to generate the index of names using `biblatex`» (Cevolani, 2010). Cevolani's procedure could be usefully integrated in the present program, while Beccari's instruction is used in the output produced by this program.

2 Program outline

2.1 General interface

The program interacts with the user through a graphical user interface. After the first window, displaying the name of the program and the current version (see fig. 1), the user is presented with a dialogue window, asking to choose between the two provided functions (see fig. 2). The second dialogue window asks for the name of the LaTeX file, as described below.

2.2 Function `doindex`

`doindex` reads a LaTeX file, using a list file, and enters index entries in the file according to it. Previously entered index entries are left unchanged, allowing for further indexing.

The input LaTeX file must have extension `tex`, in whatever case combination. If the user makes a wrong choice, the program outputs an error message and the same dialog window is displayed again. The user can interrupt the program at any time hitting the cancel button.

The list file is meant to contain all the words to be indexed. It must have the same name of the LaTeX file and extension `wls`; sub-entries are separated by tabs: up to 3 levels are possible, which seems reasonable for most documents[4]. Such a file can be easily prepared with any spreadsheet program, such as `Libreoffice calc`, `Excel`, and the like (or obviously with a text editor). See `test.wls` as example:

animals		
foreign words	French	ça
animals	cats	
animals	dogs	
foreign words	German	drücken
foreign words	Italian	évita
		MacIntyre
		Cicero
		food
languages	nouns	house
		Aristotle
		Maritain
sleeping@sleep		

No particular order in this file is required. Some users will prefer alphabetical order, others different arrangements: therefore, the program has no requirements concerning order/sort in this file. Entries as `sleeping@sleep` use the standard `makeindex` syntax and are left unchanged. The "logic" of this syntax is the same of `makeindex`, namely:

Class (level 1) – Class (level 2) – Item

In the previous table, some entries are placed in the 3rd column, the 1st and the 2nd being empty. We could imagine to fill the empty values as such, for example:

philosophers	Scottish	MacIntyre
philosophers	Roman	Cicero

However, writing "standalone" items in the first column is more natural, and therefore the program treats the following two fragments as equivalent:

		MacIntyre
		Cicero

MacIntyre		
Cicero		

Some people may prefer to organise their index thinking the other way round, for example:

		animals
ça	French	foreign words
	cats	animals
	dogs	animals
drücken	German	foreign words
évita	Italian	foreign words
MacIntyre		
Cicero		
food		
house	nouns	languages
Aristotle		
Maritain		
sleeping@sleep		

4. See De Keyser (2012, 13-14): he dislikes what he calls «indirect indexing», and proposes «to rely on good software that can 'explode' from a narrower term to a broader or vice versa when needed».

This taxonomy is opposite to the internal logic of `makeindex`[5]. For `doindex`, both systems are acceptable. It's a matter of personal choice: you can think either (1) or (2):

1. Plato belongs to the class "philosophers"
2. the class "philosophers" owns Plato

Plato ∈ Philosophers or Philosophers ∋ Plato is a matter of thinking FROM MEMBER TO SET or FROM SET TO MEMBER. The same applies to a possible intermediate class (such as PLATO – ANCIENT PHILOSOPHERS – PHILOSOPHERS). Just let the program know which system are you following, and be consistent, of course. See fig. 3.

The original LaTeX file is left unchanged. A new file is written, identified by `-ind`. For example, from `file.tex` you will get `file-ind.tex`. Of course, you'll have to run `makeindex` as usual, but Beccari's procedure is used in the file, reducing the need of manual runs.

Another useful function is called replacement. For example, in the text of a book it is normal to use surnames instead of full names: I may write

> Many modern philosophers – e.g. Annas and MacIntyre – notice that ancient ethics is a fascinating field of research.

but in the index of names I will not refer simply to *MacIntyre* but to *MacIntyre, Alasdair*[6]. A replacement file lists the simple form – the form that `doindex` would use without further instruction – and the full form. The two values will be separated by tabs:

```
MacIntyre	MacIntyre, Alasdair
Cicero	Cicero, Marcus Tullius
Annas	Annas, Julia
```

Naturally you could use this also for other substitutions, such as geographical names:

```
Monaco	Monaco, Principauté de
Westminster	Westminster, City of
```

The replacement list is optional (see fig. 4). There are no restrictions on filenames and extensions: this is because a scholar may wish to have several lists of replacements and use them for all of his works, not only for a particular publication. A dialogue window asks the user to select a replacement list, if any—if the selected file does not contain a replacement list, the program fails.

5. DE KEYSER (2012, 13) calls this indexing organisation «inverted index terms» – confusing the class / member relationship with full names (to refer to his examples, «Art, Babylonian» and «Joyce, James»). The present program distinguishes among the two different structures.
6. For the general problem see MOENS (2002, 88).

2.3 The function `cleanindex`

The function `cleanindex` removes \index{...} sequences from a LaTeX file. The program can be used e.g. if a user is not happy with the indexing of a file and wants to start it all over again.

The input LaTeX file must have extension `tex`, in whatever case combination. If the user makes a wrong choice, the program outputs an error message and the same dialog window is displayed again. The user can interrupt the program at any time hitting the cancel button.

The original file is left unchanged. A new file is written, identified by `-noind`. For example, from `file.tex` you will get `file-noind.tex`. In this file, lines concerning `makeindex` are left but commented, in order to avoid an empty *Contents* section in the output. You can uncomment the lines as soon as you want to reindex the file.

3 Installation

3.1 GNU/Linux and other *nix systems

Unzip the archive in a directory of your choice. A new directory, called `Forindex`, will be created. Run `forindex` with your favourite program launcher or from terminal. If you prefer to compile `snobol4` for your particular system, download the sources from http://www.snobol4.org/csnobol4/curr/; compilation is straightforward (generally just `./configure`, `make`, `make install`), but do read the `README` file. The most recent version of the interpreter is 2.0 (2015).

3.2 Windows

The package offers an `exe` file compiled with Spitbol (see (http://www.snobol4.com). Make a directory (a subfolder of `Programs`, for example) and copy all the files there.

3.3 Macintosh

Still untested. The main problem is `zenity`: a new project looks promising (https://www.macports.org/ports.php?by=library&substr=zenity).

4 Test files

Please test the program on template files provided herein, named `atest.tex` and `atest.wls`, `btest.tex` and `btest.wls`. `atest.wls` uses the *Item – Class 1 – Class 2* logic, while `btest` follows the *Class 2 – Class 1 – Item* logic. The output files will be called `atest-ind.tex` `btest-ind.tex` using `doindex`, `atest-noind.tex` or `btest-noind.tex` using `cleanindex`.

5 Bugs and TODO

The program supports Unicode chars limited to Latin extended alphabet and Greek (complete). *Unicode files only*: since other encodings are rapidly

disappearing, adapting the program to latin1 or similar encodings is not a priority.

List of features that I would like to add[7]:

1. Index also included files.
2. Introduction of optional styles, such as italics for the most important locations of a word.
3. Support for several indexes (e.g. names, places, and general).
4. Option to generate a rough index of all the words (to produce a preliminary version of the `wls` file).
5. Support to index words listed with regular expressions. E.g. `read*` should index *read*, *reads*, *reading*, *readings*, all under the same heading *read*.
6. Optional separator for the `wls` file, e.g. comma.
7. Add Unicode support. % partially done, 1.0

6 Acknowledgements

6.1 Version 0.1, 2005 (slightly adapted)

The program `ixgen` gave me the idea of `forindex`. Many thanks to Oscar Lopez for this very good program[8].

Some questions sent by Carlo Pellegrino (Università di Modena e Reggio Emilia, Italy) gave me the idea of transforming a very rudimentary script into a general purpose utility. Maurizio Loreti (Università di Padova, Italy) sent me very useful remarks on the problems of automatical generations of indexes, which I made use of in the introduction to this text.

My warmest thanks to Phil Budne (`phil@ultimate.com`) for making his excellent CSNOBOL interpreter available. Many thanks to the community of Snobol users, particularly to the members of the list `snobol4@mercury.dsu.edu`, and, among them, to Gordon Peterson (`https://www.linkedin.com/in/gordonpeterson`) and to Rafal M. Sulejman (`rafal@engelsinfo.de`) whose `vim` syntax files are a daily blessing.

6.2 Addendum for version 1.0, 2016

I would like to add a word of gratitude to the members of the Italian TEX usergroup, GUIT (`http://www.guitex.org`) and particularly to Ivan Valbusa (Verona) for his generous help.

7. This is the list of desired features of version 0.1, published in 2005. Unicode, at least, is (partially) tagged as «done». Much more to do.

8. `ixgen` appears to be now (2016) a commercial program for `Framemaker`. I do not know what has happened to the LaTeX program with the same name. See `http://www.fsatools.com/index.htm`.

7 Screenshots

Figure 1: Initial message

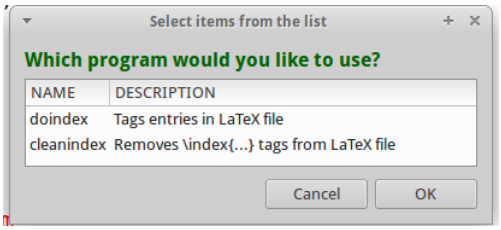


Figure 2: Choice of function

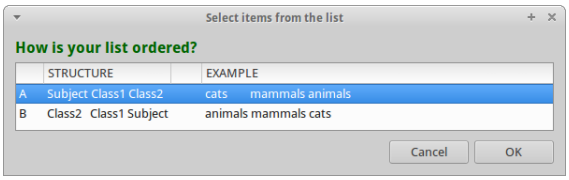


Figure 3: Order of `wls` file

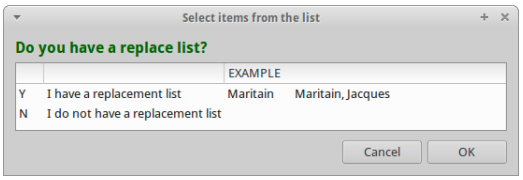


Figure 4: Optional replacement list

References

Beccari, C. (2009). «Uso del comando `\write18` per comporre l'indice analitico in modo sincrono». *ArsTEXnica*, (8), pp. 76–78. URL `http://www.guitex.org/home/numero-8`.

Browne, G. e Jermey, J. (2007). *The indexing companion*. Cambridge University Press, Cambridge. URL `http://www.loc.gov/catdir/toc/ecip0620/2006028722.html`.

Cevolani, G. (2010). «Indice dei nomi automatico con biblatex». *ArsTEXnica*, (9), pp. 31–38. URL `http://www.guitex.org/home/numero-9`.

De Keyser, P. (2012). *Indexing: from thesauri to the semantic web*. Chandos information professional series. Chandos, Oxford – Cambridge – New Delhi.

Milanese, G. (2005). *Censimento dei manoscritti noniani*. Pubblicazioni del Darficlet, N.S. 225. DARFICLET "F. Della Corte", Genova.

Moens, M.-F. (2002). *Automatic Indexing and Abstracting of Document Texts*. The Information Retrieval Series. Kluwer Academic Publishers, Boston, MA. URL http://dx.doi.org/10.1007/b116177.

Mulvany, N. C. (2005). *Indexing books*. Chicago guides to writing, editing, and publishing. University of Chicago Press, Chicago – London, 2ª edizione. URL http://www.loc.gov/catdir/toc/ecip057/2005004214.html.

Stauber, D. M. (2004). *Facing the text: content and structure in book indexing*. Cedar Row, Eugene, OR.

▷ Guido Milanese
Università Cattolica del Sacro Cuore, Milano–Brescia
guido dot milanese at unicatt dot it

Liste, cicli, LaTeX3

Enrico Gregorio

Sommario

Si esamina come eseguire cicli in LaTeX con la macro `\foreach` e come sostituirla con codice basato su LaTeX3.

Abstract

We examine how to execute loops in LaTeX with the macro `\foreach` and a new implementation based on LaTeX3 code.

1 Introduzione

In molte occasioni è necessario o conveniente ripetere un comando più volte su una lista di oggetti o su interi in un certo intervallo.

Con plain TeX è possibile eseguire cicli con `\loop`, che ha la forma

```
\loop
  ⟨codice⟩
⟨if⟩⟨test⟩
  ⟨codice⟩
\repeat
```

Ovviamente è anche possibile definire nuove macro per i propri cicli. Più esteso è il supporto del nucleo di LaTeX, che mette a disposizione

```
\@for
\@tfor
\@whilenum
\@whiledim
\@whilesw
```

Si vedano Goossens *et al.* (1994) e Braams *et al.* (2016).

Il pacchetto etoolbox fornisce parecchie macro dedicate allo scopo. È anche possibile trovare codice girando per la rete, in particolare per estendere le funzionalità di `\loop`.

Nell'articolo si presume l'uso di LaTeX, ma per alcune macro sarà adoperato codice plain TeX.

2 `\foreach`

Un comando molto comodo è fornito da PGF, con il pacchetto pgffor, caricato automaticamente da Ti*k*Z(Tantau, 2015). La sintassi di base è

```
\foreach \i in {⟨lista⟩}{⟨codice⟩}
```

oppure

```
\foreach \i in ⟨macro⟩ {⟨codice⟩}
```

ed è possibile dare alcune opzioni per ottenere effetti particolari.

