CSRD (2012)27(2): 113–126
DOI 10.1007/s00450-010-0145-x

SPECIAL ISSUE PAPER

Reconstituting typeset Marriage Registers using simple software tools

David F. Brailsford

Received ???? 2010 / Accepted ??? 2010

Abstract In a world of fully integrated software applications, which can seem daunting to develop and to maintain, it is sometimes useful to recall that a system of loosely-linked software components can provide surprisingly powerful and flexible methods for software development.

This paper describes a project which aims to re-typeset a series of volumes from the Phillimore Marriage Registers, first published in England around the turn of the last century. The source material is plain text derived from running Optical Character Recognition (OCR) on a set of page scans taken from the original printed volumes. The regular, tabular, structure of the Register pages allows us to automate the re-typesetting process.

The UNIX *troff* software and its *tbl* preprocessor are used for the typesetting itself, but a series of simple *awk*-based software tools, all of them parsers and code generators of one sort or another, is used to bring about the OCR-to-*troff* transformation.

By re-parsing the generated *troff* codes it is possible to produce a surname index as a supplement to the re-typeset volume. Moreover, this second-stage parsing has been invaluable in discovering subtle 'typos' in the automatically generated material. With small adjustments to this parser it would be possible to output the complete marriage entries in standard XML or GEDCOM notations.

Keywords Re-typesetting · OCR · *troff* · parsing · genealogy · hyperlinking · indexing

CR subject classification

David F. Brailsford (✉)
Document Engineering Research Group
School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
email: dfb@cs.nott.ac.uk

1. Introduction

This paper focuses on the re-typesetting of a series of printed marriage registers, published by the Phillimore company, in England, in the early years of the 20th century, as an aid to genealogical researchers. These printed registers were transcribed from the original hand-written registers, kept in local parish churches throughout England and Wales. The Phillimore Marriage Registers contain useful extra material not available in other transcripts taken from the hand-written originals. In particular, for each parish, there are a few paragraphs of introduction relating to the number of original hand-written registers that were available, and their physical condition. The use of local volunteers for the transcription task meant that many of the more illegible entries could be annotated with likely interpretations of obscure surnames, place names etc.

What follows should be read as a tale of 'practice and experience' which calls upon a combination of tried and trusted techniques to bring the Phillimore Registers back to life. The project became a document engineering exercise by automating the re-typesetting (henceforth abbreviated to 're-setting') of the Phillimore registers, as a first step towards indexing them and making their enhanced content available in structured form via XML and GEDCOM [1] notations (GEDCOM is an acronym for **GE**nealogical **D**ata **COM**munication).

1.1. English Marriage Registers

In 1538 Thomas Cromwell, the Vicar General to King Henry VIII, ordered that all baptisms, burials and marriages should be recorded. These records were normally kept at the local Parish Church. Some 60 years later, during the reign of Queen Elizabeth I, an Act was passed requiring that a copy be made of these records, which was to be sent to the bishop in whose diocese the parish was situated. These are known as "Bishop's Transcripts". It is interesting to note that in these early days the ecclesiastical year started and ended with the Feast of the Assumption of the Virgin Mary (Lady Day) on 25th

March and this can result in marriages for January, February and March being recorded with a year looking like $165\frac{8}{9}$, denoting that the event took place in the ecclesiastical year of 1658 but the secular year of 1659.

For the original registers large and wealthy parishes could afford professional scribes who wrote up the register entries, and the Bishop's Transcripts, in a beautiful calligraphic hand, whereas smaller parishes were reliant on the priest's own handwriting. Thus the Phillimore sub-editors assigned to making a transcript of a particular parish's records, ready for printing, were often faced with the challenge of deciphering near-illegible entries.

The next major legal event for Parish registers was Lord Hardwicke's Marriage Act of 1754 which ordered, among many other things, that the year dates of weddings had to follow the secular calendar and not the ecclesiastical one. The home parish of the bride and groom was assumed to be that where the marriage took place, unless explicitly stated otherwise. Finally, after 1812, baptisms, burials and marriages were to be recorded in separate volumes on standardised pre-printed forms.

For the next 83 years after 1754 the Hardwicke rules were followed, but on 1st July 1837 central recording for births, deaths and marriages was introduced by the government (even though the entries, in the first instance, would almost always be made by the parish priest). For many years thereafter the archived parish registers were retained in the local churches but the enactment of the Parochial Registers and Records Measure, in 1987, means that most UK church registers have now been moved to their respective County Record Offices.

In recent times there have been some attempts to make the contents of the Bishops' Transcripts or even transcripts of the parish registers themselves, available online. These transcripts are of variable quality, often with many gaps and omissions in the original material, and the transcription having often been made by volunteers with little or no local knowledge of the parish concerned. It is very rare to find, for any parish, a set of transcripts going all the way back to Thomas Cromwell's 1538 decree.

1.2. The Phillimore Parish Registers

Well before the Internet era, in the period 1890–1922, the Phillimore company of London conducted a major effort to transcribe the archived registers that were still being kept by the parish churches. Phillimore employed local sub-editors from each of the counties and enlisted the help of the parish priests themselves in doing the transcriptions. In this way local knowledge could be brought to bear in deciphering the hand-written marriage details. Despite the help of many volunteers the sheer volume of work, even for the marriage registers, was overwhelming; the transcribed marriage registers remained un-indexed and the transcription of baptisms and burials was not even begun. In recent years several vendors have made the Phillimore Register material available as page scans. For example, there is a free set of scans available at relatively low quality [2], and at least one set of much better scans is available for purchase on CD [3].

The transcribed material prepared by the Phillimore volunteers was typeset and issued as a subscription series of bound volumes. By way of example the size of print run, for each of the 15 volumes relating to the county of Derbyshire, was 150 copies. Volume 1 was released in 1906 but Volume 15 did not appear until 1922.

