

# Verslag officieel gedeelte

## Verslag 21<sup>e</sup> NTG-bijeenkomst, 11 juni 1998

Maarten Gelderman (mgelderman@bik.econ.vu.nl)

### **aanwezig**

Alaaddin Al-dhahir, A. Bakker, Toin Bloo, Berend de Boer, H.J. Boersma, Johannes Braams, Luc de Coninck, Wybo Dekker, Gilbert van den Dobbelssteen, Wietse Dol, Cees Fortuin, Erik Frambach, Guy Geens, Maarten Gelderman (notulen), Frans Goddijn, W.J. van der Guchte, Michael Guravage, Hans Hagen, Herman Haverkort, R. vd Heijden, J.F. Hellings, Taco Hoekwater, Gert Huisman, P. de Jong, Johan Jonker, Siep Kroonenberg, C.G. van der Laan, Wietze Lise, C.C.M. Moes, P. van Mouche, Gerard van Nes, Jeroen Nijhof, Gerrit Noordzij, Gerrit Oomen, Piet van Oostrum, Simon Pepping, Jozef Pijnenburg, Roef Ragas, Sebastian Rahtz, Hans Renkema, Ronald Rietman, R. v. Rijn, J. Rusch, Kees Serier, Rein Smedinga, W. Smit, J.H. van de Stadt, Teun Tunnissen, Johannes van der Tak, Philip Taylor, Steven Tekofsky, Chris van Uffelen, E.H.M. Ulijn, Philippe Vanoverbeeke, Henk Vink, Gea Vlak, Olaf Weber, A. de Leeuw van Weenen, Jules van Weerden, Jacoline van Welden, K.H. Wesseling, Peter van de Wijden, Jos Winnink<sup>1</sup>

### **Opening**

Wegens het feit dat de voorzitter ondanks herhaalde navraag onvindbaar blijkt, gaat deze bijeenkomst, gehouden bij de ... onder gastheerschap van Kees Fortuin enigszins chaotisch van start. In afwachting van de komst van de voorzitter laat men in eerste instantie de presentielijst rondgaan en deelt de MAPS-redactie de vergaderstukken en de nieuwe MAPS uit. Uiteindelijk neemt onder het vervangend voorzitterschap van Frans Goddijn de vergadering een aanvang.

### **Notulen van de 20<sup>e</sup> NTG-bijeenkomst**

De notulen van de vorige bijeenkomst worden goedgekeurd en onwijzigd vastgesteld.

### **Verslag van de secretaris**

Na binnenkomst van Erik Frambach, maar nog onder voorzitterschap van Frans Goddijn wordt het verslag van de secretaris zonder verdere opmerkingen vastgesteld.

### **Verslag penningmeester en kascontrolecommissie**

Het verslag van de penningmeester wordt ongewijzigd vastgesteld. Gerard van Nes complimenteert de penningmeester met het feit dat de marges van realisatie ten opzichte van het budget binnen 1% blijven. De kascontrolecommissie, bij monde van J.H. van de Stadt en mede gevormd door Kees van der Laan, merkt op dat de admini-

stratie een voorbeeld is voor vele andere verenigingen. Na afloop van de vergadering zal door de commissie een steekproef worden genomen uit de administratieve bescheiden ter afronding van de controle. Bij afwezigheid van verder bericht van de commissie mag de vereniging er vanuit gaan dat ook deze bescheiden in orde zijn bevonden.

Kees van der Laan treedt af als lid van de kascontrolecommissie. Aan de scheidend secretaris wordt verzocht de ontstane vacature op te vullen. Na instemming van Gerard van Nes wordt de nieuwe kascontrolecommissie – bestaande uit J.H. van de Stadt en Gerard van Nes – door de vergadering benoemd.

### **Bestuursverkiezingen**

Erik Frambach neemt het voorzitterschap van de vergadering over en attendeert de leden op het vertrek van de huidige secretaris (Gerard van Nes). Hij stelt voor Maarten Gelderman, conform de in de vorige vergadering door het bestuur reeds uitgesproken intentie te benoemen tot de nieuwe secretaris, waar de vergadering mee instemt.

De eerste bestuurstermijn van de voorzitter is tevens ten einde. Frambach stelt zichzelf beschikbaar voor een tweede termijn en wordt herkozen.

### **Erelidmaatschap**

Vanwege zijn grote inspanningen voor de NTG wordt voorgesteld aan Gerard van Nes een erelidmaatschap toe te kennen. Van Nes is sinds de oprichting bij de vereniging betrokken verkeerdt en heeft gedurende 10 jaar zonder mankeren de taak van secretaris uitgevoerd. Daarnaast heeft hij zich op talloze andere manieren met niet aflatend enthousiasme voor de vereniging en de promotie van T<sub>E</sub>X ingezet.

Zoals gezien de verdiensten van Gerard te verwachten viel wordt het voorstel bij acclamatie aangenomen. Gerard van Nes krijgt de bij het erelidmaatschap behorende gouden badge uitgereikt. Hiermee is het aantal ereleden van de vereniging op vier uitgekomen: Kees van der Laan, Johannes Braams, Piet van Oostrum en Gerard van Nes genieten tezamen het voorrecht van een levenslang kostenloos erelidmaatschap.

---

Dit stuk kwam mede tot stand dankzij de inspanningen van Frans Goddijn, natuurlijk blijft de onder de titel gemelde auteur verantwoordelijk voor de juistheid van de inhoud. Het zou echter niet juist zijn de credits alleen naar de officiële geheimschrijver te laten gaan.

1. Gebaseerd op handtekeningen op presentielijst.

# Van de voorzitter

Erik Frambach  
(E.H.M.Frambach@eco.rug.nl)

De NTG viert dit jaar zijn tiende verjaardag en het lijkt wel of daar een enorme impuls van uitgaat. Het gonst van de activiteiten. Op het gebied van  $\text{T}_{\text{E}}\text{X}$  & friends zijn belangrijke ontwikkelingen te melden, maar ook op verenigingsniveau gebeurt veel. Ik zal hieronder de belangrijkste zaken op een rijtje zetten.

## Nieuwe MAPS-opzet

Wellicht heb je al opgemerkt dat de MAPS dit keer wat dunner is dan gewoonlijk. Maar we zijn we ook erg verwend. De jubileum-MAPS die dit voorjaar uitkwam was maar liefst bijna 350 pagina's, en ook nog deels in kleur. Een mijlpaal in de geschiedenis van de NTG.

Van die MAPS zijn er een hoop extra gedrukt die als promotiemateriaal worden gebruikt. Op de TUG-conferentie in Toruń in Polen was er veel belangstelling voor.

Het maken van de MAPS blijft een tijdrovende bezigheid, zeker als je er echt iets nuttigs en moois van wil maken. Het bestuur van de NTG en de MAPS-redactie hebben daarom de koppen bijeen gestoken om ideeën te ontwikkelen om de MAPS-productie enerzijds zeker te stellen immers een van de peilers van de NTG), en anderzijds zo te stroomlijnen dat het geen bovenmenselijke inspanningen vergt.

Zekerstellen van MAPS-productie kunnen we realiseren door kennis en vaardigheden te delen. Ook als de hoofdredacteur (Taco Hoekwater) om wat voor reden dan ook uitvalt moet zijn taak overgenomen kunnen worden. Siep Kroonenberg is daar inmiddels toe in staat. Maar om ongelukken nog verder te voorkomen gaat de redactie ook een draaiboek opstellen voor de MAPS-productie, zodat vrijwilligers die een steentje bij willen dragen ook snel door krijgen hoe zoiets werkt.

Verder is ervoor gekozen de historische koppeling tussen 'Minutes' en 'Appendices' los te weken. In den beginne was die koppeling praktisch en logisch, maar gezien de omvang van de 'Appendices' de MAPS naar niet-NTG-leden menen we dat het nu tijd wordt die zaken te splitsen. De NTG-leden kunnen voortaan vier keer per jaar materiaal van de NTG verwachten. Kort voor de voorjaars- en najaarsbijeenkomst zal een dunne MAPS uitkomen die de 'Minutes' bevat die nodig zijn voor de vergadering; drie maanden later zal een dikkere MAPS uitkomen met artikelen en verslagen van o.a. de laatste bijeenkomst. Dat geeft degenen die een lezing hebben gegeven op een bijeenkomst

ook meer ruimte om een bijbehorend artikel te schrijven, wat weer interessant is voor degenen die de bijeenkomst niet konden bijwonen.

Deze constructie maakt het ook mogelijk beter in te spelen op actuele zaken. Verspreiding van cd-roms en ander materiaal wordt gemakkelijker. Een bijkomend voordeel is dat de 'Appendices'-MAPS interessanter wordt voor niet-NTG-leden en buitenlanders.

We hopen de NTG-leden met deze nieuwe opzet een betere dienst te kunnen leveren.

## PR-set

Tien jaar betekent ook: eens stilstaan bij de manier waarop we als gebruikersgroep naar buiten treden. Daar kan nog best wat aan verbeterd worden. Een van manieren waarop we ons presenteren is van oudsher de 'PR-set', een uitgebreide folder die uitlegt wat  $\text{T}_{\text{E}}\text{X}$  is, wat de NTG is, met wat aardige voorbeelden van wat er met  $\text{T}_{\text{E}}\text{X}$  & friends zoal voor moois te realiseren is.

Die PR-set moet nodig opgefrist worden. De MAPS-redactie heeft die taak op zich genomen. Binnenkort zal een geheel vernieuwde versie worden uitgebracht. Ook via Internet zal die beschikbaar worden gesteld.

## WWW

Ook onze www-site heeft onderhoud nodig. Daar zal ook tijd en energie in worden gestoken. Zo zal de MAPS-bibliografie helemaal bijgewerkt worden, compleet met PDF-versies van de MAPS. Verder zal er meer en betere informatie komen voor wie nog niet of nauwelijks weet wat TEKS is.

## De toekomst van $\text{T}_{\text{E}}\text{X}$

In NTG-kringen heeft zich een kern van mensen gevormd die hard aan het nadenken zijn over wat er schort aan de huidige versie aan  $\text{T}_{\text{E}}\text{X}$ , en hoe een betere  $\text{T}_{\text{E}}\text{X}$  ("T<sub>E</sub>X for the next millennium") er uit zou moeten zien. Dat levert heel interessante ideeën op. En niet alleen ideeën, ook producten. Er zijn al concrete voorstellen en prototypes. Elders in de MAPS lees je er meer over.

## Werk aan de winkel

Zoals gewoonlijk kosten al die activiteiten veel tijd en energie. Zoiets moet je dus gezamenlijk doen. Schroom daarom niet als een van de activiteiten binnen de NTG je aanspreekt: ga erop af en doe mee. Hoe meer zielen hoe meer vreugd.

# Redactioneel

Siep Kroonenberg  
Taco Hoekwater

“Is dat nou alles?” lijkt ons een voor de hand liggende reactie bij de lezers die zojuist deze laatste MAPS op de deurmat gevonden hebben. Het is nogal een dunnetje deze keer, en dat heeft een aantal elkaar beconcurrerende oorzaken.

Eén van de redenen is een beleids-beslissing van het bestuur van de NTG: de MAPS zoals-we-die-allemaal-kennen verschijnt straks op een ander tijdstip (namelijk eind februari en eind augustus), en in de overgangs-periode betekent dat dat we vóór het verschijnen van de MAPS #22 vrij weinig tijd hebben om artikelen te werven. Er zijn dus een paar dingen alvast blijven liggen voor de volgende MAPS, die weer van ‘normale’ dikte zal zijn.

Een andere reden is een beleids-beslissing van uw redactie: deadlines zijn tegenwoordig ook echte ‘dead’-lines. Alles wat te laat ten burele verschijnt wordt gewoon niet afgedrukt. Dat zelfs het bestuur geen uitzondering vormt voor deze regel blijkt uit het feit dat er in deze MAPS geen begroting staat voor 1999.

Een groot voordeel van zo’n klein MAPS-je is natuurlijk dat de productie een makkie was. Het is nu 25 september, krap een week na de deadline, en we hadden tijd over!

Even wat over de vormgeving van deze MAPS: er zijn weer wat kleine veranderingen ten opzichte van de vorige keer. We hebben een nieuw logo gekregen. Het oude was niet meer bruikbaar nu er feitelijk een ‘M’ en een ‘APS’ gaan verschijnen. Het nieuwe logo (in Palatino) is zowel simpel als doeltreffend.

De layout van de colofon hebben we omgedraaid, omdat dat beter aansluit bij de 1-koloms artikelen in de MAPS zelf. Verder zijn de problemen met de kop- en voetregels nu verholpen, dus het is gelukkig weer mogelijk om op pagina- of bijlagennummer te zoeken naar artikelen. Er is ook wat meer ruimte gekomen tussen de titels van de artikelen en de tekst.

En dan nu de inhoud. We hebben het verslag van de vorige bijeenkomst alvast gesplitst in een officieel stuk (verslag van de vergadering) en een informeel stuk (verslag van de lezingen). Voor de volgende bijeenkomst beloven we dat er ook een paar foto’s bij komen.

‘Het Weten Waard’ is flink gekrompen, onder andere omdat de informatie over andere gebruikersgroepen een

eigen artikel heeft gekregen van de hand van Erik; de ‘Agenda’ bevatte deze keer niets interessants tot ruim na het verschijnen van de volgende MAPS; en de NTG-winkel is tijdelijk gesloten wegens gebrek aan inventaris. We zoeken nog iemand om de verklarende woordenlijst aan te passen en uit te breiden.

We hebben aandacht voor de toekomst van T<sub>E</sub>X met de twee artikelen van de T<sub>E</sub>X Future Group. Deze artikelen zijn gepresenteerd tijdens TUG’98 in Toruń, waarvan we natuurlijk ook een verslag hebben.

Verder in deze MAPS aandacht voor een ruime verzameling verschillende macro-pakketten: De ‘Toolbox’ gaat dit keer over het gebruik van plain’s macros binnen L<sup>A</sup>T<sub>E</sub>X, twee artikelen van Kees van der Laan over BLUeT<sub>E</sub>X (over tabel- en minimale markup), een artikel over ConT<sub>E</sub>Xt’s eenheden-module door Hans Hagen en Ton Otten, en een artikel over JadeT<sub>E</sub>X door Sebastian Rahtz.

Van de hand van Kees van der Laan ook nog een artikel over het gebruik van Cyrillisch op een Macintosh, en als laatste ook nog een overzicht over hoe T<sub>E</sub>X past in de moderne productie-processen zoals PDF en SGML. Kees, nog van harte bedankt voor deze verzameling artikelen.

Het laatste artikel in deze MAPS is ook het enige artikel waar we het moeilijk mee hadden tijdens de productie: Het artikel van Thierry Bouche gebruikt EPS bestanden die zijn gemaakt met dvips. Bijna al deze EPS figuren zijn met behulp van GhostScript en Adobe’s Photoshop omgezet in bitmaps, omdat anders de PDF Distiller van slag raakte bij het invoegen van de gebruikte fonts.

De redactie van deze MAPS bestond uit: Siep Kroonenberg, Jos Winnink en Taco Hoekwater.

De MAPS wordt gezet met gebruikmaking van een L<sup>A</sup>T<sub>E</sub>X class file en een ConT<sub>E</sub>Xt module.

De gebruikte fonts zijn Adobe Times-Roman en Frutiger, met een versmalde Courier voor de ‘verbatim’ omgevingen. Wiskundige formules worden gezet met de MathTime fonts van Y&Y.

Compilatie gebeurt met web2c versie 7.2 onder Windows95, en we drukken van een PDF bestand dat wordt gemaakt met dvips versie 5.78 en Adobe’s Distiller 3.01, op 70-grams coated papier.

Het versturen gebeurt deels met de ptt partijpost (Bastiaan wordt weer bedankt voor het stempelen en sjouwen), deels door de internationale bezorging van Kluwer Academic Publishers (waarvoor onze hartelijke dank).

# Het Weten Waard

## Diverse afkortingen

*CTAN* – Comprehensive T<sub>E</sub>X Archive Network; sites waar men ‘anonymous ftp’ kan gebruiken om T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X-achtig materiaal te verkrijgen. CTAN is de ‘home’ voor de officiële versie van L<sup>A</sup>T<sub>E</sub>X etc. CTAN sites zijn: ftp.dante.de, ftp.tex.ac.uk en ftp.cs.tug.org

*FGBBS* – NTG’s Bulletin Board

*AllT<sub>E</sub>X* – T<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X, etc T<sub>E</sub>X

*ltxiii* – L<sup>A</sup>T<sub>E</sub>X 3.0

*4T<sub>E</sub>X* – Het volledige T<sub>E</sub>X runtime systeem voor MS-DOS PC systeem, gebaseerd op emT<sub>E</sub>X en 4DOS.

*4allT<sub>E</sub>X* – De 4T<sub>E</sub>X applicatie plus alle mogelijke gerelateerde files en utilities, gedistribueerd op CD.

*AMS* – American Mathematical Society

*SGML* – Standard Generalized Markup Language

## NTG’s T<sub>E</sub>X Bulletin Board Systeem

Op het T<sub>E</sub>X Bulletin Board van de Nederlandstalige T<sub>E</sub>X Gebruikersgroep (FGBBS) is een zo volledig en actueel mogelijke T<sub>E</sub>X, emT<sub>E</sub>X, L<sup>A</sup>T<sub>E</sub>X, TEX-NL en MusicT<sub>E</sub>X collectie beschikbaar voor alle bezitters van een modem. Het BBS is kosteloos toegankelijk voor iedereen en er zijn geen beperkingen aan de hoeveelheid bestanden die kunnen worden opgevraagd. Het systeem is aangesloten op een modem die zowel ISDN als lagere snelheden aankan.

De beheerders zijn Frans Goddijn en Henk de Haan. FGBBS is te bellen op 026-3217041.

## NTG/TUG lidmaatschap

Het blijkt soms dat nieuwe NTG/TUG leden na ongeveer een half jaar nog geen TUGboat of TTN van TUG hebben ontvangen. Ondanks dat men een TUG lidmaatschap via NTG aanvraagt, blijkt in bijna alle gevallen de administratie- en verzendproblemen bij TUG zelf te liggen. Mocht na enige maanden tijd geen post van TUG ontvangen worden, dan worden de betreffende NTG/TUG leden dringend verzocht om contact op te nemen met het secretariaat van de NTG.

## MAPS 99.1

Sluitingsdata voor het inleveren van artikelen, bijlagen, en/of mededelingen voor de volgende MAPS uitgaven zijn:

*15 januari '99 (MAPS 99.1; #22)*

*1 juli '99 (MAPS 99.2; #23)*

De voorjaars-MAPS verschijnt op 1 maart, de najaars-MAPS op 1 september.

Aanleveren kopij voor de komende MAPS:

- *Bij voorkeur* in gebruikmakend van de L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> class file maps.cls of de ConT<sub>E</sub>Xt module m-map-01. Beide files zijn via de redactie te verkrijgen en beschikbaar op de TEX-NL fileserver, archive.cs.ruu.nl (ftp-site) en FGBBS (026-3217041).
- Daarnaast kunnen bijdragen ingestuurd worden gemaakt met ltugboat.sty of article.sty / report.sty.
- Verder zijn bijdragen vanzelfsprekend ook welkom in *plain-T<sub>E</sub>X* of ongeformatteerd.
- Plaatjes bij voorkeur als (Encapsulated) PostScript file plus het oorspronkelijke formaat; dit laatste om eventuele problemen beter te kunnen oplossen.

Daar MAPS bijdragen in *plain T<sub>E</sub>X* altijd worden omgezet naar L<sup>A</sup>T<sub>E</sub>X of ConT<sub>E</sub>Xt, verdient vanzelfsprekend aanbieding van materiaal in een van deze formaten de voorkeur.

Eventuele nadere richtlijnen voor auteurs zijn op te vragen bij de redactie.

Bijdragen kunnen gestuurd worden naar:

Taco Hoekwater,  
Singel 191  
3311 PD Dordrecht  
Email: taco.hoekwater@wkap.nl

Personen met een modem maar zonder internetaansluiting kunnen hun bijdrage ook via modem/PTT lijn naar de redactie sturen. Gaarne hiervoor eerst contact opnemen met Taco Hoekwater, tel. 078-6137806.

# Verslag niet-officieel gedeelte

## Verslag 21<sup>e</sup> NTG-bijeenkomst, 11 juni 1998

Maarten Gelderman (mgelderman@bik.econ.vu.nl)

Met de verkiezing van Gerard van Nes tot erelid is het officiële gedeelte van de vergadering ten einde. Namens de vereniging krijgt Gerard nu een kado als dank voor zijn jarenlange trouwe dienst: een zweefvliegles. Vervolgens is het tijd voor koffie. Tijdens de koffie wordt door het 4AllT<sub>E</sub>X-team aan de NTG en daarmee aan alle aanwezigen een taart aangeboden.

### Het echte handwerk (Gerrit Oomen)

De eerste ‘officiële’ activiteit tijdens het niet-officiële gedeelte van de bijeenkomst (waarvoor we per slot van rekening eigenlijk allemaal naar Arnhem zijn gekomen) is een lezing door Gerrit Oomen. Gerrit is bij de in-crowd van de NTG met name bekend als de baas van Taco (de hoofdredacteur van de MAPS). Nu krijgt hij de kans om voor zichzelf te spreken en dat blijkt de moeite waard te zijn. Gerrit dateert nog uit de tijd van het lood en verhaalt op boeiende wijze en met vele plaatjes over de wijze waarop nog maar een paar decennia geleden teksten uit lood werden gezet. We horen alles over trucs met toetsenborden van zetmachines, het handmatig nawerken van kopij en het gratis verstrekken van melk aan loodzetters teneinde vergiftigingsverschijnselen te komen. De komst van T<sub>E</sub>X blijkt de wereld verlost te hebben van een ongezond en naar mijn inschatting zenuwslopend beroep.

### Typografie (Gerrit Noordzij)

De goden waren de NTG op haar lustrum bijeenkomst niet gunstig gezind. Waar eerst de voorzitter verstek liet gaan bleek vervolgens de tweede spreker onvindbaar. Naar later bleek stond deze op de gang te praten met de secretaris in afwachting van het aflopen van de lezing van Oomen, waarna oorspronkelijk immers een koffiepauze was gepland. De lezing bleek echter alleszins de moeite van het wachten waard. Met de uitspraak ‘Ik behoor al 40 jaar tot de avant-garde’, een verwijzing naar de manier waarop het Holland Festival een Amerikaans toneelgezelschap introduceerde, plaatste de spreker zichzelf even onbescheiden als humoristisch in de typografische geschiedenis. Hierna voerde Noordzij de zaal mee naar de wereld van de typografische wetten, regelspatiëring, de onhebbelijke eigenschappen van Tschichold en het ontwerp van letters. Toen hij zich omdraaide en met een half krijtje even achteloos

als meesterlijk een aantal letters op het schoolbord zette, oogste hij een spontaan applaus uit de zaal.

De spreker was goed voor meer prikkelende visies dan zijn gehoor kon noteren. Zo zette hij uiteen dat de drukpers *technologisch* helemaal niets nieuws was. De vervaardiging van gegoten klokken en munten was al ouder en van een technologisch eender principe. Het bedrukken van textiel was er ook al lang. Nieuw was het *papier* als alternatief voor het kostbare perkament. De typografie bestond al en bestaat nog steeds, ondanks het verdwijnen van de matrijs in het drukproces.

Over voetnoten: ‘die zijn voor negen/tiende onnodig en voor tien/tiende schadelijk.’ Noordzij eerde met talrijke anecdoten zijn vriend en rivaal Jan Tschichold, de typograaf die de vormgeving van de bekende Penguin-reeks bepaalde en publiceerde over vorm en verhouding van boek en bladspiegel. Typografie, vond Noordzij, is de moeite waard om over te strijden en ‘Tschichold en ik waren het zo verschikkelijk oneens dat er een onverbreekelijk hechte vriendschap uit is ontstaan’. Tschichold had voor een van zijn lezingen eens uitdrukkelijk aangegeven dat ‘eine Diskussion findet nicht statt’, toch waren er twistpunten te over waar beide grote ego’s zich hartstochtelijk aan overgaven.

‘Ik kan deze lezing gemakkelijk in 25 minuten geven, maar er is geen bezwaar tegen uitweidingen’, sprak Noordzij halverwege zijn betoog. En zo was het ook. Het was zoals Vladimir Nabokov na de openingsregels van zijn roman ‘Laughter in the Dark’ vaststelde: ‘This is the whole of the story and we might have left it at that had there not been profit and pleasure in the telling; and although there is plenty of space on a gravestone to contain, bound in moss, the abridged version of a man’s life, detail is always welcome.’

### 10 jaar ntg

Kort schetste Erik Fambach de geschiedenis van de NTG die in MAPS 20 uitgebreider door hem is beschreven.

### How T<sub>E</sub>X does Arabics (Klaus Lagally)

Na de geschiedenis van de NTG kwam de geschiedenis van het Arabisch aan het bod. Na de zaal eerst lekker te heb-

---

Dit stuk kwam mede tot stand dankzij de inspanningen van Frans Goddijn, natuurlijk blijft de onder de titel gemelde auteur verantwoordelijkheid voor de juistheid van de inhoud. Het zou echter niet juist zijn de credits alleen naar de officiële geheimschrijver te laten gaan.

ben gemaakt met het tonen van prachtige stukjes Arabische typografie, waarin alle letters van een woord of zin tot één geheel lijken samen te smelten, werden de deelnemers wakkergeschud met kopiën van recente kranten. De gebrekkige mogelijkheden die de redacteurs hier tot hun beschikking staan leiden tot afschrikwekkende typografie die slechts hier en daar in de vorm van een handgeschreven kop of advertentie een klein lichtpuntje toont. Een lichtpuntje werd getoond in de vorm van het werk van Lagally zelf dat het dan wel niet mogelijk maakt de vaardigheden van de ouderwetse typografen aan een willekeurige computeraar ter beschikking te stellen, maar in ieder geval de mogelijkheid biedt tot het maken van in de ogen van een leek prachtige ‘plaatjes’. Zie het vorige nummer van MAPS voor verdere toelichting.

### Typografie in T<sub>E</sub>X (Phil Taylor)

Ook het verhaal van Phil Taylor hoeven de afwezigen niet te missen. Dit is onder de titel ‘Book Design for T<sub>E</sub>X Users’ te vinden in MAPS #19. Wat zij wel hebben moeten missen was de inspirerende presentatie van dit verhaal, waarin de puntjes nog eens op de 1 werden gezet. In een uitvoerige dialezing werden de door Noordzij net zo verfoeide wetten van de typografie uiteengezet.

### T<sub>E</sub>X en www (Sebastian Rahtz)

Aan Sebastian Rahtz, T<sub>E</sub>X-tovenaar van Elsevier en samensteller van de T<sub>E</sub>XLive CD, was de ondankbare taak om de aandacht van een murw geprate zaal net voor de borrel vast te houden. We hoeven er niet aan te twijfelen dat hij hier uitstekend in is geslaagd. Benadrukkend dat hij niet

namens Elsevier praatte liet Sebastian een aantal nieuwe (op hyperref gebaseerde) producten zien en toonde hoe het mogelijk was uit één enkel document niet alleen DVI en PS, maar ook PDF en HTML te genereren. Een indrukwekkende prestatie.

Ook Sebastian inspireerde met bondige uitspraken die bijna literair spitsvondig waren. Zoals de ‘puncline’ van zijn beschouwing over de zwakte van T<sub>E</sub>X waar regels uit de maat gaan lopen zodra er een verticale witruimte van enige rek verschijnt. Een lelijke trek van T<sub>E</sub>X die je niet vindt bij fraaie boeken. ‘Negative leading is possible, but can lead to rooftop lines – hard to read but waterproof.’

Phil gaf als commentaar op de volgende oplossing:

```
\parskip = 0.8ex minus -0.2ex
(dus zonder PLUS...)
```

‘Possible, but I’d advise against it.’ We mochten wat hem betreft de tip doorvertellen ‘but if you run into trouble, don’t come to me for help! T<sub>E</sub>X listens to you (plus n, minus n) but fails to obey you, even though it issues warnings that something *bad* is happening...’

Bij de geringste waarde voor plus gaat T<sub>E</sub>X immers tot in het oneindige oprekken, ook al heb je dat nooit gewild, en T<sub>E</sub>X geeft alleen in de logfile en op het scherm aan met badness meldingen dat er iets verkeerd gaat.

Sebastian toonde ons echter ook zijn zorgen over de toekomst van T<sub>E</sub>X. Naar zijn mening is een aantal essentiële veranderingen aan T<sub>E</sub>X noodzakelijk om het bij voorbeeld geheel zonder supervisie van een gebruiker tekst te laten zetten. Het mag duidelijk zijn dat het er tijdens de op deze lezing volgende borrel verhit aan toe ging.

# De NTG en het Internet

Jules van Weerden  
email: Jules.vanWeerden@let.uu.nl  
mmv Maarten Gelderman  
email: mgelderman@bik.econ.vu.nl

## abstract

De NTG is niet alleen met webpagina's op het Internet aanwezig, maar ook middels een aantal discussielijsten. In dit artikel wordt kort aangegeven wat een discussielijst is, hoe je je aan- en afmeldt en wat je te wachten staat na aanmelding.

Tevens wordt een overzicht gegeven van de in Nederland aanwezige T<sub>E</sub>X-gerelateerde lijsten.

## keywords

discussielijsten, tex-nl

Het bekendste onderdeel van het Internet is ongetwijfeld het World Wide Web. Elders in deze MAPS staat een bijdrage van Piet van Oostrum over de web-pagina's van de NTG. In deze bijdrage wil ik het hebben over een onderdeel van Internet dat vaak, te vaak, over het hoofd wordt gezien: discussielijsten. Eerst zal ik globaal een beeld schetsen van wat zo'n lijst is en hoe een discussielijst werkt. Vervolgens kom ik toe aan het bespreken van de Nederlandse T<sub>E</sub>X-gerelateerde discussielijsten.

## Discussielijsten

Veel mensen zijn zich niet bewust van het feit dat er discussielijsten bestaan. De reden hiervoor is eenvoudig. In tegenstelling tot bij voorbeeld het World Wide Web of ftp heb je voor het gebruik van discussielijsten geen aparte software nodig. Een email-programma (en natuurlijk toegang tot Internet) volstaan. De werking van een discussielijst zelf is eigenlijk te simpel voor woorden. Een discussielijst is voor de gebruiker niets anders dan een email-adres, bij voorbeeld `discussielijst@lijstenccomputer.nl`. Alle mail die naar dit email-adres wordt gestuurd wordt automatisch doorgestuurd naar alle leden van de discussielijst. Het email-adres van de belangrijkste lijst in Nederland, `tex-nl@nic.surfnet.nl`, is bij voorbeeld `tex-nl@nic.surfnet.nl`. Alle mail die naar dit adres wordt gestuurd komt bij alle 183 leden aan.<sup>1</sup> Op de lijstcomputer draait een softwarepakket (listserv en Majordomo zijn de populairste pakketten) dat er voor zorgt dat dit automatisch gebeurt. De op deze wijze gecreëerde lijst kan gebruikt

worden om over van alles en nog wat te discussiëren. De T<sub>E</sub>X-lijsten hebben veelal een vraag- en antwoordkarakter. De leden van deze lijsten stellen op de lijst vragen over T<sub>E</sub>X-gerelateerde problemen en de andere leden (en met name Piet van Oostrum die hiermee terecht een erelidmaatschap van de NTG heeft verdiend) proberen deze vragen te beantwoorden.

Tot zover erg aardig natuurlijk, maar dan moet een lijst wel leden hebben. Ook dit proces is geautomatiseerd. Naast het adres `discussielijst@lijstenccomputer.nl` bestaat er een adres waarmee het programma direct benaderd kan worden, bij voorbeeld `listserv@lijstenccomputer.nl`. Het is niet de bedoeling dat er naar dit adres gewone berichten worden gestuurd, daar kan de software niets mee. In plaats daarvan moet de mail korte commando's bevatten. Je kan bij voorbeeld een mailtje met als inhoud

```
HELP  
END
```

versturen naar `listserv@nic.surfnet.nl`, je krijgt dan een mailtje met uitleg over de beschikbare commando's terug.<sup>2</sup> Om je aan te melden als lid van bij voorbeeld `tex-nl`, kan je het volgende mailtje versturen naar `listserv@nic.surfnet.nl`:

```
SUBSCRIBE TEX-NL Voornaam Achternaam  
END
```

De listserv-software kan vervolgens drie dingen doen.

1. Een mailtje terugsturen met de mededeling dat je geen lid kunt worden. Bij `tex-nl` is het uiterst onwaarschijnlijk dat dit gebeurt, maar bij andere lijsten (b.v. de bestuurslijst van de NTG) kan dit voorkomen.
2. Een mailtje terugsturen met de mededeling dat je lid bent geworden van de lijst. In dit mailtje wordt tevens uitgelegd hoe je je weer af kunt melden. Bewaar het!

---

1. Dit is niet helemaal waar. Indien het mailtje door één van de 183 leden wordt verstuurd, komt het normaal gesproken alleen bij de overige leden aan. Wil je het ook zelf weer terughebben stuur een melding aan het beheersadres (zie verder) met als commando `'repro <lijstnaam>'`. Op sommige lijsten is het bovendien alleen aan leden van de lijst of soms zelfs alleen aan een lijstbeheerder toegestaan om mail te verzenden.

2. Helaas zijn de commando's voor listserv en Majordomo niet identiek.

3. Een mailtje terugsturen met de mededeling dat je voor de zekerheid nog even moet bevestigen dat je echt lid wilt worden van de lijst. Dit is met name om te voorkomen dat ongeldige emailadressen aan een discussielijst worden toegevoegd en om te zorgen dat mensen niet ongewenst aan een lijst worden toegevoegd.

Na deze eenvoudig actie ben je lid van tex-nl en kan je mail (met vragen over  $\TeX$  en  $\LaTeX$  naar deze lijst versturen. Tevens ontvang je vanzelfsprekend alle mail die anderen naar deze lijst zenden (gemiddeld zo'n 25 mailtjes per week). Om je weer af te melden stuur je een mailtje met het commando UNSUBSCRIBE TEX-NL naar de listserver.

## Lijsten in Nederland

Ook op het gebied van de elektronische berichtenuitwisseling gaat het goed met de NTG. Op dit moment is een zevental lijsten geïnitieerd vanuit Nederland (voor zover ik weet ;-).

**tex-nl** Algemene discussie lijst over  $\TeX$  in Nederland. [aantal abonnees: 183].

**4tex** Lijst voor vragen en mededelingen betreffende het 4TeX-systeem van Erik Frambach en Wietse Dol. [aantal abonnees: 275].

**ntg-sgml** Discussielijst van de SGML-werkgroep binnen de NTG. [aantal abonnees: 9].

**ntg-tex-tools** Vragen-, antwoorden- en opmerkingenlijst over allerlei programmatuur die gebruikt kan worden om de resultaten van  $\TeX$  nog beter te maken. Of zelfs om  $\TeX$  te genereren. [aantal abonnees: 27].