Mi occuperò per ora del caso base, in cui la lista è di valori separati da virgole, in cui è anche possibile adoperare `...` per indicare ripetizione secondo uno schema; per esempio, invece di

```
\foreach \i in {1,3,5,7,9,11,13,15}{⟨codice⟩}
```

è più semplice scrivere

```
\foreach \i in {1,3,...,15}{⟨codice⟩}
```

specificando il valore iniziale, quello finale e il passo. Si può anche scrivere `{1,...,10}` e il passo sarà implicitamente uno. I valori possono anche essere numeri in virgola mobile, si dirà più avanti qualcosa al riguardo. La lista può anche essere arbitraria, come

```
\foreach \i in {cane,gatto,criceto}{%
  ⟨codice⟩%
}
```

In questo caso, ovviamente, non è possibile la notazione ellittica.

La seconda possibilità si ottiene con codice come

```
\newcommand{\intervallo}{1,3,...,17}
\newcommand{\lista}{cane,gatto,criceto}
\foreach \i in \intervallo {⟨codice⟩}
\foreach \i in \lista {⟨codice⟩}
```

In altre parole, se la parola chiave `in` è seguita da una graffa aperta, l'argomento denota la lista di valori, altrimenti viene espansa la macro che deve essere passata e che deve espandersi a una lista lecita.

Il ⟨*codice*⟩ normalmente conterrà il "segnaposto" `\i`. Va osservato che la macro, qui denotata con `\i` nell'esempio di sintassi, è del tutto arbitraria e può essere adoperata nel codice da ripetere, dove avrà come espansione il valore corrente preso successivamente dalla lista. Ovviamente bisogna assicurarsi che la macro scelta come segnaposto non venga adoperata nel codice.

Per consentire di adoperare qualsiasi macro come segnaposto, ciascun ciclo viene eseguito in un gruppo. Una tikzpicture è costruita un passo alla volta e ciascuna dichiarazione come `\draw` viene tradotta in linguaggio PGF e aggiunta *globalmente* a una macro interna che poi viene eseguita da `\end{tikzpicture}`.

Il principale difetto di `\foreach` è proprio di eseguire i cicli in un gruppo, anche se ciò consente grande libertà nel dare un nome alla variabile 'locale'. Un altro difetto, meno ovvio, è che gli spazi sono rilevanti. Per esempio

```
\foreach \i in { cane, gatto, criceto }
  {X\i X\par}
```

produrrà

XcaneX
XgattoX
Xcriceto X

da cui si vede che gli spazi iniziali sono ignorati, mentre quelli successivi alla stringa non lo sono. Il problema è mascherato nel caso in cui la lista è numerica, perché la sintassi di TEX ignora spazi dopo costanti nelle assegnazioni.

Un altro difetto, meno evidente a un primo esame, è che il segnaposto non viene espanso nel codice da ripetere. Consideriamo il seguente esempio[1]

```
\def\list{}
\foreach \x in {1,2,3,4} {%
  \gappto\list{\color{color\x}}%
}
```

che ha lo scopo di costruire una macro che, alla fine, contenga

```
\color{color1}\color{color2}%
\color{color3}\color{color4}
```

Ovviamente si tratta di un esempio ridotto al minimo. Se però si esegue questo codice (che richiede etoolbox), la macro costruita avrà come espansione

```
\color{color\x}\color{color\x}
\color{color\x}\color{color\x}
```

e \x non sarà definita. C'è anche da dire che adoperare \list come nome della macro da costruire è un grave errore, ma questo è secondario. Il problema è di espandere \x, ma se provassimo con \expandafter scopriremmo che ne dovremmo scrivere una quantità inverosimile: in questo caso si devono saltare dieci token e quindi occorre scrivere $1023 = 2^{10} - 1$ \expandafter prima di \gappto, $511 = 2^9 - 1$ prima di \list e così via fino a uno prima di r. L'alternativa con \xappto invece di \gappto è ancora peggiore, perché causa una notevole quantità di errori, ma non è tanto meglio

```
\newcommand\mylist{}
\makeatletter
\foreach \x in {1,2,3,4} {
  \protected@xdef\mylist{\color{color\x}}
}
\makeatother
```

anche se è efficace. Non sarebbe troppo difficile correggere il problema, con una seconda macro

```
\newcommand\mylist{}
\makeatletter
\foreach \x in {1,2,3,4} {
  \def\y##1{%
    \g@addto@macro\mylist{%
      \color{color##1}%
    }%
  }
  \expandafter\y\expandafter{\x}
}
\makeatother
```

La macro ausiliaria potrebbe essere definita fuori dal ciclo; così è più semplice e forse più chiaro.

Questo problema affligge anche \@for e \@tfor; a parte la sintassi meno flessibile, si avrebbe il problema descritto prima anche con

```
\newcommand{\mylist}{}
\makeatletter
\@for\next:={1,2,3,4}\do{%
  \g@addto@macro\mylist{%
    \color{color\next}%
  }%
}
\makeatother
```

e il rimedio sarebbe molto simile a quanto visto per \foreach.

Al di fuori dell'impiego in tikzpicture, \foreach è spesso adoperato per costruire macro per "accumulo", come nel caso precedente. Un caso interessante è definire comandi personali per lettere speciali in formule matematiche; l'esempio è

```
\foreach \x in {A,B,...,Z} {%
  \let\xp\expandafter
  \xp\gdef\csname c\x\xp\endcsname\xp{%
      \xp\mathcal\xp{\x}%
  }%
}
```

che può evitare \expandafter con

```
\foreach \x in {A,B,...,Z} {%
  \expandafter\xdef\csname c\x\endcsname
    {\noexpand\mathcal{\x}}%
}
```

Qui si sfrutta l'abilità di \foreach di completare anche liste alfabetiche e non solo numeriche con la notazione ellittica. C'è sempre il problema dell'espansione del segnaposto, ma solo nel testo di sostituzione della macro, perché \csname esegue l'espansione completa da sé.

Non è difficile scrivere lo stesso con \@whilenum:

```
\makeatletter
\count@=\z@
\@whilenum\count@<26 \do{%
  \advance\count@ by \@ne
  \expandafter
  \edef\csname c\@Alph{\count@}\endcsname
    {\noexpand\mathcal{\@Alph{\count@}}}%
}
\makeatother
```

anche se questo potrebbe non funzionare se sono caricati certi moduli di babel che modificano la definizione di \@Alph e il metodo sarebbe più sicuro con

1. http://tex.stackexchange.com/q/211234

```
\makeatletter
\count@=`A \advance\count@\m@ne
\@whilenum\count@<`Z \do{%
  \advance\count@ by \@ne
  \begingroup\lccode`x=\count@
  \lowercase{\endgroup
    \@namedef{cx}{\mathcal{x}}%
  }%
}
\makeatother
```

Non c'è dubbio che `\foreach` sia più semplice, anche se non troppo.

3 Liste

Senza entrare troppo nei dettagli tecnici, una lista ordinata può essere realizzata in due modi: con un separatore fra un elemento e il successivo oppure dando ciascun elemento come argomento di una macro:

```
\def\listaA{cane,gatto,criceto}
\def\listaB{%
  \foo{cane}%
  \foo{gatto}%
  \foo{criceto}%
}
```

Sia `\@for` sia `\foreach` accettano solo liste del primo tipo; è facile capire però che le liste del secondo tipo sono molto più flessibili, perché il significato della macro adoperata per delimitare ciascun elemento può cambiare quando ci pare.

Per esempio, possiamo facilmente trasformare la seconda lista nella prima, ma con elementi separati da `;` invece che da una virgola, con

```
\def\listaB{%
  \foo{cane}%
  \foo{gatto}%
  \foo{criceto}%
}
\begingroup
\def\foo#1{;\unexpanded{#1}}
\def\gobble#1#2\gobble{\unexpanded{#2}}
\edef\next{\listaB}
\edef\next{\endgroup
  \def\noexpand\listaC{%
    \expandafter\gobble\next\gobble
  }%
}\next
```

Oppure possiamo contare gli elementi della lista con

```
\newcount\listcount
\begingroup
\listcount=0
\def\foo#1{%
  \global\advance\listcount by 1
}
\listaB
\endgroup
```

Se non desideriamo assegnazioni globali, si può adoperare un registro temporaneo

```
\newcount\listcount
\begingroup
\count0=0
\def\foo#1{\advance\count0 by 1 }
\listaB
\expandafter\endgroup
\expandafter\listcount\the\count0\relax
```

Qui il valore di `\count0` è ottenuto prima che il gruppo venga terminato; solo in seguito `\endgroup` riassegna al registro il valore precedente.

Non è difficile fare lo stesso con `\foreach` operando su `\listaA`, basta ricordare che occorrono assegnazioni globali:

```
\newcount\listcount
\foreach \x in \listaA {%
  \global\advance\listcount by 1
}
```

oppure usare un contatore LaTeX:

```
\newcounter{listcount}
\foreach \x in \listaA {%
  \stepcounter{listcount}%
}
```

Qui si sfrutta il fatto che `\stepcounter` esegue un'assegnazione globale.

Una lista può essere costruita passo passo; per le liste del secondo tipo la faccenda è semplice:

```
\newcommand{\apptolistII}[2]{%
  \toks0=\expandafter{#1}%
  \edef#1{%
    \the\toks0
    \unexpanded{\foo{#2}}%
  }%
}
\newcommand{\listaD}{}% init
\apptolistII{\listaD}{cane}
\apptolistII{\listaD}{gatto}
\apptolistII{\listaD}{criceto}
```

Per quelle del primo tipo è un po' più complicato perché occorre distinguere se la lista è vuota:

```
% syntactic sugar
\providecommand{\expandonce}[1]{%
  \unexpanded\expandafter{#1}%
}
\newcommand\emptylist{}

\newcommand{\apptolistI}[2]{%
  \ifx#1\emptylist
    \def#1{#2}%
  \else
    \edef#1{%
      \expandonce{#1},%
      \unexpanded{#2}%
    }
  \fi
}
\newcommand{\listaE}{}% init
\apptolistI{\listaE}{cane}
\apptolistI{\listaE}{gatto}
\apptolistI{\listaE}{criceto}
```

Più complicato è poi adoperare una lista. Supponiamo che si voglia stampare la lista della frutta che si è mangiata. Per ogni pasto, si può eseguire `\addtolistI{\fruits}{⟨frutto⟩}` e terminare con

```
Ho mangiato
\foreach \x in \fruits {%
  \x, % spazio
}.
```

con il problema di avere una virgola di troppo. Possibile soluzione: contare prima il numero di elementi e produrre una virgola se non siamo all'ultimo passo:

```
\documentclass{article}
\usepackage{pgffor}

\newcounter{fruits}
\newcommand{\fruits}{%
  pera,
  mela,
  banana,
  pesca%
}

\begin{document}

Ho mangiato
\setcounter{fruits}{0}%
\foreach \x in \fruits{%
  \stepcounter{fruits}%
}%
\foreach \x [count=\y]
  in \fruits{%
  \x
  \ifnum\y<\value{fruits}%
    , % spazio
  \fi
}.

\end{document}
```

L'opzione `[count=\y]` fa in modo che all'interno di ciascun ciclo, `\y` si riferisca al numero dell'iterazione.

Per esercizio, si estenda il codice in modo che ci sia una 'e' tra banana e pesca, invece della virgola come nell'esempio che segue.

> Ho mangiato pera, mela, banana e pesca.

Se però si volesse seguire la convenzione detta *Oxford comma*, secondo la quale si dovrebbe scrivere

> mela e pera
> mela, pera, banana, e pesca

il codice dovrebbe essere molto più complicato.

È venuto il momento di procurarsi una cassetta degli attrezzi più fornita, cioè expl3.

Prima un po' di *syntactic sugar* per maneggiare liste. Si supporrà che i pacchetti expl3 e xparse siano caricati (basta il secondo). Per maggiori informazioni si consultino The LaTeX3 Project (2016b,a,c).

```
% nel preambolo
\ExplSyntaxOn
% definire una lista
\NewDocumentCommand{\newlist}{m}
 {
  \manual_list_new:n { #1 }
 }
% azzerare una lista
\NewDocumentCommand{\clearlist}{m}
 {
  \manual_list_clear:n { #1 }
 }
% aggiungere uno o più elementi
\NewDocumentCommand{\addtolist}{mm}
 {
  \clist_map_inline:nn { #2 }
   {
    \manual_list_append:nn { #1 } { ##1 }
   }
 }
\NewDocumentCommand{\uselist}{mmmO{#3}}
 {
  \seq_use:cnnn { g_manual_list_#1_seq }
   { #3 } { #2 } { #4 }
 }
% comandi interni
\cs_new_protected:Nn \manual_list_new:n
 {
  \seq_new:c { g_manual_list_#1_seq }
 }
\cs_new_protected:Nn \manual_list_clear:n
 {
  \seq_clear:c { g_manual_list_#1_seq }
 }
\cs_new_protected:Nn \manual_list_append:nn
 {
  \seq_gput_right:cn { g_manual_list_#1_seq }
   { #2 }
 }
\ExplSyntaxOff

% nel documento
\newlist{fruits}
\addtolist{fruits}{mela,pera}

Ho mangiato
\uselist{fruits}{, }{ e }.

\addtolist{fruits}{banana}
\addtolist{fruits}{pesca}

Ho mangiato
\uselist{fruits}{, }{ e }.

Ho mangiato
\uselist{fruits}{, }{ e }[, e ].
```

La lista viene chiamata "per nome"; il secondo argomento di `\uselist` è il separatore normale, il terzo il separatore fra gli ultimi due elementi; se viene specificato l'argomento facoltativo, il terzo argomento viene adoperato solo se la lista è di due soli elementi, altrimenti fra gli ultimi due apparirà l'argomento facoltativo. Si verifichi che il codice produce

Ho mangiato mela e pera
Ho mangiato mela, pera, banana e pesca.
Ho mangiato mela, pera, banana, e pesca.