The idea for this project began with my borrowing the first three volumes of the Derbyshire Phillimore Marriages series from a local Nottingham library. For reasons that will be apparent from the author's surname, initial interest focused on Volume 2 which contained records for the parish of Brailsford[1]. Figure 1 shows the title page for the parish of Brailsford, firstly as a bitmap scan and then in its re-typeset form resulting from the present work.

The most striking feature of the published entries was the disciplined tabular nature of the typesetting. All the records were set on a standard measure of 3.7 inches, using overflow lines, if needed, with a 6-en indent.

Within each record, the details for groom, bride and parish were set in a column width of 2.8 inches with the remainder of the measure allocated to day, month and year. The names of groom and bride were invariably linked by an & character and optional subsidiary information was indicated either by position (e.g. prefixed titles such as "Mr.", "Mrs.") or by comma delimited subordinate clauses (e.g. ", yeoman,", ", of Derby,"). Such was the regularity that one could write down a context-free grammar for the entries, thereby prompting the thought that Phillimore's original aim of producing per-volume indexes might be achievable.

The plan that emerged was, first, to acquire plain and unformatted text from OCR treatment of bitmap Phillimore Register pages. The pages would then be reformatted, and the tabular records re-set, using UNIX-based software tools, as a first step towards checking the internal structure and consistency of the records. An incidental benefit of doing this is that a PDF version (or even a hard-copy, bound, version) of each re-set volume would be straightforward to produce. Moreover, after re-setting, further content enhancement and repurposing would become relatively easy. However, if re-setting is to be a step on the way to syntactic verification, and greater abstraction, then we need to review, briefly, the usual techniques available for re-setting a book that has gone out of print.

[1] Brailsford is a Derbyshire village situated about half way between the city of Derby and the town of Ashbourne.

1.3. Resetting a published book

A large amount of the book-length material still available as hard-copy volumes was created prior to the era of standardised text processing software, and the availability of PostScript. Some of it may have been set on a variety of second- and third-generation typesetting machines, some will have been set on hot-metal machines and yet more will have been hand typeset. It would certainly be possible to re-key and re-set a Phillimore volume, provided the necessary typefaces could be identified, using software such as Quark Express or Adobe InDesign. But this is a seriously labour-intensive process and one has to constantly cross refer to the original material to check that line spacing, measure and general layout characteristics are being faithfully reproduced. It is also a problem in WYSIWYG page layout programs that tabular layout capabilities are not generally as powerful as those found in markup-driven systems such as *troff* / *tbl* and LaTeX.

An alternative way to achieve some form of 'electronic' product is simply to scan in the pages of the book to a standard format such as JPG or TIFF. Type legibility then largely depends on the quality of the original material and then on the resolution of the scanner and the degree to which file compression has been applied within lossy formats such as JPG. Until relatively recently the drawback for electronic books consisting just of bitmap pages has been their lack of full-text searchability. In the mid-1990s Adobe Systems Inc announced a product called Capture, which capitalised on the fact that the PDF format could not only display conventionally typeset pages but could also handle bitmap pages, by importing TIFF images into PDF's internal equivalent of the PostScript bitmap format.

Capture starts from a set of bitmapped PDF pages and applies an Optical Character Recognition (OCR) engine to the task of acquiring the underlying text. There is then a second software engine capable of analysing the fonts in use within the document. With the aid of an interactive interface Capture replaces bitmapped text with the typeset equivalent in the appropriate font. The user is prompted to supply corrected words for all partially-recognised 'suspects'. Any suspects not replaced in this way are left as bitmap inserts in the surrounding re-typeset text.

Given sufficient time and persistence Capture can recreate a very close approximation to the original. However, such is the level of editing effort required that it occupies a niche market; the stand-alone Capture product has not developed beyond the 3.0 version, which was issued in 2000. Instead, a very popular, reduced-functionality, version has been added as a plugin to Acrobat itself. When used on a PDF file consisting of bitmapped pages the Acrobat Capture plugin has an OCR facility and a rough ability to gauge the metrics of the fonts in use. However it lacks the detailed font recognition and editing facilities of the full Capture product. When the plugin is invoked it OCRs the bitmapped pages to hidden typeset text, using generic fonts that are metrically similar to the scanned bitmap glyphs. Accurate page coordinates are calculated for each recognised word and the hidden text is then placed in exact registration with the original bitmapped text (technically, PDF talks about rendering this hidden material in Text Mode 3).

Once the OCR is complete the PDF document becomes searchable, limited only by the quality of the OCR recognition; the search function highlights the place on the screen where the search phrase occurs in the hidden text. Provided that registration is accurate, the perceived effect of the highlighting is to illuminate the corresponding area in the visible bitmapped text. In this way Capture seemingly achieves the near-miracle of 'searchable bitmapped text'.

For the present project the use of Acrobat Capture would be wholly inappropriate. The full Capture product with its in-built suspects editor could be equipped with the correct fonts but there is no way that it could be trained to exploit the repeated regular structure of the marriage register pages. By contrast the *tbl* pre-processor for UNIX *troff*, or the tabular environment in TeX and LaTeX, are well suited to this kind of material. Most important of all the *troff* and *tbl* coding is auto-generated from *sed* and *awk* scripts working on OCR-acquired raw text. In this way the various replacement rules for 'suspects' can be made explicit rather than residing in hidden data structures inside an integrated document recognition application such as Capture.