**ntg-context** Discussie lijst over ConTeXt, een parameter-gestuurd  $\TeX$  macro pakket. [aantal abonnees: 11].

**ntg-ppchtex** Deze lijst gaat over PPCHTeX, een  $\TeX$  macro pakket dat gebruikt kan worden om chemische structuurformules te zetten. [aantal abonnees: 14].

**ntg-toekomsttex** Discussielijst over de toekomst van  $\TeX$ . [aantal abonnees: 7].

De eerste twee lijsten zijn te gast op de email-server LISTSERV van SURFNET. De lijsten die beginnen met 'ntg-' zijn te gast bij de MajorDomo email-server van de faculteit der Letteren in Utrecht.

Verder zouden nog de lijsten genoemd kunnen worden:

**teTeX** -lijst, waarop gediscussieerd wordt over het totaalpakket dat Thomas Esser heeft samengesteld van de basis-programmatuur die je nodig hebt om  $\TeX$  op UNIX te draaien. Is de basis van de  $\TeX$  -live CD. [aantal abonnees: 668].

**CTAN-Ann** - De beheerder(s) van CTAN gebruiken deze lijst voor de aankondigingen van nieuwe bestanden op CTAN. Handig om op de hoogte te blijven van het uitkomen van de nieuwste macro's en programma's. Je kunt er zelf (dus) niet posten.

Op alle lijsten wordt levendig gediscussieerd. Voor de lijsten tex-nl, 4TEX en teTeX houd ik (JvW) zelf een archief bij en omdat het niet erg geheim is, kan iedereen daar ook bij via de ftp-server ftp.let.uu.nl in de directory: /pub/tex/<lijst> [lijstnamen in kleine letters].

Let verder op het verschil tussen de LISTSERV- en de MajorDomo-software. Voor de eerste kun je op de regel 'subscribe <lijstnaam>' ook nog een willekeurige tekst toevoegen. Bv

```
subscribe tex-nl Jules van Weerden, CIM, Fac \
                                der Letteren, UU, NL
```

De vrije tekst wordt als commentaar in de verzendlijst opgenomen en door de programmatuur genegeerd. Bij MajorDomo mag die extra informatie er NIET staan. Die wordt nl beschouwd als het email adres waar de berichten heen moeten.

Als je je voor een lijst wil aanmelden moet je een email-bericht sturen aan het bijbehorende adres. Het onderwerp is niet echt van belang, maar je krijgt het in de reactie terug, dus je kunt het er aan herkennen. In de tekst van het bericht neem je de regel op: subscribe <lijst>.

De verschillende aanmeld-adressen:

- Voor tex-nl en 4TEX: LISTSERV@nic.surfnet.nl.
- Voor ntg-tex-tools, ntg-ppchtex, ntg-context en ntg-toekomsttex: Majordomo@ntg.nl.
- voor teTeX: Majordomo@informatik.uni-hannover.de
- voor CTAN-Ann: listserv@urz.uni-heidelberg.de

De reden dat de lijsten met 'ntg-' beginnen is dat we (nog) geen scheiding hebben aangebracht tussen de lijsten van de faculteit der Letteren en de lijsten van de NTG. Hopelijk kunnen we binnenkort alles splitsen. Dan krijg je ook de reactie terug van MajorDomo@ntg.nl en niet zoals nu van MajorDomo@let.uu.nl.

Voor alle lijsten geldt dat als je een probleem met de lijst hebt dat je een bericht kunt sturen aan owner-<lijst>@<email-server> waar een 'echt' persoon achter zit, die kan helpen met het oplossen van het probleem.

Geniet van de lijsten en bestrijd de SPAM (ongewenste reclame).



# TeX user groups around the world

Erik Frambach  
email: E.H.M.Frambach@eco.rug.nl

All over the world TeX users have formed networks of *user groups* on an informal basis. These user groups consist of enthusiastic TeX users who share their problems and solutions with anyone who wants to join the TeX community. They usually communicate through e-mail and often produce printed periodicals with contributions from members.

The periodicals and discussions on e-mail are essential for users who want to be informed about the latest developments. E-mail is important for getting information on, e.g. what is the newest version of program *x*, where do I find a macro for problem *y*, how do I install program *z*. For this purpose, a list of frequently asked questions ('FAQ') is maintained, and is distributed regularly.

Below we have listed all TeX user groups currently known to us.

**AsTeX** (French-speaking)  
Michel Lavaud, President  
Association pour la diffusion de logiciels scientifiques lies à TeX Association AsTeX  
BP 6532  
45066 Orleans cedex 2  
France  
tel: 33 2 38 64 09 94  
e-mail: astex-admin@univ-orleans.fr  
discussion list astex@univ-orleans.fr

**CsTUG** (Czech and Slovak Republics)  
Petr Sojka, President  
Československé sdružení uživatelů TeXu  
CsTUG, c/o FI MU  
Botanická 68a  
CZ-602 00 Brno  
Czech Republic  
fax: +420-5-755896  
e-mail: cstug@cstug.cz  
WWW page: <http://www.cstug.cz/>  
periodical: *Zpravodaj CSTUG*

**CyrTUG** (Russia)  
Irina Makhovaya, Executive Director  
Associaciia Pol'zovatelej Kirillicheskogo TeX'a

Najaar 1998


## CSTUG - Czechoslovak TeX Users Group - Československé sdružení uživatelů TeXu

"Poslaním sdružení je vytváření předpokladů pro všestranné využívání a další rozvoj jazyka počítačové typografie TeX a příbuzného programového vybavení pro stolní tisk, zejména mezi českými a slovenskými uživateli." (Stanovy sdružení/ CSTUG bylaws: [Postscript](#), [PDF](#)).

- Bulletin of CSTUG/ Sdružení vydává časopis [Zpravodaj](#).
- CSTUG membership form/ Chcete-li se stát členy sdružení (členství je buď individuální nebo kolektivní), staci vyplnit elektronicky [přihlášku](#) nebo namapiste
- Zapis ze schuze vyboru CSTUGu 6.10.97 v Brne [Postscript](#), [pdf](#)
- Zapis z valne hromady CSTUGu 12.10.97 v Brne [textove bez hacku](#), [Postscript](#), [pdf](#)
- CSTUG electronic discussion list archive (searchable)/ [Archiv diskusni skupiny cstex@cs.felk.cvut.cz](#) (prohledavatelny)
- [contact addresses/kontaktni adresy](#)
- <sup>NEW</sup> knihy o TeXu nabizene sdruzenim [Postscript](#), [PDF](#)
- [CSTeX basic info/ zakladni informace](#), [latest version/ posledni verze ke stazeni](#)
- [CSTUG FAQ - casto kladene otazky a odpovedi](#)

CyrTUG

Ассоциация пользователей  
кириллического TeX'a  
(Cyrillic TeX Users Group)



English version [here](#)  
 Russian version (Win 1251) [here](#) ([ААаа ааААаааа](#))  
 Russian version (K.O18-R) [here](#) ([+АУА ЁІАФОІ•ЁА](#))

000234 visitors since January 9, 1998

Please send comments to [cyr tug@cemi.rssi.ru](mailto:cyr tug@cemi.rssi.ru)

Mir Publishers  
2, Pervyj Rizhskij Pereulok  
Moscow 129820  
Russia  
tel: +7 95 286 0622, 286-1777  
fax: +7 95 288 9522  
e-mail: [cyr tug@cemi.rssi.ru](mailto:cyr tug@cemi.rssi.ru)  
WWW page: <http://www.cemi.rssi.ru/cyr tug/>

**Dante e.V.** (German-speaking)  
Marion Neubauer, President  
Deutschsprachige Anwendervereinigung  
TeX e.V.

## Willkommen beim WWW-Server der DANTE e.V.

You can find a short [english introduction here](#).

### DANTE e.V. - kurz vorgestellt

DANTE, Deutschsprachige Anwendervereinigung TeX e.V., wurde am 14. April 1989 in Heidelberg gegründet. Der Zweck des Vereins ist die Betreuung und Beratung von TeX-Benutzern im gesamten deutschsprachigen Raum. Dazu gehört die Beratung sowohl mittels *electronic mail* als auch gelber Post, was die Anschaffung, Implementation und Anwendungsprobleme von TeX angeht, die Verteilung von Software an Mitglieder und die Informationsübermittlung dessen, was in der TeX-Welt geschieht. Außerdem werden Entwicklungen im Bereich von TeX, LaTeX, Metafont, BibTeX... national und international initiiert, gefördert und koordiniert.



Polish T<sub>E</sub>X Users Group

## Welcome to the Greek TeX's Friends Group's Home Page!



# GUTenberg

L'association GUTenberg

Le Groupe Francophone des Utilisateurs de TeX est une association de loi 1901 qui a pour but de promouvoir l'utilisation de TeX dans les pays francophones, ainsi que d'offrir à ses adhérents un ensemble de services aidant à la connaissance et à l'utilisation de TeX et de son environnement.

Ce site a pour but de vous permettre d'accéder à l'ensemble des [services offerts](#) par GUTenberg, d'obtenir des [informations](#) sur l'association, ainsi que de nous [contacter](#).

Postfach 101840  
D-69008 Heidelberg  
Germany  
tel: +49 6221 29766  
fax: +49 6221 167906  
e-mail: [dante@dante.de](mailto:dante@dante.de)  
WWW page: <http://www.dante.de/>  
periodical: *Die TeXnische Komödie*

### Estonian User Group

Enn Saar, Tartu  
Astrophysical Observatory, Toravere  
EE 2444 Estonia  
e-mail: [saar@aai.ee](mailto:saar@aai.ee)

### Greek TeX Friends Group (Greek speaking)

Apostolos Syropoulos, President  
366, 28th October Str.  
GR-671 00 Xanthi  
Greece  
tel: +30 541 28704  
e-mail: [apostolo@platon.ee.duth.gr](mailto:apostolo@platon.ee.duth.gr)  
WWW page: <http://obelix.ee.duth.gr/eft/>

### GUST (Poland)

Tomasz Przechlewski (President)

Polska Grupa Użytkowników Systemu TeX  
Instytut Matematyki Uniwersytetu  
Gdańskiego  
ul. Wita Stwosza 57  
80 - 952 Gdańsk  
Poland  
e-mail: [ekotp@univ.gda.pl](mailto:ekotp@univ.gda.pl)  
WWW page: <http://www.gust.org.pl/>  
periodical: *GUST Bulletin*

### GUTenberg (French-speaking)

Michel Goossens, President  
Groupe francophone des Utilisateurs  
de TeX  
Association GUTenberg  
BP 10  
F-93220 Gagny principal  
France  
tel: +33 1 44 32 37 96  
fax: +33 1 44 32 20 80  
e-mail: [gut@irisa.fr](mailto:gut@irisa.fr)  
WWW page: <http://www.ens.fr/gut/>  
periodical: *Les Cahier GUTenberg*

### GUTH (Grupo de Usuarios de TeX Hispanoparlantes)

No formal user group yet.  
Public mailing list: [spanish-tex@eunet.es](mailto:spanish-tex@eunet.es) (Send

subscription requests to this list).

Julio Sanchez  
 GMV SA  
 Isaac Newton 11  
 PTM Tres Cantos  
 E-28760 Madrid  
 Spain  
 e-mail: jsanchez@gmv.es  
 WWW page:  
<http://gordo.us.es/Actividades/GUTH/>

#### ITALIC (Irish)

No formal user group yet.  
 Public mailing list: ITALIC-L@irlearn.ucd.ie (send  
 subscription requests to  
[listserv@irlearn.ucd.ie](mailto:listserv@irlearn.ucd.ie)).

Peter Flynn  
 Computer Centre  
 University College Cork  
 Ireland  
 e-mail: pflynn@www.ucc.ie

#### iT<sub>E</sub>Xnici (Italian)

(Unofficial) Italian T<sub>E</sub>X Users Group  
 Giovanni Pensa  
 e-mail: [pensa@dsi.unimi.it](mailto:pensa@dsi.unimi.it)

#### JTUG (Japan)

Nobuo Saitoh, Chairman  
 Japan T<sub>E</sub>X Users' Group  
 Faculty of Environmental Information  
 Keio University  
 5322 Endo, Fujisawa-shi  
 JP-252 Japan  
 tel: +81 466 47 5111  
 e-mail: [ns@keio.ac.jp](mailto:ns@keio.ac.jp)

#### Lithuanian T<sub>E</sub>X Users Group

Vytas Statulevičius, Chair  
 Akademijos 4  
 LT-2600 Vilnius  
 Lithuania  
 tel: +370 2 359 609  
 fax: +370 2 359 804  
 e-mail: [statulevicius@mii.lt](mailto:statulevicius@mii.lt)  
 WWW page: <http://www.vtex.lt/tex/>

#### Nordic T<sub>E</sub>X Users Group (Scandinavian countries)

Dag Langmyhr, Chair  
 Nordic T<sub>E</sub>X Users Group  
 Department of Informatics  
 PO Box 1080 Blindern

Najaar 1998

#### Lietuvos T<sub>E</sub>X'o Vartotojų Grupė

##### Lithuanian TeX Users Group


##### Diame lape:

##### ☛ Lietuviškas TeX / Lithuanian TeX

Eia surasite [Sigitto Tohušo](#) ir [Vytio Statulevičiaus](#) paruoštą programė rinkinį darbui su TeX sistema lietuviškai:

- Virtualūs lietuviški šriftai (paremti standartiniais šriftais DC kodavime);
- Lietuviškas skiemėnavimas TeX sistemai. Kuriant skiemėnavimo lentelę neįkainojamą pagalbą suteikė TeX burtininkas (wizard) [Yannis Haralambous](#) ir Scandinavian TeX Users Group
- Draiveriai, leidžiantys naudoti ávairias lietuviškas þenkles kodavimo lenteles
- Latex 209, Latex 2e komandos darbui su lietuviškais tekstais

##### ☛ Lietuviški TeX dokumentai, instrukcijos

• [Vytas Statulevičius](#). LaTeX2e: Matematiniai simboliai ir šriftai (PDF failas, 150Kb) 

##### ☛ TeX WWW resursai

Rasite nuorodas á TeX archyvą tinklą (CTAN), ádomesnius TeX serverius bei nuorodas á FAQ lapus bei USENET grupę

##### ☛ TeX Live CD

TeX Live CD yra puikus TeX programų archyvas. Ją paruošė Thomas Esser ir Sebastian Rahtz. Programė versijos UNIX, Linux, MS DOS, Windows aplinkoms, metalinės programos. Pilnas stilių ir ávairių makro paketų rinkynys. Paruoštas pagal CTAN archyvą 1997 gegužę, platinamas kartu su TUGboat þurnalu.

#### Nordic T<sub>E</sub>X User Group

The Nordic TeX Users Group (NTUG) is, according to its *bylaws*, an association devoted to *promote the development and use of TeX and related programs, particularly in the Nordic countries.*

#### Organization

Membership in NTUG is open to everybody, and there is currently no membership fee. To be registered as a member, just send a short note by e-mail to the chairman Dag Langmyhr ([dag@ifi.uio.no](mailto:dag@ifi.uio.no)).

NTUG is a very informal group. We try to arrange an annual meeting every year, but otherwise the communication is mostly via e-mail and WWW.



University of Oslo  
 N-0316 Oslo  
 Norway  
 tel: +47 22 85 24 50  
 fax: +47 22 85 24 01  
 e-mail: [dag@ifi.uio.no](mailto:dag@ifi.uio.no)  
 WWW page: <http://www.ifi.uio.no/~dag/ntug/>

NTG (Dutch-speaking)  
 Erik Frambach, Chair

**Grup d'usuaris  
catalanoparlants  
de TeX**

**Catalan  
TeX Users Group**

**Tirant lo TeX**



**Welcome to the  
TeX Users Group  
Home Page**

The TeX Users Group (TUG) was founded in 1980 to provide leadership for users of TeX, Donald Knuth's revolutionary typesetting system. It represents the interests of TeX users worldwide: if you use TeX in any of its forms, you should join TUG (see the [Aims and Benefits](#)).

**The UK TeX User Group and CTAN Archive  
(Archive hosted by the University of  
Cambridge Computer Lab)**



Nederlandstalige TeX Gebruikersgroep  
Postbus 394  
NL-1740 AJ Schagen  
The Netherlands  
e-mail: [ntg@ntg.nl](mailto:ntg@ntg.nl)  
WWW page: <http://www.ntg.nl/>  
periodical: *MAPS*

**TeXCeH** (Slovenian TeX User Group)  
Vladimir Batagelj  
Jadranska 19  
SI-61111 Ljubljana  
Slovenia  
e-mail: [texceh@uni-lj.si](mailto:texceh@uni-lj.si)

**Tirant lo TeX** (Grup d'usuaris catalanoparlants de TeX)  
No formal user group yet.  
e-mail: [valiente@lsi.upc.es](mailto:valiente@lsi.upc.es)  
Mailing list: [catala-tex@aligna.cesca.es](mailto:catala-tex@aligna.cesca.es)  
WWW page: <http://www-lsi.upc.es/~valiente/tug-catalan.html>

**TUG** (International user group)  
Mimi Jett, President  
TeX Users Group  
1466 NW Front Avenue, Suite 3141  
Portland, OR 97209  
USA  
tel: +1 503 223 9994  
fax: +1 503 223 3960  
e-mail: [tug@tug.org](mailto:tug@tug.org)

WWW page: <http://www.tug.org/>  
periodical: *TUGboat*

**TUGIndia** (Indian)  
Prof. (Dr.) K. S. S. Nambooripad, Chairman  
Indian TeX Users Group  
Kripa, TC 24/548, Sastha Gardens  
Thycaud, Trivandrum 695014  
India tel: +91 471 324341  
fax: +91 471 333186  
email: [tugindia@mailexcite.com](mailto:tugindia@mailexcite.com)

**UK TUG** (United Kingdom)  
Philip Taylor, Chairman  
UK TeX Users' Group  
For information:  
Peter Abbott  
1 Eymore Close  
Selly Oak  
Birmingham B29 4LB  
England  
e-mail: [uktug-enquiries@tex.ac.uk](mailto:uktug-enquiries@tex.ac.uk)  
WWW page: <http://www.tex.ac.uk/UKTUG/>  
periodical: *Baskerville*

**T<sub>E</sub>X in 2003: Part I****Propositions and conjectures on the future of T<sub>E</sub>X**

NTG T<sub>E</sub>X future working group  
 P.O. Box 394,  
 1740 AJ Schagen,  
 The Netherlands  
 ntg-toekomsttex@ntg.nl  
 http://www.ntg.nl

**Introduction**

In the last year, there has been a lively discussion within the Dutch T<sub>E</sub>X Users Group about the future of T<sub>E</sub>X. This discussion was initialized by a couple of posts to the TEX-NL e-mail list by Hans Hagen and Taco Hoekwater, but it soon spread to a much larger group of correspondents.

Eventually, this resulted in a meeting between the most interested people in December 1997. The current articles are a re-working of the long-term proposals and requests formulated by this group of people. The short-term requests were passed on to the eT<sub>E</sub>X team.

**Our views on current work**

At the moment, there are at least three distinct projects available to current T<sub>E</sub>X users that are working to extend T<sub>E</sub>X: Omega, P<sub>d</sub>ftex and eT<sub>E</sub>X.

The first two of these are in a sense niche products: If you don't need either non-latin language typesetting or PDF output, there is little point in learning how to use these two programs. The third project, eT<sub>E</sub>X follows the more general approach, and is potentially of interest to every current user of T<sub>E</sub>X.

The work done in eT<sub>E</sub>X is nicely thought out, and the result is both stable and virtually bug-free, but it is hardly ever used in real applications. The reason is simple: package writers will not use eT<sub>E</sub>X primitives until they can be certain that eT<sub>E</sub>X is indeed available everywhere. On the other side, eT<sub>E</sub>X cannot develop without input from package writers that intent to use eT<sub>E</sub>X. There is a chicken-egg situation, and it leads to the following conclusion:

1. *eT<sub>E</sub>X is a nice idea with too little momentum to make a difference.*

Another important problem is the fact that people that need the functionality of either eT<sub>E</sub>X or P<sub>d</sub>ftex or Omega and one of the other two extensions, cannot do so from within one document. All three have their own specific syntax extensions, that are hard to fake in one of the other extensions. This is unsolvable in the current situation, and leads us to the following statement:

2. *Omega, P<sub>d</sub>ftex and eT<sub>E</sub>X should be merged as soon as possible.*

Then there is a fourth project that has just started: the New Typesetting System (NTS).

The NTS development group hopes to increase the chances of general acceptance of NTS by guaranteeing compatibility with T<sub>E</sub>X for a number of years to come. We feel that this is a error, because most of the more fundamental issues that NTS should deal with to live up to the 'New' in it's name cannot be done without sacrificing that compatibility. Issues like grid-based typesetting and better insertion control are very likely to require a completely new algorithm, resulting in a completely new implementation. Of course it is possible to do these things in parallel, but trying to implement something new while having to be really careful not to break the old implementation unnecessarily complicates development: people that want to use T<sub>E</sub>X should stay with T<sub>E</sub>X anyway.

Therefore, we urge the NTS group to reconsider their decision to stay compatible with T<sub>E</sub>X for at least the next five years.

3. *NTS will be pointless if it intends to be compatible with T<sub>E</sub>X82*

The second remark we have deals with the proposed modularity of the system, which is facilitated by the use of Java:

4. *NTS is a step forward and a step backward at the same time.*

A great feature of NTS will be its extensibility. This is similar in many ways to current L<sup>A</sup>T<sub>E</sub>X packages, albeit much more advanced. Since NTS will be written in Java, one can

easily extend NTS with its own classes. We presume there will be an easy interface to extend NTS (if not, someone will just hack the sources).

In all likelihood, this will result in precisely the same problems that current L<sup>A</sup>T<sub>E</sub>X has:

- Users are not aware of the packages available, and so keep asking questions like: How can I make this work in L<sup>A</sup>T<sub>E</sub>X?
- Furthermore, the portability of source documents (the .tex or .nts? input file) will be seriously endangered. We expect to see things like:

```
Error: this .nts style-file requires
module x.y which has not been
installed on your system.
```

The NTS team should give very strict rules for these extensions, otherwise we'll end up with another \special- similar situation. A central registry and a "head maintainer" are needed to keep track of extension modules in order to prevent these problems. It would be wise to turn this work into a full-fledged job under the control of (probably) TUG.

**5. *We need time to experiment and must not fall into the every year a new version trap.***

An interesting common aspect in all current work is that only experience can lead to useful functional specifications. It is likely that NTS functionality will follow the same track.

This means that when we deal with the next generation T<sub>E</sub>X programs, common users must be patient until the developers of extensions and macro packages trust the new features and can guarantee upward compatibility. It also means that it will take some years until eT<sub>E</sub>X as well as NTS will be accepted as descendants.

We have to keep in mind Knuth completely rewrote his first T<sub>E</sub>X!

## Packaging of Distributions

Over the last 5 years T<sub>E</sub>X has become a lot easier to install. The most important reasons for this are:

- Cd-roms have become available at large. These can easily hold a complete T<sub>E</sub>X system. The old-fashioned piles of diskettes gave far too much trouble, and tape is only for professionals.
- Recently hard disk space has become so cheap that complete installations on hard disk are not unusual anymore.
- Installation scripts were made to shield users from tedious setup and configuration issues.

Still a number of problems remain because they are inherent to the way that T<sub>E</sub>X systems work:

- A typical T<sub>E</sub>X system consists of an incredible number of files (more than 31415). No one really knows which parts are essential and which parts are not. In other words: every system is too large.
- "Everything" can be found on CTAN but only the most recent version. Old versions can be necessary to run old documents. Old CTAN dumps on cd-rom can be used to track down older versions, but we really need more professional version control.
- Maintenance is only feasible for professionals. Others are better off replacing the entire system, even though this will undoubtedly cause problems. The draw-back of 'plug & play' systems is that users have no idea anymore of the inner workings of the system. Is that a good thing or a bad thing?
- There is no such thing as an easy upgrade path. It's usually very hard if not impossible to simply add some files to a system and make them cooperate.
- Initial configuration can be automated, but reconfiguring is usually very hard. Any typical T<sub>E</sub>X system contains dozens of configuration files in almost as many completely different flavours. As a rule they are scattered all over, and only an absolute expert can deal with this.

This leads to a number of conjectures:

**6. *The number of files in a typical T<sub>E</sub>X system should be reduced by a factor 100.***

We can achieve this by redefining the way any program finds its resources. A central database should be queried for any resource. This database should physically contain all resources. And of course it should be able to report (in any required level of detail) what's available. The database may even connect to CTAN (another database application) to retrieve resources not available locally.

This setup would allow for a minimal local installation to grow as necessary using Internet.

**7. *Configuration of a T<sub>E</sub>X system should be centralized and automated.***

If we can realize the previous issue this one will not be too hard. Programs should specify formal descriptions of the configuration details they need. These could then be generated through menus or automatically by scanning the current setup, i.e., querying the database.

**8.** *Installation and maintenance should require far less expertise.*

The database may occasionally query CTAN for any updates. The administrator would get short descriptions of these, with links to complete documentation. He/she could then select which ones should be installed. This could even be done silently (overnight) if you want an up-to-date system all the time. If necessary, programs will be signaled to reconfigure themselves.

This setup should also take care of the endless problems with non-portable DVI files. We should all be using the same resources and if we are not, the system should warn us about possible mismatches. If we decide to make T<sub>E</sub>X produce DVI files that require no virtual fonts at all (i.e., T<sub>E</sub>X reads VF's itself instead of the DVI driver) an important source of problems can be eliminated.

**9.** *CTAN should have a complete index with descriptions of everything and cross-links to anything related to anything.*

This is obvious now if we want the systems to interact. Uploads to CTAN will have to be checked more carefully: descriptions, specifications, version number, relations to other packages, dependencies on other resources, etc. must be supplied. Any item that doesn't comply to this convention should be moved out ('not supported') and deleted after a certain period.

We realize that this might cause a cultural shock in the T<sub>E</sub>X world, but we feel this is necessary to keep T<sub>E</sub>X alive & kicking in the next millenium:

**10.** *Anarchy is what made T<sub>E</sub>X great, and it's anarchy again that will kill T<sub>E</sub>X.*

Let's try to prevent this!

## On-line Publication Wishlist

With the increasing growth of the internet, a whole new branch of documents has appeared: documents that are only or primarily intended for screen viewing. The used formats differ, but it is easy to see that there are some common issues involved in all of those: file download

sizes, hyperlink support and ease-of-use are important points for all of these formats.

**11.** *T<sub>E</sub>X is rather well suited to cater for those needs as it is, but some extensions are needed to make sure that T<sub>E</sub>X will stay/become in the leading position in this arena*

For about 15 years T<sub>E</sub>X was only capable of producing DVI output. The limitations in both T<sub>E</sub>X and the DVI format mainly concerned direct graphic support and color typesetting, but color printers were rare and the lack of graphics support could be worked around.

Although originally T<sub>E</sub>X was more or less supposed to handle everything itself, those 15 years of use have demonstrated that many applications, like color and graphic inserts, heavily depend on the DVI postprocessing stage. To a large extent, this is not feasible nor desired in on-line publication. On-line formats are all rather device independent themselves: otherwise people would have to publish several versions of the same document.

Theoretically, both pdftex the current trajectory using and DVI to PDF processing through dvips and the Distiller can offer similar functionality, given that post processors are available to help out in the second case, but we can imagine both methods drifting apart, and we feel that the use of external programs to solve intrinsic problems adds a great deal of unnecessary complexity to the system.

**12.** *On-line publishing needs primitive support*

In fact, most of the conceptual extensions like hyper referencing can be implemented using DVI and `\specials`. However, usage can be far more robust in e.g. current pdftex, simply because hyper referencing is build in, and there is no longer a need to run various programs in turn. The same goes for object reuse, fill-in forms, scripting (Java), and graphic inclusion.

But systems like pdftex also create new problems. Take for instance graphics inclusions: where originally T<sub>E</sub>X macros only had to bother with the dimensions of the needed box, on-line publishing backends have to include the file directly.

Another conceptual extension is hyper referencing. Although clever tricks can give acceptable results, all approaches based on current T<sub>E</sub>X interfere with either the explicit wishes of the author or the line and paragraph break mechanisms present in T<sub>E</sub>X.

**13.** *T<sub>E</sub>X objects should be easily re-usable*

When we look at object reuse, we see that this concept never surfaced in DVI (using `\specials`). This is probably due to the fact that especially screen designed documents need these features, and it hardly matters for paper output.

From the users point of view, reuse may look rather straightforward (a sort of variant on copying boxes), but from the implementors eyes, object definitions are just another interfering kind of *<whatsit>*. And why is it interfering? Simply because T<sub>E</sub>X has no particular mode which

suppresses all interference. Yes, we can use a box, and we can let things happen at certain locations in the document that don't do any harm, but the situation is far from optimal.

When applied to for instance figure inclusion, reuse can quite easily be implemented in original T<sub>E</sub>X (pure DVI, using Gilbert's DVIVIEW), the traditional DVI-dvips-Acrobat trajectory or Thanh's pdftex. But PDF fill-in fields support demands for more.

To give you a real life example where objects are needed: in PDF one can define a check field with several appearances like on, off, mouse down, etc. Technically this means something like this (in pdftex syntax):

```
\setbox0=\hbox{$\star $} \pdfform0
  \edef\on {\the\pdfastform}
\setbox0=\hbox{$\bullet$} \pdfform0
  \edef\off {\the\pdfastform}
\setbox0=\hbox{$\times $} \pdfform0
  \edef\down{\the\pdfastform}
```

When defining the check field, we then can refer to \on, \off and \down, as in the following code:

```
\pdfannot{ ... /On \on\space0 R ... }
```

Currently pdftex only flushes forms to the output file when are accessed. (this feature is needed because we want to be able to try out things, without ending up with redundant objects, like in a macro that tries three different methods and takes the best result).

Back to the three objects, these won't end up in the file when we refer to them in the field definition above, because the field definition is handled like a \special: pdftex just passes the information through.

Therefore, we end up with invalid references: the object is referred to, but never passed to the file. What do we learn from this:

#### 14. T<sub>E</sub>X needs a real object model.

One with immediate as well as deferred definitions, that do not interfere with the internal lists that T<sub>E</sub>X builds and that permits forward *and* backward referencing.

Another typicality that surfaces often in on-line documents is the fact that screen layouts tend to use a lot more page decorations and colors than traditional typesetting. This is an area where a lot of disagreement is possible, but in the real world there are lots of practical applications of this.

At TUG97 there were several presentations on graphics. The related discussions invoked a BOF session on graphic primitives. Direct inclusion of METAPOST output (in pdftex) had already proven that a relatively small subset of

PostScript primitives could be used for advanced graphics and therefore the discussion focussed on those primitives.

These graphic primitives in T<sub>E</sub>X are not meant for drawing free hand graphics like one would do in programs like Illustrator, CorelDRAW, or indeed Freehand. Instead, they are most often (to be) used for things like visualizing statistical results, plotting functions and drawing almost-mathematical shapes that can be used to emphasize certain layouts. In these graphics, text plays a important role, and this text must preferably be typeset by T<sub>E</sub>X. It follows that inclusion of an external file will not do, and the conclusion is:

#### 15. T<sub>E</sub>X needs a reliable system for in-line graphics and colors

The most important outcome of the '97 BOF session was an agreement on the way to go: define a set of extensions that permit direct METAPOST output inclusion. It was felt that this set could also suffice the needs of the mainstream graphic macro packages written in T<sub>E</sub>X.

During the NTG 'future of T<sub>E</sub>X meeting' the participants made the exact specification of these graphic primitives (currently to be implemented as \specials) one of its main goals. To this end, we had to create a formal specification of the syntax involved, and that put us right in the middle of the \special problems.

Our final proposal on that matter will appear somewhere else in these proceedings, but Gilbert has already done some of the groundwork. Below is his explanatory text on the \specials that are currently included in DVIVIEW. This text is kept here because it demonstrates very well that only a few primitive commands are enough to give almost full in-line graphics capabilities.

To allow for instance METAPOST drawings to be inlined in T<sub>E</sub>X you need several things:

- A macro to interpret METAPOST's POSTSCRIPT output. Hans Hagen wrote a set of macros for PDF<sub>T<sub>E</sub>X</sub> using \pdfliteral commands. These macros are easy to adapt to another standard using \special syntax
- A primitive sub-set of POSTSCRIPT commands is needed. METAPOST uses only a few POSTSCRIPT commands to draw it's figures.

To actually test the inline graphics standard we needed a viewer where this support was easy to include. DVIVIEW was coming to life at that time so it was logical to use that as a test and development environment.

All primitives are easy to interpret, except for a few things like clipping and the like. The syntax will probably change in the future when the new special syntax is standar-



ized. Converting these specials to POSTSCRIPT output (e.g. modifying dvips) is easy to do, since the commands hardly need any translation.

Specials and stuff for inline graphics in DVIVIEW:

```
\special{dv:startgraphic}
\special{dv:stopgraphic}
\special{dv:moveto x y}
\special{dv:lineto x y}
\special{dv:curveto x1 y1 x2 y2 x3 y3}
\special{dv:stroke}
\special{dv:setlinejoin j}
\special{dv:setlinecap c}
\special{dv:setdash offset values}
\special{dv:setlinewidth w}
\special{dv:setmiterlimit m}
\special{dv:rotate r}
\special{dv:translate x y}
\special{dv:concat x1 y1 x2 y2 x3 y3}
\special{dv:newpath}
\special{dv:closepath}
\special{dv:clip}
\special{dv:fill}

\special{dv:gsave}
\special{dv:grestore}
```

As you can see the amount of commands needed to support METAPOST output is in fact quiet small.

Some explanations:

#### **dv:startgraphic**

Starts a graphics figure. It saves the current position and context of the DVI interpreter. The current location is marked as (0, 0). As in POSTSCRIPT positive  $x$ ,  $y$  draws to the right and up.

#### **dv:stopgraphic**

Stops a graphics figure and restores the context.

#### **dv:moveto x y**

Moves the current position to  $x$ ,  $y$ .

#### **dv:lineto x y**

Draws a line to  $x$ ,  $y$ . This does not actually draw the line but only remembers the coordinates. The actual drawing is performed by `stroke`.

#### **dv:curveto x1 y1 x2 y2 x3 y3**

Draws a Bézier curve starting at the current point to  $(x3, y3)$ . The control points are given as  $(x1, y1)$  and  $(x2, y2)$ .

#### **dv:stroke**

Performs the actual drawing using the current pen-style, color and width.

#### **dv:setlinejoin j**

How lines are joined.  $j$  can be 0, 1 or 2.

#### **dv:setlinecap c**

How the line-endings will look like.  $c$  can be 0 1 or 2.