Per divagare un po', si noti come, nella specificazione degli argomenti di `\uselist`, il valore di default per l'argomento facoltativo è identico al valore del terzo argomento. Una specificazione equivalente sarebbe

```
\NewDocumentCommand{\uselist}{mmmo}
 {
  \IfNoValueTF{#4}
   {
    \seq_use:cnnn { g_manual_list_#1_seq }
     { #3 } { #2 } { #3 }
   }
   {
    \seq_use:cnnn { g_manual_list_#1_seq }
     { #3 } { #2 } { #4 }
   }
 }
```

ed è chiaro il risparmio in termini di codice. Il metodo più lungo deve essere usato se si vuole che la macro sia espandibile:

```
\DeclareExpandableDocumentCommand{\uselist}
 {mmom}
 {
  \IfNoValueTF{#3}
   {
    \seq_use:cnnn { g_manual_list_#1_seq }
     { #4 } { #2 } { #4 }
   }
   {
    \seq_use:cnnn { g_manual_list_#1_seq }
     { #4 } { #2 } { #3 }
   }
 }
```

ma la sintassi sarà diversa, perché l'argomento facoltativo non può andare per ultimo in questa situazione.

La macro `\addtolist` accetta anche più di un elemento alla volta, rimuovendo gli spazi prima e dopo; quindi sono del tutto equivalenti

```
\addtolist{fruits}{mela,pera}
\addtolist{fruits}{mela, pera}
\addtolist{fruits}{mela , pera}
\addtolist{fruits}{ mela, pera }
```

e le altre possibili variazioni sul tema.

La struttura dati adoperata è quella delle "liste del secondo tipo", anche se non è necessario saperlo.

Viene subito in mente l'idea di costruire liste da dare in pasto a `\foreach` con metodi simili. In tal caso, basta adoperare la struttura dati *clist* e sostituire `\seq` con `\clist` e `_seq` con `_clist` nel codice mostrato.

4 Un nuovo `\foreach`, anzi due

La sintassi di `\foreach` per indicare cicli basati su interi è comoda, ma non più di una sintassi del tipo

```
\nforeach{
  start=1,
  step=2,
  end=13
}{<codice>}
```

invece di

```
\foreach \x in {1,3,...,13}{<codice>}
```

dove l'opzione `step` è facoltativa, nel caso il passo scelto sia 1.

Il lettore attento noterà una differenza importante: nel codice per `\nforeach` non è menzionata la 'variabile' che riceve il valore corrispondente in ciascun ciclo. La spiegazione è semplice: invece di

```
\begin{tikzpicture}
 \foreach \x in {1,...,10}{
   \draw (\x,0) circle (0.4cm);
 }
\end{tikzpicture}
```

(il passo è 1), possiamo scrivere

```
\begin{tikzpicture}
\nforeach{start=1,end=10}{
  \draw (#1,0) circle (0.4cm);
}
\end{tikzpicture}
```

La variabile è semplicemente indicata con `#1`, che diventerà `##1` in un ciclo annidato, `####1` al terzo livello e così via raddoppiando.

Potremmo facilmente definire una versione con argomenti invece della sintassi chiave-valore con

```
\NewDocumentCommand{\simpleforeach}{mmmm}
 {
  \int_step_inline:nnnn { #1 } { #2 } { #3 }
   { #4 }
 }
```

dove gli argomenti sono, nell'ordine, il punto di partenza, il passo, il punto finale e il codice da eseguire. Tuttavia la sintassi proposta prima permette di estendere la macro per avere a disposizione cicli basati su numerazione alfabetica o incrementi su numeri a virgola mobile. Un esempio:

```
\nforeach{start=0,end=10}{%
  $2^{#1}=\fpeval{2^#1}$, %
}
```

produce

$2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, $2^7 = 128$, $2^8 = 256$, $2^9 = 512$, $2^{10} = 1024$,

dove `\fpeval` è un'ovvia versione 'utente' di `\fp_eval:n`.

Il caso di numeri in virgola mobile:

Tabella 1: Logaritmi naturali

$\log 1 = 0$	$\log 2 = 0.6931$	$\log 3 = 1.0986$	$\log 4 = 1.3863$	$\log 5 = 1.6094$
$\log 1.1 = 0.0953$	$\log 2.1 = 0.7419$	$\log 3.1 = 1.1314$	$\log 4.1 = 1.411$	$\log 5.1 = 1.6292$
$\log 1.2 = 0.1823$	$\log 2.2 = 0.7885$	$\log 3.2 = 1.1632$	$\log 4.2 = 1.4351$	$\log 5.2 = 1.6487$
$\log 1.3 = 0.2624$	$\log 2.3 = 0.8329$	$\log 3.3 = 1.1939$	$\log 4.3 = 1.4586$	$\log 5.3 = 1.6677$
$\log 1.4 = 0.3365$	$\log 2.4 = 0.8755$	$\log 3.4 = 1.2238$	$\log 4.4 = 1.4816$	$\log 5.4 = 1.6864$
$\log 1.5 = 0.4055$	$\log 2.5 = 0.9163$	$\log 3.5 = 1.2528$	$\log 4.5 = 1.5041$	$\log 5.5 = 1.7047$
$\log 1.6 = 0.47$	$\log 2.6 = 0.9555$	$\log 3.6 = 1.2809$	$\log 4.6 = 1.5261$	$\log 5.6 = 1.7228$
$\log 1.7 = 0.5306$	$\log 2.7 = 0.9933$	$\log 3.7 = 1.3083$	$\log 4.7 = 1.5476$	$\log 5.7 = 1.7405$
$\log 1.8 = 0.5878$	$\log 2.8 = 1.0296$	$\log 3.8 = 1.335$	$\log 4.8 = 1.5686$	$\log 5.8 = 1.7579$
$\log 1.9 = 0.6419$	$\log 2.9 = 1.0647$	$\log 3.9 = 1.361$	$\log 4.9 = 1.5892$	$\log 5.9 = 1.775$

```
\nforeach{
  type=fp,
  start=1.1,
  step=0.1,
  end=1.9
}
{ciclo~#1, }
```

ha come risultato

ciclo 1.1, ciclo 1.2, ciclo 1.3, ciclo 1.4, ciclo 1.5, ciclo 1.6, ciclo 1.7, ciclo 1.8, ciclo 1.9,

Nella tabella 1 c'è il risultato di

```
\nforeach{
  start=1,
  end=5,
  step=1,
}{%
  \nforeach{
    type=fp,
    start=0,
    end=0.9,
    step=0.1
  }{%
    $\log\fpeval{#1+##1}=
     \fpeval{round(ln(#1+##1),4)}$%
    \par
   }%
}
```

cioè la tabella dei logaritmi dei numeri da 1 a 5.9 con un passo di 0.1 arrotondati alla quarta cifra decimale. Si sarebbe potuto scrivere un unico ciclo, ma in questo modo si dimostra come i cicli possano essere annidati. Non occorre l'opzione `type=integers` nel caso la variabile sia intera, ma può essere scritta per maggiore chiarezza.

Se vogliamo definire tutti insieme i comandi come `\calA` per `\mathcal{A}`, problema che abbiamo già risolto prima,

```
% syntactic sugar
\newcommand\namenewcommand[1]{%
  \expandafter\newcommand\csname #1\endcsname
}
\nforeach{type=Alph,start=A,end=Z}{%
  \namenewcommand{cal#1}{\mathcal{#1}}%
}
$\calA$, $\calC$, $\calZ$.
```

e il risultato è

$\mathcal{A}$, $\mathcal{C}$, $\mathcal{Z}$.

Il codice è nelle tabelle 2 e 3. Non è possibile commentarlo completamente, ma l'idea fondamentale è che con

`\nforeach{`⟨*opzioni*⟩`}{`⟨*codice*⟩`}`

il ⟨*codice*⟩ diventa il testo di sostituzione di una macro con un argomento il cui nome dipende dal livello di annidamento e che viene applicata a ciascun numero (o lettera) che viene passato successivamente dalla funzione che ripete i cicli. Il livello di annidamento è conservato in una variabile globale che viene incrementata a ogni chiamata e riportata indietro quando il ciclo termina. A dire il vero la faccenda è un po' più complessa, ma non è il caso di entrare nei dettagli. Per i cicli base, con interi, viene adoperata la funzione `\int_step_inline:nnnn` già fornita da expl3, per i cicli con numeri a virgola mobile e alfabetici l'intera serie viene memorizzata in una variabile di tipo *sequence* e si impiega la funzione `\seq_map_inline:Nn`.

Nel manuale di Ti*k*Z, pagina 902, c'è l'esempio

```
\foreach \x in {0,0.1,...,0.5} {\x, }
```

che produce

0, 0.1, 0.20001, 0.30002, 0.40002,

Con `\nforeach` avremmo

```
\nforeach{
  type=fp,
  start=0,
  step=0.1,
  end=0.5}{#1, }
```

che produce

0, 0.1, 0.2, 0.3, 0.4, 0.5,

e si vede come la precisione sia notevolmente maggiore.

Il codice è in una versione preliminare e richiederà parecchi perfezionamenti; per esempio non è ancora possibile sapere in quale passo del ciclo si sia, ma verrà aggiunta una chiave del tipo `cyclecount` e, con `cyclecount=var` si potrà adoperare `\cyclecount{var}` per ottenere (dopo l'espansione) il numero corrispondente al numero del ciclo attuale.

Per i cicli basati su liste è prevista una macro diversa, chiamata `\lforeach`; sarebbe forse possibile adoperare una sola macro, ma la mia opinione è che sia meglio separarle. Le opzioni sono molto diverse.

La macro ha una variante asterisco per indicare che il primo argomento è a sua volta una macro che contiene la lista da elaborare; a parte questo, la sintassi è

`\lforeach[`⟨*opzione*⟩`]{`⟨*lista*⟩`}{`⟨*azione*⟩`}`

dove, come per `\nforeach`, ci si può riferire al valore attuale nel ciclo con `#1`. Quindi

```
\lforeach{cane,gatto,criceto}{Animale: #1\par}
```

produrrà

Animale: cane
Animale: gatto
Animale: criceto

Le opzioni sono `single`, `double`, `triple` e `format`. La prima è implicita e indica che è presente solo una variabile, come nell'esempio precedente. Per illustrare l'opzione `double` si consideri

```
\lforeach[double]{
  cane/bau,gatto/miao,criceto/squit
}{Il #1 fa #2\par}
```

che produce

Il cane fa bau
Il gatto fa miao
Il criceto fa squit

Questo ciclo ha due variabili, i cui valori sono indicati separandoli con una barra. L'opzione `triple` si applica in modo analogo a liste i cui elementi siano del tipo `A/B/C`. L'opzione `format` permette invece di indicare le variabili (massimo nove) in modo molto libero:

```
\begin{tikzpicture}
\lforeach[format=(#1;#2)]{
  (0;0),(0;1),(1;1),(1;0)
}{\draw (#1,#2) circle (0.1cm);}
\end{tikzpicture}
```

per ottenere

Si noti che le coordinate non possono essere separate da una virgola, per non confondere il *parser* della lista. Con `\foreach` si potrebbe scrivere lo stesso codice come

```
\begin{tikzpicture}
\foreach \x in {(0,0),(0,1),(1,1),(1,0)}{
  \draw \x circle (0.1cm);
}
\end{tikzpicture}
```

che però limita la possibilità di adoperare separatamente le due variabili. Se la lista di coordinate è ottenuta da qualche sorgente esterna, non sarà difficile modificarla in modo da poter adoperare `\lforeach`.

Naturalmente è possibile annidare `\lforeach` e `\nforeach` in ogni combinazione:

```
\lforeach{cane,gatto,criceto}{%
  \nforeach{start=1,end=3}{%
    #1--##1 }\par
}
```

produce

cane–1 cane–2 cane–3
gatto–1 gatto–2 gatto–3
criceto–1 criceto–2 criceto–3

Riferimenti bibliografici

Braams, J., Carlisle, D., Jeffrey, A., Lamport, L., Mittelbach, F., Rowley, C. e Schöpf, R. (2016). «The LaTeX 2ε sources, 2016/03/31 patch level 3». `texdoc source2e`.

Goossens, M., Mittelbach, F. e Samarin, A. (1994). *The LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA.

Tantau, T. (2015). «The Ti*k*Z and PGF packages (version 3.0.1a)». `texdoc tikz`.

The LaTeX3 Project (2016a). «The LaTeX3 interfaces». `texdoc interface3`.

— (2016b). «The expl3 package and LaTeX3 programming». `texdoc expl3`.

— (2016c). «The xparse package; document command parser». `texdoc xparse`.