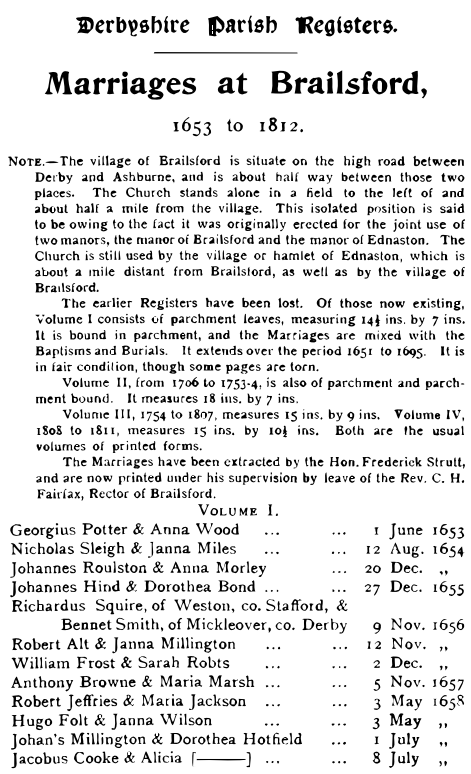
2. TOOLS AND TECHNIQUES

2.1. Typeface choices

If a book is to be completely reset then achieving a high-quality result depends on being able to find appropriate fonts for *all* of the pages in the original. Any material that can be replicated only by insertions of scanned-in bitmap will greatly degrade the overall appearance of the product.

Analysis of an original Phillimore Register page, as shown in Figure 1(a), shows three typefaces in use. The body text was identified (from the structure of the & and A characters, and from other clues) as being Caslon, set at 10 pt. on 11.5 pt. line spacing. Now, given that the matrices for typecasting Caslon on Monotype machines were not cut until 1914 it seems likely that the early Phillimore volumes (Derbyshire Marriages Volume 2 is dated 1907), were hand set. The body typeface selected for the re-setting exercise was Caslon 540 from the Adobe Type Library — this particular variant of Caslon being chosen because the Old Style figures and Small Capitals, used in the original material, were all available.

The Fraktur typeface used in the 'Derbyshire Marriage Registers' heading was submitted to "What the Font" [4] for identification. A strong match came back for Fordor Incised NF, which was revived in digital form by Nick Curtis in 2005. Finally, the bold face used for the head-



Derbyshire Parish Registers.

Marriages at Brailsford,

1653 to 1812.

NOTE.—The village of Brailsford is situate on the high road between Derby and Ashburne, and is about half way between those two places. The Church stands alone in a field to the left of and about half a mile from the village. This isolated position is said to be owing to the fact it was originally erected for the joint use of two manors, the manor of Brailsford and the manor of Ednaston. The Church is still used by the village or hamlet of Ednaston, which is about a mile distant from Brailsford, as well as by the village of Brailsford.

The earlier Registers have been lost. Of those now existing, Volume I consists of parchment leaves, measuring 14¼ ins. by 7 ins. It is bound in parchment, and the Marriages are mixed with the Baptisms and Burials. It extends over the period 1651 to 1695. It is in fair condition, though some pages are torn.

Volume II, from 1706 to 1753-4, is also of parchment and parchment bound. It measures 18 ins. by 7 ins.

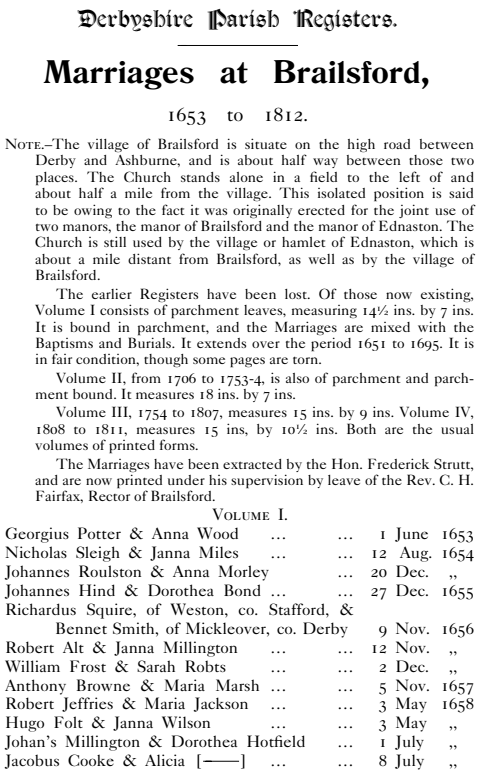
Volume III, 1754 to 1807, measures 15 ins. by 9 ins. Volume IV, 1808 to 1811, measures 15 ins. by 10¼ ins. Both are the usual volumes of printed forms.

The Marriages have been extracted by the Hon. Frederick Strutt, and are now printed under his supervision by leave of the Rev. C. H. Fairfax, Rector of Brailsford.

VOLUME I.

Georgius Potter & Anna Wood	1 June	1653
Nicholas Sleigh & Janna Miles	12 Aug.	1654
Johannes Roulston & Anna Morley ...	20 Dec.	,,
Johannes Hind & Dorothea Bond	27 Dec.	1655
Richardus Squire, of Weston, co. Stafford, & Bennet Smith, of Mickleover, co. Derby	9 Nov.	1656
Robert Alt & Janna Millington	12 Nov.	,,
William Frost & Sarah Robts	2 Dec.	,,
Anthony Browne & Maria Marsh	5 Nov.	1657
Robert Jeffries & Maria Jackson	3 May	1658
Hugo Folt & Janna Wilson	3 May	,,
Johan's Millington & Dorothea Hotfield ...	1 July	,,
Jacobus Cooke & Alicia [——]	8 July	,,

Figure 1(a), Original title page



Derbyshire Parish Registers.

Marriages at Brailsford,

1653 to 1812.

NOTE.–The village of Brailsford is situate on the high road between Derby and Ashburne, and is about half way between those two places. The Church stands alone in a field to the left of and about half a mile from the village. This isolated position is said to be owing to the fact it was originally erected for the joint use of two manors, the manor of Brailsford and the manor of Ednaston. The Church is still used by the village or hamlet of Ednaston, which is about a mile distant from Brailsford, as well as by the village of Brailsford.