#### **dv:setdash offset vals**

Sets the pen-style. `vals` is any number of values and specifies how long the pen is on and how long the pen is off. `offset` can be used to specify a starting offset in the `vals` pattern.

#### **dv:setlinewidth w**

Sets the thickness of the current pen.

#### **dv:setmiterlimit m**

Sets the miterlimit.

#### **dv:rotate r**

Modifies the current transformation matrix so that everything following this is rotated  $r$  degrees.

#### **dv:translate x y**

Modifies the current transformation matrix so everything following this is translated  $(x, y)$ .

#### **dv:concat x1 y1 x2 y2 x3 y3**

Multiplies the current transformation matrix with the given values.

#### **dv:newpath**

Discards any present paths and start a new path.

#### **dv:closepath**

Closes the current path. After this you can use `fill` to fill the closed path.

#### **dv:clip**

Selects the current path as the clipping path. All subsequent fills and strokes are clipped to the this path. The clipping path may contain one or more closed paths.

#### **dv:fill**

Fills the current path with the current color.

#### **dv:gsave**

Saves the graphics state.

**dv:grestore**

Restores the graphics state.

**dv:setrgbcolor r g b**

Sets the current color. *r*, *g*, and *b* are specified from 0 to 1.

**dv:setcmykcolor c m y k**

Sets the current color.

**dv:setgray g**

Sets the current gray-level. 0 means black, and 1 means white.

Though it is easy to extend this set and include much more POSTSCRIPT operators, this is not the intention. It should be noted that complex graphics which require the full POSTSCRIPT set of commands should be done by including the EPS file and let PostScript do the work.

**Language extension wishlist****Removal of limitations regarding fonts**

The font limitations that are inherent in the TFM format should be dropped. One fairly simple way to achieve this is to make T<sub>E</sub>X read .pl or .vpl files instead of TFMs, but it is also possible to adopt a new format like Omega's OFM files or even create a completely new specification.

An overview of limitations in current T<sub>E</sub>X shows limits in almost all places: the amount of characters present in a TFM, The number of separate width/height/depth/italics-corr values, the number of ligatures and kerning pairs, math sizing stuff, etc. Almost all of these limitations are not really needed anymore; most of them were born out of Knuth's desire to use as small an amount of memory as possible.

Especially the current implementation of math mode has some really weird demands on the used fonts (some characters get really weird places in the glyph container, like integrals and delimiters are all below the baseline, and the height of the `\sqrt` sign is used to decide the width of the extension bar). This should be fixed so that it becomes possible to use non-metafont math fonts in a reliable way, and to facilitate the creation of new math font sets. The current situation makes it impossible to use non-T<sub>E</sub>X math fonts from e.g. *mathematica* without *lots* of *vf* trickery.

These things are all very easy to fix in the executable, but it won't do any good at the moment, because we are still stuck with the TFM format.

**16. *The way TFM and VF formats are defined and implemented is the primary cause of the current font chaos***

If we want to adopt a new format, the extensibility of the syntax of PL files is to our advantage, even allowing new features to be added in the future while remaining backward-compatible. But, although there no longer is a real reason for binary file input as speed or disk space optimization, binary files *do* have the advantage of being non-editable (meaning that the chances of a user accidentally breaking them is very small).

**17. *We need symbolic names for characters***

T<sub>E</sub>X currently uses encoding instead of glyph names. Encoding is old-fashioned and merely a speed optimizing thing. The coupling of glyph-name-character should be a T<sub>E</sub>X internal operation.

The used named characters from the fonts should be deductible from the output (DVI) file, to prevent reencoding issues in postprocessing applications. To reach this goal, it is very likely that T<sub>E</sub>X needs an internal naming scheme for glyphs that does not depend on font encoding. Work in this area is already being done by the eT<sub>E</sub>X team. It is considered unlikely that using unicode will solve the problem, but it might well be that a solution based on the predefined set of unicode names (the road taken by Omega) is the right way to go.

**18. *Ligatures and kern info should be independent of the character metrics***

Ligatures can be present in the current font definitions, but we would like to be able to modify the lig-table internally from within T<sub>E</sub>X. This request has already be passed on to the eT<sub>E</sub>X group, but it needs a more general solution than the primitives that were proposed to eT<sub>E</sub>X(`\noligs` and `\nolig<char>`). Likewise for the kerning tables.

The mechanism by which a user loads fonts into T<sub>E</sub>X's memory is much too simple. It should be possible to specify encodings, kerning info and ligature tables separate from the actual glyph dimensions. The ligature problem actually comprises two very different problems.

The simple case is most noticeable in typesetting verbatim stuff in non-tt fonts, something that is often needed for textbooks on programming languages.

The hard case comes from the fact that ligatures depend on the language, not on the used font itself. The spanish quotation e.g. is never needed outside of Spain, and we are all stuck with it now. Ideally, every language should have it's own ligature table, that is part of the language attributes just like `\patterns` are.

**19. *Metafont is becoming outdated, even if T<sub>E</sub>X itself isn't***

A new version of metafont is needed that can generate acceptable outline fonts instead of the now used .pk format, and the use of non-metafont fonts (Postscript, TrueType) should be simplified. As stated in a previous article, T<sub>E</sub>X should take care of the virtuality of fonts itself. But that does not *have* to imply using .vf files. There are some other possible solutions that may not be as powerful as .vf, but are a lot less confusing: The only widely used applications of virtual fonts are reencoding and creation of composite characters.

## User interface

Currently, T<sub>E</sub>X shows a weird duality: while mostly a batch tool, there are still a number of places where user intervention is needed.

On one side, if T<sub>E</sub>X wants to survive as a batch tool (either as a stand-alone typesetter or as back-end for e.g. SGML processing systems), it will need extensions so that it is 100% safe to run the program unattended. Things like breaking math formulas and placement of figures cannot be left to T<sub>E</sub>X on its own.

On the other end of the spectrum, T<sub>E</sub>X needs a real-time graphical user interface to satisfy interactive users (maybe this can be a partial implementation, like having GUI-based equation- or table-editors). This goal can only be reached if the GUI-based tools have fool-proof T<sub>E</sub>X input format that they can rely on.

There are two probable roads we envisage:

- Moving a large number of current macros into the executable itself will avoid confusion of macro formats, but there are still problems to be solved relating to redefined primitives.
- Allowing a tokenized input in a precompiled format would probably be better since it circumvents these problems. The idea is that, assuming we are an external program that tries to generate T<sub>E</sub>X code, we want to be very sure that `\par` really means `\par`.

But there are some other idiosyncracies in T<sub>E</sub>X's language that needs to be dealt with as well. Sometimes optional, sometimes not optional keywords and characters like equal signs; arguments with braces versus arguments that are space-delimited; confusing rules for spaces; etc.

**20.** *At all events, the language should be cleaned up drastically.*

The syntax should definitely be cleaned out. Anybody who has ever tried to write a non-trivial macro will know that even if your approach in itself is correct, chances are that the macro still won't work, because of a stupid mistake with `\expandafter` or extra/too few spaces. Solutions that use markup in the style of SGML or lisp would be vastly preferable over the current situation. The current syntax often justifies the following statement:

**21.** *T<sub>E</sub>X's macro language encourages writing garbage*

We can safely say that many sources look awful in terms of formatting, just take a look at the sources of the style used to typeset this article. (Or look at the sources of the T<sub>E</sub>Xbook: the output is beautiful, the input just ugly.) In the hands of common users, bad input becomes bad output.

**22.** *We would profit from better programming primitives*

Finally, experience shows that format files are never simple and small, like Knuth presumed they would be. Instead, format files are complex programs with numerous interactions between the various parts. T<sub>E</sub>X's macro language was never supposed to support this, and as a result has virtually no programming support. Among the missing things are data structures like lists and queues; name spaces; control structures (like cases and while loops); signals; and reliable `\if` tests.

# TEX in 2003: Part II

## Proposal for a `\special` standard

NTG TEX future working group  
 P.O. Box 394,  
 1740 AJ Schagen,  
 The Netherlands  
 ntg-toekomsttex@ntg.nl  
 http://www.ntg.nl

### abstract

The text of this article is a proposal for an “endorsed” `\special` specification, to be voted on by the assembly of the TUG98 meeting. Portions of this text are reworks of an original article by Nelson Beebe, and indeed large portions of the proposal itself are also based on original work done by Beebe.

### Introduction

Most existing drivers have chosen an arbitrary syntax for the `\special` strings they support. This is undesirable, for at least these reasons:

- The chosen syntax is usually unique to a particular driver, and therefore seriously compromises document portability.
- The syntax is usually not extensible in an easy way.
- The syntax cannot always be unambiguously parsed.
- The output device, or driver, to which the `\special` applies is not determinable.
- The capabilities are weak, and fail to address many of the potential uses of the `\special` command.

The `\special` syntax that we have developed, which is really an extension and modification on the work done by Nelson Beebe, resolves these objections. It has the following features:

- The `\special` string is defined to contain a program written in a small language that consists of an identification string and a command, followed by sequences of assignment statements, possibly with embedded comments.

- The `\special` language is *rigorously defined* by a programming language grammar (available on request).
- The language is *extensible*. An assignment statement consists of a keyword/value(s) pair. Several keywords are already defined, and new ones can be added without invalidating existing uses of the language.
- Keywords are typed, and constant values assigned to them must be of the correct type. The supported types are names, strings, numbers, and dimensions.
- Value string concatenation is supported in the style of ANSI C, avoiding the often severe line length limitations of text editors, operating systems, and file systems.
- Provision is made for encoding all 8-bit characters in the host character set, so that, e.g. binary printer control sequences can be incorporated as *printable*, and *portable*, text in TEX documents.
- A particular keyword, language, is provided to permit the user to specify the output device language, or the DVI driver, to which the `\special` command is directed.

By suitable abstractions, it is possible to create a recursive-descent parser for the language in which commands, keywords and value storage locations are provided in a table passed to the parser. The parser code is therefore completely portable, and independent of the commands and keywords in the language it parses.

We will write a table-driven parser that will accept all the commands and keywords we have defined, and this parser, written in the C language, will be included in the DVIVIEW program that will serve as a reference implementation. The parser itself will be available in the public domain soon, and patches will be made to at least `dvips` and `xdvi` to support this proposed standard.

### A proposed syntax for the `\special` command

What does the language look like? Some examples will give the general flavor before we describe the details of the grammar. Here are some fragments of hypothetical TEX input which show some of the `\special` commands:

```
% Display a picture with the upper-left
% corner at the current point
\special{**include pict.eps}

% Display a picture at its original
% absolute page position
\special{**overlay "pict.001",
         filetype metapost}

% Display a figure at half size
\special{**include "pict.eps",
         scale 0.5 0.5}

% Switch to a different colour
\special{**colour .09 .06 .6,
         model rgb}
```

Naturally, the details of a `\special` command invocation should be hidden away in suitable macros that are easy to use.

## The language grammar

In the original article, Beebe gave a formal grammar for his language. In the interest of keeping this article as short and readable as possible to a non-programmer, that grammar has been deleted and we have not inserted our own. If you are interested in it, it is available upon request. We will suffice here with giving a textual explanation.

We will start by defining the various primitive types that are supported:

### Spaces

Whitespace is ignored except as delimiting characters, so the specification can be formatted for readability, or for compactness. Token may not contain embedded blanks (except strings of course).

### Comments

Comments are from percent to end-of-line, like in T<sub>E</sub>X. Comments cannot occur inside of strings or keywords, so this is not a comment:

```
\special{**message "Here % is some text"}
```

and this is in fact illegal:

```
\special{**mes% neat eh?
         sage "Here % is some text"}
```

### Names

The grammar states that an extended letter is a digit, letter, hyphen, dot, or underscore. These are the characters

that are allowed in commands, keywords, alternative values and unquoted strings. Letter case is not significant in these cases.

The characters permitted are chosen such that for instance simple filenames can be used without surrounding quotes (see below for more info on strings and alternative values).

An “alternative value” is actually a string with some pre-defined values.

### Numbers

Numeric constants are parsed by the ANSI C library routine, `strtod()`, which expects to see numbers in the form:

```
[whitespace][sign][digits][. digits]
[{e|E}][sign]digits
```

### Dimensions

Dimensions can be given in any absolute unit known to T<sub>E</sub>X (bp cc cm dd in mm pc pt sp). Note that the font-specific `em` and `ex` are not allowed. Since tokens may not contain embedded blanks, 210mm is legal input, but 210mm is not.

Any keyword that accepts dimensions as arguments will also accept numbers. In the absence of a dimensional unit, a default value will be used. This default can be defined with a separate `\special` (see below under `defaultunits` for important usage information), or, in the absence of that `\special`, the driver will presume scaled points (`sp`).

### Strings

The grammar supports unquoted strings and two kinds of quoted strings.

An unquoted string has to be one word only (since there are no spaces allowed), and can only use the characters that are legal extended letters as defined above.

The *normal* kind of quoted string is delimited by double quotes, and inside it are recognized all the escape sequences supported by the C language. The *raw* kind is delimited by single quotes; only escape-single-quote pairs are recognized inside it. This is more convenient when it is necessary to have strings with several backslashes, since it then avoids having to double all of them. Once normal and raw strings are parsed, they are stored identically.

Backslashes in literal strings and filenames pose a small problem for the user, because T<sub>E</sub>X will ordinarily try to interpret control sequences triggered by backslashes in the argument of the `\special` command. Although it would have been possible to choose another escape character than backslash for such strings, this would likely prove confusing to those users who are used to C and UNIX, where the backslash escape character is firmly entrenched.

Fortunately, the solution is not difficult, because T<sub>E</sub>X does not have backslash hardcoded as a control sequence prefix; you can change it by altering T<sub>E</sub>X's catcodes.

In the descriptions of the `\specials` below, the character `n` and `m` used to indicate a value from a fixed set of alternatives, `s` is used to indicate all sorts of strings, `x`, `y` and `z` (possibly with numeric tags) are used for dimensions, and `a` through `j` are used for numbers.

Now let us move to the portions of a `\special` that actually define things. The structure of a `\special` command is as follows:

### ID bytes

The first 2 characters in every `\special` are to be the two tokens `**`. The rationale behind this is that a convention like this makes it easier to adjust programs that have to remain backward-compatible with their old private syntax. As far as we know, this particular sequence of tokens is never used in current `\specials`.

### Command

The next word is the principal command for this `\special`. Depending on the command itself, it may have arguments or it may be a single command.

### Assignments

Optionally, the command can be followed by a series of keywords that supply extra information. Keywords follow the same syntax as commands, so there can be zero or more arguments to a keyword.

In a series of assignment statements, the order of the keywords is not significant, except that if duplicate keywords are specified, the value of the last one is used.

Every keyword-value group needs to be separated from the previous one by a separator, which may be either a semicolon or a comma. This is correct:

```
\special{**include "pict.eps";
          scale 0.5 0.5}
```

And this is not:

```
\special{**include "pict.eps"
          scale 0.5 0.5}
```

### Separating items

Finally, the assignment statement may use either the equals or colon operator, or the operator may be omitted altogether. This supports the common forms:

```
\special{**include=pict.eps}
\special{**include:pict.eps}
\special{**include pict.eps}
```

Because the values have very limited syntactical possibilities, there is no ambiguity created by this.

## The `\special` language

The preceding section defined the grammar for the `\special` language. We now need to define what commands and keywords will be recognized. As emphasized above, the language is *extensible*, and the parser that we will implement for it makes it easy to add new commands and keywords *without touching a single line of the parser code itself*.

However, we presume that there will be a maintainer or maintenance group assigned to take care of this specification, and this person has the right to refuse to accept extensions that do not fit in.

### Generic keywords

The full set of commands and keywords that are recognized is given below, but we will start off with some general keywords. These keywords can be used within any `\special`, and also be used as a command. They will not be mentioned separately in the descriptions of the other `\specials`:

Keyword	Value	Action
message	s	Supply an operator message to be sent to the terminal and log file.
id	n	Supplies a name that uniquely identifies this <code>\special</code> .
use	n	Supplies a name that identifies a previously defined <code>\special</code> .

The message string provides a means for operator communication; for example,

```
message "Thesis bond paper for this job"
```

The message is sent verbatim to the terminal and the log file.

The `id` allows identification of the `\special` it occurs in. The command and the keywords and values associated with this `\special`, are saved and available for later reuse through `use`. The current location in the file is also saved, for later retrieval by one of the cross-link `\specials`.

The usage of `use` is as follows: first, all of the data from the `\special` it refers minus the `id` value are inserted in the current `\special`, and other any values that occur in the current `\special` are used to override the inherited options. An example is probably the best way to show this.

After

```
\special{**include "pic1.eps";
        scale 0.5 0.5;
        id mypic}
```

The following command re-does precisely the same in a later portion of the document:

```
\special{**use mypic}
```

and

```
\special{**use mypic;
        scale 1 1;
        id mypic2}
```

inserts the same figure, but at a different scale. It also assigns a new *id* to this current `\special`. The following is also allowed

```
\special{**include "pic2.eps";
        use mypic;
        id mypic2}
```

but it is *not* legal to switch to an entirely different command, like `overlay`.

Drivers are allowed to set an upper limit to the number of *distinct* *ids* that can be used in a document, but this limit should not be lower than 256. There is never a point to limit the total amount of *ids*, since later definitions will just overwrite the previous one with the same name.

There is at the moment exactly one command that affects the `\special` parser itself:

Keyword	Value	Action
defaultunits	n	Sets the default units to one of the defined dimension types instead of <i>sp</i> .

### Commands for graphics inclusion

There are three possible ways of including a graphic figure file from disc:

Keyword	Value	Action
include	s	Insert file contents with relative page positioning.
overlay	s	Insert file contents with absolute page positioning.
underlay	s	Insert file contents with absolute page positioning.

The filename string can be used for normal local files, but it can also be used for URLs, following the normal rules for URL specification. If no explicit protocol (like `http` or `ftp`) is given, the name is assumed to be a local file. Even non-networked drivers are required to correctly handle one protocol: `file://`.

In all these three cases, drivers can opt to give a default search path for figure files with relative path names, but this is not required nor encouraged. The driver is not required to include any file type except `dvi`.

`overlay` and `underlay` are supposed to start from the lower-left corner of the physical page, with coordinates as in PostScript: up and right are positive values for *x* and *y*. In cases where there is no obvious lower-left corner (as may be the case for on-line backends), the lower-left corner is defined to be at the end of the output medium.

`include` places the top-left corner of the image at T<sub>E</sub>X's current point. Here coordinates are as in DVI: down and right are positive values for *x* and *y*.

The difference between `overlay` and `underlay` should be clear: overlays can actually obstruct other images and text on the page (depending on where precisely on the page the `\special` was given), underlays can never do this, but a second underlay might be on top of the previous one.

If the file cannot be opened, or for relative positioning, the bounding box cannot be determined, a warning message is issued and the `\special` command is ignored.

There is also a `\special` command available for the inclusion of literal drawing commands:

Keyword	Value	Action
graphics	s	Execute the graphics primitives in string (defined below).

The `graphics` keyword value is used to insert simple generic graphics commands in one of the existing (mini-)languages for graphics. These are properly handled by using the `graphics` and `type` keywords together.

```
\special{**graphics = "...",
        type = tpic }
```

The driver will issue an error if there is a `graphics` command without a `type` specified as well, and the corresponding `\special` will be ignored. The driver is not required to execute `graphics` except if the `type` is `dvi`.

All four `graphics \specials` accept the following options:

Keyword	Value	Action
boundingbox	x1 y1 x2 y2	Defines the four dimensions

		of the lower-left and upper-right corners of the box which bounds the figure.
clipbox	x1 y1 x2 y2	If present, clipping to the specified four dimensions should occur.
position	n m	Specify the reference point on an inserted figure which is to be mapped to the current page position.
size	x y z	three values that are absolute dimensions for the size of the figure.
type	s	gives a way to specify the type for files with non-standard extensions.

boundingbox also applies to graphics, since it can be used to decide whether and where clipping should occur. Note that this is essentially the same value as the PostScript BoundingBox for (E)PS figures. For clipping purposes, this statement overrules the in-file version of such a BoundingBox. In the absence of a boundingbox keyword, (E)PS and similar file formats where it is legal to draw outside the box should *always* be clipped to the in-file values.

The position keyword specifies two values. The first should be one of top, middle, or bottom, and the second should be one of left, center, or right. These words may be abbreviated to a single letter if desired. Together, they select on the bounding box one of nine points (four corners, four edge centers, and the box center) which is to be placed at the T<sub>E</sub>X current point. If this keyword is not given, the default is

```
position = top left
```

The point selected by this keyword (or by default) will be the *reference point* for the insertion of the graphic file.

In the values of size, negative dimensions means that size in that direction should be ignored.

The string argument to type is used to give information about the type of file or graphics. This value should be either the 'normal' three-letter extension for this type of file or the name of a graphics description language. The following language names are predefined: dv, dvi (ordinary binary dvi commands), epic, encapsulated postscript (also eps), eepic, emtex, fig, metapost (also mp), pcl, pdf, postscript (also ps), tektronics, tpic, xpic.

### Generic graphics keywords

There are three keywords that define transformations. Actually these belong to the graphics language, but they can also appear inside figure \specials, which is why they are explained here.

Keyword	Value	Action
translate	x y	Defines two dimensions that shift the figure's reference point from the default value.
scale	a b	one or two numbers that are relative to the 'normal' size of the figure.
rotate	a	rotation angle in degrees. Counterclockwise is positive.

These three keywords can be used as stand-alone commands, in which case they apply until explicitly stopped by means of one of the commands we will define below, or they can be included inside one of the four \specials for figure inclusion, in which case they only apply to the subject of that \special.

The keyword size is processed before taking any transformation commands within the same \special into account.

Rotations etc. that were in force at the time the figure \special was encountered, *are* taken into account before the calculations for inclusions are done. Here is a small example that demonstrates possible usage:

```
\special{**gsave}
\special{**scale=2 2}
  Some large text here
\special{**rotate=45}
  Large and rotated text
  \special{**include test.eps,
    rotate = 45}
  This figure is rotated 90 deg CCW
  and twice as large.
\special{**grestore}
  Back to normal
```

### Command for colour specifications

There is only one command defined for colour specification (well, actually two, since the American spelling "color" is also accepted), and one optional keyword:



Keyword	Value	Action
<code>colo(u)r</code>	?	The value should be the numbers or tokens that specify the color in the defined colour model
<code>model</code>	s	The value should be a recognizable color model name

Every driver is required to recognize the following six named values for the option string of `model`. These are the ones that define the four most commonly used colour models: `rgb`, `cmymk`, `gray`, (also known as `grey`) and `mono` (bitmap).

For all these predefined colour models, a colour is defined as one or more real numbers between 0 and 1. In the absence of a `model` keyword, drivers should take the following guess as default action: if there is one number in `colour`'s value, the colour model is `grey`. If there are three numbers, the model is `rgb`, and if four, the model is `cmymk`. All other non-qualified values signify a syntax error.

#### Commands for the in-line graphics language

First there are the commands that change the state of the graphics system's default values:

Keyword	Value	Action
<code>setlinejoin</code>	n	Select method of joining lines.
<code>setlinecap</code>	n	Selects the line ending method. One of <code>butt</code> , <code>round</code> , <code>square</code>
<code>setdash</code>	offset values	Select the dashing pattern for drawing lines.
<code>setlinewidth</code>	x	Selects the line-width.
<code>setmiterlimit</code>	a	Sets the miter limit for drawing.
<code>setoverprint</code>	n	Value is yes or no
<code>setvisible</code>	n	Value is yes or no

Note that the commands `scale`, `translate`, `rotate` and `colour` also belong to this category.

`setvisible` and `setoverprint` are supposed to compensate for overlays and underlays as well as for the background colour of the page (defined below in the section on paper settings).

Then there are commands that draw stuff:

Keyword	Value	Action
<code>moveto</code>	x y	Moves the cursor position to (x,y).
<code>lineto</code>	x y	Draws a line to (x,y).
<code>curveto</code>	x1 y1 x2 y2 x3 y3	Draws a Bézier curve where (x1,y1) and (x2,y2) are the control points and (x3,y3) is the end-point.

All three commands draw relative to the current point, and in fact, they even move the driver's idea of 'current point' just like the regular DVI commands do. If this side-effect is undesirable, the commands should be part of an explicit drawing, which is defined and drawn with one of the following commands:

Keyword	Value	Action
<code>startgraphic</code>		Indicates the beginning of a graphic.
<code>stopgraphic</code>		Analogously ends a graphic.

Inside one of those explicit figures, the drawing commands do not actually draw anything. Instead, one of the following commands should be used:

Keyword	Value	Action
<code>newpath</code>		Discards any present paths and start a new one.
<code>closepath</code>		Closes the current path.
<code>stroke</code>		Draws all the lines with the current selected pen.
<code>clip</code>		Selects the current path as the clipping path.
<code>fill</code>		Fills the current path with the current selected color.

Of course you are allowed to use the other commands too, and there might be intermixed text. Page breaks are not allowed though, since the entire graphics state will be restored to its default state at the beginning of the page. Usage of these commands is analogous to PostScript.

Alternatively, the graphics state can be saved and restored explicitly, again as in PostScript:

Keyword	Value	Action
<code>gsave</code>		Saves the graphics state. Position, current color, current

path, current clipping path, current transformation matrix, and the current pen-type is saved.

grestore Restores the graphics state.

---

**Commands for hyper-referencing**

There are not that many keywords explicitly involved with hyperlinks, since they can use the keyword `id` to mark either pages or locations. in the document. The link specification decides whether the specific `id` indicates a location marker or a page marker.

Linking re-uses the option keywords `position`, `size`, `filename` and `type` that are defined elsewhere in this paper.

Keyword	Value	Action
<code>linktopage</code>	n	The name has to be defined though an <code>id</code> elsewhere
<code>linktoloc</code>	n	The name has to be defined though an <code>id</code> elsewhere
<code>linkend</code>		Ends an HTML style link
<code>position</code>	n m	Specify the reference point of the link area.
<code>size</code>	x y z	three dimensions that are width, height and depth of the link area.
<code>filename</code>	s	This is the url in case an external file is linked to
<code>type</code>	s	gives a way to specify the type for files with non-standard extensions. The value should be the 'normal' three-letter extension for this type (like <code>pdf</code> or <code>dvi</code> ).

The value of `size`, if available, gives the borders of the 'clickable area'. An example:

```
\special{**id=1}This is a
\special{**linktoloc=1,
size=16pt 6pt 1pt}link.
```

If `size` is not explicitly given, `linkto...` functions analogous to the HTML style syntax, and `linkend` is used to stop the area. Here is an example of the this approach:

```
\special{**id=1}This is a
\special{**linktoloc 1}link%
\special{**linkend}.
```

It is a syntax error to end a link with `linkend` if that link was started with an explicit `size`, and the entire link specification will be ignored by the driver.

It is *not* an error if there is a line or even line break in the case that is supposed to end with `linkend`. These cases have to be handled correctly by the driver (the clickable area will probably have to be split in separate parts).

**Commands for meta-information**

A number of keywords is available to pass information to the processing application. This information can be used to fill `<meta>` tags or for debugging purposes.

Keyword	Value	Action
<code>info</code>	n	Value can be either meta, debug, or comment
<code>title</code>	s	Name of the current document
<code>subject</code>	s	Subject of the document
<code>author</code>	s	The (probably human) author
<code>creator</code>	s	The generating program
<code>version</code>	s	Version information
<code>keywords</code>	s	Keywords for this document
<code>abstract</code>	s	Short abstract for this document
<code>filename</code>	s	Original filename
<code>lineno</code>	a	Records original line number in source
<code>charno</code>	a	Records character location in line
<code>byteno</code>	a	Records location in file
<code>date</code>	a b c	Date in a fixed format (dd mm YYYY)
<code>time</code>	a b	Generation time in fixed (hh mm) format, assumed to be GMT

The meanings should be clear from the names. These commands can all be used inside of any other `\special` in this same group, and they can be used in the optional part of the three figure file inclusion `\specials` and as part of the `linktoloc` and `linktopage` commands if they refer to an external file, where they can be used to request a specific version of a file. (The driver does not have to honour these latter cases in order to comply, but it is required to give the usual warning about failing to process the `\special` entirely).

**Handling paper.**

Device initialization can be a complicated business, so it will usually require the `language` keyword as well (see below), but some of the more common keywords can be

defined without problems. Paper is fairly simple. There are two commands available, `paper` and `screen`.

Keyword	Value	Action
<code>paper</code>	s	paper form name
<code>screen</code>	s	screen form name
<code>height</code>	x	paper or screen height
<code>width</code>	y	paper or screen width
<code>colo(u)r</code>	?	The value should be the numbers or tokens that specify the color in the defined colour model

The `paper` and `screen` keywords defines a name that is used to tag the collected parameters. If the form name already exists, assignments will replace previous values. Otherwise, a new form is created. `screen` is intended for on-line formats, and is a synonym for `paper` that feels more natural in this case.

The use of `colour` here defines the background colour of the paper or screen. Printer drivers (or any other driver where execution of this command might lead to very expensive output) are supposed to ask confirmation from the user before executing this `\special`.

#### Other processing options

are

Keyword	Value	Action
<code>imaging_type</code>	n	The type of imaging that is applied
<code>resolution</code>	x	Gives the required resolution for device where there are more possible resolutions
<code>tray</code>	n	Tray number for devices with more then one input tray
<code>duplex</code>	n	Either on or off

The `imaging_type` can be one of the words `normal`, `negative`, `mirror` or `mirrornegative`.

The commands are used for for instance typesetter output, and they always apply to at least one full page (the page the `\special` appeared one)

#### Other device options

Certain drivers might require certain extra commands that only they understand. There is one command reserved to handle these things.

Keyword	Value	Action
<code>language</code>	n	Name the output-device language for which this <code>\special</code> is intended.
<code>literal</code>	s	Insert literal output device code.
<code>options</code>	s	Insert driver option.

The `language` keyword determines whether the DVI driver will process this `\special`, or ignore it.

Drivers are not required to understand *any* kind of `language special`, and are free to ignore that `\special` right after it has seen the `language` command. However, any driver that is willing to support this `\special`, even in a very minor way, *must* recognize a generic language choice relevant to its output device, such as PostScript or Epson. Also, each driver that tries to handle this `\special` *must* recognize its own name as a language value.

`literal` is allowed to occur *only* in combination with `language`, and is used to insert literal portions of the command language used by the `language` in question.

The `options` keyword can be used to supply device-dependent information to the driver; this is only allowed if the `language` is the name of a driver.

#### Correct driver behaviour

Drivers are supposed to correctly interpret and execute all of the `\specials` defined in this document, except were we specifically indicated that this is not needed.

If the program that processes the DVI file *does not* know how to handle a specified `\special` (other than those defined in this document), it is allowed to issue *at most one* warning to the user per unrecognized `\special` type.

Since there is a reasonable chance that this DVI file at hand should have been processed by another program altogether, one warning seems prudent, but that should be enough. This rule prevents the appearance of miriads of “unknown special” warnings in documents that have parallel `\specials` for various drivers.

If the program that processes the DVI file *does* know how to handle a certain `\special`, it is allowed to issue messages, warnings or errors as it sees fit. It is always *required* to give warnings in the case of a `\special` that can only be partly obeyed. It is also *required* to give user errors for all `\specials` that have syntax errors (assuming the driver is aware of the right syntax, which may not always be the case, but is definately the case for the `\specials` defined here)

# Toolbox: let's keep things plain

Maarten Gelderman

## abstract

This Toolbox follows the eclectic approach that most readers will know from previous ones. Without aiming to give a comprehensive oversight of the logic behind them, I present some plain  $\TeX$ -commands that can be used directly in  $\LaTeX$ .

As the editors of this journal decided to make English the preferred language for contributions, this column is no longer in Dutch. However, as far as my limited command of the language allows, I will try to keep the tone informal and the discussion accessible to novice users.

Keywords: plain  $\TeX$  in  $\LaTeX$

## Some background

Most readers of this journal will be familiar with the fact that  $\TeX$  and  $\LaTeX$  are not the same thing.  $\TeX$ , the program is the ‘machine’ that runs  $\LaTeX$ .  $\LaTeX$ , the macro-package, is the system that is run by most of us in order to process our documents. Instead of  $\LaTeX$ , we may run any other macro package. The most well-known are Context, eplain, and plain. Context is a monolithic system which instantaneously provides functionality  $\LaTeX$  can only offer if combined with a whole bunch of packages. Plain is the original  $\TeX$  macro-package developed by Donald Knuth and eplain is an extension of plain that provides some  $\LaTeX$ -like functionality without sacrificing any of the original plain possibilities

The last sentence already indicates that other macro-systems may make some plain- $\TeX$  functionality inaccessible, and indeed this is the case.  $\LaTeX$ -makes some of the plain  $\TeX$  macro's unavailable or unusable. Fortunately, the majority of those commands still works and  $\LaTeX$ -users may well use them to their advantage. This column will sketch the possibilities of some plain- $\TeX$  macro's. If will mainly focus on some commands and parameters that influence low-level formatting. The information in this column is based on documentation in `ltpplain.dtx` and the  $\TeX$ -book (especially Chapter 14).

## Line breaking and overfull boxes

One of the most advantageous areas for application of plain macro's is in manipulating hyphenation and justification.

As long as you stick to the computer modern fonts most of the standard settings work just fine. However, when you switch to some other font (say “times” by e.g. issuing the command `\usepackage{times}` or `\rmdefault{ptm}`) you may almost feel drowned by the amount of overfull boxes. Of course, you will finally get rid of those by carefully checking all hyphenation possibilities. However, while on a document this is too much work and the overfull boxes can become a real nuisance. One possible measure is to use  $\LaTeX$ 's `sloppy`-environment. However, this will affect the look of your document rather negatively. A more subtle correction can be made by `\emergencystretch`. `Emergencystretch` is the amount by which the spaces in a single line may be enlarged in order to avoid overfull boxes when the ordinary line-breaking algorithm fails. By issuing `\emergencystretch=1em` you should be able to get rid of the majority of those annoying messages.

Another way to avoid messages without improving your document is adapting the value of `\tolerance`.  $\TeX$  will only produce overfull boxes if it finds no way to break a paragraph into lines without creating ugly lines like the ones you just have been reading. In order to determine whether a line is too ugly to be acceptable,  $\TeX$  compares the ugliness of a line with the value of `\tolerance`. Standard  $\LaTeX$  sets tolerance at 200, by setting e.g. `\tolerance=400` uglier lines (more like the ones commercial word-processors produce) are permitted and fewer overfull boxes will be produced.