▷ Enrico Gregorio
Dipartimento di Informatica
Università di Verona
`enrico DOT gregorio AT univr DOT it`

Tabella 2: Il codice per `\nforeach` e `\lforeach`

```
\ExplSyntaxOn
\providecommand\fpeval{\fp_eval:n}

\NewDocumentCommand{\nforeach}{ m +m }
 {
  \int_gincr:N \g__manual_foreach_map_int
  \tl_clear:N \l__manual_nforeach_type_tl
  \keys_set:nn { manual/nforeach }
   {
    type=integers,start = 1, step = 1, end = 0,
   }
  \keys_set:nn { manual/nforeach } { #1 }
  \__manual_nforeach_exec:n { #2 }
  \int_gdecr:N \g__manual_foreach_map_int
 }

\int_new:N \g__manual_foreach_map_int
\int_new:N \g__manual_fp_map_int
\tl_new:N \l__manual_nforeach_type_tl

\keys_define:nn { manual/nforeach }
 {
  type .choice:,
  type .value_required:n = true,
  type/integers .code:n = \tl_set:Nn \l__manual_nforeach_type_tl { integers },
  type/fp       .code:n = \tl_set:Nn \l__manual_nforeach_type_tl { fp },
  type/alph     .code:n = \tl_set:Nn \l__manual_nforeach_type_tl { alph },
  type/Alph     .code:n = \tl_set:Nn \l__manual_nforeach_type_tl { Alph },
  start .tl_set:N = \l__manual_nforeach_start_tl,
  step  .tl_set:N = \l__manual_nforeach_step_tl,
  end   .tl_set:N = \l__manual_nforeach_end_tl,
 }

\cs_new_protected:Nn \__manual_nforeach_exec:n
 {
  \str_case:Vn \l__manual_nforeach_type_tl
   {
    {integers}{\__manual_nforeach_exec_integers:n { #1 }}
    {fp}      {\__manual_nforeach_exec_fp:n { #1 }}
    {alph}    {\__manual_nforeach_exec_alph:Nn \int_to_alph:n { #1 }}
    {Alph}    {\__manual_nforeach_exec_alph:Nn \int_to_Alph:n { #1 }}
   }
 }
\cs_generate_variant:Nn \str_case:nn { V }

\cs_new_protected:Nn \__manual_nforeach_exec_integers:n
 {
  \int_step_inline:nnnn
   { \l__manual_nforeach_start_tl }
   { \l__manual_nforeach_step_tl }
   { \l__manual_nforeach_end_tl }
   { #1 }
 }
\cs_new_protected:Nn \__manual_nforeach_exec_alph:Nn
 {
  \cs_set:cn { __manual_nforeach_alph_ \int_use:N \g__manual_foreach_map_int :n } { #2 }
  \cs_generate_variant:cn
   { __manual_nforeach_alph_ \int_use:N \g__manual_foreach_map_int :n }
   { f }
  \int_step_inline:nnnn
   { \int_from_alph:f { \l__manual_nforeach_start_tl } }
   { \l__manual_nforeach_step_tl }
   { \int_from_alph:f { \l__manual_nforeach_end_tl } }
   {
    \use:c { __manual_nforeach_alph_ \int_use:N \g__manual_foreach_map_int :f }
     { #1 { ##1 } }
   }
 }
\cs_generate_variant:Nn \cs_generate_variant:Nn { c }
\cs_generate_variant:Nn \int_from_alph:n { f }

\cs_new_protected:Nn \__manual_nforeach_exec_fp:n
 {
  \manual_fp_step_inline:nnnn
   { \l__manual_nforeach_start_tl }
   { \l__manual_nforeach_step_tl }
   { \l__manual_nforeach_end_tl }
   { #1 }
 }
```

Tabella 3: Il codice per \nforeach e \lforeach (continua)

```
% a replacement for \fp_step_inline:nnnn
\seq_new:N \l__manual_fp_step_seq
\fp_new:N \l__manual_fp_step_start_fp

\cs_new_protected:Nn \manual_fp_step_inline:nnnn
 {
  \int_gincr:N \g__manual_fp_map_int
  \seq_clear_new:c { l__manual_fp_step_ \int_use:N \g__manual_fp_map_int _seq }
  \fp_compare:nTF { #2 < \c_zero_fp }
   {
    \__manual_fp_step_make_neg:nnn { #1 } { #2 } { #3 }
   }
   {
    \__manual_fp_step_make_pos:nnn { #1 } { #2 } { #3 }
   }
  \seq_map_inline:cn { l__manual_fp_step_ \int_use:N \g__manual_fp_map_int _seq } { #4 }
  \int_gdecr:N \g__manual_fp_map_int
 }
\cs_new_protected:Nn \__manual_fp_step_make_neg:nnn
 {
  \fp_set:Nn \l__manual_fp_step_start_fp { #1 }
  \fp_do_while:nn { \l__manual_fp_step_start_fp >= #3 }
   {
    \seq_put_right:cx { l__manual_fp_step_ \int_use:N \g__manual_fp_map_int _seq }
     { \fp_eval:n { \l__manual_fp_step_start_fp } }
    \fp_add:Nn \l__manual_fp_step_start_fp { #2 }
   }
 }
\cs_new_protected:Nn \__manual_fp_step_make_pos:nnn
 {
  \fp_set:Nn \l__manual_fp_step_start_fp { #1 }
  \fp_do_while:nn { \l__manual_fp_step_start_fp <= #3 }
   {
    \seq_put_right:cx { l__manual_fp_step_ \int_use:N \g__manual_fp_map_int _seq }
     { \fp_eval:n { \l__manual_fp_step_start_fp } }
    \fp_add:Nn \l__manual_fp_step_start_fp { #2 }
   }
 }

\NewDocumentCommand{\lforeach}{ s O{} m +m }
 {
  \IfBooleanTF{#1}
   {
    \manual_lforeach:non { #2 } { #3 } { #4 }
   }
   {
    \manual_lforeach:nnn { #2 } { #3 } { #4 }
   }
 }

\cs_new_protected:Nn \manual_lforeach:nnn
 {
  \int_gincr:N \g__manual_foreach_map_int
  \keys_set:nn { manual/lforeach } { single }
  \keys_set:nn { manual/lforeach } { #1 }
  \clist_set:Nn \l__manual_lforeach_list_clist { #2 }
  \__manual_lforeach_define:n { #3 }
  \clist_map_inline:Nn \l__manual_lforeach_list_clist
   {
    \use:c { __manual_lforeach_ \int_use:N \g__manual_foreach_map_int _action:w } ##1 \q_stop
   }
  \int_gdecr:N \g__manual_foreach_map_int
 }
\cs_generate_variant:Nn \manual_lforeach:nnn { no }

\cs_new_protected:Nn \__manual_lforeach_define:n
 {
  \exp_last_unbraced:NcV
   \cs_set:Npn
   { __manual_lforeach_ \int_use:N \g__manual_foreach_map_int _action:w }
   \l__manual_lforeach_format_tl
   \q_stop
   {#1}
 }

\keys_define:nn { manual/lforeach }
 {
  format .tl_set:N = \l__manual_lforeach_format_tl,
  single .code:n = \tl_set:Nn \l__manual_lforeach_format_tl { ##1 },
  double .code:n = \tl_set:Nn \l__manual_lforeach_format_tl { ##1/##2 },
  triple .code:n = \tl_set:Nn \l__manual_lforeach_format_tl { ##1/##2/##3 },
 }
\ExplSyntaxOff
```

Towards a New Bibliography Format

Jean-Michel Hufflen

Abstract

MlBibTeX, our reimplementation of BibTeX, uses an enriched format for bibliography database files. Due to some features of this format, it is not backward-compatible with the conventions recognised by 'old' BibTeX. We explain why and make precise the requirements for a modern bibliography format and the specification of our new format.

Keywords MlBibTeX, bibliography processor, Unicode, character encodings, bibliography format.

Sommario

MlBibTeX, la nostra reimplementazione di BibTeX, usa un formato ampliato per il database bibliografico. A causa di alcune particolarità di questo formato, non è retrocompatibile con le convenzioni del 'vecchio' BibTeX. Spieghiamo perché e mostriamo con precisione i requisiti per un formato bibliografico moderno e le specifiche del nostro nuovo formato.

Parole chiave MlBibTeX, elaboratore bibliografico, Unicode, codifche dei caratteri, formato bibliografico.

Preliminary note The bibliography of the present article has been processed with MlBibTeX, as a showcase for this program.

Introduction

As mentioned in Mittelbach and Goossens (2004, § 13.1), the LaTeX typesetting system has deeply evolved since its first versions whereas BibTeX, the bibliography processor usually associated with LaTeX, has remained stable for many years. Let us recall that this program reads an auxiliary (.aux) file built when LaTeX typesets a source text (.tex file). The citation keys—used by the `\cite` command throughout .tex files—are extracted from this auxiliary file, and BibTeX looks in bibliography database (.bib) files for these keys. The auxiliary file also gives the pathnames of the .bib files to be searched and the information needed for the layout of a 'References' section. With BibTeX, such layouts are put into action by means of *bibliography styles*.

BibTeX is ageing, and we observe some possible replacement for several years. In particular, the biber program (Kime and Charette, 2014), generating references suitable for the biblatex package (Lehman et al., 2014). Let us recall that such references are marked up by means of LaTeX commands; accurate redefinitions of these commands allow bibliography styles to be put into action. Another example of a possible replacement is our reimplementation, MlBibTeX[1] (Hufflen, 2015). Since there is a *huge* number of .bib files—especially on the Web—BibTeX's successors still use this format for bibliography database files.

In Hufflen (2015) we introduced a new version of MlBibTeX, presently in beta-test. We recalled some syntactic extensions for .bib files. But these extensions induce some backward-compatibility problems: if end-users of MlBibTeX have written .bib files with this enriched syntax, what happens if they have to revert to 'old' BibTeX? Such a *scenario* is possible: nowadays, the whole process of publishing research papers in conference proceedings is often controlled by Web programs[2]. On such sites, all of the successive steps are controlled: electronic submission, acceptance or rejection, and depositing the final version if the paper has been accepted. Concerning this last step, often BibTeX is the only usable bibliography processor when articles are typeset with LaTeX.

This is clearly a drawback of using MlBibTeX[3], but in our personal opinion this drawback is the price to pay to get more expressive power within .bib files. We think that this format should be redefined and extended in next years, since there are requirements—especially modern ones—difficultly implementable within 'original' .bib format. In fact, this article aims to show that most of our syntactic extensions are debatable solutions out of .bib format expressive power. A comparable work about abbreviating authors' and editors' first names within bibliographies has already been presented in Hufflen (2016). The present work aims to give a more general view. In the first section, we recall this .bib format's origin and show how some LaTeX commands may be used as workarounds in quite simple cases. That is suitable for a 'basic' use of BibTeX, but makes other applications more complicated: for example, formatting the contents of .bib files and displaying the result on the Web by means of HTML[4] pages. Section 2 shows other examples where putting LaTeX commands does not apply or is unsatisfactory. As an alternative, the technique used by the biblatex package

1. **M**ulti**L**ingual BibTeX.
2. The most famous site for Computer Science conferences is indisputably `http://www.easychair.org`.
3. Some analogous drawbacks exist for biblatex users, as shown in § 3.
4. **H**yper**T**ext **M**arkup **L**anguage.

```
@BOOK{de-camp1991,
        AUTHOR = {Lyon Sprague de Camp and
                  Catherine Crook de Camp},
        TITLE = {The Incorporated Knight},
        PUBLISHER = {Baen Books},
        YEAR = 1991,
        MONTH = feb}
```

FIGURE 1: Example of a bibliographical entry.

to increase .bib files' expressive power is described in § 3. Then our choices are discussed in § 4. Reading this article requires some good knowledge of BIBTEX; advanced technical details can be found in Mittelbach and Goossens (2004, Ch. 13).

1 The .bib format

An example of a bibliographical entry usable by BIBTEX is shown in Fig. 1. In reality, this program (Patashnik, 1988) was initially designed to work with Scribe (Reid, 1984). This historical point explains some features of BIBTEX: Scribe has influenced LATEX—in particular, the notion of *document style* originates from Scribe—and was one of the first *markup languages*. However its syntax is close to LATEX's but is not identical. In Scribe, the '@' character is used at the beginning of a command name, such a command argument may be surrounded by braces, parentheses or double quote characters; '\' is not a special character[5], as in (LA)TEX. We see the origin of the '@' characters used within entry types such as @ARTICLE or @BOOK, the association of field names with corresponding values being surrounded by braces—mostly used—or parentheses. Even though BIBTEX's syntax is close to LATEX's and braces can be used to group some consecutive characters, accent commands are not recognised.

As LATEX has succeeded whilst Scribe has just had some historical value, the former has become the only word processor targeted by BIBTEX: for many years, bibliography styles built 'References' sections for LATEX, not for Scribe. So writers get used to put LATEX commands inside values associated with fields, for example, to specify typographical effects:

```
TITLE = {\emph{Babylon Babies}}
```

(this work's title uses italicised characters for foreign words, the book being written in French), or reach accented letters:

```
AUTHOR = {Herbert Vo{\ss}}
```

(the \ss command gives the German letter 'ß').

5. Analogous syntax is still used within Texinfo (Chassell and Stallman, 2008), the markup language used to document the products of the GNU (GNU's Not UNIX) projects.

These two examples are suitable for LATEX, but not for ConTEXt (Hagen, 2001), another format built out of TEX: the \emph command is unknown and the \ss command causes a switch towards a sans-serif font[6]. The second problem should disappear with the use of Unicode-compliant .bib files; however the 'ß' letter is quite frequent in German, and this \ss command would be still used in .bib files already developed. To correctly process these cases in ConTEXt the parser of .bib files should include a kind of mini-TEX parser for such commands. The same point holds if you are interested in building texts according to other formats, e.g., HTML pages.