The earlier Registers have been lost. Of those now existing, Volume I consists of parchment leaves, measuring 14½ ins. by 7 ins. It is bound in parchment, and the Marriages are mixed with the Baptisms and Burials. It extends over the period 1651 to 1695. It is in fair condition, though some pages are torn.

Volume II, from 1706 to 1753-4, is also of parchment and parchment bound. It measures 18 ins. by 7 ins.

Volume III, 1754 to 1807, measures 15 ins. by 9 ins. Volume IV, 1808 to 1811, measures 15 ins, by 10½ ins. Both are the usual volumes of printed forms.

The Marriages have been extracted by the Hon. Frederick Strutt, and are now printed under his supervision by leave of the Rev. C. H. Fairfax, Rector of Brailsford.

VOLUME I.

Georgius Potter & Anna Wood	1 June	1653
Nicholas Sleigh & Janna Miles	12 Aug.	1654
Johannes Roulston & Anna Morley ...	20 Dec.	,,
Johannes Hind & Dorothea Bond	27 Dec.	1655
Richardus Squire, of Weston, co. Stafford, & Bennet Smith, of Mickleover, co. Derby	9 Nov.	1656
Robert Alt & Janna Millington	12 Nov.	,,
William Frost & Sarah Robts	2 Dec.	,,
Anthony Browne & Maria Marsh	5 Nov.	1657
Robert Jeffries & Maria Jackson	3 May	1658
Hugo Folt & Janna Wilson	3 May	,,
Johan's Millington & Dorothea Hotfield ...	1 July	,,
Jacobus Cooke & Alicia [——]	8 July	,,

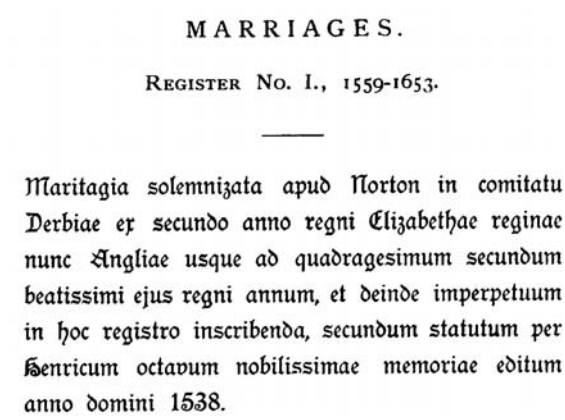
Figure 1(b) Re-typeset title page

ing 'Marriages at Brailsford' did not exactly match any typeface known to "What the Font", but it showed similarities to ITC Bookman Bold and Bitstream's Romana Bold. The Bookman face was a slightly better match for weight but Romana was ultimately chosen because of the closer similarities of its letter shapes to those in the original material.

Figure 1(b) shows how well the substitutes performed in practice. Fortunately, Caslon 540 has a smaller x-height, and slightly smaller set widths, than the original Caslon font of Figure 1(a). So, for the 9 pt. descriptive material about the Brailsford parish, it was possible to stretch the lines to fit the measure using the *troff* `\p` command. Equally, as described in section 2.3, the *troff* 'fields' facility enabled each line in the tabular material to be padded and justified.

Having identified the three main typefaces needed for the Phillimore Marriage Register series, it was with a slight sense of dismay that yet another font challenge was discerned in Volume 5 of the Derbyshire Marriages. In this volume the registers for the parish of Norton go back to 1559 and at the start of the very first register is an inscription, in Latin and in a calligraphic hand, recording that the marriage entries were commencing in the second year of the reign of Queen Elizabeth and would be recorded, in perpetuity, as laid down in the edict of Henry VIII in 1538. The Phillimore editors and printers took up the challenge of replicating this opening page as best they could and a scanned version of what they produced appears in Figure 2(a). A cursory glance shows that the typeface is some sort of cursive blackletter, much like the early types used for English printing by Caxton in the late 15th century. Now, the style of handwriting on which Caxton's types were based is called Bâtarde, and more particularly, Burgundian Bâtarde, owing to its popularity among the scribes of the Duchy of Burgundy. The Bâtarde terminology denotes that it is cursive, or semi-cursive, and hence bastardized, in comparison to the upright, fully hexagonalized, blackletter typeface found in the Gutenberg Bible.

Blackletter digital typefaces are certainly available that mimic Gutenburg's originals, or which follow the Fraktur style. But faces that emulate Caxton's blackletter are very few and far between and are unlikely to exactly replicate the face used in this particular Phillimore register of 1909. (The typeface was undoubtedly obtained from one of the English type foundries of the day.) Fortunately a modern re-interpretation of the Bâtarde style is available in the form of Lucida Blackletter from Bigelow and Holmes which has rather more ornate capital letters than the original material but otherwise bears a remarkable resemblance to it. With permission from Bigelow and Holmes the stems of the lower-case letters `b`, `l`, `h` and `d` were altered, in Fontographer, to give a better match to the original face. The results are shown in Figure 2(b).



MARRIAGES.

REGISTER No. I., 1559-1653.

Maritagia solemnizata apud Norton in comitatu Derbiae ex secundo anno regni Elizabethae reginae nunc Angliae usque ad quadragesimum secundum beatissimi ejus regni annum, et deinde imperpetuum in hoc registro inscribenda, secundum statutum per Henricum octavum nobilissimae memoriae editum anno domini 1538.

Figure 2(a), Original Norton parish title page

MARRIAGES.

REGISTER No. I., 1559–1653.

Maritagia solemnizata apud Norton in comitatu Derbiae ex secundo anno regni Elizabethae reginae nunc Angliae usque ad quadragesimum secundum beatissimi ejus regni annum, et deinde imperpetuum in hoc registro inscribenda, secundum statutum per Henricum octavum nobilissimae memoriae editum anno domini 1538.