A final small trick that may save you the trouble of manually correcting a lot of bad hyphenation is `\slash`. It is simply used to produce a slash (/) that functions like a hyphen. Hence “`automatisering\slash` mechanisering” will be hyphenated as “`automatisering/` mechanisering”, if necessary (I forced it to be necessary here by adding yet another `\break`). You may make commands like `\slash` yourself. Just use `\def\en{-\discretionary}{\{}}` to get `xxxxxxxxxxxx-`

## Changing the length of paragraphs

In the previous section we were concerned about the ease with which acceptable output can be produced. This sec-

**Table 1.** Automatically generating a font-inventory.

```
\input multido
{\font\test pzdr at 10pt \multido{\i=0+1}{255}
{ \i: {\test\char\i},}\ 255: \char255}.
```

0: , 1: , 2: , 3: , 4: , 5: , 6: , 7: , 8: , 9: , 10: , 11: , 12: , 13: , 14: , 15: , 16: , 17: , 18: , 19: , 20: , 21: , 22: , 23: , 24: , 25: , 26: , 27: , 28: , 29: , 30: , 31: , 32: , 33: ✂, 34: ✂, 35: ✂, 36: ✂, 37: ✂, 38: ✂, 39: ✂, 40: ✂, 41: ✂, 42: ✂, 43: ✂, 44: ✂, 45: ✂, 46: ✂, 47: ✂, 48: ✂, 49: ✂, 50: ✂, 51: ✂, 52: ✂, 53: ✂, 54: ✂, 55: ✂, 56: ✂, 57: ✂, 58: ✂, 59: ✂, 60: ✂, 61: ✂, 62: ✂, 63: ✂, 64: ✂, 65: ✂, 66: ✂, 67: ✂, 68: ✂, 69: ✂, 70: ✂, 71: ✂, 72: ✂, 73: ✂, 74: ✂, 75: ✂, 76: ✂, 77: ✂, 78: ✂, 79: ✂, 80: ✂, 81: ✂, 82: ✂, 83: ✂, 84: ✂, 85: ✂, 86: ✂, 87: ✂, 88: ✂, 89: ✂, 90: ✂, 91: ✂, 92: ✂, 93: ✂, 94: ✂, 95: ✂, 96: ✂, 97: ✂, 98: ✂, 99: ✂, 100: ✂, 101: ✂, 102: ✂, 103: ✂, 104: ✂, 105: ✂, 106: ✂, 107: ✂, 108: ●, 109: ○, 110: ■, 111: □, 112: □, 113: □, 114: □, 115: ▲, 116: ▼, 117: ◆, 118: ◆, 119: ◆, 120: l, 121: l, 122: l, 123: 6, 124: 9, 125: “, 126: ”, 127: , 128: , 129: , 130: , 131: , 132: , 133: , 134: , 135: , 136: , 137: , 138: , 139: , 140: , 141: , 142: , 143: , 144: , 145: , 146: , 147: , 148: , 149: , 150: , 151: , 152: , 153: , 154: , 155: , 156: , 157: , 158: , 159: , 160: , 161: ♣, 162: ♣, 163: ♣, 164: ♥, 165: ♠, 166: ♠, 167: ♠, 168: ♣, 169: ♠, 170: ♥, 171: ♠, 172: ①, 173: ②, 174: ③, 175: ④, 176: ⑤, 177: ⑥, 178: ⑦, 179: ⑧, 180: ⑨, 181: ⑩, 182: ①, 183: ②, 184: ③, 185: ④, 186: ⑤, 187: ⑥, 188: ⑦, 189: ⑧, 190: ⑨, 191: ⑩, 192: ①, 193: ②, 194: ③, 195: ④, 196: ⑤, 197: ⑥, 198: ⑦, 199: ⑧, 200: ⑨, 201: ⑩, 202: ①, 203: ②, 204: ③, 205: ④, 206: ⑤, 207: ⑥, 208: ⑦, 209: ⑧, 210: ⑨, 211: ⑩, 212: →, 213: →, 214: ↔, 215: ↓, 216: ▲, 217: →, 218: ↗, 219: →, 220: →, 221: →, 222: →, 223: →, 224: →, 225: →, 226: →, 227: →, 228: →, 229: →, 230: →, 231: →, 232: →, 233: →, 234: →, 235: →, 236: →, 237: →, 238: →, 239: →, 240: →, 241: →, 242: →, 243: →, 244: →, 245: →, 246: →, 247: →, 248: →, 249: →, 250: →, 251: →, 252: →, 253: →, 254: →, 255: →.

tion takes the opposite approach. In the final stage of document preparation fine-tuning of the appearance of pages will be required. On a global level you may for instance want to influence the length of your document. When a book is printed, 16 pages are processed at a time. If the length of the text you are working on is 194 rather than 192 pages, sixteen additional pages have to be printed of which only two will be used. You may try to reduce the length of your text by setting `\linepenalty=nnn` (where *nnn* is a number between -10000 en 10000).  $\LaTeX$  sets this parameter at 10. By e.g. raising it to 100 you discourage the line-breaking algorithm of  $\TeX$  from making additional lines and hence may be able to reduce the overall length of your printout.

In some cases it may be desirable to manipulate the length of individual paragraphs. You can use `\looseness` to achieve this. If you set `\looseness=1`  $\TeX$  will try to make the paragraph one line longer than it would otherwise do. If you use a negative value, paragraphs will be shortened. In this way widows and orphans can be eliminated manually (see also the final paragraph of this section).

On a micro level, footnotes may cause problems. If you typeset a book with  $\LaTeX$ , and this book contains footnotes, almost inevitably some of them will be split across pages. Often for, to the human mind, no good reason at all. Setting `\interlinepenalty=10000` *locally* in the footnote, will assign infinite badness to such breaking and hence will keep the footnotetext together (if you set this variable *globally* not a single paragraph in your document will be split across pages, which may not be exactly what you want). It is not necessary to set the value to infinite

(10000) you may choose any large value in order to discourage breaking of footnotes (and paragraphs in general) across pages.

Similar to the footnote-problem is the occurrence of widows and orphans (when the first line of a paragraph is placed on another page than the remainder, this line is called a widow, if the same accident occurs to the last line, it is called an orphan). By setting `\widowpenalty` and `\clubpenalty` to a large value, such behaviour is discouraged.  $\LaTeX$  fixes both values at 150.

## Easy font changes

The subject matter discussed above is mainly concerned with low level formatting that cannot be done from  $\LaTeX$  directly. The issue presented in this section, font selection, can be done rather adequately by  $\LaTeX$  itself. However, in some cases the  $\LaTeX$ -approach is just too heavy. Take the case where we have a font file from which we want to use just a single character (this may be a dingbats font—a font with symbols rather than letters—or a Metafont-file generated by GnuPlot). Setting up such a font using the New Font Selection Scheme (NFSS) requires that you generate font definition files. The plain  $\TeX$  approach is far more simple: you just need your font and the associated TFM-file.<sup>1</sup> For the Zapf DingBats font, the name of the TFM-file is `pzdr.tfm`. Issue the command `\font\test pzdr at 10pt` (omitting the `.tfm` part of the file name)

1. TFM-files for PostScript font can be generated by the utility `AFM2TFM`.

to load this font at the required size (10 points in this case) and the font is ready for use. `\test 1234567890` will generate the following output: *Œ œ ✓ ✕ XXXX †*. A small example of the beautiful Zapf Chagnery font is produced just as easily: look up the name of the TFM-file involved in the directory `/texmf/fonts/tfm/adobe/zapfding` (it turns out to be `pzcmi`) and enter: `\font\test pzcmi at 10 pt \test This is a beautiful Hamburfont, isn's it? which produces This is a beautiful Hamburfont, isn's it?`

## Multido

In order to be able to use a DingBats font, you will need to find out where each symbol can be found. You may use the individual symbols by their corresponding letters (as was done in the example above), a more straightforward approach, however, is to access the individual symbols by the command `\charnnn`, where `nnn` is a number between 0 and 255 (a single TFM-file cannot contain more than 256 glyphs). Manually trying out all possibilities wouldn't be

too exiting of course. Fortunately the (plain) T<sub>E</sub>X-package `multido` is available. Just load it, and let T<sub>E</sub>X do the work (see Table 1). A 'translation' of the commands in Table 1 would read: input the file `multido.tex`, load the TFM-file `pzdrr` at a size of 10 points and next start repeating the command `i = i + 1` 255 times, each time executing the third argument of the `\multido`-command, which tells T<sub>E</sub>X to print `\i` and typeset character `i` from the selected font. After finishing the iterations, print 255 and the corresponding character.

## A warning

In applying the suggestion mentioned above be warned that most documents simply are not suited to be perfectly formatted. T<sub>E</sub>X will search for the best solution it can find, given the badnesses you provide it with. However, most times you will be trading of one badness against another; simply fixing all penalties and demerits at infinite will most probably result in an ugly document.

# The 19<sup>th</sup> annual T<sub>E</sub>X Users Group Meeting

Kaveh Bazargan  
Philip Taylor  
kaveh@focal.demon.co.uk  
chaa006@vms.rhbnc.ac.uk

## abstract

The 19th annual T<sub>E</sub>X Users Group Meeting took place in Toruń, Poland, in August of this year. The following is the impression of the authors who tried to attend all the talks. Unfortunately a couple of the talks were not attended by either of the authors. Apologies to those authors.

Daniel Taupin described his L<sup>A</sup>T<sub>E</sub>X to RTF converter, `ltx2rtf`. His problem is that he wants to transfer L<sup>A</sup>T<sub>E</sub>X documents to colleagues who may not have L<sup>A</sup>T<sub>E</sub>X, and not even a PostScript printer. His conclusion is that the great majority have MicroSoft Word installed, and therefore that should form the basis for the format in which the files are transferred. Since RTF can be read in directly by Word and by other word processors, and the specifications are published, it makes a logical choice for the conversion process. Of course the limitations of Word cannot be overcome. For example equations are saved and placed as bitmaps. Daniel's colourful personality and his vociferous comments at most presentations livened up the proceedings enormously.

RWD Nickalls is a consultant anaesthetist in a hospital. He uses L<sup>A</sup>T<sub>E</sub>X in conjunction with GNUPLOT to display and print charts in the operating theatre. The use of ASCII input for both GNUPLOT and L<sup>A</sup>T<sub>E</sub>X makes for a reliable system.

Marcin Woliński presented a L<sup>A</sup>T<sub>E</sub>X(2e) package for 'pretty-printing' program codes such as Pascal and Prolog directly through T<sub>E</sub>X. The results are quite amazing, with the code printing with the usual conventions of bold, italic, correct indentation, and cleverly using maths mode to position mathematical constructs. It is a good use of the unique nature of T<sub>E</sub>X, being a text processor and a programming language in one.

Metapost seems at last to have opened the door to the untapped power of Knuth's METAFONT. Metapost uses only a small subset of PostScript operators. In particular, there is no access to the PostScript pattern operators through

Metapost. Piotr Bolek's solution is to allow the user to use routines in MetaPost to define patterns, but then to post-process the file using a PERL script to insert the correct code according to PostScript level 2. His `mpattern` package consists of the metapost code to define the patterns, and the PERL script for post-processing.

Taco Hoekwater, of Kluwer Academic, laid out the procedure to generate PostScript Type 1 fonts from METAFONT sources. The major incentive for this work was to allow the inclusion of these fonts in PDF files generated by PdfT<sub>E</sub>X (Adobe Acrobat is not good at viewing bitmap fonts). The translation is not straightforward. For one thing, METAFONT has a richer set of primitives for describing font shapes and characteristics than the definitions of Type 1 fonts. Secondly, METAFONT allows the use of the equivalent of PostScript stroked paths, whereas Adobe Type 1 fonts can only be described as outlines. A reliable translator would open the way to more widespread use of METAFONT, which is undoubtedly the most powerful tool for generating typefaces. At the moment the procedure is quite long, as several manual steps are needed. Taco's paper, incidentally, is an excellent introduction to Metafont, and to Type 1 fonts.

Using his Calculator demo, Hans Hagen posed a philosophical question which emphasized the changing world of document production. 'calculator.pdf' (available from <http://www.ntg.nl/context>) is a PDF file, viewable in Adobe Acrobat Reader, which looks and behaves like a scientific calculator. So is it a program or is it a document? Well, it is a kind of intelligent document, which goes one step further than having hypertext links. Its usefulness may be trivial, but the method used to produce it is extremely significant. The 'document' was produced by allowing T<sub>E</sub>X to embed Metapost and JavaScript code, and produce one composite PDF file. Metapost produces graphics parametrically on the fly, JavaScript does the calculations behind the buttons, and T<sub>E</sub>X does the overall typesetting and management. The whole process is processed by Hans's powerful Context package, which uses PdfT<sub>E</sub>X to produce the output directly (rather than taking the DVI-PostScript-Distiller route). The Calculator demo is perhaps the single most powerful example of how relevant T<sub>E</sub>X still is in the world of electronic publishing.

Another little gem from Hans Hagen is the idea of ‘visual debugging’.  $\TeX$  has a subtle, and sometimes unconventional way of using vertical and horizontal space. When writing macros, a programmer often has to get into the mind of  $\TeX$  to work out why certain spaces have appeared in strange places, or why a paragraph is indented, etc. This can take quite a while to work out, and the answer is usually obvious in hindsight! It turns out that by carefully redefining a lot of  $\TeX$ ’s placement commands, such as `\hfil`, `\vfil`, `\kern`, `penalty` etc,  $\TeX$  can be persuaded to show graphically all spaces, fills, glue, and boxes. Hans has designed different solid and dashed lines to signify each type of box or glue placed by  $\TeX$ , thus providing a visual picture of what he calls ‘an endoscopic view of  $\TeX$ ’s stomach’.

Bogusław Jackowski and his colleagues in BOP s.c. are masters of PostScript, especially when applied to  $\TeX$ . In the first of his most entertaining and gripping deliveries, Bogusław described his CEP and COP packages for compressing PostScript files. Most people who deal with PostScript files are aware of the enormous file sizes. What is not well known is that it is possible to reduce these files substantially by using PostScript’s own compression algorithms. Most applications do not bother to use this compression when generating PS files. The most efficient compression is achieved when the type of data in the PS file is known a priori. Depending on the type of data, one of the compression programs (CEP and COP) is chosen.

In his second talk, Bogusław described a suite of unusual PostScript tools for use with  $\TeX$ . `Tiff2ps` is a program that converts Tiff files to PostScript files, the program itself having been written in PostScript. `Pf2afm` tries to extract AFM (Adobe Font Metrics) files from PFM (Printer Font Metrics) files. AFM files are needed for  $\TeX$  users, normally converted to TFM. PFM files are subsets of AFM files, and are used in Windows. Another utility, `Ttf2pf` converts TrueType fonts to AFM and Type 42 (PostScript’s equivalent of TrueType). Finally, `Colormap` is a  $\TeX$  macro package that makes simple colour and grey scale changes to EPS files that are called by the  $\TeX$  file, without modifying the EPS files themselves. The two papers have a lot of useful material regarding PostScript and fonts, and are recommended reading. One thing that comes through is the power of the freeware GhostScript program which is used in all this work.

Hàn Thê Thành, author of Pdf $\TeX$  was not present, but his paper was read by Jiří Zlatuška. His idea is to improve the look of text set on a narrow measure by horizontally scaling the text in Acrobat after  $\TeX$  has finished its work, thus reducing the large variation in word spacing. This scaling

has to be limited to only a small percentage, as otherwise the different shapes of the letters become noticeable. Also, the differing stem widths make the letters appear lighter or bolder, a generally undesirable effect.

Miroslava Misáková takes this idea of postprocessing one ingenious step further. (In fact she got the idea by looking at Gutenberg’s 42-line Bible!) She solves the problem of the differing stem width by regenerating the font using Metafont, and varying the width of the font but keeping the stem width constant—something that probably can only be done with Metafont. This makes for a more uniform looking block of text. The procedure is first to set a paragraph by allowing the right margin to vary by some 5%. The right margin is therefore ragged, but the word spacing is more uniform. Using  $\TeX$  macros and `\specials`, marks are embedded in the DVI file, signifying the positions of the ends of the lines. A Perl script then interrogates the DVI file, and for each line, works out the correct font to use for setting. Usually, it would look at a set of 10 fonts already generated, and choose the closest match. The results presented are most impressive. Even when adjoining lines vary in letter width by some 7%, the text looks uniform, and is comfortable to read. Of course the ideal solution would be to add this capability of variable font width to  $\TeX$  as one of its parameters, but that becomes a more complicated procedure.

Richard Kinch’s contribution was a new math font, Belleek, that he put into the public domain. The font was produced manually, using Fontographer, and designed to work with Times as the body text. His conclusion from the experience of creating this font is that Metafont is the best tool for font creation, but that the main stumbling block for its more popular use is that it creates bitmap output, as opposed to outlines. He discussed some methods for getting outlines from Metafont.

The presentation that created the most heated discussion was kept for the last day. This was the joint presentation by the Dutch contingent (NTG), who had arrived in an orange coloured van at the conference. The Dutch team, led by Hans Hagen, presented some radical ideas. These ideas were a culmination of lively discussions among Dutch  $\TeX$  users during the months prior to the conference, and led to the ‘NTG  $\TeX$  future working group’. They felt that they had to share these ideas with the  $\TeX$  community. Here are some of the highlights of their views:

- ▣ It is proposed that the three projects currently under way to extend the capabilities of  $\TeX$ , namely Omega, Pdf $\TeX$  and e- $\TeX$ , should merge into one project. NTS (the New Typesetting System) is a project to



rewrite T<sub>E</sub>X in Java, with a guaranteed backward compatibility with T<sub>E</sub>X82, at least for the first 5 years. The Dutch team feel that this is a mistake, and that backward compatibility should be discarded.

- ▣ The group feels strongly that current distribution and implementation methods for T<sub>E</sub>X are too complicated, incompatible, and involve too many files. A typical complete T<sub>E</sub>X system can comprise some 30,000 files, which the Dutch group feel should be reduced by a factor of 100. This would be achieved by setting up an intelligent central database which could be queried, and which would install only the necessary files according to a user's specifications. This database could have access to CTAN to obtain any files not available locally. It is also proposed that CTAN be more closely regulated with files and packages being put up with more information accompanying them.
- ▣ Another suggestion is to extend the capabilities of T<sub>E</sub>X to allow the production of electronic documentation. These include more support for colour and hypertext linking within T<sub>E</sub>X, rather than relying on `\specials` to produce these capabilities at the back end. To allow the direct inclusion of Metapost graphics, a set of `\specials` is proposed which allows the use a larger subset of PostScript commands than is used in Metapost.
- ▣ Fonts have always caused problems for all T<sub>E</sub>X users apart from real experts. The NTG T<sub>E</sub>X working group recommend an overhaul of the font system, including replacing the way TFM and VF formats are defined, separation of ligatures and kerning information from the character metrics of fonts, and the facility for direct output of outline fonts from Metafont. (The latter echoes the wishes of Richard Kinch.) The group also call for a friendly user interface for mathematics, and a general cleaning up of the T<sub>E</sub>X syntax.
- ▣ The final proposal is a standard for the specification of `\specials` accross all systems. The proposal is based on that of Nelson Beebe, and the aim is to make T<sub>E</sub>X truly portable.

Laurence Finston presented what should have been an extremely interesting paper on the generation of concordances using a combination of T<sub>E</sub>X & Lisp. This paper was rather difficult to understand by reading it, so we looked forward to Laurence's oral presentation. Unfortunately, he chose to *read* his paper rather than presenting it extempore, as a result of which his audience (including your humble scribes) started to sneak out after the first five minutes. We learn that M. Taupin asked at the end if Laurence considered TUG'98 merely a practice run for his dissertation...!

Janusz Nowacki, speaking through a translator, presented his work on Antykwa Toruńska, a modern digital adaptation of a traditional Polish font. Janusz commenced his talk by recounting the motivation for his work: the simple fact that since the DTP/EP revolution, no Polish text had been produced using a Polish font! Despite the problems of translation, Janusz kept his audience enthralled, and his talk was one of the most interesting and stimulating delivered. Antykwa Toruńska in its current version originates as scanned bitmaps; it is then post-processed in Corel Draw and fine-tuned in Fontographer. It is available in regular, italic and bold versions.

Karel Skoupý presented a report on the current status of NTS ("a New Typesetting System"). This project, which was first mooted in 1992 and which has only recently sprung into life, involves a complete re-implementation of T<sub>E</sub>X using Sun's JAVA programming language. Rather than simply being a re-implementation, however, the project's real aim is to completely restructure T<sub>E</sub>X/NTS internally so as to make the inter-module interfaces far clearer and to provide a framework within which current (for example e-T<sub>E</sub>X, Omega and PdfT<sub>E</sub>X) and future developments can be easily and straightforwardly embedded. Karel emphasized that the choice of JAVA had many benefits, of which guaranteed portability and network awareness were the greatest, but also had its drawbacks (for example, the lack of rigorous compile-time type-checking for derived types).

Although the majority of papers took place on schedule, the unanticipated absence of John Plaiice (originator of the Omega project) caused some gaps in the schedule. These were filled by extempore talks by Kaveh Bazargan ("Can T<sub>E</sub>X create Farsi scripts?"), Daniel Taupin ("L<sup>A</sup>T<sub>E</sub>X<sub>2</sub>HTML for the i'x86 family of operating systems), and Gilbert van den Dobbelen ("DVIVIEW : a reference previewer").

The four days surrounding the conference proper were occupied by tutorials. Hans Hagen had bravely volunteered to give a split two-day tutorial entitled "Actually making an electronic document"; Philip Taylor & Jiří Zlatuška offered a preview of a tutorial originally prepared for IFIP'98 ("Document design, document markup, and the converging worlds of computer typesetting and electronic publishing"); and Bogusław Jackowski offered a one-day tutorial on "T<sub>E</sub>X and PostScript integration". All three were well attended, but the Taylor/Zlatuška tutorial rather fell apart when the participants agreed that it was either too advanced or too basic but certainly not the right level... Hans' and Bogusław's tutorials, on the other hand, both proved very successful.

Of course, a conference is considerably more than a series of tutorials and talks: the social aspects are in many ways as important if not more so. From this perspective TUG'98 was a great success: the organizers had carefully planned the social activities, and the gala concert and banquet at Artus' Hall ("King Arthur's Hall") was one of the high spots. Equally memorable was the outing to the mediæval castle at Golub-Dobrzyn, where the guest were greeted by the sight of an ancient and wizened serf, clad in chain mail, slowly roasting a pig on the spit. There was limitless beer to accompany this feast, and a guided tour of the castle; guests of a weak disposition were probably somewhat discomfitted to learn of the mildly unpleasant ways in which "guests" of an earlier age were accommo-

dated: simple pleasures such as being tied down under a slow but continuous water drip, or having salt rubbed into fresh wounds, were among the many ways in which their hosts passed away the weary hours... Despite (or perhaps because of) this, the sight of the pig being slowly roasted conjured up in several minds an analogous scene in which the central participant was rather more anthropoid than porcine... Simpler pleasures were also to be found every evening, when beer, vodka, guitars, whistle and voice mingled together to create an atmosphere of great conviviality and warmth. All in all we were impressed by the genuine hospitality that we received by from the local organizing party.

# Bijlage 6

## Parameterized data for tables in T<sub>E</sub>X Dynamics, aha!

Kees van der Laan  
cg1@hetnet.nl

### abstract

The issue of generating and using parameterized data for tables in T<sub>E</sub>X is elaborated upon. The automatic insertion of markup along with the use of parameterized markup and the `\btable` macro is not the issue, but are prerequisites. `\btable`'s use, along with the automatic insertion of markup in the data as such, can be seen as tools, stepping stones, which made it possible to concentrate on pure data generation, or its use in typesetting tables by T<sub>E</sub>X.

### keywords

BLUe, `btable`, code tables, data generation, dynamical markup, education, macro writing, parameterization, tables, tail recursion

## Introduction

Tables comprise a bewildering variety. A glimpse of this variety was presented at the EuroT<sub>E</sub>X 92 at Prague, and included as examples in the tables chapter of my PWT.<sup>1</sup>

For this note I like to discriminate between two kind of parametrizations

- parametrization of the markup in tags, like `\ruled`, `\framed`, and similar things, and
- parameterization of the data, that is data which are generated or taken from a database as function of a parameter.

In this note I'll not discuss the parameterization of the markup. That has been done along with my `\btable` macro, which is at the heart of BLUeT<sub>E</sub>X's tables. However, because `\btable` is used I'll discuss its use.

### A little about bordered tables

In BLUeT<sub>E</sub>X the philosophy behind table markup is to separate markup of the data from the markup of the border, or as I put it to separate the markup of the first column and row from the data proper. The markup for the data proper is independent from whether the table is ruled or not, and so on. This is parameterized in the column and row separators. The reason behind 'bordering' is similar to the reason to treat bordered matrices as a separate group, I guess. A border of a table, especially the first row, is usually more

complicated to format than the proper entries of the table. In the PWT user guide I introduced a model for a bordered table.

**Use of `\btable`** The above ideas about bordering of tables have been implemented in `\btable`, borrowed from BLUeT<sub>E</sub>X. To explain the coding of the `\btable` macro is not at stake; to tell a little about how to use it might be handy in understanding this note.

In the following example the use of `\btable` and its parameterization of the markup are shown in a nutshell.

**Example:** Use of `\btable` and parameterizations

			Caption		
11 12 21 22	1 <sup>st</sup> row 2 <sup>nd</sup> row	Header 11 12 21 22	Header 11 12 21 22	1 <sup>st</sup> row 2 <sup>nd</sup> row	11 12 21 22
			Footer		

It is important to realize that the core, and invariant, markup of all the three representations is just the `\data`<sup>2</sup>

```
\def\data{11\cs12\rs
          21\cs22}
\btable\data
```

where `\cs` denotes column separator, and `\rs` denotes row separator, of which the appearance in print depends on tags like `\framed` or `\ruled`. Whether the table takes an explanatory first row, a caption, rules, or similar things, these have no influence on the markup of the data proper. I consider this as a sort of abstract markup, which is related to dynamical markup, because variations of representation have been accounted for, while the specification of the

---

<sup>1</sup> PWT denotes Publishing with T<sub>E</sub>X, the user guide which comes with BLUeT<sub>E</sub>X.  
<sup>2</sup> The `\cs` and `\rs` can be omitted, see my note on 'Minimal Markup.'  
<sup>3</sup> This abstraction from plain T<sub>E</sub>X's `&` as separator is also useful

data remains invariant. This approach helps among other things to recognize in the markup more easily the number of columns of a table.<sup>3</sup>

The essentials for its use can be distilled from the example given above.<sup>4</sup> The data are expected as replacement text of the macro `\data`, and separated by `\cs` and `\rs`, the (pre-fab, already provided for) tags for column and row separation. Important is that the table is thought of as a bordered table, similar to the idea of bordered matrix as treated in TUGBOAT. The border consists of the ‘`II`’-element first, to be optionally specified as a def `\first`, the border top row to be optionally specified as the def `\header`, and the first column to be optionally specified as replacement text of the def `\rowstblst` as ‘groups’<sup>5</sup> without further separation. If the optionals are not defined `\btable` won’t complain. (This holds too for `\btablecaption` and `\footer`.) The invoke is just `\btable\data` preceded by optional tags like `\ruled`, `\framed` or similar things, eventually within a math display. So, it looks like an intelligent, robust macro, which won’t complain when you specify `\first` without a `\rowstblst` or a `\header`. In that case the results will most likely not look like what you had in mind. `\btable` tries to make something nice out of what you supplied.

#### Parameterized data

The data for the tables treated in this note are either provided by

- a program, or
- read from a file.

The program will not be recognized as a table, nor will it be easy to tell how many columns and rows the table will consist of.

#### Why?

Data as programs comes in naturally when we think of a bunch of tables, which are closely related. Examples are the guide cards for a bridge tournament—which actually triggered this note—or Wietse’s data—his (employer’s database of tables—the real life examples.

Roughly seven years ago, I coded my first dynamical and parameterized ‘table’ in T<sub>E</sub>X: the various stadia of the towers of Hanoi game, while attending the advanced T<sub>E</sub>X class of David Salomon.<sup>6</sup> In that ‘markup’ I made heavily use of the concept of an active list separator, which I encountered for the first time in TUGBOAT, and have used happily ever since. Moreover, the data are generated and parametrized over the size.

Disclaimer. This note is not about the look-and-feel of tables. The appearance of the discussed tables is straightforward, simple and time-proven, and belong to the class of bordered tables.

## Multiplication table

This example was introduced in the T<sub>E</sub>X world by Pittman in the eighties may serve to illustrate the issue of data generation for tables.<sup>7</sup> Maybe, it forms the simplest example of a parameterized, bordered table.

**Example:** Multiplication table

×	1	2	3
1	1	2	3
2	2	4	6

#### What is the problem, doc?

I guess that people would argue that the numbers and lines could just be marked up for in whatever T<sub>E</sub>X flavour you wish, and that is it. Agreed, that is true. See for example the straight markup à la BLUe given below.<sup>8</sup>

```
\def\first{$\times$}
\def\header{1\cs2\cs3}
\def\rowstblst{12}
\def\data{1\cs2\cs3\rs
          2\cs4\cs6}
\btable\data
```

But... what if we wish to vary for the size or to vary with respect to the rules of the table? And in general, what can be learned from the example? My answer is positive. Indeed, some paradigms in T<sub>E</sub>X macro writing can be distilled from the example marked up as function of its size.

---

for complicated tables with blocks in it. See the PWT guide for examples.

- 4 There is also a two-part variant of the macro but this will not be used in this note. I have not included the model picture for the `\btable` macro either, because this note as such is not about the macro. For a picture of the model and more elaborate description of the use of `\btable`, if needed, see the chapter on tables of the PWT guide.
- 5 Well, as arguments for a macro. The empty bordered table, without data is something I did not foresee. One can think of these as a sort of fill-in form, roster or calendar.
- 6 See TUGboat, 1992, or MAPS 92.1, or the Tables chapter of the PWT guide. If people don’t wish to look upon this as a table, that is fine with me. IMHO, with all respect, a taxonomy of tables is not yet in sight, because of the diversity and ambiguity of the subject.
- 7 Pittman used it to illustrate nested loops in T<sub>E</sub>X. I consider it as a table with data generated (tail) recursively, which is equivalent to looping.
- 8 The separation between the header and row-stub-list is default a rule in BLUe. Because of the different functions it is quite natural, IMHO, with all respect, to separate the header and rowstubs from the table proper.

The drawback is that the resulting markup is hardly accessible for an ordinary, non-programmer user. Don't be afraid, hang-on, and go for it.

### Tail recursion in T<sub>E</sub>X

On several occasions I found it rather useful that I knew how to code tail recursion—and its termination—in T<sub>E</sub>X.<sup>9</sup> The following generates the numbers 0–9. The simplest example I could think of in order to illustrate the idea.

```
\def\0{\the\i\advance\i1 \ifnum\i=10 \9\fi
\0}
\def\9#1\0{\fi} %The recursion terminator
\0
```

Explanation. The macro `\0` invokes itself, and before doing so formats and increases the value of the counter `\i`. So far an infinite loop, and simple to understand. Now comes the tricky, well unusual, part. The condition is also straightforward but what happens in the true branch? The macro `\9` is invoked, which takes as argument all up to and including `\0`, meaning there is no further invoke, aha... termination. Nice, but ... in the termination process also the `\fi` is eaten, and therefore the replacement text of `\9` must provide a `\fi`. Amazing, isn't it? Confusing? After a while, you will be used to it, don't worry too much, it is not that complicated IMHO, ... just unusual.

To check your understanding, and demonstrating en-passant that it is worthwhile to spend some time on it, the following example of a loop as a real tail recursion encoding in T<sub>E</sub>X, due to van der Goot.<sup>10</sup>

```
\def\loop#1\pool{#1\loop#1\pool}
\def\break#1\pool{}
```

The toy example can be marked up by the above loop as follows

```
\loop \the\i \advance\i1
\ifnum\i=10 \expandafter\break\fi
\pool
```

Remark. Sometimes, especially in nested loops, it is better to have `\fi` as replacement text of `\break` instead of the use of `\expandafter`.

### The multiplication table via tail recursion

I like to split up the generation of what is needed in three.

How to generate the

- ▣ header row
- ▣ data proper, and
- ▣ first column.

Next to the generation of the data I inserted en-passant the markup tags for column and row separation, `\cs` and `\rs`, respectively.

**The header row** The first row, which is called `\header` in BLUe, is a straightforward elaboration of the toy example of generating the digits 0-9 via tail recursion, treated above. However, now we must increase globally, because we are within plain T<sub>E</sub>X's `\halign` reign. The problem is to generate the numbers 1, 2, ... (up to the order `\n`), separated by the markup for the column separator `\cs`.

```
\def\header{\the\j \global\advance\j1
\ifnum\j>\n \redaeh\fi
\cs\header}
\def\redaeh#1\header{\fi}
```

**The data proper** This is a little harder because we have to apply the tail recursion nested: on the outer level for the rows, and within each row for the columns.

```
\newcount\i%row index
\newcount\j%column index
\newcount\n%order
\newcount\entry
\def\rows{\global\j1 \cols\global\advance\i1
\ifnum\i>\n \swor\fi
\rs\rows}
\def\swor#1\rows{\fi}
\def\cols{\entry\i \multiply\entry\j \the\entry
\global\advance\j1
\ifnum\j>\n\sloc\fi
\cs\cols}
\def\sloc#1\cols{\fi}
%
```

```
\global\i1 \n3 \framed\table\rows
```

By the way, `\framed` does what its name suggests: the table is framed.

**Are you still there?** To finish up the dynamical 'markup' we have to code for the first column, also called row-stub-list in BLUe. This is tricky, admitted. One has to know how `\rowstblst` is processed. Once this is known, it is not that difficult anymore.

```
\def\rowstblst{Anything, just phoney}
\def\nxtrs{\the\i\rss}%overriding definition
```

Explanation. The phoney definition is there for fooling `\btable`, to let it think that a row-stub-list is there. Agreed, a real hack. Then, the real thing, redefine `\nxtrs` to yield the rowstubs on turns.

<sup>9</sup> By the way, I had to unlearn never thinking in infinite loops. In this case, however, it was useful to code an infinite situation first and then account for the termination.

<sup>10</sup> Loops have been dealt with in my 'Paradigms: Loops,' MAPS 17.

### What I needed once

Of late I was asked as a tournament director for a bridge drive and faced the problem to provide for guide cards.<sup>11</sup> The scheme I used—Mitchell—is defined as follows.

For a tournament with  $n$  tables and  $n$  rounds, that is  $2n$  pairs

- at table 1 pair 1 as NZ meets pair 2 as EW,
- at table 2 pair 3 as NZ meets pair 4 as EW, etc.
- After each round NZ moves up and EW moves down 1 table.

In the concrete case  $n$  was 14. I took the number of tables (half the number of pairs) as parameter. Below I have taken for convenience  $n = 7$ .