It is well-known that the AUTHOR and EDITORS fields are structured: co-authors or co-editors are separated by the 'and' keyword, as shown in Fig. 1, names are structured into four parts: *First*, *von* (a particle), *Last* and *Junior*[7]. As suggested by our notation, the *First*, *Last* and *Junior* parts begin with an uppercase letter whereas the *von* part begins with an uppercase letter. The complete rules for parsing names in BIBTEX are quite complicated[8], but the first author's name given in Fig. 1 is easily processed as follows:

```
first => Lyon Sprague, von => de,
last => Camp
```

(by using MlBIBTEX's alternative notation). If the particle begins with a lowercase letter, a workaround is needed as shown in Mittelbach and Goossens (2004, p. 766–768):

```
Maria {\MakeUppercase{d}e La} Cruz
```

Because of the 'd' letter, BIBTEX considers that the second token begins with a lowercase letter—the command name being irrelevant—and the dummy \MakeUppercase command is only used to put an uppercase letter when the bibliography is typeset. Here also, we need a mini-TEX parser to handle such cases for formats other than LATEX.

If you build a 'References' section where first names are to be abbreviated, let us recall that in most cases, only the first letter is retained, but some abbreviations use several letters, e.g., 'Clive', abbreviated into 'Cl.' In addition, some authors drop out their middle name, so 'Clive Eric Cussler' should be abbreviated into 'Cl. Cussler'. You can specify such *modus operandi* in BIBTEX:

```
{\relax Cl}{ive Eric} Cussler
```

6. In the first case, you have to use the '{\em ...}' construct. It exists in LATEX, but it is preferable to use the \emph command. Concerning the 'ß' letter, the ConTEXt command to get it is \SS, the namesake command in LATEX causes the case of 'ß' to be raised and the result is 'SS'.
7. This part is also known as *Lineage*.
8. You can find them in Hufflen (2006).

but from our point of view, it is actually a *dirty* trick. In such a case, additional syntax to specify a non-standard abbreviation is missing in the .bib format. The solution of MlBibTeX is more readable:

```
AUTHOR = {Clive Eric Cussler,
          abbr => Cl.}
```

An alternative solution could be:

```
Cussler,, Clive Eric, Cl.
```

—an empty *Junior* part being specified between the first two commas—but such notation using three commas is incorrectly processed by BibTeX and causes biber to crash.

2 More features

2.1 Difficultly with LaTeX commands

If we go on with fields for person names, it is impossible to specify *additional collaborators* if you use only standard fields[9]. Either collaborator names are dropped out, or viewed as co-authors. You can add specific fields, but they will not be recognised by standard bibliography styles. The syntax put into action by MlBibTeX just uses another connector, ignored by standard bibliography styles:

```
AUTHOR = {Robert Silverberg with
          Karen Huber}
```

(another example is given by Mittelbach and Goossens (2004)).

Sometimes there should be no space between a particle and the *Last* part:

Guy d'Antin

As far as we know, there is no satisfactory solution for specifying this French name in BibTeX, in order for it to be typeset nicely.

Let us go back to abbreviating first names, some authors retain their middle name when their first name is abbreviated, e.g.:

Henry Rider Haggard ⟶ H. Rider Haggard

Some books about LaTeX and BibTeX—e.g., Mittelbach and Goossens (2004, Ch. 13)—propose the use of square brackets:

```
H[enry] Rider Haggard
```

but only a few styles are able to handle them. Here also, MlBibTeX's additional syntax is unusable with 'old' BibTeX, but seems to us to be clearer:

```
Henry Rider Haggard, abbr => H. Rider
```

9. Such specification is allowed within the bibliographies managed by the DocBook system (Walsh, 2010).

```
<personname>
  <first>Guy</first>
  <von space-after-f="no">d'</von>
  <last>Antin</last>
</personname>
```

Figure 2: MlBibTeX's internal format.

2.2 Considerations about sorting

In Hufflen (2014), we mentioned that BibTeX was only able to perform *lexicographic* sorts, the YEAR field is just concatenated to other information. An analogous problem complicates the use of organism names as authors or editors. The following specification, using only a *Last* part:

```
AUTHOR = {{\GuIT}}
```

aims to put the GuIT logo of the Italian TeX Users Group as the author of a collective work. Unfortunately, the commands at the beginning of a BibTeX token are pruned when such a token is used as a sort key, as done about the \MakeUppercase command mentioned in § 1. As a consequence, the sort key derived from this specification is empty.

3 The solutions of biblatex

The biblatex package did not extend the look of existing fields, but added new ones. This is successful for sort operations since some information can be redefined at the sort step. For example, the fields SORTNAME and SORTYEAR take precedence over the fields AUTHOR and YEAR. In particular, the example given in § 2.2 can be extended as follows:

```
AUTHOR = {{\GuIT}},
SORTNAME = {GuIT}
```

whereas MlBibTeX's solution is:

```
AUTHOR = {org => \GuIT,
          sortingkey => GuIT}
```

The other problems mentioned above are not handled by the biblatex package. In addition, let us remark that biblatex end-users also know some backward-compatibility problems if they are to revert to 'old' BibTeX. A simple example is given by the DATE field, which extends the specification of dates—you can specify not only a simple date, but also a *range* of dates—: this field takes precedence over the predefined fields YEAR and MONTH. Obviously, these end-users prefer to use this DATE field, which increases the expressive power of .bib files, but its drawback is that it is not recognised by standard bibliography styles.

4 Discussion and conclusion

BibTEX successors have developed interesting extensions, but often these extensions are mutually incompatible. That is regrettable, but we may think that these experimentation steps should lead to a new format for bibliographical entries in the near future. Maybe the .bib format reached its limits and it is now difficult to extend it. So, the best solution for a new bibliography format could be based on another syntax, e.g., XML[10]. In particular, this format should implement some LaTeX command used within .bib files, in order to ease their translation into other formats, as mentioned in § 1. Obviously, it should provide solutions to problems described in § 2.

When we designed MlBibTEX, we decided for such an internal format, XML-based. We have been able to define our extensions, implement the features of interest for us. Since we would like to get access to these extensions, we defined some concrete syntax, as extensions within .bib files. Most often, but not always. For example, we can specify that there should be no space between a name's *von* and *Last* part in our internal format—as shown in Fig. 2—but we have not proposed yet some concrete syntax for this point within .bib files[11]. If a new format is defined with a comparable expressive power, we think that our functions could be working; we would have just to translate this new format into our internal one. We hope that such a new standard for bibliography database files will be carried out. In the meantime, we show that some syntactic extensions of .bib files—even if they are not backward compatible—can lead to interesting and useful results.

Acknowledgements

I thank Claudio Beccari for his Italian translations of the abstract and keywords.

References

Robert J. Chassell and Richard M. Stallman. *Texinfo. The* GNU *Documentation System. Version 4.13*, September 2008. http://www.gnu.org/software/texinfo.

Hans Hagen. *ConTEXt, the Manual*. http://www.pragma-ade.com/general/manuals/cont-enp.pdf, November 2001.

Jean-Michel Hufflen. Names in BibTEX and MlBibTEX. TUG*boat*, 27(2):243–253, November 2006. TUG 2006 proceedings, Marrakesh, Morocco.

Jean-Michel Hufflen. Dealing with ancient works in bibliographies. *ArsTEXnica*, 18:81–86, October 2014. In Proc. GUIT meeting 2014.

Jean-Michel Hufflen. MlBibTEX 1.4: the new version. *ArsTEXnica*, 20:35–39, October 2015. In Proc. GUIT meeting 2015.

Jean-Michel Hufflen. Abbreviating first names. In *Proc. BachoTEX 2016*, April 2016.

Philip Kime and François Charette. *biber. A Backend Bibliography Processor for biblatex. Version biber 1.9 (biblatex 2.9)*. http://ftp.oleane.net/pub/CTAN/biblio/biber/documentation/biber.pdf, May 2014.

Philipp Lehman, Philip Kime, Audrey Boruvka, and Joseph Wright. *The biblatex Package. Programmable Bibliographies and Citations. Version 2.9a*. http://ctan.mirrorcatalogs.com/macros/latex/contrib/biblatex/doc/biblatex.pdf, June 2014.

Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig, and Joachim Schrod. *The LaTeX Companion*. Addison-Wesley Publishing Company, Reading, Massachusetts, 2 edition, August 2004.

Oren Patashnik. *BibTEXing*. Part of the BibTEX distribution, February 1988.

Brian Keith Reid. SCRIBE document production system user manual. Technical report, Unilogic, Ltd., 1984.

Norman Walsh. *DocBook 5. The Definitive Guide*. O'Reilly & Associates, Inc., May 2010. Edited by Richard L. Hamilton.

▷ Jean-Michel Hufflen
FEMTO-ST (UMR CNRS 6174) &
University of Franche-Comté,
16, route de Gray,
25030 BESANÇON CEDEX
FRANCE
jmhuffle at femto-st dot fr

10. e**X**tensible **M**arkup **L**anguage.

11. More precisely, we have not found yet a satisfying solution. Let us go back to Fig. 2, the `spacing-after-f` attribute obviously defaults to `yes`, that is, if a bibliography style orders the insertion of a space character after a *von* part, this space character should not be removed.

Il formato PDF archiviabile con gli aggiornamenti di TEX Live 2016

Claudio Beccari

Sommario

La distribuzione TEX Live 2016 ha portato importanti novità per quel che riguarda la produzione di documenti PDF archiviabili a lungo termine. La novità maggiore è che ora i pacchetti disponibili permettono di usare anche XeLaTeX e LuaLaTeX oltre che pdfLaTeX. Con XeLaTeX bisogna fare un poco di attenzione ma le difficoltà che questo programma mostra nella produzione di documenti archiviabili si possono superare.

L'archiviabilità a lungo termine è importante in ogni settore professionale e disciplinare; le discipline umanistiche, specialmente quelle legate alle lingue, hanno avuto finora dei grossi problemi con gli alfabeti diversi da quello latino; ora sembrano superati specialmente se ci si affida a LuaLaTeX che permette di gestire agilmente tutti i tipi di font OpenType.

Abstract

The TEX Live 2016 distribution contains important advances for what concerns the production of long term archivable PDF documents. The most important new functionality consists in the fact that now it is possible to use XeLaTeX and LuaLaTeX, not only pdfLaTeX. With XeLaTeX it is necessary to pay some special care, but these program features may be overcome.

Long term archivability is very important in any professional or research area and in any discipline; up to now humanities, especially those dealing with ancient languages that are written with alphabets different form Latin, had some real problems that now appear to be overcome if LuaLaTeX is used, thanks to the fact that it manages any kind of OpenType fonts.

1 Introduzione

La produzione di documenti PDF archiviabili a tempo indefinito interessa ogni attività professionale e accademica in ogni possibile settore disciplinare. Si pensi soltanto alla grande quantità di documenti legali che si producono durante i vari processi; ai progetti edili con particolare attenzione a quelli che hanno una rilevanza giuridica; alle certificazioni di ogni genere; a testi oggetto di archiviazione per l'uso libero da parte di ricercatori di ogni disciplina; all'archiviazione elettronica delle tesi di laurea e di dottorato. Si potrebbe continuare a lungo l'elenco, ma è evidente che non esiste un solo settore in cui l'archiviazione a lungo termine non sia una necessità assoluta.

Fino a poco tempo fa il sistema TEX disponeva di poche risorse; esisteva il pacchetto pdfx del 2007, ma per farlo funzionare era necessario metterci le mani dentro, come ho mostrato in un precedente articolo (Beccari, 2009). Pur tuttavia qualcosa si riusciva a fare solo con la compilazione mediante il programma di tipocomposizione `pdflatex`; purtroppo il risultato conforme alla norma PDF/A-1b non era sempre positivo.

Esisteva anche la via attraverso il programma `ghostscript` che provvedeva alla trasformazione di un file PDF "normale" in uno conforme alle norme ISO 19005:2005-1 ma a spese di molti compromessi e con probabilità di riuscita incerte.

Scarso (2010) ha poi presentato su ArsTEXnica i suoi risultati ottenuti con il programma di tipocomposizione ConTEXt; riusciva a produrre i file PDF nella versione *tagged PDF*, che sono la base per le norme più restrittive PDF/A-1a; ma anche con ConTEXt il risultato era incerto. Forse oggi ConTEXt offre maggiori possibilità, ma il suo uso non è così vasto come quello dei programmi basati sul mark up LaTeX.

Oggi il pacchetto pdfx rinnovato ed esteso permette di comporre documenti PDF/X conformi alle norme per la stampa in quadricromia sottrattiva secondo una moltitudine di successive norme che continuano ad aggiornarsi anno dopo anno. Similmente quel pacchetto permette di comporre documenti conformi agli standard PDF/A di diversi livelli secondo i successivi aggiornamenti della norma.

La documentazione del nuovo pacchetto pdfx (Radhakrishnan *et al.*, 2016) descrive tutti gli standard che grazie a esso sarebbe possibile creare (al verificarsi di altre condizioni nel seguito) e mette in evidenza anche quegli standard che il pacchetto potrebbe aiutare a comporre quando sarà possibile usarlo con i programmi di tipocomposizione, basati sul mark up LaTeX, capaci di produrre file di tipo *tagged PDF*.

Il pacchetto ghostscript riesce ancora a fare il lavoro che faceva prima ma la documentazione della versione 9.16 dovrebbe venire aggiornata, perché le istruzioni sono lacunose.