Figure 2(b) Re-typeset page using Lucida Blackletter

2.2. Scanning and OCR

A series of 300 dpi, bitonal, TIFF test scans, from the chosen Phillimore volume, were made on an HP Scanjet 8250 flatbed scanner. Two possible OCR programs were tested. Firstly the TIFF pages were imported into Adobe Acrobat, converted to PDF bitmap format, and then subjected to OCR analysis via the Capture plugin.

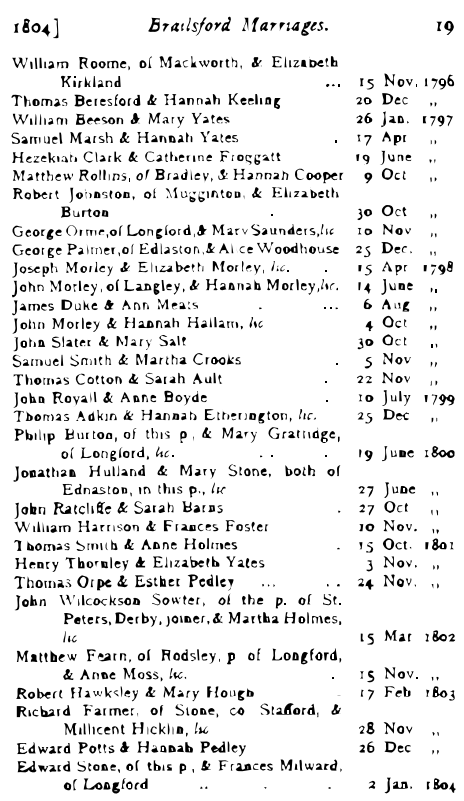
The quality of the resultant 'hidden text' could be analysed by saving it from Acrobat as plain text. For comparison, the TIFF pages were also OCR-analysed using the Read Iris Pro OCR software supplied with the 8250 scanner. The recognition performance of the two OCR engines was broadly similar but the Capture software had to be rejected because of the simple fact that there was no way to disable its page zoning and structural analysis features. About 50% of the time a typical page would be represented as a single text block. Another 40% of the time it would recognise the tables as two-column text, with main material in the left column and the final three tabular columns rolled up into a single textual column. For the remaining 10% of the scanned pages a wide variety of bizarre page structures were inferred, which led in turn to some equally bizarre reading orders within the saved text. Whether rendering order is the same as reading order is usually of little concern in PostScript and PDF, so long as the final displayed page "looks right". But if saved plain text from an OCR process is being analysed and re-processed then correct reading order is essential.

For all the above reasons the serious OCR work for this project was conducted using ReadIris Pro, in which it was possible to drag out a single-block template and apply it to all scanned pages, prior to OCR processing into plain text. Like many programs of this sort, ReadIris offers an interactive option to signal suspect words or phrases that lie outside the confidence limits for recognition. After correcting the suspect one can either ask that this correction be 'learned', in which case it will be applied silently thereafter, or that it be 'not learned' which means that the same correction will be proposed if the suspect recurs, but it will require confirmation on every occasion. In practice this latter style of correction is the option that has to be chosen most of the time; the former sort requires complete confidence that it is always safe, in all contexts, to make the indicated correction. Even if such universal replacement rules are valid they become part of the knowledge-based data structures hidden inside the recognition software. Just as in Adobe Capture there is no way to externalise these rules, nor to have close control over the degree to which they are context free or context sensitive.

Thus, after some experimentation it was decided to dispense with interactive suspect editing during OCR and to simply scan the pages to plain text, followed by processing with the *sed* and *awk* scripts described in a later section. The division of labour between *sed* and *awk* allows exactly the level of context-free/context-sensitive control that has just been referred to.

2.3. Core typesetting software

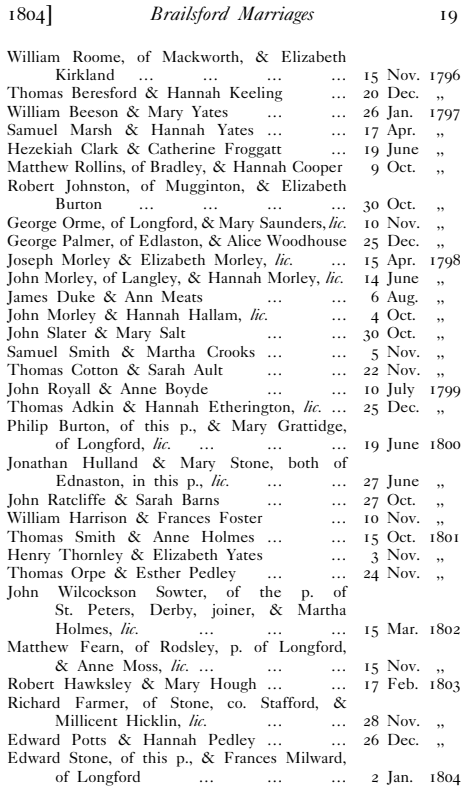
The choice of UNIX *ditroff* [5] (the device-independent version of *troff*), for the re-setting work, rather than the equally suitable TEX or LATEX, was largely for historical reasons. In the early 1980s I led a project to perform, 'in house', the typesetting of all examination papers for the University of Nottingham [6]. At the time there was much debate as to whether *troff* or TEX should be chosen for this task. TEX was available for free but it needed a DEC VAX (which we did not possess) in order to run successfully. On the other hand the *ditroff* suite cost $4000 (a substantial sum at the time) but the software could run quite happily on the PDP 11/44 and PDP 11/70 machines we already possessed. Thus, our decision in favour of the *troff* suite was confirmed by simple economic considerations.