Mitchell 14, 7 rounds

T1	T2	T3	T4	T5	T6	T7
1-2	3-4	5-6	7-8	9-10	11-12	13-14
13-4	1-6	3-8	5-10	7-12	9-14	11-2
11-6	13-8	1-10	3-12	5-14	7-2	9-4
9-8	11-10	13-12	1-14	3-2	5-4	7-6
7-10	9-12	11-14	13-2	1-4	3-6	5-8
5-12	7-14	9-2	11-4	13-6	1-8	3-10
3-14	5-2	7-4	9-6	11-8	13-10	1-12

How to program this table? Of course we can provide all the data explicitly, with the advantage of simplicity and readability, but with the disadvantage of susceptibility for errors and that it has to be redone for each value of the parameter. More elegant is to program the table in the same spirit as the multiplication table above. Because it is so similar I'll just give the code, modulo some syntactic sugar.

```

\newcount\NZ\newcount\EW
\newcount\NZ\newcount\ew
\newcount\i\newcount\j
\newcount\N\newcount\T\newcount\T2\newcount\T3\newcount\T4\newcount\T5\newcount\T6\newcount\T7
\NZ3 \EW0 \i0 \j0 \N7 \T2\T3\T4\T5\T6\T7 \multiply\T\newcount\T2\newcount\T3\newcount\T4\newcount\T5\newcount\T6\newcount\T7
\def\btblecaption{Mitchell 14, 7 rounds}
\def\header{\global\advance\j1 T\the\j
\ifnum\j=\N \global\j0 \redaeh\fi
\cs\header}
\def\redaeh#1\header{\fi}
%data
\def\rows{\global\advance\i1 \global\j0
\global\advance\NZ-2
\ifnum1>\NZ \global\advance\NZ\T\newcount\T2\newcount\T3\newcount\T4\newcount\T5\newcount\T6\newcount\T7

```

```

\NZ\NZ
\global\advance\EW2
\ifnum\EW>14 \global\advance\EW-\T\newcount\T2\newcount\T3\newcount\T4\newcount\T5\newcount\T6\newcount\T7
\ew\EW
\cols
\ifnum\i=\N\swor\fi
\rs\rows}
\def\swor#1\rows{\fi}
\def\cols{\the\NZ--\the\ew
\global\advance\NZ2
\ifnum\NZ>\T\newcount\T2\newcount\T3\newcount\T4\newcount\T5\newcount\T6\newcount\T7
\global\advance\ew2
\ifnum\ew>\T\newcount\T2\newcount\T3\newcount\T4\newcount\T5\newcount\T6\newcount\T7
\global\advance\j1
\ifnum\j=\N\sloc\fi
\cs\cols}
\def\sloc#1\cols{\fi}
%There you go
\ruled\framed\btble\rows

```

In a Mitchell scheme the pairs don't really need a guide card, because of the simple process of going from one table to the next, c.q. the previous. The guide card is generally given to each pair in order that they know at which table in what direction they play in each round, despite the heat of the tournament. Below such a guide card has been included.

**Example:** Guide card parameterized over pair number

Paar 13 – NZ

Ronde	Tafel	Tegen	Spellen
1	7	14	25-28
2	8	18	1-4
3	9	22	5-8
4	10	26	9-12
5	11	2	13-16
6	12	6	17-20
7	13	10	21-24

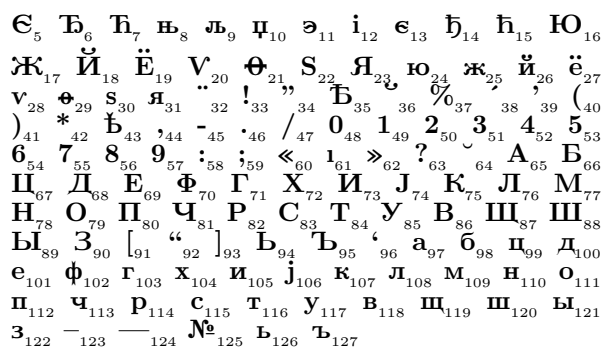
Remark. It is tempting to ponder about such a scheme as a database from which the required tables can be generated via for example the commands

<sup>11</sup> Guide cards guide pairs through the tournament, prompted by the competition and the scheme chosen. One class is called the Mitchell scheme.

```
\mitchell{14}
\pair13\mitchell{14}
```

### Font chart alternative

Along with BLUe Sky's TeXtures I found an alternative concise coding for the 'font charts.' In TUGBOAT the code tables are parameterized over the fonts and the entries are generated by `\normalchart`. Below a variant of 'normalchart' is included.<sup>12</sup>



The above has been obtained as follows.

```
\font\test=wncyb10 \relax
\def\c#1{\setbox0=\hbox{\test\char#1}%
\ifdim\wd0>0pt\box0\lower3pt
\hbox{\fiverm\the#1}
\fi}
\noindent{\count0=0
\loop\c{\count0}
\ifnum\count0<128 \advance\count0 by1
\repeat}
```

### Data from a file

Suppose you have data on a file separated by spaces and by lines. How to typeset these with TeX?

I did read the data line by line to insert the `\rs` and for each line between the data I inserted the `\cs`. For a tiny in-principle example it worked. This has more to do with automatically insertion of markup by TeX than with data generation.

### Acknowledgements

As usual Jos Winnink proofed the paper and helped me in coercing the note into MAPS format. His remarks and suggestions are always well-taken. To say the least he reflects what despite the intention did not come across. I consider the approach of a friendly eye better than a referee in

warranting quality.

Wietse Dol suggested to consider 'data-driven' tables, or as he did in Pascal to typeset tables from a database of table data.

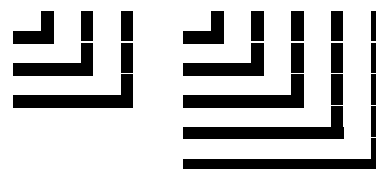
### Conclusions

The unusual, dynamical and parameterized markup of tables is completely different from the example material as treated in TUGBOAT in the alignment chapter.<sup>13</sup> I encountered in practice the need for parameterization table markup next to the generation of the data, when I had to create bridge fill-in forms, score sheets, and guide cards for each pair, all of varying size.

In the PWT guide the Pascal triangle, for example, has been marked up as function of its order. Crosswords are parameterized over whether the puzzle or the solution is wanted, and are data-driven. Moreover, the following gnomons, which have been discussed in TUGboat some years ago, can be found in the PWT guide as another example of generated data and unusual markup in general, not to mention the included charts and trees. Do have a try in parameterization of your table markup and generation of data.

1	3	5	7	9
1	4	7	10	13
1	5	9	13	17
1	6	11	16	21
1	7	13	19	25

Or its graphical counter part.



And, . . . let me know about your examples. I welcome comments.

My case rests. Have fun, and all the best

<sup>12</sup> I don't know who the author is, I'm sorry, but found it handy myself to find out what should be keyboarded or put in a chart for unusual fonts.  
<sup>13</sup> However, the markup for the font tables is parameterized over the fonts.

# Poor man's cyrillics on a Mac

## T<sub>E</sub>Xing English and Russian

Kees van der Laan  
cg1@hetnet.nl

### abstract

The typesetting in T<sub>E</sub>X of English mixed with Russian is discussed for a Macintosh. Starting from the WNCY-fonts only a few extra control sequences have to be remembered for the keyboarding. The approach is suitable for all machines which have the WNCY-fonts. Email in cyrillics can be handled by forming via pdfT<sub>E</sub>X and sending the .pdf file as attachment in a MIME message.

### keywords

Cyrillics, MIME, Macintosh, PDF(T<sub>E</sub>X)

## Introduction

Что такое? Это, ... посмотрим.

How to typeset in T<sub>E</sub>X English mixed with Russian on a Macintosh, that is the problem.

It is not standard available, and it has become difficult because fonts are treated a bit confusing on a Macintosh.

In the beginning all fonts were bitmaps. Then with Apple Laserwriter PostScript fonts were added, and because of their use for printing they were nicknamed printer fonts. Then Apple in a joint venture with Microsoft started with their TrueType fonts, which are similar to, but different from, PostScript.

Although Adobe's Type Manager allows PostScript fonts to be displayed on a screen, one bitmap is still needed to locate the fonts. Next to that Macintoshes come with TrueType fonts. All-in-all quite confusing. An extra inconvenience is that T<sub>E</sub>X requires TFM (T<sub>E</sub>X Font Metric) files while PostScript fonts have their AFM (Adobe's Font Metric) files (It is true that AFM to TFM tools exist via the editor edmetrics, but it is not trivial to perform the transformation).

So what to do? Ну вот?

I asked friends, but alas all came with partial solutions. For some the bitmap was not there or the PostScript fonts were lacking. Others just provided me with TrueType fonts, accompanied by tools to assist keyboarding. Some were aimed at a WorkPerfect environment. And so on. The confusion for me was too great.

### To whom it may concern

In this note I'm not aiming at a fool-proof solution but just report about my ability to typeset simple cyrillics along with my english texts (of a PrePrinT nature) with anyT<sub>E</sub>X, Allowing me to view and print the result with available drivers with little overhead with respect to keyboarding (the latter not specific to Macintoshes). It worked under T<sub>E</sub>Xtures and CMacT<sub>E</sub>X.

### The public domain WNCY family

When I extended my Powerbook 150 and Classic II (which together with my LaserWriter 4/600 formed a local apple-talk network, just as extra and for free) with a real PowerMac 5500/275, I also looked into which T<sub>E</sub>X & Co environment would be beneficial for some time to come. I was happily surprised by CMacT<sub>E</sub>X's availability of pdfT<sub>E</sub>X, and MetaFont/Post to name but two tools which interested me of late. However, T<sub>E</sub>Xtures also provided the fonts WNCYB, WNCYI, WNCYR, WNCYSC, WNCYSS, in PostScript, along with bitmaps at 10pt, and apparently the good metric files: I could view them in the menu option 'showfonts,' telling me that a bitmap was available and that they were scalable towards all sizes for a printer, meaning that the PostScript variants and the metric TFM and AFM files were available too. A few finger exercises made me happy.

### Keyboarding

A print of a font table was easily done. I figured out that the keystrokes correspond to the phonetical equivalent of the cyrillic character. The т, pronounced as t, was under the t-key and so on. Only a few characters I had to cope with differently. I chose to handle them by T<sub>E</sub>X control sequences, for example я by \ja, which is implemented as `\def\ja{\char31 }`.

I made the following table for myself which would assist me to remember what key or control sequence corresponds to a certain cyrillic character. The lookup is roman alphabetical and from the cyrillics to the key or command. The right part of the table contains the characters that are not mapped under the 26 characters of our a b c . . .



a	a	э	\e
б	b	ю	\ju
ц	c	ж	\zh
е	e	й	\j
д	d	ё	\jo
ф	f	ь	\s
г	g	ъ	\h
х	h	э	\E
и	i	Ю	\Ju
ј	j	Ж	\Zh
к	k	Й	\J
л	l	Ё	\Jo
м	m	Б	\S
н	n	Ъ	\H
о	o	№	\No
п	p	Ѓ	+
ч	q		
р	r		
с	s		
т	t		
у	u		
в	v		
щ	w		
ш	x		
ы	y		
з	z		

wncyr10 lookup table

### Stressings

Russian is a difficult language. All problems I know of from Dutch, German, English, French and Spanish, come back in Russian, and that is just the beginning. Like in English one can hardly guess the pronunciation and therefore, for foreigners text takes stressings. This is especially

true in the Easy Readers series. Unfortunately the accent in the WNCY font table is used for different purposes, so I defined the accent within \cy's reign appropriately as an active character.

Undoubtedly, the precise positioning of the stress mark is wrong, but for my purposes it is good enough. My knowledge of Russian is such that I commit more severe offences against the language.

Text with stressings is for example

Ба́ю-Ба́юшки-Ба́ю

with input {\cy B'a\ju-B'a{\ju}xki-Ba'\ju}.

### Acknowledgements

As usual Jos Winnink helped me in coercing this note into MAPS format. Большой спасибо!

### Conclusions

It seems that the WNCY group allowed me to typeset by T<sub>E</sub>X simple cyrillics and English mixed on a Macintosh, portable at the expense of that I have to remember a few control sequences to account for characters not available in the roman alphabet. However, it is not the most pleasing font I can think of.

With pdfT<sub>E</sub>X I was able to send my first real Russian-English email: a PDF (Portable Document Format) file, sent by MIME (Multipurpose Internet Mail Exchange).

A fool-proof and richer mixing of cyrillics and roman needs more attention.

My case rests. Have fun, and all the best

До свидания

# Bijlage 8

## Minimal markup expansion in the gullet, aha!

Kees van der Laan  
cgl@hetnet.nl

### abstract

A plea is made for a reappraisal of T<sub>E</sub>X's capabilities of expansion in the gullet of minimal marked up scripts into complete marked up scripts. Attention is focused on expansion of implicitly marked up table data by spaces and e-o-l-s into data separated by `\cs` and `\rs`, the abstract but explicit column and row separators, respectively. The ultimate aim is that the processing tool can't be distilled from the 'marked up' script.

### keywords

BLUe, crosswords, expansion, fifo, look ahead, macro writing, minimal markup, mouth processing, preprocessing, tables, tail recursion

## Introduction

Who cares about minimal markup as long as the results in print look beautiful?

Well, . . . I do.

The reason why I care about minimal markup comes from various experiences.

The main reason is: it's elegant, IMHO with all respect, when a user can prepare his copy in such a way that we can hardly tell that he works with (La)T<sub>E</sub>X. This comes close to abstraction from the processing tool, and I for one consider this beneficial, if not for the entailed flexibility, or the ease in converting a BLUe script into a MAPS submission, with the ultimate goal that conversion is no longer needed. Second, I like to read marked up scripts next to the results in print. Redundancy, to say the least, in markup does not contribute to readability nor clean scripts.

Third, many a (La)T<sub>E</sub>X result does not look beautiful at all, and maybe because authors got lost in the complexity of too much and redundant markup.

Fourth, now and then I could just generate the data for tables as function of its size by T<sub>E</sub>X, and so the user does not have to supply the data for those tables, let alone the markup.<sup>1</sup>

And last but not least, it allows you to become 'lazy,' to be able to forget about most of the details of the tool, but . . . only after you have matured, gained a thorough understanding.

In BLUe I adopted, similar to Knuth, that the `\blueheads`

take as end separator of the heading title a blank line, and so on.<sup>2</sup>

There is one example I'm particular fond of: the minimal markup for crosswords.<sup>3</sup> For crosswords I considered it particularly convenient that a user only has to provide the data between the tags `\bdata` and `\edata`,<sup>4</sup> without explicit separators. Just the letters and the \* to denote a black cell, with the column separators implicit because the elements consist of only one character, and as row separators the end of lines, e-o-l-s for short. T<sub>E</sub>X's mouth-gullet while expanding should insert the tags required by subsequent macros.<sup>5</sup> In general I found it beneficial to abstract from the markup of the table data in `\cs` and `\rs`, the column and row separators, respectively.

Disclaimer. Alas, as yet I can't read the T<sub>E</sub>X WEB source<sup>6</sup> so I don't know what `\noexpand\cs` really entails. Obviously, the gullet refrains from expansion. But eventually it has to be expanded and processed, of course. All below is biased by my intuition and has been verified, interactively.

**Example (Crosswords)** For the discussion at the moment concentrate your attention on how the data between `\bdata` and `\edata`, are specified, that is each row as such on a line.

<sup>1</sup> See my note 'Parameterized data for tables in T<sub>E</sub>X.'

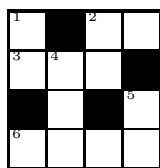
<sup>2</sup> A trifle in markup, but it paved the way towards the approach of starting without markup. More advanced is the need for processing on-the-fly, that is with catcodes to be fixed while processing. For that some more markup is needed: the two-part variant consisting of the pair `\beginhead \endhead`, with its `\head{. . .}` variant.

<sup>3</sup> I know of work done and published in TUGboat by Brian Hamilton Kelley, and I myself published a note about it in the EuroT<sub>E</sub>X'92 proceedings, in MAPS 92.2, and included a crossword example in the Publishing with T<sub>E</sub>X user guide, PWT for short, which comes with BLUeT<sub>E</sub>X. Crossword macros are included as one of the tools in BLUe's tools.dat.

<sup>4</sup> An even more minimal variant, a `\bluedata` macro with a blank line as end separator of the data can be provided on top as follows: `\def\bluedata#1\par{\bdata#1\edata}`.

<sup>5</sup> A white lie. T<sub>E</sub>X has coupled the mouth-gullet-stomach-gastro-intestinal track by 'pipes.' As soon as some expansion has been done the stomach already starts promulgating it, and so on. In the old computer terminology T<sub>E</sub>X's mouth functioning can be seen as a preprocessor.

<sup>6</sup> METAFONT and PostScript ranked higher on my priority list.



The (left) puzzle above is obtained essentially via<sup>7</sup>

```
\bdata%
P*On
DEk*
*n*S
Edit
\edata \crw
```

Note that the numbers for the clues in the puzzle are generated automatically. The user must take care that these numbers correspond with the clues by using capital characters in the data at the right places—these places get their numbers on turns, rowwise—and that the clues take the correct numbers.

While glancing through this note the markup for the crossword might look so natural to you that you might wonder

Is that all?

Indeed for a typewriter the above supplied data, for example, is all what is needed, if we consider \* as black cell. However, for the puzzle with numbers for the clues inserted we are at loss, definitely when we wish not to bother about it as user. But, ... it needs a touch of knowledge, awareness, wisdom and discipline not to pollute the data by markup, while  $\TeX$ ing for beautiful results, such as black cells, automatic generation and typesetting of numbers for the clues, rules, scaled variants to be placed anywhere on the page, and so on, while preserving simplicity and flexibility.

Because the minimal markup approach in  $\TeX$  is not widespread, we have to push advanced (plain)  $\TeX$ ies to write the macros for minimal markup. To use the macros once they are there is simple by nature. Although this note also deals with how to write these macros I hope this will not put off the push-the-button type of user.

### Why?

In the preface of *The  $\TeX$ book* Knuth already stated

If you are preparing a simple manuscript, you won't need to learn much about  $\TeX$  at all;

with the consequence of few markup.

Apart from the lovesong on the elegance of minimal markup, I'll inevitably touch on the subject of macro writing. To my experience there are no two  $\TeX$ ies alike who

write macros in the same spirit.<sup>8</sup> This is a weakness of macro writing in  $\TeX$ , IMHO with all respect, because it is much easier to write macros from scratch of your own than to understand the (perceived) idiosyncrazies of your colleagues. Some would call this the strength of  $\TeX$ 's macro language, this richness of possibilities, but I think it really inhibits to create for example a library of  $\TeX$  macros.<sup>9</sup>

I favour a discipline in  $\TeX$  macro writing and I hope that this note will contribute towards the macro writing for generating markup automatically, if not for a library of  $\TeX$  macros. In absence of this library<sup>10</sup> I collect these tools and macros in *BLUe $\TeX$* , all within the framework of a coherent spirit and consistent philosophy.

I like to code macros for general paradigms in programming, such as FIFO—First-In-First-Out. The FIFO principle implemented as an expandable macro, together with its variants with looking ahead functionality, will be the main  $\TeX$ niques to be used in this note.<sup>11</sup>

Disclaimer. The quotation taken from the preface of *The  $\TeX$ book* continues with

... on the other hand some things that go into printing of technical books are inherently difficult, and if you wish to achieve more complex effects you will want to penetrate some of  $\TeX$ 's darker corners.

So, what we can reasonably go for is that

simple scripts should be marked up simple

and that for inherent difficult scripts complexity will cross our way. But, ... IMHO, with all respect, do hide complexity in macros, such that only the invokes will appear in the markup, or even better that these invokes be inserted automatically by  $\TeX$ 's expansion mechanism, wherever feasible.

<sup>7</sup> With default `\puzzlettrue`. With `\puzzlefals`e the right figure—the solution—will be obtained.

<sup>8</sup> Except for the  $\LaTeX$  team, I presume.

<sup>9</sup> The  $\LaTeX$  users are not so much interested in how it is done as long as it does what you want. That is different from what I'm up to. I like to understand and be able to read the code as well.

<sup>10</sup> The  $\TeX$  archive—CTAN—provides mostly style files for  $\LaTeX$ , or other collections like my *BLUe*. A detailed taxonomy for a library is still lacking. How would you classify FIFO macros for example? (Of course under `fifo`, but ...)

<sup>11</sup> Knuth already had his `\dolist` macro, but because of assignments it is not completely processed in the mouth. Maybe that is not so relevant because of the coupling via pipes of  $\TeX$ 's digestive phases. Knuth's `\dolist` is more general for sure, but for down to earth use my straight implementation of `fifo` can do a lot. See *The  $\TeX$ book*, ex 11.5, or `\ctest The  $\TeX$ book`, 376. In *MAPS 92.2*, I have recasted `\dolist` in FIFO terminology, while preserving its assignments, of course, in 'FIFO and LIFO sing the BLUes.'

**What is on?**

I aim at that the reader after completion will understand how  $\TeX$  can be used for expansion, to transform for example

$$ABC \longrightarrow \backslash ls A \backslash ls B \backslash ls C$$

or

$$ABC \longrightarrow A \backslash cs B \backslash cs C$$

That is, to insert before each element a list separator— $\backslash ls$ —or to insert between elements a control sequence— $\backslash cs$ . These eventually nested, which is actual for the case of crosswords, and for the data proper part of tables.<sup>12</sup>

Expansion in  $\TeX$ 's mouth, well gullet, is a very powerful tool for transforming the minimal, or implicit, markup into the complete representation as required by subsequent macros, such as  $\backslash halign$ . In other words to get the script ready for further execution.

I assumed that the reader starts from the same viewpoint as me: to supply data is one thing, to format them is another.  $\TeX$ 's expansion in the mouth–gullet is a way to bridge the gap.

**Minimal markup by Knuth**

Knuth in his markup has amply made use of blank lines, see for example *The  $\TeX$ book* file. Nice because of the implicit structure, pleasing for the eye.

A  $\TeX$  paragraph starts usually by its first character and is ended by the first blank line.<sup>13</sup> No explicit markup at all, not to mention that kerning goes automatically!

The markup for a chapter of *The  $\TeX$ book* starts with  $\backslash beginchapter$  followed by the chapter title and ended by a blank line.

Another occurrence of minimal markup has to do with options. Knuth's  $\backslash begindisplay \endisplay$  pair is quite flexible with variations there when needed, which otherwise won't hinder. Just one line? That is fine. More lines? Separate them just by  $\backslash cr$ . More columns? OK too, separate columns by  $\&$ . Then there is the possibility for inclusion by supplying after  $\backslash begindisplay$  on the same line.

In general Knuth implemented options via the use of token variables—his  $\backslash every . . . s$ —to be inspected at appropriate places.

If you don't need options, Knuth's implementation of them won't hinder, you don't have to pay for what you don't use, also known as Samelson's principle.<sup>14</sup>

Maybe you will consider the above not so relevant, but IMHO, with all respect, when paying attention to minimal markup your scripts will become cleaner and cleaner, if not for clearer and clearer. Moreover, it gives more and more

pleasure in (re)reading them. You won't no longer be a victim of the curly braces mania, with its drawback of the 'non-matching braces' error message.

Disclaimer. There is one approach of Knuth which IMO is not minimal. It is about that input has to be repeated when the result in print as well as how this was achieved are displayed, as is the case on many places in *The  $\TeX$ book*.

**Automatic inclusion of markup for list processing**

I needed this for the first time when typesetting the Tower of Hanoi game.<sup>15</sup> Knuth used the active list separator on several occasions, see for example *The  $\TeX$ book*, Appendix D.2 List Macros. Maybe, the most notorious practical use is his writing of the answers of the exercises to a file preceded by the list separator  $\backslash ansno$ , see *The  $\TeX$ book*, 422. As we all know this file is the input for the Appendix A: Answer to All the Exercises. I used the (active) list separator in my database approach of  $\TeX$  tools and formats, not to mention my literature database, from where references can be selectively loaded, and of course in selecting addresses from my 'addressbook.'

Maybe, the viewpoint that this functionality is of importance to allow minimal markup has not been recognized at large.

Within NTG, next to myself, Piet Tutelaers has used the active list separator, to name but one. He prescribed chess positions in a minimal way, with  $\TeX$  to expand these into the complete markup. See later on.

<sup>12</sup> A prerequisite is to distinguish between the header, first column and the data proper, for the markup of tables. In doing this the generally more complex header part of a table—the legenda—is separated from the data.

<sup>13</sup> My Polish friends nicknamed paragraphs by 'From  $\backslash indent$  to  $\backslash par$ ,' in their contribution about how paragraph parameters work for a Euro $\TeX$  some years ago.

<sup>14</sup> This in contrast with the handling of options by parsing of arguments. In the old  $\LaTeX$  for example markup for options requires embracing them by square brackets. The disadvantage is that these square brackets must be there whether you need an option or not. Definitely redundant, IMHO, with all respect. (I was told that this no longer the case.) In  $\LaTeX$ 's graphics chapter brackets like ( and ) are used to embrace optional items, which is inconsistent and confusing. Inconsistency is inevitable with macros emerging from all over the world. What I borrowed I generally recasted in my BLUE's philosophy in order to enhance consistency, and to allow myself to forget about the details of the tool.

<sup>15</sup> An predecessor paper on the issue overlooked the inclusion of (active) list separators and therefore the author had to go explicitly through the data recursively again, all within a LISP flavour. Definitely unintelligible, especially for a non-LISP programmer like me.

### What is the problem?

The problem in its simplest form is that we have for example the string ‘ABC’ and we wish to replace this by ‘\ls A \ls B \ls C.’

### How to do this?

The algorithm is straightforward: walk along the elements and insert \ls before each element. The programming details are a bit tricky in T<sub>E</sub>X, however.<sup>16</sup> Let us assume that ‘ABC’ is stored as replacement text of the definition \data, and that we’ll deliver the result under the name \markedupdata for educational purposes. The latter allows to use T<sub>E</sub>X’s \show for inspecting the result. For simplicity, I assume further that the elements are not expandable and consist of only single characters.<sup>17</sup>

### Example (Insertion of control sequences before)

```
\def\fifo#1{\ifx\ofif#1\ofif\fi
  \noexpand\ls{#1}\fifo}
\def\ofif#1\fifo{\fi}
\def\data{ABC} \show\data
\xdef\markedupdata{\expandafter\fifo\data\ofif}
\show\markedupdata
```

Explanation. I implemented going through the list by tail recursion, via the \fifo macro. Once you understand this macro you are able to program tail recursion on-the-fly. It might be handy to forget about termination first—yes, the infinite loop nightmare—and next concentrate on the termination. I presume that the infinite situation is rather straightforward: all what follows \fifo in the markup is taken one argument at a time, and reinserted preceded by \ls. In order to terminate \fifo we have to append a token to the data, a so-called sentinel in these kinds of loop programming, for the moment without a specific meaning. I chose \ofif. If this \ofif token as argument is recognized then the tail recursion is ended by an invoke of—why not \ofif?—in the true branch. Only now the meaning of \ofif becomes relevant. It should be defined with \fifo as end separator of its argument, in order to ‘eat’ all tokens up to and including the \fifo token, meaning no next level of tail recursion will take place. However, also the closing \fi is eaten and therefore the replacement text of \ofif must reinsert this token, et voilà.

Remark. T<sub>E</sub>X’s \show control sequence displays the (replacement text of the) macro, supplied as argument, in the log file. By the use of \show we can verify what happened.

### Minimal markup in chess

Tutelaers 1992, as mentioned by Eijkhout 1991, faced the problem of inputting a chess position. The problem is characterized by an unspecified number of positions of pieces, with for the pawn positions the identification of the pawn

generally omitted. Let us denote the pieces by the capital letters K(ing), Q(ueen), B(ishop), (k)N(ight), R(ook), and P(awn), with the latter symbol, P, default. The position on the board is indicated by a letter a, b, c, . . . , or h, followed by a number, 1, 2, . . . , or 8. The goal is that, for example

```
\pieces Ke1 e2 e4
```

should expand into

```
\piece Ke1 \piece Pe2 \piece Pe4
```

with \piece to be defined for further processing, which is not relevant for our purpose in this note. I assumed a meanest-and-leanest one line input.

The transformation can be done by an appropriate definition of \pieces, and an adaptation of the \fifo template, as follows.

```
\def\pieces#1\par{\xdef\markeduppieces
  {\fifo#1\ofif} }}
\def\fifo#1 {\ifx\ofif#1\ofif\fi
  \process#1\ssecorp\fifo}
\def\ofif#1\fifo{\fi}
\def\process#1#2#3\ssecorp{\if\relax#3\relax
  \noexpand\piece P#1#2\else
  \noexpand\piece#1#2#3\fi}
%Test
\pieces Ke1 e2 e4 Ra1

\show\markeduppieces
\bye
```

Remarks. A few subtleties have been introduced in the adaptation of the \fifo template. Use has been made of the implicit e-o-l, which is converted by T<sub>E</sub>X into a space. Therefore the added sentinel \ofif in the replacement text of \pieces must immediately follow #1, no space in between, because this space has already been added by T<sub>E</sub>X. On the other hand we need a space after the sentinel \ofif and therefore the sentinel is embraced and followed by a space.<sup>18</sup> The argument, #1, as seen by \process is followed by \ssecorp as end separator to cope with the default, implicit identification for pawns. Only recently I polished this variant, sigh, . . . details matter.<sup>19</sup>

<sup>16</sup> . . . not that difficult really, once you get the hang of it.

<sup>17</sup> Of course there are a lot of variations thinkable, but not so relevant for the basic part, I guess. For example accented characters can be accounted for via modifications, such as enclosing them between braces at the appropriate places. I have omitted those confusing details and will concentrate on the main issues.

<sup>18</sup> This sentinel is debraced when taken as argument.

<sup>19</sup> This note is all about expanding minimal markup into explicit markup. In practice the creation of an explicit \markeduppieces can be omitted.

## Automatic inclusion of markup between elements

Inclusion of markup in between is a little more cumbersome. If we think of processing each element and to insert the in between token after each element, except for the last, then this entails that we must decide for each element whether it is the last one or not, that is to look ahead for the sentinel. Below the control sequence, `\cs`, will be inserted in between.

### Example (Insertion of control sequences in between)

```
\def\fifo#1#2{#1\ifx\ofif#2\ofif\fi
  \noexpand\cs\fifo#2}
\def\ofif#1\ofif{\fi}
\def\data{ABC} \show\data
\xdef\markedupdata{\expandafter\fifo\data\ofif}
\show\markedupdata
```

Explanation. As usual `\ofif` has to be appended as sentinel to the data, `\fifo` has two arguments, the second is used to look (ahead) for the sentinel. The first is reinserted and processed—in this simple case just reinserted—and the second is tested against `\ofif` and reinserted at the end of the replacement text. If the test yields true, `\ofif` is invoked which in this variant of looking ahead must ‘eat’ all tokens up to and including the `\ofif`, that is the reinserted `#2`. Again a `\fi` must be inserted to compensate for the eaten `\fi`.

Remark. A variant implementation—albeit, more clumsy and less efficient—of the above is to ‘look ahead’ not just by one element, but to split the total list into the first element and the rest—also called head and tail in programming—where the (shrinking) rest has to be copied each time. This opens the Pandora box of coding variants. For an average T<sub>E</sub>Xie these kinds of variants are quite confusing, I guess. I pay so much attention to as straight as possible implementations in order to use them over and over in my macros, with confidence, enhancing conciseness and more importantly, correctness.

### Example (Splitting into head and tail)

```
\def\fifo#1#2\ofif{#1\ifx\empty#2\empty\ofif\fi
  \noexpand\cs\fifo#2\ofif}
```

This clumsy code is not only less efficient but also more tricky, see the test for the empty argument, which by the way can be a useful trick at times.

## Automatic inclusion of markup for tables

Starting from data as such it will be shown how T<sub>E</sub>X can insert the markup tags `\cs` and `\rs`, to separate columns and rows.

The basic idea has been borrowed from *The T<sub>E</sub>Xbook*, 249, from the dubble dangerous bend remark about omission of `\cr` in the input data. I have extended this with omission in the input of `&`.

### One-character data with implicit row separators

The idea in its simplest form is to transform for example<sup>20</sup>

```
P*On      P\cs *\cs O\cs n\rs
...      → ...
Edit      E\cs d\cs i\cs t
```

In each row elements are not separated, the elements consist of just one character. Macros for the above read as follows.

```
\def\fifol#1 #2 {\fifo#1\ofif
  \ifx\lofif#2\lofif\fi
  \noexpand\rs\fifol#2 }
\def\lofif#1\lofif{\fi}
\def\fifo#1#2{%\prc
  #1\ifx\ofif#2\ofif\fi
  \noexpand\cs\fifo#2}
\def\ofif#1\ofif{\fi}
\def\bdata#1\edata
  {\xdef\markedupdata{\fifol#1{\lofif} }}
\bdata P*On
      DEk*
      *n*S
      Edit
\edata%\show\markedupdata%check it
\framed\ruled\bttable\markedupdata
```

Explanation. The storing of the (marked up) data is done by `\bdata`, with the sentinel `{\lofif}` added, that is embraced and followed by a space.<sup>21</sup> Each row to be further processed by `\fifo` ends by a space, because T<sub>E</sub>X converts e-o-l-s into spaces, and the sentinel is also followed by a space. The rows, two at a time, are arguments of `\fifol`. The first row is processed. The second row (debraced one level) is tested whether it is the sentinel or not,<sup>22</sup> and reinserted, appended by a space. If the test yields false the process is recursively repeated. If the test yields true the

---

For fun compare this recent minimal variant with the one I launched in ‘FIFO and LIFO sing the BLUES.’ Of course I have updated my version of that note, with some more examples I found interesting since then. I’m pondering about my www page with all my notes and BLUE’s format. Keep fingers crossed.

<sup>20</sup> I abstracted `&` into `\cs` and `\cr` into `\rs`.

<sup>21</sup> If the braces are omitted the space is neglected because T<sub>E</sub>X neglects spaces after control sequences.

<sup>22</sup> Note that the order of the arguments to the test matter.

invoke of `\lofi` eats all tokens up to and including the (reinserted) `\lofi`, that is the #2, and compensates for the eaten `\fi` by the replacement text of `\lofi`. Similar is the processing of each row, via the invoke of `\fifol#1` with the sentinel `\ofif` added. I like to call the latter nested use of the FIFO idea.

The next step towards typesetting crosswords is that `\prc` formats the cells in each row, that is insert a `\prc` before the reinserted element, and give meaning to `\prc`.

However, in the PWT version of my crossword macros I decided that I could better not use `\haling`, or my `\btable`, but process each element directly within a box, and stagger these boxes appropriately, with the total framed. But, the typesetting is not of our concern for the moment, we are concentrating on the expansion of minimal markup, aren't we?

I think that the insertion of the `\cs-s` and `\rs-s` by  $\TeX$ , can be useful. Whatever the value might be, the thinking along the lines I have elaborated on above helped definitely to develop much nicer and clearer codes.

Remarks.