Nel frattempo anche `pdflatex` con il pacchetto pdfpages potrebbe fare un lavoro equivalente o

migliore di quello che si può fare con `ghostscript`; l'unico vantaggio di quest'ultimo è che, volendo, riuscirebbe a trasformare direttamente un file in formato PS in un file conforme alla norma PDF/A-1b.

Inoltre con la distribuzione 2016 di TEX Live, è stata distribuita anche la versione 0.95 del programma `luatex` che introduce notevoli modifiche rispetto alla versione distribuita con TEX Live 2015; alcune di queste sono funzionali all'uso di LuaLATEX per comporre documenti conformi allo standard PDF/A.

In questo articolo cercherò di esporre succintamente come si può usare il pacchetto pdfx con i programmi di tipocomposizione basati sul mark up LATEX; aggiungerò qualche nota per descrivere il procedimento che si può seguire con il programma `pdflatex` insieme al pacchetto pdfpages e con il programma `ghostscript` per creare documenti PDF/A compatibili; questi procedimenti possono essere utili non solo per documenti interi, ma anche per singole parti da includere nei documenti completi, per esempio disegni, tabelle, e simili, da riportare come citazioni da altri documenti.

2 Caratteristiche dei documenti PDF/A

Perché un documento PDF sia conforme alla norma ISO 19005:2005 e alle sue successive modifiche, esso deve essere composto in modo che soddisfi a certi requisiti, diversi per le varianti 'a' o 'b' di ciascuna norma. La variante 'a' richiede che il file sia *tagged PDF*, mentre la variante 'b' non ha questa esigenza. Siccome a mia conoscenza, tranne ConTEXt che però non si basa sul mark up LATEX, nessuno dei programmi di tipocomposizione del sistema TEX è in grado (per ora[1]) di produrre file di tipo *tagged PDF*, la specifica 'a' è fuori discussione. Mi limiterò dunque a descrivere solo la produzione di documenti conformi alla norma PDF/A-1b; questi devono avere le seguenti caratteristiche.

1. Il file PDF deve essere compilato con il linguaggio PDF di livello 1.4. Secondo standard approvati dopo il 2005, ora si potrebbero usare anche versioni più avanzate.
2. Il file deve contenere un insieme di metadati, alcuni obbligatori, altri facoltativi; il pacchetto pdfx provvede ai metadati obbligatori e ad alcuni facoltativi.
3. I metadati devono essere contenuti nel file PDF in chiaro, non in forma compressa né criptata.
4. Il file deve contenere una specificazione del profilo di colore per le immagini che contiene; il pacchetto pdfx provvede un profilo di colore RGB abbastanza generico; se però l'utente desidera usare un profilo di colore particolare lo può fare usando i comandi appositi messi a disposizione dal pacchetto pdfx. I profili di colore RGB vanno bene sia per le immagini colorate sia per quelle in tonalità di grigio.
5. Il file PDF deve essere autosufficiente per quel che riguarda i font, perciò deve contenere almeno tutti i glifi dei vari font usati nel documento.
6. Sono assolutamente vietati i font bitmapped (o *raster*, o a *matrici di punti*); nessun glifo deve avere larghezza nulla.
7. Non sono ammessi overlay di glifi; in particolare non sono ammessi gli accenti e gli altri segni diacritici ottenuti per sovrapposizione, come quando si usa la codifica dei font OT1.
8. Esistono delle limitazioni per quel che riguarda i collegamenti ipertestuali, se non quelli verso punti interni allo stesso documento.

Esistono altre limitazioni, descritte in documenti avanzati relativi alla norma ISO 19005, ma i punti importanti sono quelli elencati sopra.

3 Il pacchetto pdfx

Ciò premesso, quello che fa il pacchetto pdfx consiste proprio nel garantire che le prescrizioni siano soddisfatte, ma non può fare miracoli. Per esempio non dipende da pdfx se l'utente inserisce un'immagine con profilo di colore diverso da RGB, magari col profilo CMYK; non può correggere i metadati errati, anche se è in grado (specialmente con `pdflatex`, che può usare solo font codificati a 8 bit, in particolare font Type 1, con codifiche molto diverse le une dalle altre) di trasformare i glifi di entrata in modo da renderli compatibili con lo standard[2]. Non ci sono problemi a questo proposito se si usa LuaLATEX, ma... Problemi simili si possono verificare con l'importazione di file esterni, ma si tratta sempre di sviste dell'utente.

Per i metadati facoltativi il pacchetto pdfx offre un certo numero di parole chiave documentate per

1. In realtà da tempo si trova nel sito indicato nella documentazione di pdfx un insieme di file da compilare sulla propria macchina al fine di produrre l'eseguibile di `pdftex` versione 1.50, capace di inserire i metadati richiesti; pur essendo disponibile da alcuni anni, questa versione non è mai stata resa pubblica nel senso di essere pubblicamente distribuita con TEX Live che, invece, ancora oggi contiene la versione 1.40. Quella versione sperimentale probabilmente aveva ancora dei piccoli bachi che non le consentivano di essere distribuita se non come versione beta assolutamente sperimentale e da usarsi a proprio rischio e pericolo. Io l'usai tempo fa, senza incontrare problemi particolari, ma non l'ho più né scaricata, né compilata, né integrata con il formato del linguaggio LATEX e non mi sento di raccomandarla proprio perché l'autore Hàn Thế Thành non l'ha resa pubblica lui stesso.

2. Molti anni fa ho creato i font greci a 8 bit con una codifica che di fatto è diventata quella (semi)ufficiale nel sistema TEX, cioè la codifica LGR; con mia grande sorpresa questi miei font greci non hanno mai dato nessun problema con le norme per l'archiviabilità; invece ho incontrato problemi seri con i font Computer Modern (CM) matematici usati da `pdflatex`.

esteso in Radhakrishnan *et al.* (2016). Ma la documentazione stessa richiede di inserire questi metadati in un file con il nome uguale a quello del file principale del documento ma con l'estensione `.xmpdata`. Il pacchetto consiglia però di usare sempre la codifica d'entrata utf8 sia per i metadati sia per il documento e di creare il file dei metadati mediante l'uso dell'ambiente filecontents* in questo modo:

```
% !TEX encoding = UTF-8 Unicode
\begin{filecontents*}{\jobname.xmpdata}
\Author{Mario Rossi}
\Title{My archivable document}
\Publisher{University of Einstenia}
\Keywords{PDF/A\sep
pdfx.sty package\sep
long term archiving\sep
LuaLaTeX}
\end{filecontents*}
\documentclass[...]{...}
\usepackage[a-1b]{pdfx}
...
\begin{document}
...
\end{document}
```

In questo modo si è sicuri che il documento e i metadati sono composti con la stessa codifica utf8; questo metodo va bene per tutti e tre i principali programmi di tipocomposizione. Solo per i file da compilare con `pdflatex` è necessario specificare nel preambolo l'uso del pacchetto inputenc con l'opzione utf8, mentre con gli altri due programmi non è necessario specificare nulla del genere, ma è necessario usare solo o prevalentemente font OpenType, ricordando che i font a 8-bit, se mai dovessero assolutamente venire usati, devono essere richiamati nei preamboli in modo particolare, rispettando certe priorità che sono descritte nella documentazione del pacchetto fontenc.

Nei limiti del possibile raccomanderei di usare la lingua inglese per i metadati per due motivi: (*a*) l'inglese di solito non usa parole con diacritici, e (*b*) se questi metadati devono essere usati per ricerche biblioteconomiche in giro per il mondo, è meglio che siano scritti nella lingua franca, l'inglese, piuttosto che in qualsiasi lingua nazionale.

4 Font problematici

Se si evitano i font Computer Modern sia per il testo sia per la matematica si è già a buon punto; si noti che tutti i font matematici diversi dai CM che si possono usare tramite appositi pacchetti, soffrono degli stessi inconvenienti dei CM, perché i loro file metrici sono ricavati dai file metrici di quelli. Per evitare di usare i file matematici difettosi si deve evitare di usare `pdflatex`; perciò si crei il documento con `lualatex`[3]: è un po' più lento di `pdflatex`, ma è decisamente consigliabile per la creazione di file PDF archiviabili. I font OpenType matematici non sono ancora molto numerosi, ma sono sufficienti per ogni evenienza, anche se l'occhio (molto) esercitato riesce a individuare alcune piccolissime anomalie di disegno di questi font rispetto ai font testuali con cui si dovrebbero accoppiare. Qui è troppo lungo scendere nei dettagli; posso solo dire in base alla mia esperienza che io non ho mai incontrato problemi seri con i font matematici OpenType.

Se bisogna importare brani di testo, formule, tabelle o simili, per citazioni da altri testi, è possibile che anche se si traggono da file PDF, quegli elementi, una volta estratti, non siano compatibili con i requisiti dei file PDF/A. In questi casi se si tratta di testo non dovrebbe essere un problema ricomporlo; se si tratta di tabelle la ricomposizione è più laboriosa ma è fattibile; se si tratta di formule che contengono simboli strani la cosa diventa difficile se non impossibile. In questi casi conviene trasformare il codice estratto in una figura; sarebbe preferibile il formato PNG, ma siccome questo potrebbe contenere delle trasparenze, vietate nel file PDF/A compatibili, è necessario ricorrere al formato JPG con un'adeguata densità di pixel.

Talvolta si riesce a mantenere il formato vettoriale ricorrendo al programma `inkscape` che è capace di trasformare i glifi presenti nel testo nei loro disegni vettoriali; lo fa molto bene e così facendo non si perde in qualità, ma l'operazione non produce sempre un file PDF compatibile con la norma PDF/A per cui non serve a niente incorporarlo nel proprio documento se è irrimediabilmente incompatibile.

5 Figure problematiche

Gli stessi discorsi valgono per le figure problematiche quando sono in formato vettoriale e se ne vuole mantenere la vettorialità. Oppure per le figure in formato PNG che contengano trasparenze. Per questi ultimi tipi di figure si perdono le trasparenze ma non si perde quasi nient'altro se le si trasforma in formato JPG. Le figure potrebbero avere un profilo di colore diverso da profilo RGB: per controllarlo ed eventualmente modificarlo si può usare il programma freeware e open source `GIMP` in grado di eseguire queste verifiche e le eventuali correzioni.

6 Usare `pdflatex`

Per produrre un file PDF/A compatibile basta usare un preambolo e un file `.xmpdata` come mostrato nel listato della pagina 84; naturalmente perché il file sia veramente compatibile con la norma bisogna che siano rispettate tutte le cautele indicate nei paragrafi precedenti.

3. Per i motivi esposti più avanti si eviti di usare `xelatex`.

7 Usare lualatex

Con lualatex le cose sono sotto certi aspetti più semplici, nel senso che la codifica con cui il file sorgente viene registrato sul disco *deve* essere utf8, non si può scegliere un'altra codifica. Un discorso simile vale per i font, che *devono* essere di tipo OpenType, anche se con particolari precauzioni si potrebbero usare anche font Type 1; lo si sconsiglia vivamente perché i font OpenType coprono la quasi totalità delle necessità. Per imitare i font di default usati con pdflatex, i font OpenType CMU (con grazie, senza grazie, a spaziatura fissa, latini, greci, cirillici, eccetera) sono praticamente completi. Possono non essere sufficienti per l'arabo, l'ebraico, il cinese, il giapponese, il coreano, ma con ogni sistema operativo sono installati anche moltissimi font OpenType, e serve solo la pazienza di trovare quelli giusti per comporre in una lingua esotica; alla peggio si possono scaricare dalla rete altri font OpenType gratuiti o commerciali, così che si possano soddisfare tutte le esigenze. I font CMU non sono ancora dotati della collezione corrispondente di font matematici, ma i font OpenType matematici XITS Math[4] vi si accoppiano abbastanza bene. Questo testo è composto con i font CMU testuali e i font XITS matematici: si confronti la formula seguente per constatare le minime differenze di stili fra i font matematici e quelli testuali.

$$x_{1,2} = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} & \text{se} \quad b^2 - 4ac > 0 \\ \dfrac{-b}{2a} & \text{se} \quad b^2 - 4ac = 0 \\ \dfrac{-b \pm \mathrm{i}\sqrt{4ac - b^2}}{2a} & \text{se} \quad b^2 - 4ac < 0 \end{cases}$$

8 Usare xelatex

Con xelatex le cose sono un po' più complesse, perché questo programma di suo non produce in uscita un file PDF, ma questo viene ottenuto dalla trasformazione di un file in formato DVI esteso mediante il programma xdvipdfmx. Questo è un programma completamente distinto da xelatex e non sa a che cosa il file DVI esteso è destinato a essere usato. Siccome il file PDF/A compatibile deve avere i metadati in formato non compresso, bisogna che xelatex passi il suo file di uscita a xdvipdfmx in modo che nulla sia compresso.

Per fare questo bisogna usare un comando dalla finestra comandi che esegua questo compito senza comprimere il file finale; in questo modo nemmeno i metadati vengono compressi. Il comando è il seguente (da scrivere in una sola riga):

```
xelatex -shell-escape
   -output-driver="xdvipdfmx -z 0"
   <filename>.tex
```

Esso è configurato per usare la modalità shell escape che consente al compilatore di eseguire comandi che si potrebbero dare in una finestra comandi; come tale, quell'intera riga può venire digitata in una finestra comandi e il sistema operativo provvede a eseguire il comando principale xelatex; al momento opportuno questo si interrompe e ripassa il controllo al sistema operativo per trasferire il suo file di uscita all'output-driver a cui fornisce anche l'opzione -z 0; questa opzione è quella che ordina a xdvipdfmx di *non comprimere* il file PDF che esso produce. Ovviamente si potrebbe configurare il proprio *shell editor* perché disponga di un bottone o di una voce di menù per eseguire questa operazione, senza doversela ricordare a memoria o senza dover andare a consultare nessuna documentazione per ricavare la sintassi corretta.