1804] *Brailsford Marriages.* 19

William Roome, of Mackworth, & Elizabeth Kirkland ... 15 Nov. 1796
Thomas Beresford & Hannah Keeling 20 Dec „
William Beeson & Mary Yates 26 Jan. 1797
Samuel Marsh & Hannah Yates . 17 Apr „
Hezekiah Clark & Catherine Froggatt 19 June „
Matthew Rollins, of Bradley, & Hannah Cooper 9 Oct „
Robert Johnston, of Mugginton, & Elizabeth Burton . 30 Oct „
George Orme, of Longford, & Mary Saunders, *lic* 10 Nov „
George Palmer, of Edlaston, & Alice Woodhouse 25 Dec. „
Joseph Morley & Elizabeth Morley, *lic.* . 15 Apr 1798
John Morley, of Langley, & Hannah Morley, *lic.* 14 June „
James Duke & Ann Meats . ... 6 Aug „
John Morley & Hannah Hallam, *lic* 4 Oct „
John Slater & Mary Salt 30 Oct „
Samuel Smith & Martha Crooks . 5 Nov „
Thomas Cotton & Sarah Ault . 22 Nov „
John Royall & Anne Boyde . 10 July 1799
Thomas Adkin & Hannah Etherington, *lic.* 25 Dec „
Philip Burton, of this p., & Mary Grattidge, of Longford, *lic.* . . . 19 June 1800
Jonathan Hulland & Mary Stone, both of Ednaston, in this p., *lic* 27 June „
John Ratcliffe & Sarah Barns . 27 Oct „
William Harrison & Frances Foster 10 Nov. „
Thomas Smith & Anne Holmes . 15 Oct. 1801
Henry Thornley & Elizabeth Yates 3 Nov. „
Thomas Orpe & Esther Pedley ... . . 24 Nov. „
John Wilcockson Sowter, of the p. of St. Peters, Derby, joiner, & Martha Holmes, *lic* 15 Mar 1802
Matthew Fearn, of Rodsley, p. of Longford, & Anne Moss, *lic.* . 15 Nov. „
Robert Hawksley & Mary Hough . 17 Feb 1803
Richard Farmer, of Stone, co. Stafford, & Millicent Hicklin, *lic* 28 Nov „
Edward Potts & Hannah Pedley 26 Dec „
Edward Stone, of this p., & Frances Milward, of Longford .. . . 2 Jan. 1804

Figure 3(a), Original sample page



1804] *Brailsford Marriages* 19

William Roome, of Mackworth, & Elizabeth Kirkland ... ... 15 Nov. 1796
Thomas Beresford & Hannah Keeling ... 20 Dec. „
William Beeson & Mary Yates ... ... 26 Jan. 1797
Samuel Marsh & Hannah Yates ... ... 17 Apr. „
Hezekiah Clark & Catherine Froggatt ... 19 June „
Matthew Rollins, of Bradley, & Hannah Cooper 9 Oct. „
Robert Johnston, of Mugginton, & Elizabeth Burton ... ... 30 Oct. „
George Orme, of Longford, & Mary Saunders, *lic.* 10 Nov. „
George Palmer, of Edlaston, & Alice Woodhouse 25 Dec. „
Joseph Morley & Elizabeth Morley, *lic.* ... 15 Apr. 1798
John Morley, of Langley, & Hannah Morley, *lic.* 14 June „
James Duke & Ann Meats ... ... 6 Aug. „
John Morley & Hannah Hallam, *lic.* ... 4 Oct. „
John Slater & Mary Salt ... ... 30 Oct. „
Samuel Smith & Martha Crooks ... ... 5 Nov. „
Thomas Cotton & Sarah Ault ... ... 22 Nov. „
John Royall & Anne Boyde ... ... 10 July 1799
Thomas Adkin & Hannah Etherington, *lic.* ... 25 Dec. „
Philip Burton, of this p., & Mary Grattidge, of Longford, *lic.* ... ... 19 June 1800
Jonathan Hulland & Mary Stone, both of Ednaston, in this p., *lic.* ... ... 27 June „
John Ratcliffe & Sarah Barns ... ... 27 Oct. „
William Harrison & Frances Foster ... 10 Nov. „
Thomas Smith & Anne Holmes ... ... 15 Oct. 1801
Henry Thornley & Elizabeth Yates ... 3 Nov. „
Thomas Orpe & Esther Pedley ... ... 24 Nov. „
John Wilcockson Sowter, of the p. of St. Peters, Derby, joiner, & Martha Holmes, *lic.* ... ... 15 Mar. 1802
Matthew Fearn, of Rodsley, p. of Longford, & Anne Moss, *lic.* ... ... 15 Nov. „
Robert Hawksley & Mary Hough ... ... 17 Feb. 1803
Richard Farmer, of Stone, co. Stafford, & Millicent Hicklin, *lic.* ... ... 28 Nov. „
Edward Potts & Hannah Pedley ... ... 26 Dec. „
Edward Stone, of this p., & Frances Milward, of Longford ... ... 2 Jan. 1804

Figure 3(b) Re-typeset sample page

In the subsequent 25 years a large amount of expertise has been built up with the *ditroff* suite generally, and hence also with the *tbl* pre-processor, so essential to the present project. Expertise and experience with one's chosen software is every bit as important as the software itself in a project of this kind, in which a large number of subtle typographical effects are required if the original material is to be replicated accurately.

The *tbl* template for the register entries was:

```
lf1w(2.8i) 2 nf3 1 lf1 1 lf3.
```

in which `f1` denotes use of the standard Caslon 540 font and `f3` the variant of Caslon 540 containing Old Style figures and small capitals. The 2.8 inch width of the first column can clearly be seen, and the letters `l` or `n`, attached to each column specifier, denote either left or 'numeric' justification, respectively. The interleaved digits such as `2` and `1` show the widths of the inter-column gutters, measured in ens.