`\btable` is BLUe's (bordered) table macro on top of plain's `\halign`. It abstracts from `\halign`'s `&` and `\cr` into `\cs` and `\rs`, allowing for example the use of flags like `\ruled`. It is beyond the scope of this note to explain `\btable` or its use. Roughly speaking, `\btable` formats the data as you would expect. The above code is not robust with respect to extra blank lines in the data, however.

The code is biased by the e-o-l $\rightarrow$  substitution of  $\TeX$ .

### Data with implicit column and row separators

Sometimes it is convenient to supply data for a table as such, row by row with the columns separated by spaces.

To expand table data without markup<sup>23</sup> into marked up data ready for use with  $\TeX$ , is a bit more cumbersome. Have a try.<sup>24</sup>

**Example (*Young tableaux*)** The Young tableaux as given for example in the PWT guide, could have been provided, marked up simply, as follows.

```
\bdata 7 8 9 10
      9 11
      16
\edata
```

with (aimed at) results

7	8	9	10
9	11		
16			

The coding for transforming the input as supplied between `\bdata` and `\edata` reads as follows, where

`\obeylines` has been used.

```
{\obeylines%
 \gdef\fifol#1
   #2
  {\fifol#1 {\ofif} \ifx\lofi#2\lofi\fi%
   \noexpand\cr\fifol#2
  }%
 \gdef\lofi#1\lofi
  {\fi}%
}
\def\fifol#1 #2 {#1\ifx\ofif#2\ofif\fi
 \noexpand&\fifol#2 }
\def\ofif#1\ofif{\fi}
\def\bdata{\bgroup\obeylines\store}
{\obeylines
 \gdef\store#1\edata{\egroup%
 \xdef\markedupdata{\fifol#1\lofi
 }}}
\bdata 7 8 9 10
      9 11
      16
\edata \show\markedupdata
$$\young\markedupdata$$
```

Explanation. The new issues here are the recognition of e-o-l-s within `\obeylines`' reign. Maybe you don't know what `\obeylines` does precisely. Don't worry, whatever it does is fine, as long as we apply its use consequently it should work fine. That is what has been done in the code above, and indeed it works fine, though I know what `\obeylines` does. The rest is similar to and discussed along with the earlier provided codes.

Remarks. The use of `%-s` is critical. Extra blank lines are handled robustly: blank cells will show in the table. Because this note is not about formatting of tables but only on minimal markup and inserting markup automatically, consult the PWT guide for the source and use of `\young`. Again, its result in print is what you expect, in agreement with tradition.

### Table data from a database

As suggested by Wietse Dol the inclusion of markup by  $\TeX$  can be of practical value too. He told me of his database of table data, which he typesets for the time being by Pascal. Is it feasible to do this by  $\TeX$ ?

The idea is that for example a file contains

<sup>23</sup> Meaning just visual ASCII: line by line and within each line elements separated by (one or more) spaces.

<sup>24</sup> Hint: In order to discriminate between ordinary spaces and spaces coming from converted e-o-l-s, make use of `\obeylines`.

11 12  
21 22

and that you will end up with let's say

```
\def\markedupdata{11\cs12\rs21\cs22}
```

That is the data together with markup is the replacement text of `\markedupdata` ready to be used by `\btable`

### In principle solution

The insertion of markup is done similar to the example treated above. However, the extra complication is that the data are on a file. Below I read the file line-by-line, and could therefore insert `\rs` naturally, well, . . . more or less.

Assume that the data are in a file called `data`.

```
\openin1=data
\def\markedupdata{}
\def\addr{\def\addr{\ea
\def\ea\markedupdata\ea{\markedupdata\rs}}
\loop\read1 to\data
\ifeof1 \break\fi
\addr
\edef\mdata{\ea\fi\data{\ofif} }
\ea\ea\ea\def\ea\ea\ea\markedupdata\ea\ea\ea
{\ea\markedupdata\mdata}
\pool
\framed\ruled\btable\markedupdata
\bye
```

The above requires the following auxiliaries.

```
\let\ea\expandafter
%Loop macros due to van der Goot
\def\loop#1\pool{#1\loop#1\pool}
\def\break#1\pool{\fi}
%FIFO variant for this case
\def\fi#1 #2 {#1\ifx\ofif#2\ofif\fi
\noexpand\cs\fi#2 }
\def\ofif#1\ofif{\fi}
```

For those who don't have `BLUeTeX` available what is going on can be followed in the log file.<sup>25</sup> Therefore insert `\show\markedupdata` before the `\pool`, and don't forget to push the return key after the `def` has been shown in order to continue.

The results for the data

1 2 3  
21 22 23  
31 32 333

are

I	2	3
21	22	23
31	32	333

Remarks. Because of the loop I had to do something special with `\addr`. Maybe it is possible to read the file and deliver the data, with an appropriate separation between the 'rows,' as replacement text of `\data`, let's say. If so we can apply the earlier treated mechanism for inserting markup.<sup>26</sup>

At a lower level there is flexibility in handling the look-and-feel of the typeset data. But as said earlier this is not our concern at the moment. Consult the PWT guide chapter about tables.

## Acknowledgements

As usual Jos Winnink proofed the paper and helped me in coercing the note into MAPS format. His remarks and suggestions are always well-taken. To say the least, he reflects what despite the intention did not come across. I consider the approach of a friendly eye better than a referee in warranting quality.

Wietse Dol suggested to consider data stored in a file, or as he put it to typeset tables from a database of table data.

## Conclusions

It's hoped for that the use of the FIFO principle, implemented as an expandable macro, for inserting markup during `TeX`'s mouth-gullet processing, will contribute to a more general use of minimal markup.

Minimal markup is the royal road to more readable scripts and alleviates conversion problems such as from a `BLUe` script into a `MAPS` submission.

The expansion by `TeX`'s gullet of minimal marked up scripts into completely marked up scripts, made me realize the power and relevance of `TeX`'s gullet expansion capabilities.

Although `TeX` has been used abundantly and intensively for nearly 20 years already, the awareness of the elegance and convenience of minimal markup to be expanded by `TeX` is only just emerging.

What astonishes me still is that it is very hard to really get at the simplest codes. Apparently Knuth had the same experience as can be distilled from the following from the preface in *The TeXbook* and . . . *The METAFONTbook*

...and there are always better ways to do what you've done before.

My case rests. Have fun, and all the best.

<sup>25</sup> It's all about insertion of `\cs` and `\rs` at the appropriate places.

<sup>26</sup> A request for this on `TeX-nl` did not provide an answer; even stronger it was believed it was not possible. Hmmm. . .



# Bijlage 9

## Catching up

### PDF and HTML at the heart

Kees van der Laan  
cg1@hetnet.nl

#### abstract

New hardware not only urged me to get T<sub>E</sub>X & Co running again, in a richer environment, but I had also to catch up with developments since. Most noteworthy in relation with T<sub>E</sub>X and documents, are the acceptance of the PDF exchange format and the HTML format, next to the realization of multi-media.

#### keywords

internet, HTML, multi-media, PDF, pdfT<sub>E</sub>X, PostScript, WWW

### Introduction

When I complemented my (Mac) hardware with the PowerMac 5500/275 a complete new world opened up. For only a year or three I have T<sub>E</sub>X-ed and emailed happily on my Powerbook 150, next to an odd 5 years on my Mac Classic II. I have used the UNIX mail system for more than 11 years the same holds for T<sub>E</sub>X & Co... And see, ... the world has really changed of late.

The new emailers in the internet are much more convenient and more economical. The email is no longer handled at a remote machine via terminal access, or via telnet. But by a POP, Post Office Protocol, and a SMTP, Simple Mail Transfer Protocol, on whatever machines they are located. I can just collect and send my email with a snip of the fingers That fast. Off-line, email can be prepared, sorted, read and answered at leisure. No real need to have it on paper, one can increase the size of the font for easy reading and so on. Moreover, MIME (Multipurpose Internet Mail Exchange) allows fancy enclosures as attachments, to be sent encoded, that is economical, at will. Examples? Attached photos as (economical) jpeg format, PDF (Portable Document Format) documents uuencoded.

The latter brings us back to T<sub>E</sub>X & Co. For Knuth T<sub>E</sub>X & Co meant T<sub>E</sub>X and METAFONT, meaning tools for typesetting and creating types. Let us start from the accompanying earlier used, and T<sub>E</sub>X & Co biased, model. In this model T<sub>E</sub>X and METAFONT are at the heart, as oldest tools. But, Adobe has turned things upside down with their

‘PS on the clipboard’

and

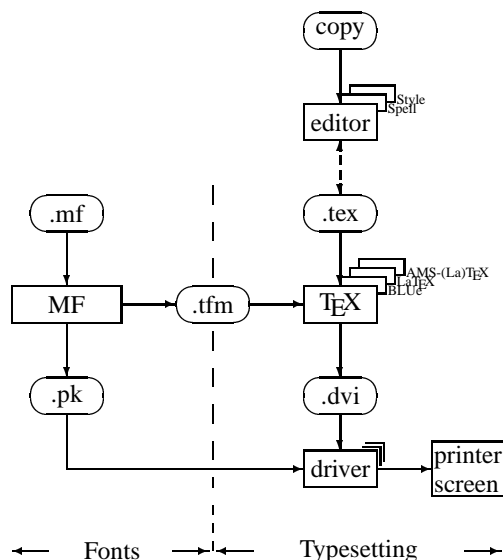
‘PDF as standard for document exchange.’

The latter not only passively to be read from, but also to be reused, modified or enriched with bookmarks, thumbnails, threading and hyperlinks

independently from the tool which created the document, and

independently from the platform or tool to be used

as long as they comply with PDF.



And then there is the internet, with the format HTML, its browsers<sup>1</sup> and its logical World-Wide Web, WWW for short. This stressed another aspect of multi-media documents. The location of (parts of) documents is no longer relevant, as long as they are connected by hyperlinks.

Something has really happened in the nineties.

In short, we have concentrated on books with their typographical traditions and concepts such as spread and so

<sup>1</sup> Netscape is a super, active HTML viewers and much more.

on. Now we really have entered the era of multi-media documents, with different, yet to be discovered rules for composing and (re)using.

In this Adobe with its PDF has earned the right of the first-born, the first prototype—PDF and Acrobat—which is catching on, IMHO, with all respect.

Adobe took the lead again.

For T<sub>E</sub>X & Co the law of diminishing returns applies.

I have to work harder and really catch up. There is much more to do than before.

### T<sub>E</sub>X and PostScript

Jacko<sup>2</sup> in one of his lectures stressed the ‘octopussy’ model:

EPS at the heart

as the central exchange format for documents, where for example T<sub>E</sub>X & Co and Adobe Illustrator could cooperate happily. In order to let this work he developed mf2eps, because MetaPost was not yet in the public domain. He followed in the footsteps of Adobe, and how right he was. But, in the meantime, the concept of hypertext got a boost when the internet took off, with virtually all computers connected into a huge web, the WWW. Adobe picked it up and developed the successor to Postscript, aimed at not only page description, but also at handling hyperlinks, or more generally, at handling and (re)using multi-media documents. In doing so, known deficiencies with respect to PostScript, such as the size of the PS files and the (non)availability of fonts, were taken care of.

### SGML and T<sub>E</sub>X

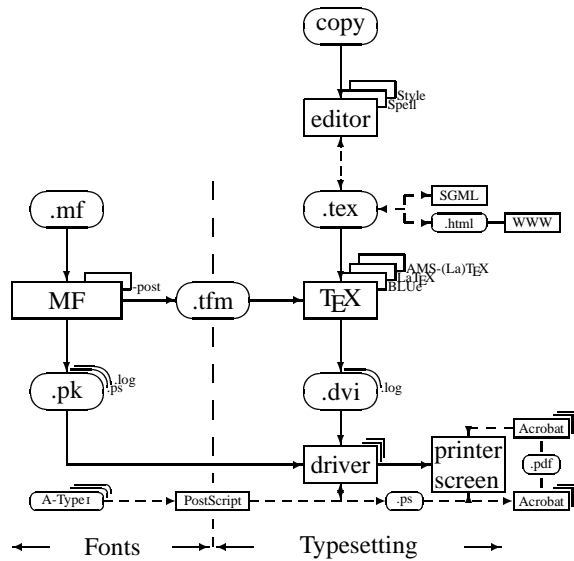
In the eighties SGML (Standardized General Markup Language) and T<sub>E</sub>X were the buzzwords in the T<sub>E</sub>X community.

SGML as frontend and T<sub>E</sub>X the backend formatter

Publishers believe in this model for several reasons. I think that SGML and T<sub>E</sub>X are alike, more two of a kind modulo some syntactic sugar, IMHO with all respect. In practical BLUe work there was no much need to go for SGML. Moreover their editors and viewers were not for free, to put it euphemistically. We also faced the problem of the Document Type Definition confusion, and the practical problems with fonts.

That HTML (HyperText Markup Language) took off with the internet. That it used an SGML-biased syntax is a mere coincidence, IMHO, with all respect.

These developments can be added to the-biased-by-T<sub>E</sub>X model as follows.



Let us look more closely at HTML and PDF and what these formats and their accompanying tools have brought us.

### PDF and T<sub>E</sub>X

This is a short-term future of T<sub>E</sub>X, if we want T<sub>E</sub>X & Co to have a niche in multi-media land.

I believe that the potential of PDF is enormous, and that it can't be overestimated.

I consider the chain

$$.tex \xrightarrow{TeX} .dvi \xrightarrow{dvi2ps} .ps \xrightarrow{ps2pdf} .pdf$$

flexible, maybe it is too flexible. On the other hand little experience with pdfT<sub>E</sub>X

$$.tex \xrightarrow{pdfTeX} .pdf$$

was heartwarming.

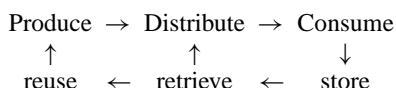
Adobe's PDFwriter to create .pdf files from without any complying application is powerful, and definitely a way to get things done by other manufacturers.

From the above we might distill that it is important to find the right balance in spending time on the tools involved. T<sub>E</sub>X is just one of the many.

With respect to T<sub>E</sub>X there is a weakness: the advantage of T<sub>E</sub>X as creator is thrown away with the advantage of

<sup>2</sup> Bogusław Jackowski

abstracting from the creating tools. Not only should there be tools to start with (all)T<sub>E</sub>X and end with PDF files, but also to go back from PDF to .tex, that is ASCII with T<sub>E</sub>X markup. I'm afraid that is too cumbersome, and not realistic. Now already we include EPS files for the graphics, for example. My model of the biological invariant of use and reuse does no longer hold for (all)T<sub>E</sub>X as such. T<sub>E</sub>X & Co must be seen with respect to cooperating tools around a common file format, the PDF. My idea of the life-cycle of multi-media documents does apply with respect to PDF, with other tools as plug-ins.



What will reasonable emerge is that a 'federation' of tools, all contributing with their strong sides to the one-and-only PDF file, be it T<sub>E</sub>X's quality of typesetting math, be it PostScript's suitability toward graphics, or whatever spreadsheet program towards spreadsheets, or whatever database tool towards database reporting, or the emerging and captivating sounds and movies. The latter two are still in their infancy with respect to multi-media, though I have watched baffling demos already.

That Adobe also provides more and more modern navigating and enriching tools only supports their PDF approach.

Blue Sky Research has in their T<sub>E</sub>Xtures 2.0 also picked up the hyperlink idea under the buzzword: synchronicity, meaning dynamic links between any place in the source window to the typeset window and vice versa, eventually controlled by the log window.

### Granularity

Some 5 years ago Adobe in their Acrobat Exchange considered pages or screens as the right amount of granularity. The building blocks of PDF format. They could be refreshed, shuffled, linked and so on. More recently on the internet pages I found that frames, as part of nested pages/screens form the right granularity and can be of any size, so even one character.

New and completely beyond T<sub>E</sub>X are digital photos or videos, which by an amateur can be processed by Adobe's PhotoDeLuxe.

Sounds can be processed, saved and sent too. As yet I have no experience with it, nor with digital video.

### T<sub>E</sub>X and HTML

HTML stands for HyperText Markup Language and came along with the internet, well its logical layer www. The homepages on internet are written in HTML and can be viewed by the ubiquitous and for free browsers like

Netscape and Internet Explorer.

New PCs or Macs come nowadays with tools like Pagemill to build Webpages, especially when subscribing to commercial providers. For transforming (All)T<sub>E</sub>X into HTML tools are available not in the least Gurari's T<sub>E</sub>X4ht, as reported by Erik Frambach in MAPS 20, spring 1998.

Erik communicated separately that HTML provide links automatically, that search engines can look into HTML code, and that downloading can be done in parts. Only those frames one is interested in, instead of the bulky PDF file.

### Daydreaming

Would it be feasible to think of MAPS as a multi-media PDF bulletin or as an HTML hypertext? Could we deliver PDF submissions instead of the stubborn and harnessing L<sub>A</sub>T<sub>E</sub>X & Co, or just supply our HTML hypertexts on our homepages and let the MAPS editors compose a symphony out of the hyperlinks in their editorial comments? No longer printing and dissemination hazzle, but it will be simply available via NTG's Web page to be downloaded on demand? Or even more liberal with only links from without on NTG's Webpage to the real material, wherever it may be?

### In the small with respect to BLUE

I got with respect to BLUE T<sub>E</sub>X feedback from Russia to parameterize over the fonts. Maybe, I should allow to easily change the CMR fonts for their PS variants. In doing so, my PS 4/600 Laser printer with 2MB RAM would no longer complain. And maybe, BLUE will be better geared toward PDF use, because we can easily adjust for the fonts given the aim. Next to that some loose ends should be handled better, not in the least to launch it as a real format, and have a clear idea about what can be scaled and what not.

Some time ago I wrote my convertor assistant which transformed

$$.bluetex \xrightarrow{TeX} .latex$$

to alleviate the retyping of commands in submitting BLUE scripts to MAPS.

Maybe I should consider

$$.bluetex \xrightarrow{TeX} .html$$

Of course, I'm familiar with transformations and know that we usually are talking about different things, for and after the transformation. For example, the concept of nested frames is not in BLUE T<sub>E</sub>X. But for the endleaves, where we like to open up documents for direct access via the internet, this tranformation might be useful. On the other

hand a  $\TeX$ ie can process a .tex file easily, although generally several source files, or databases as in  $\text{BLUe}\TeX$ 's case, are necessary.

## Acknowledgements

As usual Jos Winnink proofed the paper and helped me in coercing into MAPS format. Erik Frambach communicated the reasons why he liked .tex  $\rightarrow$  .html conversion. Thank you.

## Conclusions

Looking back, I think I did a good investment in familiarizing myself with  $\TeX$  and PostScript, and having had a look into SGML. For the future Adobe's Acrobat suite and their PDF is a must, nor can I afford it to leave HTML alone, especially when I'll come with my www homepage, some day.

I'm only too happy that I did not suboptimize in using  $\TeX$  & Co, did know when to stop.

It seems that with respect to  $\TeX$  & Co the law of diminishing returns applies.

Netscape and Internet explorer, Outlook Express, Alpha integrated with  $\text{CMac}\TeX$  on my Mac are musts, and maybe I'll have time to really delve into MPW (Mac Programmer's Workbench), or Knuth's literate programming, which has been lingering on my desk for so long already. In between some nitty-gritties about plug-ins and converters have to be worked through, and back-ups of my 'desks' secured.

$\text{BLUe}\TeX$  will be adapted, and the PWT (Publishing with  $\TeX$ ) guide will be renewed as an hypertext, in PDF and in HTML, hopefully. Maybe, all made available from my homepage to start with.

In short there is more to be done than before. . .

My case rests. Have fun, and all the best.

# Bijlage 10

## Eenheid in Eenheden

Ton Otten  
Hans Hagen  
PRAGMA ADE  
Ridderstraat 27  
8061 GH Hasselt NL  
pragma@wxs.nl  
ntg-context@ntg.nl

### keywords

SI-eenheden, eenheden, CONTEXT

### abstract

In order to support a consistent typography of units in technical documents the CONTEXT module `units` was developed. When this module is loaded all SI-units are available. Seldom used and/or complex units can be defined by the user with the command `\unit[{}]{}`. An example: `\Kilo \Gram \Meter \Per \Square \Sec` produces `kgm/s2`.

### Inleiding

In technische documenten worden vaak eenheden gebruikt. Om enige consistentie af te dwingen en om binnen het SI-eenheden stelsel (SI staat voor *Système International*) te kunnen werken is de module `units` beschikbaar. Deze module wordt in CONTEXT<sup>1</sup> als volgt geladen:

```
\gebruikmodule [eenheid]
```

Daarnaast is het commando `\eenheid` beschikbaar, waarmee zelf eenheden kunnen worden gedefinieerd. Voor de betekenis en de structuur van dit commando wordt verwezen naar de CONTEXT-handleiding. Op dit moment experimenteren we met een mechanisme om getallen consistent te zetten. Meer hierover in een volgend artikel.

### Het werken met si-eenheden

In verband met de leesbaarheid van de bronteksten is er voor gekozen om de eenheden te definiëren met begrijpelijke commando's. Een eenheid als `kgm/s2`, definiëren we daarom als volgt:

```
\Kilo \Gram \Meter \Per \Square \Sec
```

Iemand met een technische achtergrond kan zich bij het lezen van deze definitie wel een beeld vormen van de bedoelde eenheid.

Een dergelijke eenheid kan men ook in de mathematische mode realiseren. Dit zou er bijvoorbeeld als volgt uitzien:

```
$\rm kg m \cdot s^{-2}$
```

De mathematische mode is in dit geval noodzakelijk om de superscript en verhoogde punt te realiseren. In de mathematische mode wordt echter een mathfont gebruikt. Om dit te herstellen moeten we werken met `hbox`-en of zoals in dit voorbeeld met `\rm` een

---

<sup>1</sup> In de handleiding voor beginners, die in MAPS 19 en 20 is gepubliceerd, wordt verwezen naar de hier beschreven module.

omstelling doen. Een en ander is funest voor de leesbaarheid van de (bron)tekst. Een ander ‘nadeel’ van de mathematische mode is de spatiëring. Deze is anders dan in de tekst mode. Dit betekent dat deze per eenheid moet worden bekeken en gecorrigeerd, hetgeen de consistentie niet in de hand werkt. Spatiëring in samengestelde eenheden wordt in de module `units` correct afgehandeld.

Overigens kan  $/s^2$  ook worden geschreven als  $s^{-2}$  waarvoor `\ISquare \Sec` moeten worden ingetypt en waarbij de `I` staat voor Inverse. Vergelijk trouwens eens  $m^3/s^2$  met  $m^3/s^2$ , waarbij het laatste is ingevoerd als `\rm m^3/s^2`.

Soms oogt het fraaier om een `·` tussen de eenheden te plaatsen (`kg·m`). Denk bijvoorbeeld aan `Pa·s`, in plaats van `Pas`. Dit wordt gedaan met het commando `\Times`. Overigens, `Pa` is geen SI-eenheid, maar wel beschikbaar. Men kan zelf eenheden toevoegen met het commando:

```
\eenheid[Pascal]{Pa}{Pascal}
```

Een `/` in `m/s` wordt opgeroepen met `\Per`.

Het is gebruikelijk om een spatie te plaatsen tussen getal en eenheid. Een uitzondering daarop vormt een hoek van  $30^\circ$  en een temperatuur van  $30^\circ\text{C}$ . Overigens moet een temperatuur van  $30^\circ\text{C}$  worden genoteerd als `303 K`, zoals ongetwijfeld bekend is. Andere uitzonderingen vormen  $30\%$  en  $30\text{‰}$ , maar deze worden dan ook niet als eenheid aangerekend. In `CONTEXT` zijn deze tekens op te roepen met:

```
\percent  
\permille
```

De spatie tussen het getal en de eenheid is een conventie, maar bij regelovergangen is scheiding van getal en eenheid uiteraard niet wenselijk. Indien in de brontekst `30 \Degrees \Celsius` wordt ingevoerd, zal `CONTEXT` ervoor zorgen dat getal en eenheid niet worden gescheiden bij regelovergangen.

In lopende tekst wordt de eenheid afgesloten met een *backward slash*, dus: `... 300 m2 groot`, wordt ingevoerd als: `... 300 \Square \Meter\ groot`. Dit wordt gedaan om een extra spatie na de eenheid af te dwingen.

Samengevat kan men zeggen dat eenheden altijd worden samengesteld uit (eventueel) een voorvoegsel, een tussenvoegsel, een achtervoegsel en een of meerdere grootheden. In tabel 1 zijn deze ‘voegsels’ opgenomen.

Deze voor-, tussen- en achtervoegsels worden gebruikt in combinatie met de basiseenheid die past bij een grootte. Dus de massa  $m$  wordt weergegeven in `g (\Gram)` en de tijd  $t$  in `s (\Sec)`. Willen we nu `ms` en `kg /s2` als eenheden gebruiken, dan typen we:

```
\Milli \Sec  
\Kilo \Gram \Per \Square \Sec
```

Hieronder staat een overzicht van de SI-eenheden die bij de verschillende grootheden worden gebruikt.

<code>\Meter</code>	<code>\Newton</code>	<code>\Ampere</code>	<code>\Siemens</code>
<code>\Liter</code>	<code>\Pascal</code>	<code>\Coulomb</code>	<code>\Sievert</code>
<code>\Sec</code>	<code>\Joule</code>	<code>\Volt</code>	<code>\Bell</code>
<code>\Rad</code>	<code>\Watt</code>	<code>\eVolt</code>	<code>\Byte</code>
<code>\Degrees</code>	<code>\Kelvin</code>	<code>\Tesla</code>	
<code>\Hertz</code>	<code>\Degrees</code>	<code>\Celsius</code>	<code>\Farad</code>
<code>\Gram</code>	<code>\Mol</code>	<code>\Ohm</code>	

Het moge duidelijk zijn welke eenheden we met deze commando's oproepen.

### Typografische afspraken

Enkele typografische regels en afspraken (conventies, in het nederlandse taalgebied) rond eenheden zijn:

- het symbool van een eenheid wordt normaal afgedrukt (meter: m)
- het symbool van een grootheid wordt *cursief* afgedrukt (lengte: l)
- tussen getal en eenheid moet een spatie staan
- grote getallen worden gegroepeerd in drietallen en door middel van spaties gescheiden 20 000,00
- bij eenheden met meer dan één symbool achter de schuine streep (solidus), worden deze symbolen tussen haken geplaatst (belasting: m<sup>3</sup>/ (m<sup>2</sup>h))

Er dient rekening mee te worden gehouden dat bepaalde conventies per land verschillen. Een aantal van de onderstaande eenheden zijn typisch nederlands. Hoewel omwille van de consistentie alle commando's engels zijn, bieden we ook `\Graden`, `\Atoom`, `\Heure`, `\Jaar`, `\Maand`, `\Dag` en `\Uur`, mits nederlands de hoofdtal is.

In het geval dat de eenheid niet wordt vooraf gegaan door een getal, kan als loze prefix `\Unit` worden gegeven. Het is mogelijk, bijvoorbeeld uit educatief oogpunt, om de eenheden visueel te scheiden: `\spaceddimensionstrue`.

### Overzichten

De volgende tabellen geven een overzicht van de beschikbare eenheden. Bij de eenheden wordt steeds aangegeven bij welke grootheid of grootheden deze wordt gebruikt.

Voorvoegsels		Tussenvoegsels		Achtervoegsels	
commando	resultaat	commando	resultaat	commando	resultaat
<code>\Pico</code>	p	<code>\Times</code>	.	<code>\Linear</code>	1
<code>\Nano</code>	n	<code>\Solidus</code>	/	<code>\Square</code>	2
<code>\Micro</code>	μ	<code>\Per</code>	/	<code>\Cubic</code>	3
<code>\Milli</code>	m	<code>\OutOf</code>	:	<code>\Inverse</code>	-
<code>\Centi</code>	c			<code>\Linear</code>	-1
<code>\Deci</code>	d			<code>\Square</code>	-2
<code>\Hecto</code>	h			<code>\Cubic</code>	-3
<code>\Kilo</code>	k				
<code>\Mega</code>	M				
<code>\Giga</code>	G				
<code>\Terra</code>	T				

**tabel 1** Voor-, tussen- en achtervoegsels.

Grootheid	Symbool	Eenheid	Commando	Betekenis
lengte	$l$	m	<code>\Meter</code>	meter
breedte	$b$	nm	<code>\Nano \Meter</code>	nanometer
hoogte	$h$	$\mu\text{m}$	<code>\Micro \Meter</code>	micrometer
straal	$r$	mm	<code>\Milli \Meter</code>	millimeter
middellijn	$d$	cm	<code>\Centi \Meter</code>	centimeter
weglengte	$s$	dm	<code>\Deci \Meter</code>	decimeter
		hm	<code>\Hecto \Meter</code>	hectometer
		km	<code>\Kilo \Meter</code>	kilometer

**tabel 2** Lengte, breedte, hoogte, straal, middellijn en weglengte.

Grootheid	Symbool	Eenheid	Commando	Betekenis
oppervlakte	$A$	$\text{m}^2$	<code>\Square \Meter</code>	vierkante meter
		$\text{mm}^2$	<code>\Square \Milli \Meter</code>	vierkante millimeter
		$\text{cm}^2$	<code>\Square \Centi \Meter</code>	vierkante centimeter
		$\text{dm}^2$	<code>\Square \Deci \Meter</code>	vierkante decimeter
		$\text{km}^2$	<code>\Square \Kilo \Meter</code>	vierkante kilometer

**tabel 3** Oppervlakte.

Grootheid	Symbool	Eenheid	Commando	Betekenis
volume	$V$	$\text{m}^3$	<code>\Cubic \Meter</code>	kubieke meter
		$\text{cm}^3$	<code>\Cubic \Centi \Meter</code>	kubieke centimeter
inhoud		$\text{dm}^3$	<code>\Cubic \Deci \Meter</code>	kubieke decimeter
		$\text{km}^3$	<code>\Cubic \Kilo \Meter</code>	kubieke kilometer
		l	<code>\Liter</code>	liter
		ml	<code>\Milli \Liter</code>	milliliter
		cl	<code>\Centi \Liter</code>	centiliter
		dl	<code>\Deci \Liter</code>	deciliter
		$\text{l}^2$	<code>\Square \Liter</code>	literkwadraat

**tabel 4** Volume en inhoud.

Grootheid	Symbool	Eenheid	Commando	Betekenis
tijd	$t$	s	<code>\Sec</code>	seconde
		ns	<code>\Nano \Sec</code>	nanoseconde
		$\mu\text{s}$	<code>\Micro \Sec</code>	microseconde
		ms	<code>\Milli \Sec</code>	milliseconde
		a	<code>\Year</code>	jaar
		m	<code>\Month</code>	maand
		d	<code>\Day</code>	dag
		h	<code>\Hour</code>	uur
		min	<code>\Min</code>	minuten
		$\text{s}^2$	<code>\Square \Sec</code>	seconde kwadraat

**tabel 5** Tijd.



Grootheid	Symbool	Eenheid	Commando	Betekenis
hoek	$\alpha, \beta, \dots$	rad	\Rad	radialen
		°	\Angle	hoekgraden
		°	\Degrees	

**tabel 6** Hoek.

Grootheid	Symbool	Eenheid	Commando	Betekenis
frequentie	$f$	Hz	\Hertz	Hertz
kloksnelheid		kHz	\Kilo \Hertz	kilo Hertz
rotatiefrequentie	$n$	MHz	\Mega \Hertz	mega Hertz
		mHz	\Milli \Hertz	milli Hertz
		$s^{-1}$	\Inverse \Sec	per seconde
		RPS	\OmwPerSec	omwentelingen
		RPM	\OmwPerMin	omwentelingen
				per minuut

**tabel 7** Frequentie, rotatiefrequentie en kloksnelheid.

Grootheid	Symbool	Eenheid	Commando	Betekenis
massa	$m$	g	\Gram	gram
		$\mu\text{g}$	\Micro \Gram	microgram
		mg	\Milli \Gram	milligram
		kg	\Kilo \Gram	kilogram
		u	\Atom	atomaire
				massa-eenheid

**tabel 8** Massa.

Grootheid	Symbool	Eenheid	Commando	Betekenis
kracht	$F$	N	\Newton	Newton
		kN	\Kilo \Newton	kilo Newton

**tabel 9** Kracht.

Grootheid	Symbool	Eenheid	Commando	Betekenis
druk	$p$	Pa	\Pascal	Pascal
spanning		mPa	\Milli \Pascal	milli Pascal
normaalspanning	$\sigma$	kPa	\Kilo \Pascal	kilo Pascal
schuifspanning	$\tau$	MPa	\Mega \Pascal	Mega Pascal

**tabel 10** Druk en spanning.

Grootheid	Symbol	Eenheid	Commando	Betekenis
arbeid	$W$	J	\Joule	Joule
potentiële energie	$E_p$	mJ	\Milli \Joule	milli Joule
kinetische energie	$E_k$	kJ	\Kilo \Joule	kilo Joule
		MJ	\Mega \Joule	mega Joule
		GJ	\Giga \Joule	giga Joule

**tabel 11** Arbeid, potentiële energie en kinetische energie.

Grootheid	Symbol	Eenheid	Commando	Betekenis
vermogen	$P$	W	\Watt	Watt
		mW	\Milli \Watt	milli Watt
		kW	\Kilo \Watt	kilo Watt
		MW	\Mega \Watt	mega Watt

**tabel 12** Vermogen.

Grootheid	Symbol	Eenheid	Commando	Betekenis
temperatuur	$T$	K	\Kelvin	Kelvin
		°C	\Degrees \Celsius	graden Celsius

**tabel 13** Temperatuur.

Grootheid	Symbol	Eenheid	Commando	Betekenis
hoeveelheid stof	$n$	mol	\Mol	mol
		mmol	\Milli \Mol	millimol
		kmol	\Kilo \Mol	kilomol
		M	\Molair	mol/l
		eq	\Equivalent	equivalent
		meq	\Milli \Equivalent	milli equivalent

**tabel 14** Chemische hoeveelheid stof.

Grootheid	Symbol	Eenheid	Commando	Betekenis
elektrische stroom	$I$	A	\Ampere	Ampère
		mA	\Milli \Ampere	milliAmpère

**tabel 15** Stroom.

Grootheid	Symbol	Eenheid	Commando	Betekenis
elektrische lading	$Q$	C	\Coulomb	Coulomb

**tabel 16** Lading.

Grootheid	Symbool	Eenheid	Commando	Betekenis
spanning	$U$	V	<code>\Volt</code>	Volt
potentiaal verschil elektromotorische kracht	$E$	mV	<code>\Milli \Volt</code>	milli Volt
		kV	<code>\Kilo \Volt</code>	kilo Volt
potentiaal verschil	$\Delta V$	eV	<code>\eVolt</code>	elektron volt
		keV	<code>\Kilo \eVolt</code>	kilo elektron volt
		MeV	<code>\Mega \eVolt</code>	mega elektron volt

**tabel 17** Spanning, potentiaal en elektromotorische kracht.

Grootheid	Symbool	Eenheid	Commando	Betekenis
inductie	$B$	T	<code>\Tesla</code>	Tesla

**tabel 18** Magnetische inductie.

Grootheid	Symbool	Eenheid	Commando	Betekenis
capaciteit	$C$	F	<code>\Farad</code>	Farad
		pF	<code>\Pico \Farad</code>	pico Farad
		nF	<code>\Nano \Farad</code>	nano Farad
		$\mu$ F	<code>\Micro \Farad</code>	micro Farad
		mF	<code>\Milli \Farad</code>	milli Farad

**tabel 19** Capaciteit.

Grootheid	Symbool	Eenheid	Commando	Betekenis
weerstand	$R$	$\Omega$	<code>\Ohm</code>	Ohm
		k $\Omega$	<code>\Kilo \Ohm</code>	kilo Ohm

**tabel 20** Weerstand.

Grootheid	Symbool	Eenheid	Commando	Betekenis
geleiding	$G$	S	<code>\Siemens</code>	Siemens

**tabel 21** Geleiding.

Grootheid	Symbool	Eenheid	Commando	Betekenis
straling		Sv	<code>\Sievert</code>	Sievert
		mSv	<code>\Milli \Sievert</code>	milli Sievert
		Bq	<code>\Bequerel</code>	Bequerel
		MBq	<code>\Mega \Bequerel</code>	Bequerel

**tabel 22** Straling.

Grootheid	Symbool	Eenheid	Commando	Betekenis
geluidssterkte		dB	<code>\Deci \Bell</code>	decibel

**tabel 23** Geluidssterkte.

Grootheid	Symbol	Eenheid	Commando	Betekenis
opslagcapaciteit		Byte	\Byte	Byte
		kByte	\Kilo \Byte	kilo Byte
		MByte	\Mega \Byte	mega Byte
		GByte	\Giga \Byte	giga Byte

**tabel 24** Computergeheugen en opslagcapaciteit.

Grootheid	Symbol	Eenheid	Commando
snellheid	$v$	m/s	\Meter \Per \Sec
versnelling	$a, g$	$m/s^2$	\Meter \Per \Square \Sec
volumestroom	$q$	$m^3/s$	\Cubic \Meter \Per \Sec
hoeksnelheid	$\omega$	rad/s	\Rad \Per \Sec
dichtheid	$\rho$	$kg/m^3$	\Kilo \Gram \Per \Cubic \Meter
massastroom	$\Phi_m$	kg/s	\Kilo \Gram \Per \Sec
massatraagheid	$I$	$kgm^2$	\Kilo \Gram \Square \Meter
kracht	$F$	$kgm/s^2$	\Kilo \Gram \Meter \Per \Square \Sec
druk	$p$	$N/m^2$	\Newton \Per \Square \Meter
moment	$M$	Nm	\Newton \Meter
schuifspanning	$\tau$	$N/mm^2$	\Newton \Per \Square \Milli \Meter
elasticiteit	$E$	$N/m^2$	\Newton \Per \Square \Meter
dynamische viscositeit	$\eta$	Pa · s	\Pascal \Times \Sec
kinematische viscositeit	$\nu$	$m^2/s$	\Square \Meter \Per \Sec
concentratie van stof B	$c_B$	$mol/m^3$	\Mol \Per \Cubic \Meter
hardheid (H <sub>2</sub> O)	$D$	$mol/m^3$	\Mol \Per \Cubic \Meter

**tabel 25** Samen te stellen eenheden.

Grootheid	Symbol	Eenheid	Commando	Betekenis
massa	$m$	t	\Ton	ton
massa	$p$	kt	\Kilo \Ton	kilo ton
lengte	$l$	ft	\Foot	foot
lengte	$l$	inch	\Inch	inch
oppervlakte	$A$	ha	\Hectare	hectare
druk	$p$	at	\At	atmosfeer (techn.)
druk	$p$	atm	\Atm	atmosfeer (fys.)
druk	$p$	bar	\Bar	bar 1ookPa
druk	$p$	mHg	\Meter \Kwik	meter kwikkolom
kracht	$F$	kgf	\Kilo \Gram \Force	kilogramforce
vermogen	$P$	pk	\Paardenkracht	paardenkracht
energie	$E$	cal	\Cal	calorie
energie	$E$	kcal	\Kilo \Cal	kilocalorie
lading	$Q$	eV	\eV	elektronvolt
elektrisch vermogen	$P$	kWh	\Kilo \Watt \Heure	kilowattuur
hardheid H <sub>2</sub> O	$D$	D	\Duits	graad duits

**tabel 26** Niet-reguliere eenheden.

# Bijlage 11

## The T<sub>E</sub>X backend for *Jade* and the JadeT<sub>E</sub>X macros

Sebastian Rahtz  
Elsevier Science Ltd  
s.rahtz@elsevier.co.uk

### abstract

*Jade* is an implementation of the DSSSL specification, and includes a T<sub>E</sub>X backend; the JadeT<sub>E</sub>X macro package is needed to process the *Jade* T<sub>E</sub>X output. We describe how *Jade* and JadeT<sub>E</sub>X work together.