Ciò produce due effetti: (*a*) i metadati entrano nel file di uscita non compressi, ed è quello che si vuole ottenere, e (*b*) l'intero file non è compresso, quindi può avere dimensioni spropositate, anche 10 volte maggiori di quelle che lo stesso PDF avrebbe se fosse compresso. Questo inconveniente può essere "curato" come si vedrà fra poco.

Tuttavia a questo punto ci si potrebbe domandare: perché usare xelatex con le complicazioni descritte sopra, se lo stesso file sorgente potrebbe essere compilato altrettanto bene, se non meglio, con lualatex? Una risposta che riesco a trovare, ma probabilmente non è l'unica, è che non si ha abbastanza dimestichezza con lualatex e quindi non ci si azzarda a usare un programma che non si conosce bene. Un'altra risposta potrebbe essere che lualatex ha un modo suo particolare per gestire la divisione in sillabe per ogni lingua presente nel documento da comporre, e che in certi rarissimi casi questa modalità non è soddisfacente. Questa seconda spiegazione può interessare gli umanisti che si occupano di linguistica; per esempio se si occupano di latino o di greco, queste due lingue hanno ciascuna tre modi diversi di dividere in sillabe; attualmente io riesco a gestirmi il latino e il greco con le tre diverse impostazioni per ciascuna lingua ricorrendo a file di descrizione di queste lingue che mi sono costruito per mio uso e consumo, ma il gruppo di lavoro tex-hyphen sembra che non sia d'accordo ad adottare la mia soluzione (che, lo ammetto, è piuttosto artigianale, ma funziona); può darsi che in futuro anche i file ufficiali di queste lingue saranno in grado di gestire le loro tre varianti di sillabazione.

9 Produrre un file PDF/A con pdflatex e pdfpages

Se si dispone di un file PDF comunque creato e lo si vuole rendere conforme alla norma, si può

4. I font OpenType Latin Modern Math si accoppierebbero anche meglio, ma secondo la mia esperienza i font XITS, sebbene siano un pochino più scuri, si accoppiano bene e sono più completi.

usare anche un'altra tecnica. Devo dire subito che io non la userei per generare un intero documento, ma solo per rendere PDF/A compatibili semplici file da includere nel documento vero e proprio. Infatti per questa via si perdono i collegamenti ipertestuali interni e il documento finale non sarebbe "navigabile" se lo si legge con un dispositivo elettronico; però, lo stesso documento stampato andrebbe bene, perché a stampa i collegamenti ipertestuali attivi non servono.

Il semplice metodo consiste nel creare un piccolo file da elaborare con `pdflatex`; sia per esempio `mypdffile.pdf` il file da convertire nel formato PDF/A conforme alla norma ISO, diciamo `my-PDFA-file.pdf`. Si crea dunque il file di trasformazione in questo modo;

```
% !TEX encoding = UTF-8 Unicode
% !TEX TS-program = pdfLaTeX
%
%%% File my-PDFA-file.tex
%
% esempio di metadati
\begin{filecontents*}{\jobname.xmpdata}
\Author{Mario Rossi}
\Title{My PDF/A compliant document}
\Publisher{University of Einstenia}
\Keywords{PDF/A\sep
pdfx.sty package\sep
long term archiving\sep
LuaLaTeX}
\end{filecontents*}
%
\documentclass[a4paper]{book}
\usepackage[a-1b]{pdfx}
\usepackage{pdfpages}
\nofiles
\begin{document}\pagestyle{empty}
\includepdf[pages=-]{mypdffile.pdf}
\end{document}
```

Lo si compila regolarmente con `pdflatex` come ci ricorda il secondo commento all'inizio del file. Ovviamente, come ci ricordano la prima e la quarta riga di commento, questo piccolo file va registrato con la codifica `utf8` su disco con il nome `my-PDFA-file.tex`. Se il documento originale è stato composto con un formato di carta diversa da quella indicata con l'opzione `a4paper` si cambi l'opzione corrispondentemente, ma non è necessario cambiare altro, nel senso che i comandi che seguono importano nel documento che si sta componendo una alla volta le pagine del documento originale senza cambiare le loro caratteristiche grafiche né per quel che riguarda l'impaginazione né per quel che riguarda i font.

Se il documento originale non conteneva oggetti incompatibili con lo standard PDF/A, il file di uscita risulta conforme allo standard, essendo stato arricchito di tutti i metadati necessari e di alcuni metadati facoltativi.

10 Usare `ghostscript`

Anche il programma `ghostscript` consentirebbe di eseguire una procedura simile a quella indicata nel paragrafo precedente; il lettore interessato può leggere la documentazione fra quella di `ghostscript`; ne ho scritto anche in altri articoli e in altri documenti in rete, ma io stesso considero questa via come un residuo dei miei primi tentativi di creare file PDF/A compatibili; era il 2008 quando le funzionalità presenti oggi nella distribuzione del sistema TEX non erano disponibili.

Il risultato finale ha gli stessi inconvenienti che si incontrano con il procedimento descritto nel paragrafo precedente; ritengo però che la gestione dei metadati e del profilo di colore sia decisamente più semplice se si ricorre alle funzionalità del pacchetto `pdfx` che, lo ricordo, si può usare solo con i programmi di tipocomposizione del sistema TEX. L'unico vantaggio che `ghostscript` potrebbe avere, sarebbe quello di poter trasformare direttamente un file in formato PS in un altro file in formato PDF/A, ma si tratta di un vantaggio molto marginale, perché la trasformazione di un file PS in un file PDF è già eseguibile molto facilmente con il programma `pstopdf` facente parte della dotazione di programmi dei sistemi operativi UNIX e Linux, ma installabile anche per i sistemi operativi Windows. Ovviamente anche `ghostscript` può trasformare un file PS in uno corrispondente in formato PDF attraverso il suo driver `pdfwrite`.

11 Il file PDF ottenuto soddisfa i requisiti PDF/A?

Ora la domanda ovvia che ci si deve porre è se il PDF ottenuto soddisfa i requisiti della norma ISO sull'archiviabilità.

Il problema è abbastanza serio: da un lato non c'è nessuna garanzia che qualche inesattezza si sia intrufolata in quanto ho descritto, dall'altro c'è la necessità di essere sicuri di aver ottenuto il risultato voluto e di poter corredare il file con un "certificato di conformità".

Il primo aspetto è quello più delicato. Non è infrequente che si immettano nel documento finale file esterni dei quali si è sicuri che non contengano incompatibilità con lo standard PDF/A, quando invece ne contengono qualcuna.

Tuttavia il metodo per accertare questa compatibilità ed eventualmente produrre il certificato di conformità si avvale dello stesso programma: Adobe Acrobat Pro XI e versioni successive.

Esistono anche dei siti in rete che svolgono questo servizio gratuitamente; è possibile eseguire l'upload del proprio file PDF per ricevere a stretto giro di posta elettronica un messaggio che contiene la lista degli errori o il certificato di conformità. Ammetto di averne provati alcuni, ma non sono mai riuscito a ricevere verifiche positive, anche con

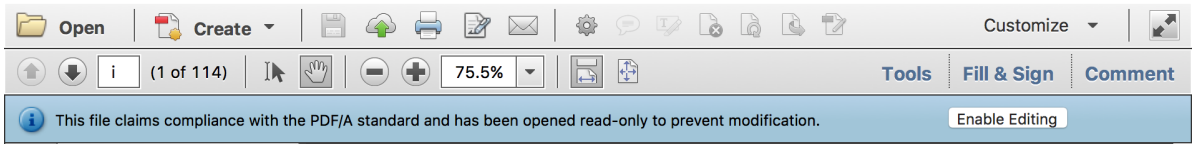


FIGURA 1: Avviso di `Acrobat Pro` sulla possibilità che il file appena aperto possa essere conforme con la norma PDF/A.

documenti che Adobe Acrobat Pro XI certificava come privi di problemi e quindi conformi con lo standard PDF/A.

Io credo che i tecnici dell'Adobe, che fanno parte della commissione che si occupa dalla norma ISO 19005, siano capaci anche di realizzare i programmi di verifica delle norme che loro stessi hanno contribuito a definire. Quindi sono propenso a fidarmi di più dell'esito della verifica effettuata con il modulo `Preflight` di Acrobat Pro XI, che non della verifica gratuita eseguita in rete.

Nello stesso tempo mi spiace che non esista un programma gratuito per verificare la conformità con la norma, come invece esistono più programmi gratuiti o commerciali che producono o dicono di produrre file compatibili. Oltre ai programmi di tipocomposizione del sistema TEX con l'aiuto del pacchetto `pdfx`, anche i word processor Word, di Microsoft Office, e Writer, di LibreOffice e di OpenOffice, sono in grado di esportare i loro file nel formato PDF e offrono anche la possibilità di specificare se si desidera la conformità non solo con la norma PDF/A-1b ma anche PDF/A-1a.

Infatti, tutte le volte che ho provato a esportare file in formato PDF da quei word processor, in entrambi i formati `A-1b` e `A-1a`, la funzione *Preflight* non mi ha mai risparmiato elenchi più o meno lunghi di manchevolezze. Con questo non voglio dire che quei word processor siano particolarmente inefficienti; semplicemente sembrano esserlo con i documenti non completamente testuali che io sono solito produrre.

Non sono in grado di dire niente pro o contro il word processor `Pages` del Mac, sebbene io usi un Mac, perché non l'ho mai usato per produrre file PDF che potessero ambire al rango di file archiviabili (talvolta vi scrivo lettere burocratiche o commerciali).

Prima di descrivere come procedere con Adobe Acrobat Pro XI, devo aggiungere che si tratta di un programma commerciale, che costa ben altro che qualche decina di euro. Tuttavia l'Adobe, come molte altre aziende dispone di un programma *Education* per vendere a prezzi assolutamente abbordabili i suoi prodotti ai docenti, al personale e agli studenti delle università e delle scuole secondarie superiori. Dipende dall'uso che se ne fa ma, visto che lo trovo molto utile, ritengo che quel programma sia consigliabile. Giudizio personale, naturalmente, ma nel parlare del programma Education non credo di aver violato alcuna norma etica.

Parlando di Adobe Acrobat posso aggiungere che consente di eseguire varie forme di editing sui file PDF; in particolare torna molto utile la sua funzione per comprimere file PDF non compressi o compressi poco; se questi file non compressi contengono i metadati per l'archiviabilità, metadati che devono rimanere in chiaro, non compressi, Acrobat Pro riesce a comprimere le parti lecite e a ridurre l'ingombro in memoria, salvando il risultato in un nuovo file; in sostanza con questo programma si supera il grosso inconveniente che si incontra quando si vuole ottenere un file PDF/A compatibile usando un file non compresso quale quello che si ottiene con `xelatex` come descritto nel paragrafo 8[5].

12 Il modulo `Preflight`

Se si apre con Adobe Acrobat (o anche con Adobe Reader) un file PDF che potrebbe essere conforme col formato archiviabile, appare una riga celeste nella parte alta della schermata, figura 1, dove compare un testo che spiega che, siccome il file pretende di essere conforme allo standard PDF/A, quel file è stato aperto in modalità protetta per evitarne qualsiasi modifica che potrebbe togliergli la qualifica di compatibilità con la norma.

Non ci si lasci ingannare: quella scritta dice espressamente che è il file che pretende di essere conforme alla norma; potrebbe esserlo, ma più frequentemente non lo è. Se ne può controllare la conformità solo usando la modalità `Preflight`. Questa si attiva dal menù "Edit" e selezionando la voce "Preflight" nel menù a tendina; vedi la figura 2. Più semplicemente basta premere simultaneamente i tasti shift, command e X. Ciò fatto appare la finestra riportata nella figura 3.

Se non fosse già selezionata, si seleziona in questo menù la riga che sceglie lo standard per il quale deve essere eseguito il controllo; come appare nella figura, si è selezionata la riga che corrisponde allo standard PDF/A-1b; ora si clicca il pulsante Analyze. Il programma si mette a lavorare e termina con il suo responso.

Se il file non è conforme, nella finestra di `Preflight` appare una *dialog box* come quella mostrata nella figura 4; vi appaiono tutti gli errori che `preflight` ha trovato; cliccando sui triangolini

5. Merita ricordare che il modo normale di produrre file PDF compressi mediante `xelatex` funziona benissimo; è solo l'introduzione dei metadati come descritto nel paragrafo 8 che impedisce la compressione.

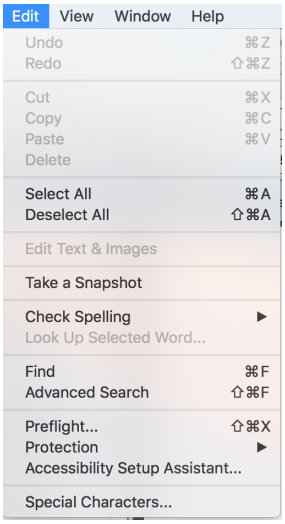


FIGURA 2: Menù a discesa della voce "Edit" dove si può selezionare il modulo `Preflight`.