Figures 3(a) and 3(b) show, respectively, an original page from the Brailsford register and its re-set version. Even though these figures are reduced in size, close inspection reveals that the measure on the re-set version is slightly wider than on the original (3.9 inches as opposed to 3.7 inches). The reason for this is that even though Caslon 540 sets more tightly than the original Caslon, huge problems arose in automating the resetting of the tables due to the occasional line that had virtually no inter-word spacing (the entries for George Orme, George Palmer and John Morley of Langley, in Figure 3(a), show this very clearly). In relaxing the measure by 0.2 inches most padding problems could be overcome, as recounted in the next sub-section.

A sample of the *tbl* encoding for the first few lines of Figure 3(b) is shown in Figure 4. The first feature of note is the use of `.fc # ^`, which is a *troff* 'field' command [7] denoting that, between the outer delimiters of `#`, the inner `^` markers are to be padded with equal amounts of space, so as to fit the declared column width of 2.8 inches. Now it was a considerable leap of faith to hope that this approach would work because fields are a *troff* facility and not a built-in feature of *tbl*. Given that *tbl* acts as a pre-processor for *troff*, and compiles *tbl* codings into lengthy streams of raw *troff* commands, it was by no means certain that the use of fields, within a table, would not interfere, in some subtle way, with the low-level coding generated by *tbl* itself. Indeed, this trick is the typesetting equivalent of trying to interleave raw assembler code with the output from a C compiler, and is almost certainly every bit as dangerous. Mercifully it worked well because, as we shall see, being able to automate spatial padding is an invaluable bonus in developing scripts for generating correct leader-dot patterns.

Further features to note are the way in which overflow lines are inset by using the `\*(6s` pre-defined string to insert a 6-en space and the use of `\*c` to call a pre-defined string for the abbreviation *lic.*, which denotes a

marriage by special licence. The string call `\*(0q` is used to insert ditto marks, consisting of two commas, whenever a year date of a wedding is the same as that on a previous line.

However, most significant of all for replicating the original material was the need for leader-dot patterns to be inserted into column 1 of the table, to complete each under-length line. Careful measurement on the original pages showed that the simplest of these leader patterns was just an ellipsis character (…) and so this has been pre-defined at the head of the marriage register *troff* file as the string `p1`, which is in turn, invoked by the call-out of `\*(p1`. The pattern `p2` is just two ellipsis characters separated by a 6-en space; more generally the leader pattern `p`*n* consists of *n* ellipsis characters with *n* – 1 sets of interleaved 6-en spaces. The `p4` pattern is the most complex one that is needed.

A more detailed discussion of how these leader patterns are calculated appears in the next sub-section.

2.4. Transformation tools

The plain-text output from the OCR process has to be processed in three stages as follows:

1. correction of frequently recurring OCR mis-reads
2. final, context-sensitive, OCR corrections plus imposition of the basic *tbl* framework
3. enhancement of the *tbl* code to insert leader-dot patterns and to induce field-based spatial padding of the material in the first column of each table.

Software tools such as Perl and Python can potentially integrate a variety of text transformation tasks at varying levels of abstraction. The adoption, in this case, of the classic UNIX tools, *sed* and *awk*, was not for reasons of nostalgia but was prompted, instead, by memories of how well a pipeline of such processes enforces a clear separation of tasks, and enables simple inter-task debugging to take place.

More specifically, the first of the above tasks is performed by a *sed* script in which the rules correct the great majority of the frequently recurring mis-reads in the OCR process. The rule-base was progressively expanded as more knowledge was gained from many dozens of page scans. It has now stabilised at about 300 simple rules. The original printed registers use Old Style figures, which have varying heights and some of which descend below the baseline. These factors cause frequent mis-readings by the OCR software resulting in the need for substitution rules such as `s/r6og/1609/`.

To keep things simple, any context sensitivity in the *sed* rule-base is limited to specifying that certain substitutions should occur at line beginnings (`^`) and line endings (`$`) only (the more complex substitutions being saved for later *awk* scripts). Examples of this kind of rule might include `s/"$/,,/` to correct a common OCR mis-reading of the two-comma ditto marks at many line ends. Another good example is the rule `s/^\]ohn/John/`, which essentially says that: " ... a closing square bracket at the start of a line is a mis-read capital J provided it is followed by `ohn`". Small-scale features, such as the ellipsis dots used in the leader patterns, are frequently mis-read, but this 'noise' has to be removed prior to a full recalculation of the correct leader pattern. No fewer than 40 *sed* rules have been accumulated to cope with this cleanup.

The second task in the list given above is performed by an *awk* script, imaginatively named `ascript`. One of its main tasks is to separate those lines that end with a marriage-date structure from those that do not. The dated lines are highly significant because they denote the end of a complete marriage record. Such lines can be filtered out very quickly given that the Phillimore Registers use the month names of `May`, `June`, `July` in full, and all other months have a four-character standard abbreviation such as `Jan.`. A few extra rules are needed to ensure that date-free lines containing surnames such as `May` or `Mayfield` do not get misinterpreted. This particular *awk* script then emits the first four table-header lines shown in Figure 3 and inserts the inter-column tab character `@` between the day, month and year components of dated lines.

Task 3 in the above list is tackled by the most complex *awk* script of all, called `dotscript`. The first step is to calculate the line length, for each of the marriage records, of the material that resides in the first column of the table i.e. the names of groom and bride and ancillary details of occupation, parish etc. To do this requires `dotscript` to have declarations for the *troff* character width (at 10 pt.) of every character in the Caslon 540 Roman font. At this stage inter-word spaces are the standard 0.5 en width but their positions in the string are detected using *awk*'s `index` command [8] and are stored as possible padding points.