This article was written for the newsletter of the International SGML Users Group.

## 1 Introduction

DSSSL (ISO standard ISO/IEC 10179:1996 — Document Style Semantics and Specification Language) is one of the great frustrations of the SGML world. On the one hand, it is the eagerly-awaited result of years of work, which finally seems to have produced a genuinely useful model of multi-lingual text transformation and formatting (Figure 1). On the other hand, its very complexity and completeness means that

- there are no full implementations of the standard;
- there are no formatting engines capable of delivering its requirements;
- the XML community has been forced to develop a new lighter-weight style language for the Web (XSL).

In addition, the style of the language used for writing specifications (more or less, but not exactly, Scheme) has had an unfortunate off-putting effect on those more used to Omnimark or C++.

However, the (publicly visible) DSSSL community is slowly developing, thanks in the main part to two things: James Clark's partial implementation, *Jade*<sup>1</sup>, and the considerable effort put by Norm Walsh into DSSSL specifications for formatting documents marked up against the Docbook DTD. The latter effort is targeted at HTML and RTF output, and has effectively demonstrated that the lack of the DSSSL transformation language in *Jade* is no barrier to very useable results.

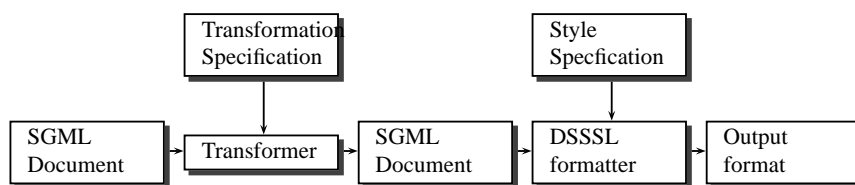


Figure 1. The DSSSL process

1. Standing for James' Awesome DSSSL Engine, by some accounts from James; and since this is the most charming, lets believe it.

But there is more to *Jade* than RTF and HTML. What if we need *real* typesetting, beyond the capabilities of Microsoft? Then we can turn to the  $\TeX$  backend. The  $\TeX$  system has many advantages

1. It is free, well-understood, and available for all machines;
2. It is designed for rule-based batch typesetting;
3. It is (pretty) good at page makeup, and very good at paragraph makeup;
4. It understands the full range of typesetting minutiae (hyphenation, fonts, math, etc);
5. It has a variant (pdf $\TeX$ ) which produces PDF directly, making it more congruent with modern pre-press;
6. It is up to date with respect to Unicode (Omega).

For many years, of course, SGML practioners have transformed their files to the input format of various formatting engines, including  $\TeX$ , but now we have a chance to write device independent specifications and use  $\TeX$ 's power to instantiate them.

## 2 $\TeX$ as a *Jade* backend

*Jade*'s  $\TeX$  backend (originally written by David Megginson, since modified by Sebastian Rahtz and Kathleen Marszalek) has a very simple model: it emits a  $\TeX$  command for the start and end of every flow object, defining any changed characteristics at the start of the command. This abstract  $\TeX$  markup can then be fleshed out by writing definitions for each of the flow object commands, and this is what the *Jade $\TeX$*  macro package provides. It is implemented on top of the widely used  $\LaTeX$  macro package, for a variety of reasons:

- $\LaTeX$  has proper support for fonts, similar to DSSSL's (the New Font Selection System);
- It has standardized multi-lingual, color, graphics inclusion, hypertext and tabular support;
- It has lots of 'functions' that one can borrow.

This means that it provides a good short cut to an implementation, to see whether  $\TeX$  can in fact meet the demands of DSSSL. It is important, however, for regular  $\LaTeX$  users to realize that no use is made of  $\LaTeX$  high-level constructs. There are no familiar sections, lists, cross-references, or bibliographies; everything is expressed in terms of vertical and horizontal space, font changes etc, explicit in the specification. Only page and line breaking is left to  $\TeX$ : the rest is up to the DSSSL code.

## 3 Installation and usage

*Jade*'s  $\TeX$  backend is available by default. The *Jade $\TeX$*  macros are delivered (at <ftp://ftp.tex.ac.uk/tex-archive/macros/jadetex/>) in a packed format; they must first be expanded, and then used to build a new  $\TeX$  format file. The sequence of command might look like this, using a modern  $\TeX$  system based on Web2c 7.2:

```
tex jadetex.ins
pdftex -ini "&pdflatex" -programe=pdfjadetex pdfjadetex.ini
tex -ini "&hugelatex" jadetex.ini
```

which produces format files `pdfjadetex.fmt` and `jadetex.fmt` which can be moved to where  $\TeX$  looks for such things. In practice, you will find a working system set up ready to go on the  $\TeX$  Live CD-ROM (see <http://www.tug.org/texlive/>).

Assuming we have a working system, usage can be as simple as

```
jade -t tex -d article.dsl article.sgml
pdfjadetex article.tex
```

which processes the SGML file `article.sgml` with the DSSSL specification `article.dsl` and writes `article.tex`; this is then run through pdfT<sub>E</sub>X, which will write `article.pdf`, which you can view or print.

## 4 Some simple examples

Let us look at what goes in and what comes out. If the DSSSL specification looks like this:

```
(root (make simple-page-sequence
      right-header: (literal "DSSSL Test")
      center-footer: (page-number-sosofo)
      font-family-name: body-font-family
      page-n-columns: 2
      page-column-sep: 16pt
      header-margin: .5in
      footer-margin: .5in
      left-margin: 1in
      right-margin: 1in
      top-margin: 1in
      bottom-margin: 1in
      page-width: 211mm
      page-height: 297mm))
```

then the intermediate T<sub>E</sub>X file (which is *not* meant to be edited by humans!), looks like this:

```
\SpS{\def\FamName{iso-serif}
\def\PageNColumns{2}
\def\PageColumnSep{16\p@}
\def\HeaderMargin{36\p@}
\def\FooterMargin{36\p@}
\def\LeftMargin{72\p@}
\def\RightMargin{72\p@}
\def\TopMargin{72\p@}
\def\BottomMargin{72\p@}
\def\PageWidth{598.11\p@}
\def\PageHeight{841.889\p@}
}
```

which clearly demonstrates the way *Jade* simply writes a macro name for the flow objects, and a series of `\def` commands for the characteristics.

Now consider some simple SGML markup

and `<it>Uncle Tom Cobbley</it>` and all

processed by this DSSSL

```
(element it
  (make sequence
    font-posture: 'italic
    (process-children-trim)))
```

from which *Jade* will write

```
and \Node{\def\Element{11}}%
\Seq{\def\fontPosture{italic}}%
Uncle Tom Cobbley
\endSeq{\endNode{}} and
all.\endSeq{\endNode{}}
```

Here we see as a side effect that almost every object that comes out of *Jade* has an ‘Element’ identifier, used for cross-referencing.

What about mathematics? This is  $\text{T}_{\text{E}}\text{X}$ ’s traditional strength, and something that few typesetting systems handle well. The intent of the following SGML markup should be fairly clear (to render as

$$\frac{X}{Y}$$

):

```
<fd><fr><nu>X<de>Y</fr></fd>
```

The DSSSL specification might look like this:

```
; displayed equation
(element fd
  (make display-group
    (make math-sequence
      math-display-mode: 'display
      min-leading: 2pt
      font-posture: 'math
      (process-children-trim))))

; fraction
(element fr
  (make fraction
    (process-children-trim)))

(element nu
  (make math-sequence
    label: 'numerator
    (process-children-trim)))

(element de
  (make math-sequence
    label: 'denominator
    (process-children-trim)))
```

and that results in the (slightly simplified)  $\text{T}_{\text{E}}\text{X}$  code:

```
\DisplayGroup{}
\MathSeq{
  \def\MathDisplayMode{display}
  \def\MinLeading{2\p@}
  \def\MinLeadingFactor{0}
  \def\fontPosture{math}
}
\FractionSerial{}
```



```

\insertFractionBar{}
\FractionNumerator{}
\MathSeq{}
X
\endMathSeq{}
\endFractionNumerator{}
\FractionDenominator{}
\MathSeq{}
Y
\endMathSeq{}
\endFractionDenominator{}
\endFractionSerial{}
\endMathSeq{}
\endDisplayGroup{}

```

For T<sub>E</sub>X aficionados, the implementation of these macros is as follows (simplified):

```

\def\FractionSerial#1{#1\bgroup}
\def\endFractionSerial{\egroup}
\def\FractionDenominator{}
\def\endFractionDenominator{}
\def\FractionNumerator{}
\def\endFractionNumerator{\over }
\def\insertFractionBar{}

```

## 5 DSSSL extensions supported in JadeT<sub>E</sub>X

The subset of DSSSL supported by *Jade* only covers ‘simple page sequences’, which do not allow such stables for the scientific publishing community as floating figures, footnotes, and multiple columns. To work around this, the T<sub>E</sub>X backend of *Jade* supports the following extra flow objects and characteristics:

```

(declare-flow-object-class page-float
  "UNREGISTERED::Sebastian Rahtz//Flow Object Class::page-float")
(declare-flow-object-class page-footnote
  "UNREGISTERED::Sebastian Rahtz//Flow Object Class::page-footnote")
(declare-characteristic page-n-columns
  "UNREGISTERED::James Clark//Characteristic::page-n-columns" 1)
(declare-characteristic page-column-sep
  "UNREGISTERED::James Clark//Characteristic::page-column-sep" 4pt)

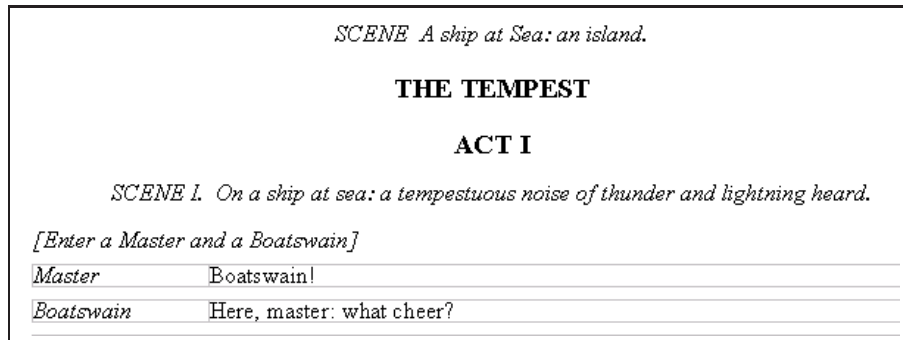
```

(the RTF backend also supports the last two.) These allow the specification author to produce simple multicolumn pages, with footnotes and floating figures.

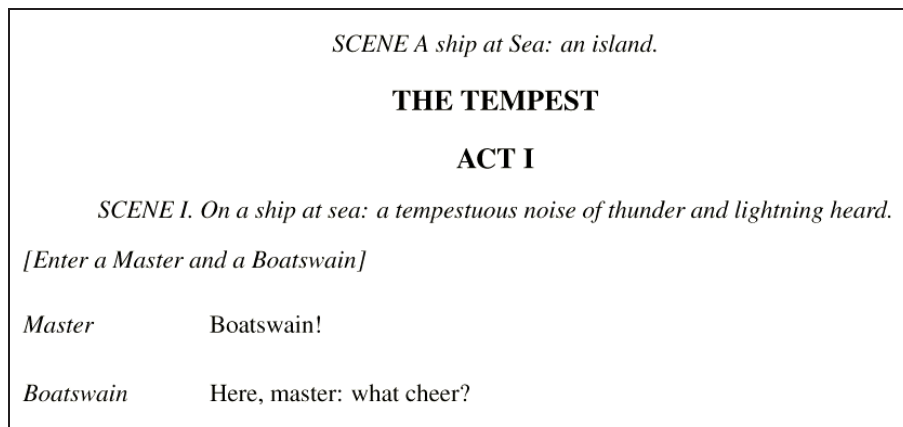
Numbered equations are still an unresolved issue, since they too require more complex objects than *Jade* supports.

## 6 Is JadeT<sub>E</sub>X useable in practice?

It is not hard to process simple texts with *Jade* and see more or less identical output from the RTF and the T<sub>E</sub>X backends (Figures 2 and 3). The pages displayed in Figures 5 and 6 are more interesting, as they demonstrate that a DSSSL specification, *Jade*, and JadeT<sub>E</sub>X can produce plausible pages of a scientific article. Figure 4 shows a portion of the math in Figure 5 as displayed in Microsoft Word, demonstrating the inadequacy of the math support in RTF (though the spacing can be adjusted for a somewhat better display).



**Figure 2.** The Tempest, formatted by Microsoft Word



**Figure 3.** The Tempest, formatted by  $\text{\TeX}$

## 7 Conclusions

The potential power of SGML/XML, DSSSL and  $\text{\TeX}$  working together is fairly awesome. Unfortunately, there are some downsides to what we have today:

- There is (perhaps) no formatter capable of dealing with the DSSSL page model;
- There is no implementation of the full DSSSL transformation language;
- There is no implementation of the full specification language;
- You cannot easily tweak the  $\text{\TeX}$  output;
- DSSSL has no equivalent of the integrated graphical languages we are beginning to appreciate in the  $\text{\LaTeX}$  world;
- DSSSL may be sidelined by the emerging XSL standard.

In addition, Jade $\text{\TeX}$  has some problems of its own:

1. The table support (while distinctly improved since its initial release) is not complete
2. The handling of white space and line-endings is hard to get right in all circumstances
3. The penalties for paragraph breaking are complicated, and not necessarily right, while DSSSL's hyphenation characteristics have not even been looked at yet.

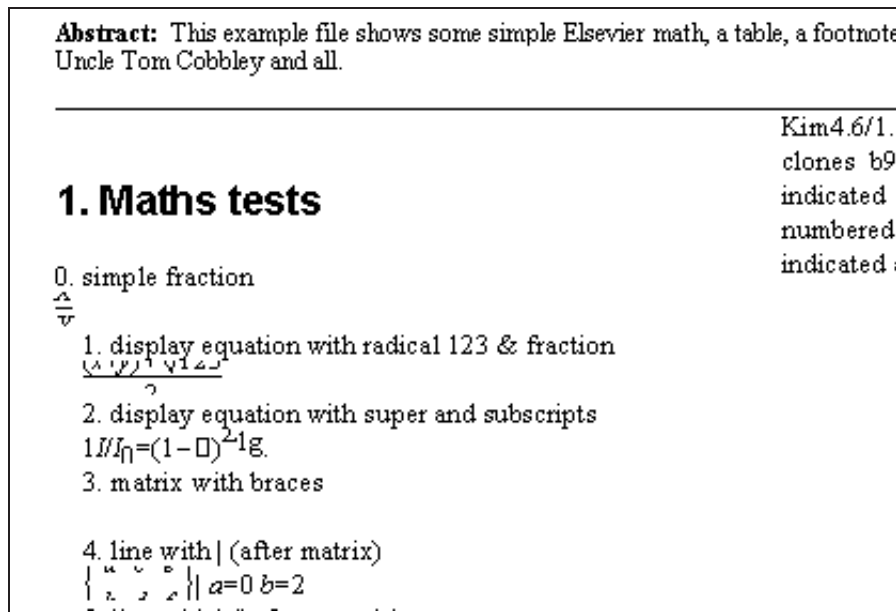


Figure 4. RTF math in Microsoft Word

We also have to consider what will happen if we get a full DSSSL implementation, where the front end will provide parallel streams of input (for the body text, footnotes, floats etc), along with information about how items in the streams have to be synchronized (e.g. appear on the same page), and each stream will have its own independent stack for inherited characteristics. The T<sub>E</sub>X backend currently handles flow objects with multiple streams by serializing the streams, i.e. giving you them each in sequence. This would not work well for column-set-sequence. You would get the main body text for a chapter followed by all the footnotes for the chapter, followed by all the floats for the chapter, plus information about which point in the body text was to be synchronized with each float/footnote. This would almost certainly be a monumental task to program in T<sub>E</sub>X, and really needs a complete rethink of how the backend works.

All this does not mean that we should despair. The *Jade* DSSSL implementation already supports a huge amount of useful transformation and specification code, and T<sub>E</sub>X is close to being a DSSSL-capable formatter. Since the T<sub>E</sub>X world knows about Unicode (in the shape of the Omega project, see <http://www.ens.fr/omega>) we are closer than many systems to dealing effectively with true multi-script typesetting.

In the medium term, it will be necessary to rewrite the font handling inside the backend, for speed, and to optimize the handling of labels and references (so many things are labelled at present that T<sub>E</sub>X can run out of memory for potential cross-references). In the longer term, it would nice to rewrite the JadeT<sub>E</sub>X macros to be independent of L<sup>A</sup>T<sub>E</sub>X, and reimplement it to use Omega and native Unicode.

DSSSL is not perfect, and neither is T<sub>E</sub>X; but they do make a very nice combination. . .

# Test file for math, multicolumns, floats, footnotes, page displays etc etc

Sebastian Rahtz

**Abstract:** This example file shows some simple Elsevier math, a table, a footnote, a float, a page display, all the entities, and *Uncle Tom Cobbley* and all.

## 1. Maths tests

0. simple fraction

$$\frac{X}{Y}$$

1. display equation with radical 123 & fraction

$$\frac{(x + y) + \sqrt{123}}{2}$$

2. display equation with super and subscripts

$$1I/I_0 = (1 - )^{\lg}$$

3. matrix with braces

$$\begin{cases} a & b \\ c & d \\ e & f \end{cases} a = 0b = 2$$

4. line with | (after matrix)

$$\begin{cases} a & b \\ c & d \\ e & f \end{cases} | a = 0b = 2$$

5. line with | (before matrix)

$$| \begin{cases} a & b \\ c & d \\ e & f \end{cases} a = 0b = 2$$

6. nested matrix with braces

$$\left( X \begin{cases} a & b \\ c & d \\ e & f \end{cases} \right) a = 0b = 2$$

7. nested fraction

$$\frac{(x + y) + \frac{X}{Y}}{2} \quad \frac{(x + y)^2 - 4a}{2}$$

8. Fence

$$\left( \begin{cases} aaa & b \\ c & d \\ e & f \end{cases} \right)^{b \times 5 = \sqrt{49202}} \sin \alpha \quad a = 0b = 2$$

9. Boxing  $A+B$

10. Some operators: summation, product, integral etc First, display math:

$$\sum_a^b \prod_c^d \int_e^f \frac{h}{g} \sin \alpha \quad \sum_{1111^{bbb}}^{2222^{3333}}^{5555}$$

Now inline math:  $\sum_a^b \prod_c^d \int_e^f \frac{h}{g} \sin \alpha \quad \sum_{1111^{bbb}}^{2222^{3333}}^{5555}$

11. A radical with a radix

$$\sqrt[3]{123}$$

## 1.1. Second-level header

12. A simple table

1	2	3	4	5
6	7	8	9	0
a	b	c	d	e

## 2. Footnotes, floats etc

13. A footnote, number 63<sup>63</sup>

14. A float occurs, related to here

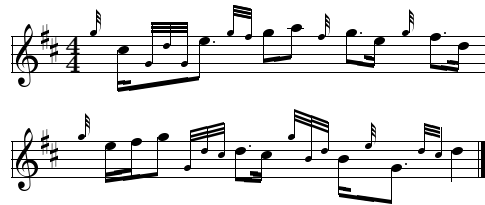
## 3. The Elsevier entity set

AElig	Æ
And	
Barwed	<8966>
Bcy	Б
Cap	Ⓜ
Colon	
Cup	☪
Dagger	‡
Dcy	
Delta	Δ
DotDot	
ETH	Ð
Ecy	Ɔ
Fcy	Φ

<sup>63</sup> Some useful text

Figure 5. Sample pages, part 1

**Figure 1.** V<sub>H</sub>3 amino acid sequences used by autoantibodies F1.1 (A) and T1.1 (B). Dashes indicate sequence homology with the most homologous germline V<sub>H</sub> region (F1.1; clones Kim4.6/1.9III of 21 and p3, 4, 7, 8 of 22; T1.1; clones b9-12, 33, 35 of 22); substitutions are indicated below. Amino acid sequence is numbered. The CDR1 and CDR2 regions are indicated above each sequence.



Gamma	Γ	VDash	
Gcy	Г	Vbar	
Gg	»»	Vdash	⦶
Gt	»	Vvdash	⦶⦶
HARDcy	Ъ	Xi	Ξ
Icy	И	YAcy	Я
Jcy	Й	YUcy	Ю
KHcy	Ч	Ycy	Ы
Key	К	ZHcy	Ж
Lambda	Λ	Zcy	З
Larr	←	acoint	
Lcy	Л	acute	´
Ll	««	aelig	æ
Lt	«	aleph	ℵ
Map		alpha	α
OElig	Œ	amalg	⦶⦶
Omega	Ω	amp	&
Or	Ø	and	^
Oslash	Ø	ang	∠
Pcy	Π	ang90	<8735>
Phi	Φ	angmsd	∠
Pi	Π	angsph	∠
Prime	''	ap	≈
Psi	Ψ	ape	≅
Rarr	→	apid	
SHCHcy	Щ	ast	*
SHcy	Ш	asym	≍
SOFTcy	Ъ	barwed	⌒
Sigma	θ	bcong	<8780>
Sub	⊆	bcy	б
Sup	⊇	becaus	∴
THORN	Þ	beta	β
TScy	Ц	beth	⦶
Theta	Θ	bowtie	⌘
Ucy	У	bprime	ʼ
Upsi	Υ	breve	<728>

**Figure 6.** Sample pages, part 2

## Diversity in math fonts\*

Thierry Bouche  
 Institut Fourier  
 Laboratoire de mathématiques pures  
 Université Joseph Fourier – Grenoble I, France  
 thierry.bouche@ujf-grenoble.fr

**abstract**

We will examine the issues raised when modifying (L<sup>A</sup>)T<sub>E</sub>X fonts within math environments, and attempt to suggest effective means of accessing a larger variety of font options, while avoiding typographic nonsense.

“Don’t mix faces haphazardly when specialized sorts are required”

— Robert Bringhurst [9]

**Stating the problem**

The advent of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> has resulted in a type of ‘standardizing’ of font selection schemes (NFSS, in other words). The advantages are many, but the main one for me is this: unlike other software that’s more expensive and of poorer quality, changing fonts is as easy as changing your socks. In fact, the ‘heroic’ days of *plain* are just a memory, where changing from the default `\textfont0` meant generating a new format, not to mention various encodings ... The temptation to play is therefore very great, especially if you want to break with the monotony of countless preprints and other (L<sup>A</sup>)T<sub>E</sub>X documents.<sup>1</sup> I won’t say much about anything other than POSTSCRIPT fonts, mainly because I can only test my hypotheses on them. Sebastian Rahtz’ *psfonts* now allows anyone equipped with a POSTSCRIPT printer to choose their text fonts for use with L<sup>A</sup>T<sub>E</sub>X: Times, Bookman, New Century Schoolbook, Palatino. You can ftp to CTAN sites to pick up everything you need to use a wide variety of commercial fonts. Alan Jeffrey’s *fontinst* program makes it easier to create the interface needed to use POSTSCRIPT fonts with L<sup>A</sup>T<sub>E</sub>X. The choices are almost limitless, with some 20,000 fonts to choose from for your document.<sup>2</sup>

Unfortunately, if your document has equations, this diversity is pretty much an illusion. There are actually very few math fonts, and of these, only a few are designed to

\* This article was previously published in TUGboat 19 (2), June 1998, pp. 121-135; translation by Christina Thiele from the original in Cahiers GUTenberg 25, novembre 1996, pp. 1-24.

work with T<sub>E</sub>X. To my knowledge, here are the font collections that provide a significant set of mathematical glyphs:

**The native T<sub>E</sub>X fonts:** these are, of course, `cmml/cmsy/cmex`, with the addition of the AMS symbol fonts (`msam/msbm`);

**Some non-native T<sub>E</sub>X fonts:** initially developed in MetaFont format to complement the Concrete text fonts by Knuth, are the Euler fonts, which aren’t coded in quite the same way as the standard T<sub>E</sub>X fonts, and do not really provide a replacement, as so many extra symbols are missing. There is an option available on CTAN, `euler.sty` by F. Jensen et F. Mittelbach, which makes installing the Euler fonts easier. However, the Eulers weren’t designed to be combined with any particular text fonts – the best you can say is that they ‘work’ with Bitstream Charter or, of course, Concrete. Karl Berry has recently used Euler with Palatino, a valid combination since both font families were designed by Hermann Zapf. U. Vieth designed a math font based on Knuth’s Concrete fonts. It is also missing many variants and glyphs, but enjoying an NFSS support package;

**MathTime:** this family is a full alternative to the CM collection, but is missing some glyphs from the AMS collection;

**Lucida New Math:** this family is as comprehensive as possible;

**PostScript Symbol font:** almost as widespread as Courier, it yields upright Greek letters, and includes a number of basic math symbols;

**Mathematical Pi:** usually used by (photo)typesetting software, this is a collection of six fonts whose glyph set is rather extensive;

**and some more:** let us also notice that many scientific software programs use proprietary fonts to display equations on-screen or print them on paper.<sup>3</sup> Not to

1. Note that ‘L<sup>A</sup>T<sub>E</sub>X’ can be understood as having two relatively independent meanings: it’s a program to typeset scientific texts, and it’s also a standard in the electronic exchange of documents. This article is concerned with the former: producing documents which are to be printed and thereby benefit from typographic programs adapted to the purpose.

2. This count, based on Unique IDs, is relatively outdated, as recent fonts IDs would imply that we’ve reached a count approaching 90,000!

3. Among them, Mathematica provides a font set with a rather rich set of glyphs. U. Vieth has made T<sub>E</sub>X virtual fonts for them, along the

mention the specific proprietary fonts used by some publishers.

In current (L<sup>A</sup>)T<sub>E</sub>X, a math font family needs to have at least three members: math italic (`cmmi` is the default), symbols (`cmsy`), and extensions for building different-sized symbols (`cmex`). Taking design consistency and glyph set exhaustivity into account, of the fonts listed above, we are effectively left with three font families, alternatives which are both complete and unified (well, one less so than the others):

- the standard fonts based on Knuth's CM<sup>4</sup>
- Lucida, a slightly more complete set from Bigelow and Holmes, is now quite extensive
- MathTime,<sup>5</sup> an alternative from Michael Spivak; nevertheless can't be as general as the previous two, since it wasn't designed to complement anything beyond the Times text font (the Times was never seen as the roman version of a family with sans serif, and typewriter versions). The current situation, where combining Times with Helvetica and Courier is seen as 'natural', is more the result of commercial suppliers making this combination available in most word-processing programs and in printers.

From this point on, I will take it as a given<sup>6</sup> that these three font families offer everyone a professional level of quality and consistency of style. The remainder of this article is for adventurous spirits or dissatisfied putterers, especially for those who have become jaded by the over-use of the currently available options. One way to describe the problem we face is "What can I do if I want to use a different font for the text, without spending a lot of time and energy designing the corresponding math symbols?"

## Typographic limitations

There are three main features or characteristics which limit font combinations: color, style, and proportions. For two fonts to work together inobtrusively, these three traits should be as close as possible. This doesn't mean avoid contrasting fonts—just use contrasts with care. For example, you may want chapter titles to be clearly separate from the text, or have visually obvious heading levels (of course, such a contrast should not be used within paragraphs). We shouldn't forget that combining similar fonts was common practice in printing and publishing. Printers of lead type had far fewer font typefaces available to them than our computers usually provide: a 17pt Caslon in the title combined with Garamond for the text, or bold italic Plantin used with Granjon were perfectly reasonable—if you had no other choice! The first example is a deliberate

*Let  $x, y, z \in \mathbb{Z}$ ; for  $f, \alpha$*

**Figure 3.** Text done in Times, math in Computer Modern.

attempt to startle the reader of today, when vendors pretend that their fonts are infinitely scalable: it's better to use a 17pt font at its design size than to scale the text font up, which is sure to yield something too bold, too round, and with too large an x-height.

## Color

A font that is more or less bold or condensed determines the grayness or 'color' of a page of text. Other parameters—interline space, interword space, margins—all affect color. Keep in mind that the L<sup>A</sup>T<sub>E</sub>X default page makeup parameters assume CM fonts. Using another font family may require adjustments to some of these parameters. Grayness is determined by the white spaces, which are therefore important parameters for typography. Variations in color are often inevitable in math: equations, for example, can change the interline spacing or force large white spaces. At the same time, though, in-line equations should have a minimum effect on the surrounding text.

A typical example would be a math article set with `times.sty`: since Times is a very "black" font, the material in math mode quite literally gives the impression that there's a hole in the page! figure 3 shows this, to a certain extent. Other than the perennial Times, books are often set with less dense fonts, such as Baskerville, Plantin, Minion, or Garamond. Depending on which one is used, these fonts have a color which is slightly darker than CM, while still being lighter than either Times or Lucida. It is the use of such font families that is at the heart of our problem.

## Style

Font *style* is what I call the design specifics of a font's characters. It's not a quantifiable feature or trait—although, one can consult classifications such as the one by Maximilien Vox to identify fonts of relatively similar styles. Strictly speaking, CM is a "Didone" font, although it has more in common with fonts of the Century/De Vinne type

---

lines of `mathptm`; see below.

4. As I am primarily interested here in font *design* rather than implementation, I won't spend much time distinguishing fonts from various vendors, or in various formats. For instance, here I don't distinguish between Knuth's CM fonts and Knappen's EC, which is largely based on the former. The slanted CM smallcaps are from the EC font.

5. Linotype's Mathematical Pi, which has no arrows or italics, is insufficient for use with T<sub>E</sub>X; we only consider it as a *complement* to MathTime.

6. This view is shared by Berthold Horn[6], whose article in TUGboat provides useful details on T<sub>E</sub>X math fonts.

CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\underbrace{\{g, \dots, g\}}_{k \text{ } g}$ ,  $\underbrace{\{h, \dots, h\}}_{\ell \text{ } h}$  taken in the basis  $(\vec{i}, \vec{j})$ .  

$$k + \ell \text{ elements}$$

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 1. The (L<sup>A</sup>)T<sub>E</sub>X default: Computer Modern.

than with Bodoni; considered Transitional Mécane (a hybrid category between Mécane and Didone, which isn't itself in the Vox AtypI classification). That's as far as any classification can help – nothing can replace education and experience. All the same, character design can be a significant obstacle to combining fonts. In particular, if we follow the standard practice of using italics in math and theorems, we run the risk of having two different styles in the same sentence, the proximity causing a rather jarring contrast, an effect which can be heightened since italics are often where design idiosyncrasies are most obvious. figure 5 shows that it's not a simple problem.

**Proportions**

The last of the three features concerns character proportions. A font style establishes the relationships of the various dimensions of its face: x-height, height of uppercase letters, ascenders, descenders. For POSTSCRIPT fonts, dimensions are specified in the afm file<sup>7</sup> in terms of 1000pts for each given character, referring to, respectively,

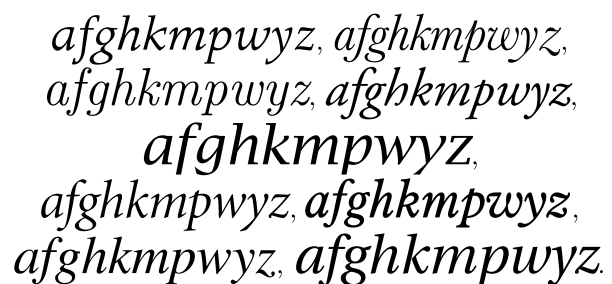


Figure 5. Apollo, Baskerville, Computer Modern, Adobe Garamond, Lucida, Minion, Plantin, Times, and Utopia (all at 21pts).

the XHeight, CapHeight, Ascender, and Descender.

7. A good qualitative description of what should be expected from the metrics of a font is provided in [10]. For a more technical approach, see [1].



CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\underbrace{\{g, \dots, g\}}_{k \text{ } g}$ ,  $\underbrace{\{h, \dots, h\}}_{\ell \text{ } h}$  taken in the basis  $(\vec{i}, \vec{j})$ .  

$$k + \ell \text{ elements}$$

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 2. The same example as Figure 1, done in Lucida.

Each of these parameters can vary independently of the others, as a quick glance through any font catalogue will prove. French printing tradition, going back to Garamont and Granjon, favors what are called 'humanist' characteristics: a fairly small x-height, with uppercase letters below the height of the tallest ascenders, and with generous descenders. In contrast, twentieth-century faces typically have short descenders, ascenders that seem almost atrophied and reduced in size, for what appear to be reasons of efficiency, rationalization of paper savings... In the italic examples in figure 5 (all fonts are in the same size), Adobe Garamond and Lucida represent diametrically opposed concepts. Clearly, one shouldn't mix fonts with x-heights that vary too widely, especially in mathematics material, where alignments must default to precise positions (superscripts, for example). You can always bring two fonts of different x-heights together by changing the scale but the results can be unpleasant if their respective proportions are too divergent. The example in figure 7 demon-

*Mixing italic fonts  
draws attention to  
their differences.*

Figure 7. Garamond and Lucida scaled to the same x-height as Lucida at 20pts.

strates yet another factor: the slope of the italic characters (ItalicAngle).<sup>8</sup>

8. T<sub>E</sub>X is satisfied with slightly less specific information, which is stored in the t<sub>f</sub>m file, for its seven \fontdimen values: the value of an em (the size of a given font, implicit in the a<sub>f</sub>m), the value of an ex (XHeight), and the tangent of ItalicAngle. The remaining dimensions concern spacing, whereas a POSTSCRIPT a<sub>f</sub>m file specifies the

CONJECTURE 1. — *Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:*

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

*Evidence for the conjecture.* — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{C}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\overbrace{\{g, \dots, g\}^k}^k \overbrace{\{h, \dots, h\}^\ell}^\ell$  taken in the basis  $(\vec{i}, \vec{j})$ .  
 $k+\ell$  elements

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 4. Again the same example, done in MathTime.

### Customizing a suitable math font

In light of these negative aspects to the problem, we will discuss two methods which each provide a “solution”. The examples have been tested, in that they provide a reasonable level of quality in documents containing mathematics. However, neither can pretend to address either the reliability or the quality of the three math font families discussed in our first section. Keeping in mind the remarks attached to its presentation, I would be willing to print a book using the second method (§ ), but I’d only make photocopies if the first method (§ ) had been used. In order of increasing difficulty in implementation, we’ll start with Alan Jeffrey’s `mathptm` option, then we’ll examine how simple NFSS commands or a virtual font created via `fontinst` allows us to choose, character by character, each font used within a math environment.

#### Mathptm

The `mathptm` distribution includes virtual fonts created by using `fontinst`, as well as the style option `mathptm.sty`, which makes it possible to use the glyphs of Times in math

mode with  $\LaTeX$ . The font is a marvel in that it manages to simulate the majority of the 384 glyphs found in the three math font families, by accessing the Times and Symbol fonts available on any POSTSCRIPT printer (the calligraphic uppercase letters, accessed via the `\mathcal` command, come out in Zapf Chancery); as a last resort, some characters are taken from Computer Modern. The style option modifies the  $\LaTeX$  defaults by invoking these various math font families, adjusting spacing parameters in math mode, and modifying the size of the type body for first- and second-order exponents. This last operation is interesting, because it pushes the ‘standard’ POSTSCRIPT fonts to their limits for typesetting mathematics. At 10pts,  $\LaTeX$  uses fonts at point sizes 10, 7, and 5, for normal text, super- and sub-scripts, and second-order super- and sub-scripts, respectively. Each of these sizes corresponds to a distinct font in the Knuth distribution, since it’s necessary to make optical corrections in a 5pt font so that it’s read-

width of the space character, but does not control the elasticity of an interword space.

CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card}\Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, O$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card}\Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^k \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = k}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\underbrace{\{g, \dots, g, h, \dots, h\}}_{k+l \text{ elements}}$  taken in the basis  $(\vec{i}, \vec{j})$ .

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 6. Again the same example, done with Mathptm (& dotlessj).

able. POSTSCRIPT printers have only one font (designed at a 12pt size) for each variant in the Times family. This means the only way to get a 5pt Times font is by applying a scaling but without optical correction, which in turn means the characters are difficult to read. Mathptm.sty redefines these sizes to 10, 7.4 and 6pts, which reduces – but does not eliminate – the visual problems. Mathematical Pi, Lucida, and MathTime will all show this flaw, hence the user will always have to adapt type body sizes with reference to readability. A few of the PostScript Multiple-Master fonts address the optical scaling issue, and while support for use with T<sub>E</sub>X is a bit tentative, I am convinced that within a few years, expert sets for these fonts will include all the refinements one could wish for.<sup>9</sup>

Mathptm is a free alternative to the MathTime fonts, but there are some drawbacks to it. The most obvious is that the Symbol font, which may be adequate for showing the characters available in the more popular word-processing programs, is decidedly smaller than the needs and possibilities available with T<sub>E</sub>X. For example, cmex has large expandable delimiters (the ones accessed via \big or \left)

whereas Symbol only has the regular parentheses, and the elements needed to create large parentheses; other expanded characters are simply scaled versions of the Symbol character. The other problem is that the lowercase Greek characters are upright. Now, almost all letters inside math mode are presented in italics: upright lowercase Greek letters may require italic corrections, which is fairly bizarre. Generating a slanted version of Symbol (using the Slant-Font operation in dvips for example), might work, but the result wouldn't be very good, especially if the slant was pushed to the values usually assigned to true italic fonts (roughly 15° vs. less than 10 for slanted fonts, to get something one could call 'acceptable'). Moreover, this would introduce yet another slope in math equations, which should be avoided as much as possible.

In summary, then, mathptm doesn't really offer a solution to the problem as outlined initially, but it does contain the kernel of two possible approaches to it: (1) a style op-

9. Provided someone feels the urge to produce the missing mathematical symbols ...

CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1)  $x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\overbrace{\{g, \dots, g, h, \dots, h\}}^{k \ g \quad \ell \ h}$  taken in the basis  $(\vec{i}, \vec{j})$ .  
 $k + \ell$  elements

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 8. Text in Apollo, math in Computer Modern.

tion relying on NFSS commands to modify math fonts, and (2) creating (via fontinst) virtual math fonts which address specific requirements. As I study the mathptm virtual fonts, I am convinced that there is no other satisfactory alternative to symbol and extension fonts: for such fonts, the issues of style, color, and proportion are not present, so the point is to ensure that their design is consistent and of good quality. My own approach is to use members of one of the three basic families in terms of what works best—using the serifs of `\prod` as a guide, for example. The Computer Modern versions are adequate for the majority of cases I've run into. The problem of uniformity of typographic characteristics is crucial for alphabets (letters, in other words) such as the Roman and Greek italic letters in the math italics fonts, and then on to the uppercase calligraphic letters, located in the symbol fonts, and uppercase Greek letters, which should appear in OT1-encoded text fonts (such as `cmr`). For example, for a professional-looking text, one might prefer to use `\mathcal` to access the uppercase cursive characters in the `rsfs` font or the Commercial Script

font (as shown in the examples on figures 11–13). It seems obvious to me that the preference will always be to choose the italic version of the text font for use in math mode. Below are two methods of achieving that goal.

**Mathfont**

I call `mathfont.sty` a 'generic' extension to access the necessary glyphs in math mode (its main features are discussed here, while the details are left for the reader to study).<sup>10</sup> The essentials are covered in the *LaTeX Companion*. L<sup>A</sup>T<sub>E</sub>X (essentially NFSS) introduces two concepts for fonts in math mode: *alphabets* and *symbols*. An alphabet is explicitly invoked by commands such as `\mathbf`. Assuming one has a text font, the math version of `\mathbf` can always be defined by means of a declaration such as:

```
\def\ED{\encodingdefault}% shorter!
```

10. It should be noted that a ready-to-use minimal adaptation of math italic for text italic can be found at: [ftp://fourier.ujf-grenoble.fr/pub/contrib-tex](http://fourier.ujf-grenoble.fr/pub/contrib-tex).

CONJECTURE 1. — *Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Fermat's last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:*

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

*Evidence for the conjecture.* — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\underbrace{\overbrace{g, \dots, g}^{k \ g} \overbrace{h, \dots, h}^{\ell \ h}}_{k+\ell \ \text{elements}}$  taken in the basis  $(\vec{i}, \vec{j})$ .

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 9. Text in Utopia, math in Computer Modern.

```
\DeclareMathAlphabet{\mathbf}{\ED}%
{\rmdefault}{b}{n}.
```

In this fashion, you can redefine `\mathcal` to access the ornamented letters one prefers. Additionally, if you want alphabets defined in this way to respond to the `\boldmath` command (to put mathematics material into boldface), you could do the following:

```
\DeclareMathAlphabet{\mathsf}{\ED}%
{\sfdefault}{m}{n}
\SetMathAlphabet{\mathbf}{\bold}{\ED}%
{\rmdefault}{b}{n}
\SetMathAlphabet{\mathsf}{\bold}{\ED}%
{\sfdefault}{b}{n}.
```

We're more interested in symbols, but note that `\mathversion` already exists (`\boldmath` is the same as saying `\mathversion{bold}`). All fonts used in math mode can have a different version, depending on what is specified by `\mathversion`: for example, a `textmathitalics`

or `textmathupright` version could be defined so that `math italics` would be accessed in a font invoked after `\mathversion{textmathitalics}`. This means it's possible to have several versions co-existing in the same document, just as it's possible to have several encodings for the text material. However, you have to be careful of  $\TeX$ 's limitations in this area: `\mathversion` can only be changed outside math mode, but never within an equation. Fonts invoked via this method must therefore be acceptable for such usage: if you've created a `textmathupright` version which replaces `math italics` by upright characters, these latter must be in a OML-encoded font in order to access such characters as lowercase Greek.

The following declarations introduce the four default symbol fonts:

```
\DeclareSymbolFont{operators}%
{OT1}{cmr}{m}{n}
\DeclareSymbolFont{letters}%
{OML}{cmm}{m}{it}
```

CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $\text{ff}i$ ,  $\underbrace{\{g, \dots, g, h, \dots, h\}}_{k+\ell \text{ elements}}$  taken in the basis  $(\vec{i}, \vec{j})$ .

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 10. Text in Utopia, with the mathfont option.

```
\DeclareSymbolFont{symbols}%
  {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}%
  {OMX}{cmex}{m}{n}
\SetSymbolFont{operators}{bold}%
  {OT1}{cmr}{bx}{n}
\SetSymbolFont{letters}{bold}%
  {OML}{cmm}{b}{it}
\SetSymbolFont{symbols}{bold}%
  {OMS}{cmsy}{b}{n}
\DeclareSymbolFontAlphabet{\mathrm}%
  {operators}
\DeclareSymbolFontAlphabet{\mathnormal}%
  {letters}
\DeclareSymbolFontAlphabet{\mathcal}%
  {symbols}.
```

To obtain the results we want, we select the most suitable symbols font (or largesymbols) font which works the best. What concerns us here are the operators and

letters. By default,  $\text{\LaTeX}$  uses the `cmr` operators font for:

1. digits 0–9
2. small delimiters (parentheses, brackets, etc.)
3. punctuation, including ; :
4. uppercase Greek letters
5. most accents
6. the + = signs

This extensive use of text characters in math mode is one of  $\text{\TeX}$ 's pitfalls (font changes are therefore very risky, particularly in plain).<sup>11</sup> While it may be natural to use the digits from the default text font, it's not likely that uppercase Greek letters will be found there. Parentheses and the + and = signs warrant a brief detour. Parentheses should be

11. This can only be addressed by the development of new font encodings, clearly differentiating text fonts (T1, for example) from text symbol complements (as in TS1) and math symbols (MC, MSP, currently being worked on by a TUG Technical Working Group).

consistent with their larger versions, and thus should come from the text font and matching extension font, all within the same font family (just as `cmr` and `cmex` are part of the CM family). The = sign is rather critical in that it joins the combinations  $\leftarrow$  and  $\rightarrow$  to produce  $\longleftrightarrow$ . Thus, it's really part of math characters; unfortunate that it's not part of a specific math font (the  $\leftarrow$  sign has the same function in simple arrows such as  $\longleftrightarrow$  even though it's part of the symbols font).

What `mathfont` does is define a second set of operators, called *textoperators*, and then it tells  $\LaTeX$  to take the digits and accents from there (which is a bit risky if you're accenting letters in math mode that aren't from the same font ...). This yields:

```
\DeclareSymbolFont{textoperators} {\ED}%
  {\rmdefault}{m}{n}
\SetSymbolFont{textoperators}{normal}{\ED}%
  {\rmdefault}{m}{n}
\SetSymbolFont{textoperators}{bold} {\ED}%
  {\rmdefault}{b}{n}
\DeclareMathSymbol{0}{\mathalpha}%
  {textoperators}{'0}
  (...)
\DeclareMathSymbol{;}{\mathpunct}%
  {textoperators}'{3B}
  (...)
% Attention: only in OT1 encoding
\DeclareMathAccent{\hat}{\mathalpha}%
  {textoperators}'{5E}
  (...)
```

Thus, using `fontmath.ltx` as a guide, it's possible to create a new symbols font, selecting the characters that will be in it.

The math italic font can be copied in the same way. By default,  $\LaTeX$  uses the `letters` font (`cmmi`) for the following:

1. regular letters (without accents)
2. a few punctuation signs, such as `,` `.`
3. the italic Greek letters
4. some letter-type symbols that are very useful, such as `\imath` ( $i$ ), `\jmath` ( $j$ ), `\ell` ( $\ell$ ), `\partial` ( $\partial$ ), some of the "harpoons" (e.g., `\leftharpoonup` ( $\leftarrow$ ))
5. some of the relatively useless symbols, such as `\smile` ( $\smile$ );
6. the only 'accent' that's not in a text font: `\vec` ( $\vec{\quad}$ )

If the selected font is a standard POSTSCRIPT font, it will only include letters and punctuation signs – the rest have to be found elsewhere. For example, in `mathfont.sty`:

```
\DeclareSymbolFont{textletters}{\ED}%
```

```
{\rmdefault}{m}{it}
\SetSymbolFont{textletters}{normal}{\ED}%
  {\rmdefault}{m}{it}
\SetSymbolFont{textletters}{bold}{\ED}%
  {\rmdefault}{b}{it}
\DeclareMathSymbol{a}{\mathalpha}%
  {textletters}'{a}
  (...)
\DeclareMathSymbol{A}{\mathalpha}%
  {textletters}'{A}
  (...)
\DeclareMathSymbol{,}{\mathpunct}%
  {textletters}'{3B}.
```

A word on the specific case of `\imath` and `\jmath`: the first is standard in POSTSCRIPT fonts (under the name *dotlessi*), whereas the second is absent.<sup>12</sup> Using the base ( $\vec{i}$ ,  $\vec{j}$ ) as a reference, it's clear that you can't use characters that are too different. As well, a word of warning about my decision to use a text font family *with its default encoding* – the choice was made purely as a way of limiting the amount of memory  $\TeX$  would allocate to the font metrics. Unlike the other characters modified up to this point in the article,  $i$  without a dot does not occupy the same slot in T1 or OT1 encodings. This method raises two additional problems:

- the number of font families declared at the same time is limited to sixteen. Each new declaration takes up one of these slots, so the method is not economic and carries certain risks.
- Math fonts and text fonts do not adhere to the same imperatives. We have to keep in mind that our initial problem revolves around the aesthetic impressions some glyphs have over others, whereas  $\TeX$  doesn't really care about glyphs, just their 'metrics', via the `tfm` file. In math mode, each character is an atom, which must be placed relative to other such characters, according to its type (relation, delimiter, etc.). This is why `\fontdimens 2, 3, 4` and `7` have a zero value in `cmmi`.<sup>13</sup> Although one can modify these global parameters dynamically from the  $(\LaTeX)$  source (in the `.fd` file, for example),

12. The 'successful' examples presented here demonstrate three reasonable alternatives for *dotlessj*. (1) Since Utopia's  $i$  is fairly similar to the  $j$  in Lucida, I used these two glyphs (of different origins) in figure 11. (2) This ruse is not possible for Apollo, so I simply edited the Apollo font and created a new character, copying  $j$  and removing the dot. (3) For figure 13, I was able to directly parameterize the POSTSCRIPT fonts, using a *header* that Bernard Desruisseaux graciously provided, and thus magically removed the dot from the  $j$  and made it the same height as a virtual  $j$ .

13. Respectively, these `\fontdimen` establish the space to put between words, the maximum space to add or take away, and the special spaces after punctuation (as used in Anglo-Saxon typography).



they would be attached to the font being loaded once only, which means it's not possible to call up the same font twice, using two different names, and assigning each one different parameters. Thus, to preserve the normal italics for text, the `mathfont` option produces atypical italics for math. The `\fontdimen` issue isn't too troubling since  $\TeX$  suppresses spaces in math mode; at the same time, though, it's not possible to suppress kerning or ligatures between letters, which can lead to some odd results for something like *Te* or *ffi*: *Te ffi*. Similarly,  $\TeX$  assumes that the side bearings (the lateral space which a designer adds to ensure that two characters of the same font don't touch) are as generous as those of its default fonts, which isn't really the general case. Super- and subscripts can end up looking like they're touching. If you use `mathfont`, you have to keep these points in mind: don't hesitate to include explicit kerning instructions (`\mkern`) to avoid inopportune ligatures and adjust the spacing.

The only way to get a math font, in terms of glyphs, similar to the one I've tried to obtain with `mathfont` (but uniform in terms of its metrics and independent of coding hazards) is to create a virtual font by using the same scheme, but making it more solid—and thus less flexible.

### Virtual fonts

Creating a virtual font with `fontinst` [7] (see [5] for many concrete examples similar to the ones presented here) essentially comes down to understanding the `\installfont` command. This generates a `vpl` file, which is a human-readable equivalent of the virtual font (`vF`). The following shows how the two fonts we're using from the `mathptm` distribution are created:

```
\installfamily{OT1}{ptmcm}{%
\transformfont{ptmr8r}{\reencodefont{8r}%
{\fromafm{ptmr8a}}}
\installfont{zptmcmr}{%
{ptmr8r,psyr,latin,zrhax,kernoff,cmr10}%
{OT1}{OT1}{ptmcm}{m}{n}{%
\installfamily{OML}{ptmcm}{%
{\skewchar\font=127}
\transformfont{ptmri8r}{\reencodefont{8r}%
{\fromafm{ptmri8a}}}
\installfont{zptmcmrm}{%
{kernoff,cmmi10,kernon,unsetalf,%
unsethum,ptmri8r,psyr,mathit,%
zrmhax}%
{OML}{OML}{ptmcm}{m}{it}{%
.
```

As you can see, the `\installfont` command has eight arguments:

- the first is the name (for the `tfm` and `vf` files for the font being generated)
- the second argument contains the set of file names (with extension `.mtx`) needed for creating *metrics* for each character
- the third indicates the internal coding used by `fontinst` for the font in question
- the next four arguments specify the parameters which allow  $\LaTeX$  to identify the font via the `fd` file, created by the `\installfamily` command
- the last argument makes it possible to configure the declaration contained in the `fd` file. This argument can be very useful for installing several virtual fonts that address optical scaling.

It's now possible to see that `mathptm` will install its operators font (replacing `cmr` in math) by using characters taken from `ptmr8r`, `psyr` and `cmr10` (that is, Times Roman re-encoded as `8r` for all the glyphs, Symbol, and Computer Modern roman). The order of these is important because all the glyphs required by an OT1 font are present in `cmr10`, yet `fontinst` follows the order in which it finds the glyphs needed for the encoding: letters are thus acquired from Times or, if they aren't there, they have to be 'simulated', thanks to macros in the file `latin.mtx`; the Greek uppercase letters come from the Symbol font. However, as mentioned previously, it can be risky using some Times glyphs, such as ( ) [ ] + =, so the `zrhax.mtx` file removes them from `fontinst`'s memory so that they're selected from `cmr10` instead. The `kernoff.mtx` file in turn suppresses the kerning that comes from `cmr10`; the resultant `zptmcmrm` font which is thus created will therefore be an *ersatz* font, providing symbols usable as operators without risk. The characters are accessed by the following declaration (which appears in `mathptm.sty`):

```
\DeclareSymbolFont{operators}{%
{OT1}{ptmcm}{m}{n}
```

The font `zptmcmrm` which replaces `cmmi` is obtained in the same way: you take all the glyphs from `cmmi10` (but leave their kerning behind), the `unsetalf` and `unsethum` files remove the lowercase Greek letters, and the italics provided by Symbol and Times Italic (respectively), `mathit` plays the role of `latin` for the OML fonts, and `zrmhax` adjusts certain spacing parameters which would not be acceptable if this font were nothing but a regrouping of characters from different sources. For the same reasons as we saw with `mathfont`, the side-bearings, which are distinctly more restricted in Times than in Computer Modern, are enlarged; accent positions in math mode, which are controlled through a special mechanism: pseudo kern pairs, with the so-called `\skewchar`, are enhanced. Thanks to `fontinst`, it's possible to correct all the shortcomings



CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\overbrace{\{g, \dots, g, h, \dots, h\}}^{k \ g \quad \ell \ h}$  taken in the basis  $(i, j)$ .  
 $k + \ell$  elements

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 11. Text in Utopia, math fonts based on Lucida and Computer Modern.

of the `mathfont` option, i.e., by specifically using glyphs chosen for aesthetic reasons (but arranged in the standard L<sup>A</sup>T<sub>E</sub>X encoding), and by adjusting all the metric parameters (kerning, side-bearings, etc.), so they can be made to work optimally in mathematics. The only thing missing from these examples is the command `scaled`, which makes it possible to adjust to the same value the x-heights of the various fonts mixed into a single font. Another advantage not to be discounted with the ‘virtual font’ solution: since it yields fonts which can replace the ‘original versions’ of Computer Modern, it is also very easy to use with all T<sub>E</sub>X dialects or formats (even `plain`). On the other hand, one could say that adjusting metric parameters is a subtle business, and should be left to a true typographer . . .

The examples I’ve presented throughout this article were produced with a certain degree of haste – they are far from being optimal. It’s not really feasible for me to distribute the virtual fonts I’ve been describing, because most are just the result of combining various bits of commercial fonts – which is why a finished package is not available. On the

other hand, I will try to share the skills I’ve acquired with difficulty. I don’t believe in a set of macros that can systematically generate virtual math fonts for, say, Palatino, Times, and New Century Schoolbook. These fonts are too different: it’s impossible for any given symbol not to clash with any one of them. As well, there are problems adjusting the x-heights and side-bearings that simply can’t be dealt with in a generic way.

I’ll finish off this presentation with a concrete demonstration (used in `plain TEX` by the secretaries at the Fourier Institute). The text font is T1 Utopia Expert scaled down to the x-height of `cmr10`. Utopia, which has a dark color to it, doesn’t really work with `cmmi`, although the symbols in `cmsy/cmex` don’t clash once they’ve been scaled down. After a few attempts, I finally chose Lucida for the upper- and lowercase Greek letters, Utopia Italic for math italics, and Utopia Expert for `oldstyle` digits. This yields the following:

```
\installfamily{OML}{putluc}%
```

CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLES, the three following formulae are very instructive:

(1)  $x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $ff_i$ ,  $\overbrace{\{g, \dots, g, h, \dots, h\}}^{k \ g \quad \ell \ h}$  taken in the basis  $(\vec{i}, \vec{j})$ .  
 $k + \ell$  elements

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 12. Text in Apollo, math fonts based on Computer Modern.

```

{\skewchar\font=127}
\transformfont{putri8r}{\reencodefont{8r}%
{\fromafm{putri8a}}}
\installfont{zputlucm}%
{kernoff,hocrim scaled 804,kernon,%
unsetalmf,unsetos,putri8r scaled 880,%
putr8x scaled 880,utmathit,zrmuthax}%
{OML}{OML}{putluc}{m}{it}{i}{}
\installfamily{OT1}{putluc}{}
\transformfont{putr8r}{\reencodefont{8r}%
{\fromafm{putr8a}}}
\transformfont{hlcr7t}{\reencodefont
{OT1luc}%
{\fromafm{hlcr8a}}}
\installfont{zputluc7t}
{putr8r scaled 880,putr8x scaled 880,%
hlcr7t scaled 840,latin,zrhax,%
kernoff,cmr10}%
{OT1}{OT1}{putluc}{m}{n}{i}{}

```

You can see that I've modified a few mathptm files, and have introduced a new unsets to suppress the oldstyle digits, so that they come from the expert font rather than from cmmi.<sup>14</sup> One final example with Apollo. Since this is a lighter face, I decided to plunder Computer Modern for all the math symbols that don't exist in Apollo.

```

\installfamily{OML}{mapcm}%
{\skewchar\font=127}
\transformfont{mapri8r}{\reencodefont{8r}%
{\fromafm{mapri8a}}}
\installfont{zmapcmm}
{kernoff,cmmi10,kernon,unsetalmf,%
unsetos,mapri8r scaled 1067,%
mapr8x scaled 1067,apmathit,zrmaphax}%
{OML}{OML}{mapcm}{m}{it}{i}{}
\installfamily{OT1}{mapcm}{}
\transformfont{mapr8r}{\reencodefont{8r}%

```

14. This manoeuvre doesn't really have any bearing on L<sup>A</sup>T<sub>E</sub>X but it does allow the plain T<sub>E</sub>X `\oldstyle` command to work.

CONJECTURE 1. — Let  $x, y, z$ , be integers; for  $\alpha \in \mathbb{N}$ , denote by  $\Omega_\alpha \subset \mathbb{N}$  the set of prime integers  $p$  (called  $p$ -primes in the sequel) such that the following equation (known as Frimas' last equation)  $x^p + y^p = z^p$  admits infinitely many solutions divisible by  $\alpha$ . We conjecture:

- $\Omega_\alpha \neq \emptyset$  ( $\Omega_\alpha$  is not empty),
- more precisely,  $\text{card } \Omega_\alpha > w$  where  $w$  is the well-known WHYLLES' constant.

Evidence for the conjecture. — Denoted by  $\mathcal{A}, \mathcal{M}, \mathcal{O}$ , the famous inferior constants of WHYLLES, the three following formulae are very instructive:

(1) 
$$x = 2\pi z \iff \text{card } \Omega_\alpha \mid \mathcal{M} \quad \text{and} \quad \varphi(t) = \frac{1}{\sqrt{2\pi}} \int_0^t e^{-x^2/2} dx$$

(2) 
$$\prod_{j \geq 0} \left( \sum_{k \geq 0} f_{jk} z^k \right) = \sum_{k \geq 0} z^n \left( \sum_{\substack{k_0, k_1, \dots \geq 0 \\ k_0 + k_1 + \dots = n}} f_{0k_0} f_{1k_1} \dots \right)$$

(3) Look at the product  $f f i$ ,  $\underbrace{\{g, \dots, g\}}_{k \text{ g}}$ ,  $\underbrace{\{h, \dots, h\}}_{\ell \text{ h}}$  taken in the basis  $(\vec{i}, \vec{r})$ .  

$$k + \ell \text{ elements}$$

Moreover, eq. (1) yields 
$$\pm \sqrt{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \end{vmatrix}} > 0.$$

Figure 13. Text in Minion MM, math fonts based on Computer Modern.

```

\fromafm{mapr8a}}
\installfont{zmapcm7t}
{mapr8r scaled 1067, mapr8x scaled 1067, %
cmlatin, zrhax, kernoff, cmr10}%
{OT1}{OT1}{mapcm}{m}{n}{ }
    
```

### Conclusion

I've illustrated different possible solutions in the above examples for Utopia and Apollo.<sup>15</sup> The principle behind these various illustrations has been the following: maintain the identical text each time, and change only the preamble, which takes care of modifying the fonts to be used via 'standard' NFSS (the equivalent of `times.sty`; see figure 8, 9), macros such as `mathfont` (figure 10), and ending up with composite virtual fonts, as described above (figure 11, 12). While the defects in `mathfont` are obvious enough (poor spacing around parentheses, the `ffi` ligature problem), you have to keep in mind that they can be fixed manually, which is a do-able operation in a document without a lot of

math formulae and typeset by someone who knows what they're doing. After demonstrating the three comprehensive font systems available – Computer Modern (figure 1), Lucida (figure 2) and MathTime (figure 4), plus `Mathptm` (figure 6) – I have shown what you can get, starting from two text fonts of incompatible design with math characters from either Computer Modern or Lucida.

The Apollo example, although somewhat marginal (I don't see it used that often), does show the benefits of the approach I use. Its style is wildly incompatible with Computer Modern yet its proportions and, above all, its color, are quite similar: once you remove the style incompatibility between them by using text italics in math, you get an undeniable uniformity and quality. Here's a list of fonts often used in books, which seem to me to lend themselves, without too much damage, to the games I've been playing with Apollo: Bembo, Adobe Garamond, Garamond Three, Granjon, Plantin *Light*, Times *Light*. Also possible, but

<sup>15</sup> A related discussion can be found in [2]. The task there was to modify not only the typography but also the layout of a  $\text{\LaTeX}$  book.

probably without the same degree of uniformity, are Adobe Caslon, Galliard, or Baskerville. To complement Palatino, Melior, Stempel Schneidler, New Century Schoolbook, I'd think of Lucida. While Stone or Rotis could prefer Math-Time symbols.

To conclude on a more pessimistic note: the French version of this article [3] was typeset in Minion—for me, one of the most beautiful fonts currently available, remarkably readable and elegant at the same time.<sup>16</sup> Today, I would choose it without hesitation for a good-quality journal. Unfortunately, the Minion design displays its acknowledgement of the Italian and French Renaissance too clearly. The initial version of this article had been prepared with the *Single Master* version (used by the journal *Libération*), which gave the page a relatively dark color, but not as dark as either Times or Lucida. And for this reason, none of the three basic fonts can complete it, even though Math-Time is probably the least problematic. Just as this article was being finished, I installed the *Multiple Master* version of Minion, which makes it possible to incrementally vary the thickness, the width, and the optical size yet still maintain a consistent design. As we've seen, this last property is crucial for the readability of smaller point sizes (superscripts, for example), and it's one of this font's undeniable advantages.

I've tried to experiment with the thinnest and widest instances so that color and proportion converged as much as possible with those of Computer Modern.<sup>17</sup> It's interesting to note that Hilmar Schlegel reports getting quite satisfactory results by using a similar method, but with a combination of a fairly bold and slightly narrowed Minion face with MathTime. figure 13 shows how this “works” quite respectably under ‘real’ conditions. Nevertheless, one can see that each glyph from the Computer Modern family is a surprise to the eye, and that there really is no alternative to it, at least regarding Greek letters.

Thanks—I went into this article without any idea where it would all end. Since I'm neither a programmer nor a typographer, nor a (L)A<sub>T</sub>E<sub>X</sub> guru (much less one in POSTSCRIPT), a certain number of unexpected roadblocks came up along the way. I'd like to thank everyone who helped me over these hurdles. In particular, I'd like to mention Jacques André, Bernard Desruisseaux and Hilmar Schlegel for their constructive criticisms and technical help, which made it possible for me to write this paper. Last but not least, it's a pleasure to thank Christina Thiele who undertook the present translation with patience & skill.

## References

1. Jacques André, Font metrics, *Visual and Technical Aspects of Types* (Roger D. Hersch, ed.), Cambridge University Press, 1993, pp. 64–77.
2. Thierry Bouche. *Approximation of one of Henri CARTAN's books: first try*, <http://www.loria.fr/tex/fontes/math/cartan-english.html>. The L<sup>A</sup>T<sub>E</sub>X Navigator, Nancy, France, 1995.
3. Thierry Bouche. *Sur la diversité des fontes mathématiques*, *Cahiers GUTenberg* **25** (1–24) novembre 1996.
4. Michel Goossens, Frank Mittelbach, & Alexander Samarin. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Addison-Wesley, Reading, USA, 1994.
5. Aloysius G. Helminck. *Contributions to fontinst*, on CTAN. 1994.
6. Berthold K.P. Horn. *Where Are The Maths Fonts ?*, TUGBOAT Vol. 14 **3**, 282–284, 1993.
7. Alan Jeffrey. *The fontinst package*, documentation accompanying the software distribution (the paper in TUGBOAT 14/3 is obsolete). June 1994.
8. Alan Jeffrey. *POSTSCRIPT Fonts in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>*, TUGBOAT Vol. 15 **3**, 263–268, 1994.
9. Robert Bringhurst. *The Elements of Typographic Style*. 2nd ed. Hartley & Marks, Vancouver. 1996
10. Lewis Blackwell. *Twentieth Century Type*. Calman & King, London. 1992

16. They say that Minion's on its way to becoming the ‘Times of the 21st century’, which is why I'm in a hurry to use it now before it becomes too passé!

17. The complete interface for production of the French version of this article will eventually become available on CTAN, as an example. A pre-version is already somewhere on my home site: see <ftp://fourier.ujf-grenoble.fr/pub/contrib-tex/psfonts/adobe>.