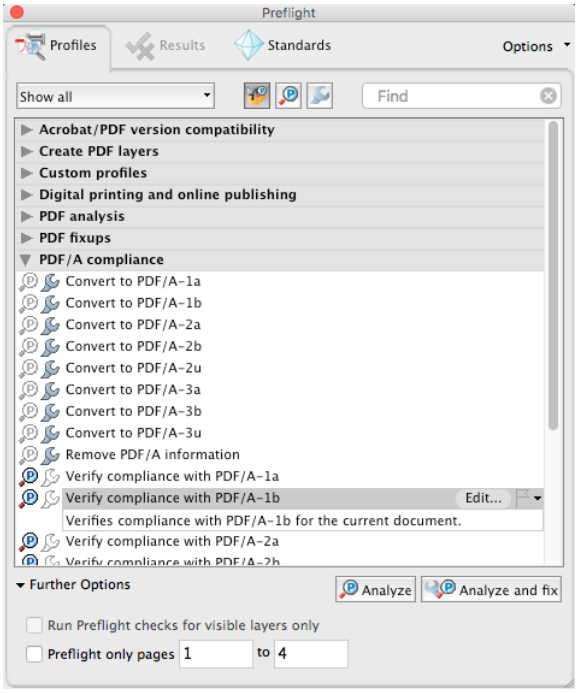


FIGURA 3: Menù delle analisi che si possono eseguire con `Preflight`.

grigi con la punta a destra si aprono e si leggono meglio le informazioni relative a ciascun errore; ammetto che l'interpretazione non è semplice, ma con un po' di pratica si riesce a fare un po' d'ordine e a capire la ragione dell'errore.

Potrebbe venire in mente di tornare alla sezione di `Preflight` mostrata nella figura 3 e di cliccare sul bottone Analyze and Fix; con questa azione `Preflight` rianalizza il file e, se ci riesce, corregge i (piccoli) errori alla sua portata. Talvolta questa azione ha successo, ma più spesso di quanto non si desideri essa fallisce; magari riesce a correggere qualche erroretto, ma non riesce a correggere gli errori importanti. La soluzione in questi casi rimane soltanto quella manuale; cercare di capire in che cosa consista ciascun errore e provvedere a eliminarlo.

Devo dire che l'esperienza che ho accumulato mi permette di affermare che la maggior parte di questi errori riguardano i font non incorporati e/o i profili di colore delle figure incluse nel testo. Se si sono seguiti i consigli indicati nei paragrafi precedenti, questi errori non dovrebbero manifestarsi, tuttavia qualche controllo sfugge all'utente, ma non sfugge a `Preflight`; l'indicazione dell'errore può essere espressa in modo criptico, ma se uno parte da una di queste due ipotesi, l'informazione di Preflight è sufficiente a localizzare l'errore per analizzarlo e correggerlo.

Se invece `Preflight` non trova errori, esso mette un'informazione del tipo di quella indicata nella figura 5; se si vuole, si può cliccare sul pulsante Create Report, e il programma provvede a comporre un documento PDF che certifica tutte le caratteristiche molto dettagliate del file appena verificato, compresa, evidentemente, la conformità e lo standard al quale il file è conforme; in sostanza compila il certificato di conformità.

13 Conclusione

I procedimenti indicati succintamente in questo articolo mostrano come si possano ottenere file PDF/A compatibili con i programmi di tipocomposizione del sistema TEX. Ammetto che l'esposizione è troppo succinta per quel che riguarda la correzione degli inconvenienti che non portano a una conversione corretta, ma ci vorrebbe un intero libro per trattare questo genere di problemi. L'esperienza conta molto e ognuno deve farsela autonomamente, perché non si può trasmettere questo tipo di esperienza attraverso lo scritto.

Devo dire, però, che con l'uso di `lualatex` mi sono sempre trovato avvantaggiato perché si eliminano alla fonte quasi tutti gli errori che dipendono dai font, dai difetti dimensionali dei glifi, e dalle codifiche inconsistenti. Il pacchetto `pdfx` fornisce correttamente i profili di colore. I pochi errori che possono rimanere sono quelli veramente subdoli, ma sono rari. Come ho detto, non userei per quanto possibile il programma `xelatex` per produrre documenti conformi all'archiviabilità, perché si può quasi sempre ottenere lo stesso risultato usando `lualatex`. Il programma `pdflatex` nonostante sia il primo con il quale il gruppo di lavoro di `pdfx` si è cimentato, dà meno sicurezze di `lualatex`.

Credo che questi progressi del sistema TEX sulla strada dell'archiviabilità vengano incontro alle necessità di ogni tipo di archiviazione, compresa quella dei documenti di tipo umanistico a cui è stata mostrata un'attenzione particolare.

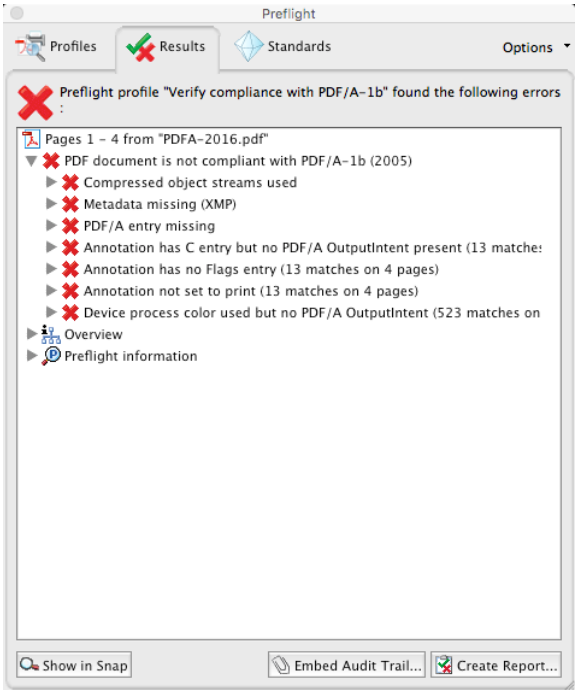


FIGURA 4: Risultato dell'analisi di `Preflight` su un file non conforme alla norma PDF/A.

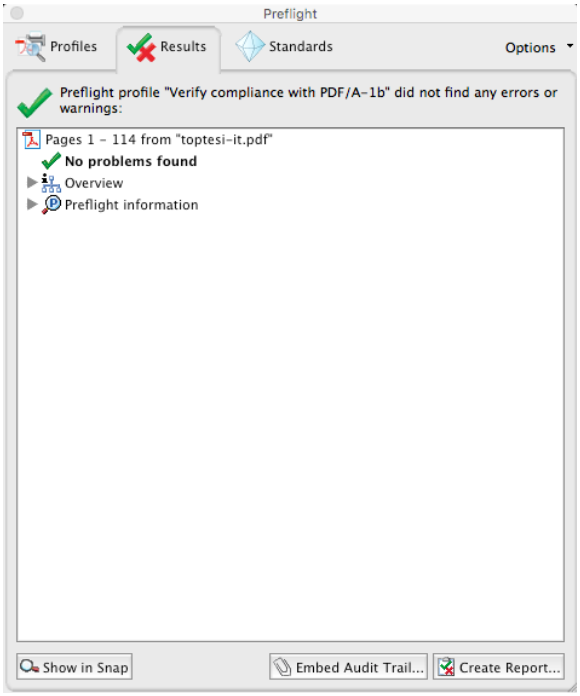


FIGURA 5: Risultato dell'analisi di `Preflight` su un file conforme alla norma PDF/A

Riferimenti bibliografici

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

BECCARI, C. (2009). «Il formato archiviabile dei file PDF». *ArsTEXnica*, (7).

DRÜMMER, O., VON SEGGERN, D. e OETTLER, A. (2007). *PDF/A in a Nutshell: Long term archiving*. Callas Software Gmbh. ISBN: 978-398116481-7.

HAGEN, H. (2016). «LuaTEX 0.90 backend changes for PDF and more». *TUGboat*, **37** (1), pp. 53–57.

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

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

HÀN THẾ THÀNH, RAHTZ, S., HAGEN, H., HENKEL, H., JACKOWSKI, P. e SCHRÖDER, M. (2007). «The pdfTEX user manual». $TEXMFDIST/doc/pdftex/manual/pdftex-a.pdf.

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

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

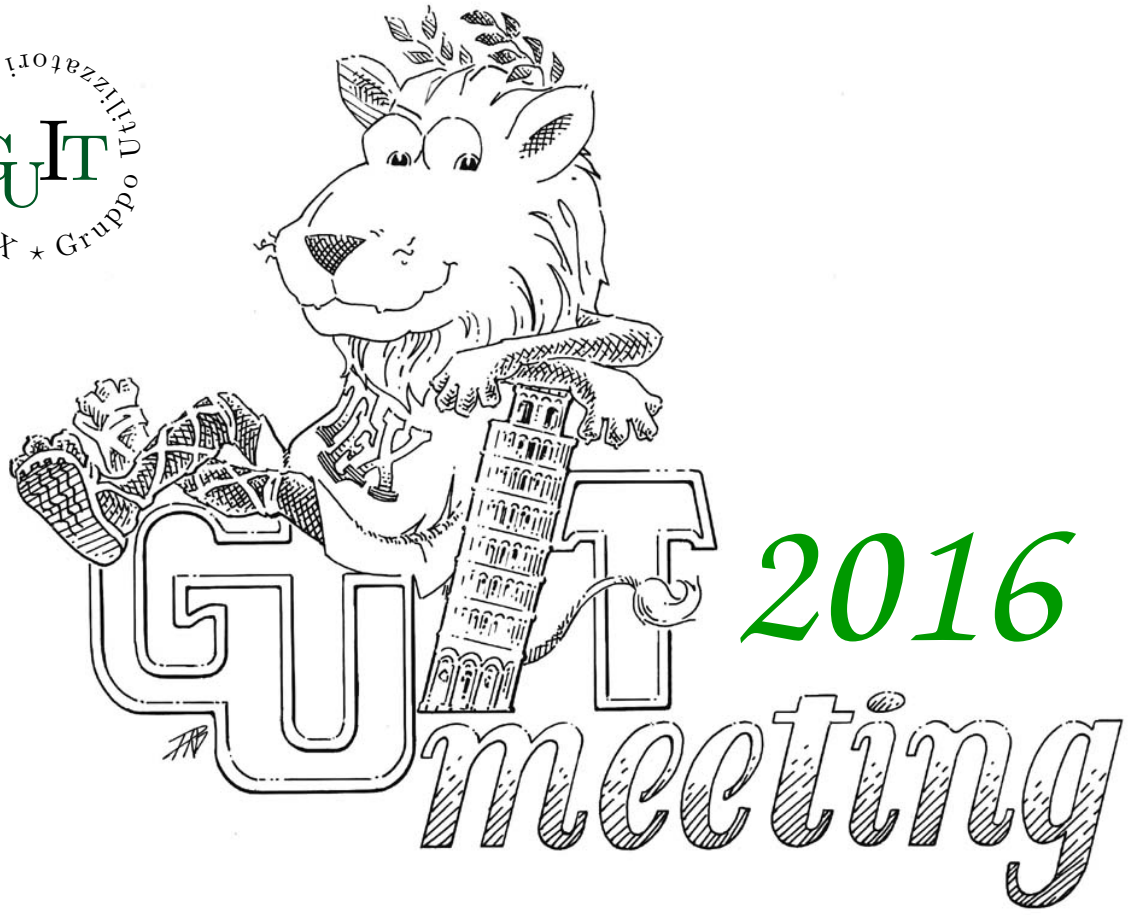
RADHAKRISHNAN, C. e HÀN THẾ THÀNH (2008). «Generation of PDF/X-1a and PDF/A-1b compliant PDF's with PDFTEX — pdfx.sty». http://tug.ctan.org/tex-archive/macros/latex/contrib/pdfx/.

RADHAKRISHNAN, C., HÀN THẾ THÀNH, MOORE, R. e SELINGER, P. (2016). «Generation of PDF/X- and PDF/A-compliant PDFs with pdfTEX – pdfx.sty». $TEXMFDIST/doc/latex/pdfx/pdfx.pdf. Version 1.5.8.

SCARSO, L. (2010). «PDF/A-1a in ConTEXt-mk4». *ArsTEXnica*, (10).

▷ Claudio Beccari
claudio dot beccari at gmail dot com

Questa rivista è stata stampata
presso Centro Stampa e Riproduzione S.r.l.,
Via di Pietralata, 157 – 00158 Roma,
su carta vellum white 80 g.
Copertina: vellum white 260 g.



Tredicesimo convegno nazionale su TEX, LATEX e tipografia digitale

Brescia, 29 ottobre 2016

Call for Papers

Sabato 29 ottobre a Brescia si terrà il tredicesimo Convegno annuale su TEX, LATEX e tipografia digitale organizzato dal Gruppo Utilizzatori Italiani di TEX. Il Convegno sarà un momento di ritrovo e di confronto per la comunità LATEX italiana, tramite una serie di interventi atti sia a contribuire all'arricchimento sia a supportarne lo sviluppo.
Maggiori informazioni sul Convegno e sulle modalità di presentazione degli interventi saranno disponibili all'indirizzo:

`http://www.guitex.org/guitmeeting/2016/`

ArsTEXnica

Rivista italiana di TEX e LATEX

Numero 22, Ottobre 2016

9 771828 236001 60022