The next step is the complicated task of padding out the first column with the appropriate leader-dot pattern and extra spacing. The four cases that can arise, for material in column 1 of the table, are as follows;

1. The material is so wide that it is over-length for the column if standard word-spaces are used. In this special case padding has to be attempted using *troff*'s 1/6 em or 1/12 em thin spaces (`\|` or `\^`).
2. The material is just wide enough to fit elegantly into the column, with *troff*'s field-based `^` padding. There is no need for any leader pattern.
3. The material fits the line with leader pattern of `p`*n* (where *n* ranges from 1 to 4) and with en-width (`\0`) inter-word spaces.
4. The material is too wide if rule 3 is used but it does still fit the column, with leader pattern of `p`*n*, if field-based padding is used at points marked `^`.

At first glance it might appear that rule 4 could always be used in place of rule 3 but if this is done some wide,

```
.TS
center,tab(@);
lf1w(2.8i) 2 nf3 1 lf1 1 lf3.
.fc # ^
#William^Roome,^of^Mackworth,^&^Elizabeth#@@@
#\*(6sKirkland^\*(p4#@15@Nov.@1796
#Thomas\0Beresford\0&\0Hannah\0Keeling^\*(p1#@20@Dec.@\*(0q
#William\0Beeson\0&\0Mary\0Yates^\*(p2#@26@Jan.@1797
#Samuel^Marsh^&^Hannah^Yates^\*(p2#@17@Apr.@\*(0q
#Hezekiah\0Clark\0&\0Catherine\0Froggatt^\*(p1#@19@June@\*(0q
Matthew\|\|Rollins,\|\|of\|\|Bradley,\|\|\^&\^\|\|Hannah\|\|Cooper@\09@Oct.@\*(0q
#Robert^Johnston,^of^Mugginton,^&^Elizabeth#@@@
#\*(6sBurton^\*(p4#@30@Oct.@\*(0q
George\|\|Orme,\|\|of\|\|Longford,\|\^&\|\^Mary\|\|Saunders,\|\*c@10@Nov.@\*(0q
George\|\|Palmer,\|\|of\|\|Edlaston,\|\|&\|\|Alice\|\|Woodhouse@25@Dec.@\*(0q
#Joseph\0Morley\0&\0Elizabeth\0Morley,\0\*c^\*(p1#@15@Apr.@1798
John\|\|Morley,\|\|of\|\|Langley,\^\|\|&\^\|\|Hannah\|\|Morley,\|\|\*c@14@June@\*(0q
```

Figure 4. Sample *tbl* coding for the marriage register page shown in Figure 3(b)

and ugly, inter-word spacing can result. Rules 3 and 4 seem to be a good approximation to the algorithm used by the compositors in hand-setting the original material.

Figure 4 illustrates all of the above four rules in action. Rule 1 is used in the line for `Matthew Hollins`; rule 2 is used in the line for `William Roome`; rule 3 is used for the `William Beeson` line (together with the `p2` leader pattern); rule 4 is used in the line for `Samuel Marsh`.

Although the table structure in the Marriage Registers is, for the most part, very regular there are occasional editorial comments inserted. These are on separate lines and are invariably enclosed in square brackets, with the text being set in italics. An example might be:

[*The above three entries are almost illegible*].

The start character of `[` enables these lines to be detected, and the padding routines of `dotscript` can then be skipped, but detailed positioning and correction has to be done by hand.

2.5. Proof-reading

Despite all of the automated table setting performed by the *sed* and *awk* scripts it is still essential to proof-read each page carefully. The OCR software is often confused by ink-bleed on the original material and yet the resultant mis-scan can still be perfectly plausible. For example the surname `Marton` might be mis-recognised as `Morton`, but since both are valid surnames one cannot devise an error-proof substitution rule. Only careful proof-reading can correct errors of this sort.

Another example is afforded by the already-described appearance of dates such as '165$\frac{8}{9}$' in the original material. Dates having this structure were so frequently mis-recognised, during OCR, as `165:` or `165;` that `dotscript` was designed to keep track of the current year from the most recently parsed previous date (1658 in the present example) and then to confect '165$\frac{8}{9}$' automatically, via the *eqn* coding of `165$8above9$`. However, very occasionally, a date such as '1651' would also be mis-recognised as `165:` and would then be erroneously typeset as '165$\frac{8}{9}$'. Here again, careful proof-reading is the only answer.

3. Parsing the Register records

A noteworthy feature of the entries in all the Phillimore registers is the grammatical regularity that is displayed by well over 90% of the entries. It is clear that firm rules were laid down by the Phillimore editors concerning the allowed sub-clauses of extra information that might be associated with the groom and the bride, or even with the wedding as a whole. Although these Registers were typeset almost 60 years before Chomsky's classification of phrase-structure grammars [9], it turns out that the standard ordering of the possible sub-clauses, which is adhered to assiduously, makes the grammar for a marriage record very close to *regular* (Chomsky Type 3).

Nowadays programming language compilers can be developed with the help of parser generators such as YACC [10] or, within the framework of XML, from a metasyntactic specification expressed as a Document Type Definition (DTD) or an XML Schema. In all these cases the grammars that can be handled extend to a class known as 'deterministic context-free' (Chomsky Type 2), and this class has the regular grammars as a proper subset. However, in all cases, it is the responsibility either of the grammar designer, or of the grammar transformation system within the parser-generator, to try and make the grammar *deterministic*, which amounts to saying that the parser must be able to determine which parsing action to take based on only a very limited amount of look-ahead on the input string. Almost any standard textbook [11] on the parsing of computer languages give proofs that all Type 3 languages can be made deterministic whereas the Type 2 languages (which allow self-embedding recursion, as well as left and right recursion) can only be made deterministic if any self-embedding

recursion has a deterministic 'mid-point'. The deterministic subset of Type 2 grammars corresponds exactly to the class of grammars that are LR(k) parsable i.e. on a left-to-right-scan of the input string, to produce a right parse, it is necessary to look only k input symbols ahead.

Many programming languages succeed in being almost context-free, apart from a few small context-sensitive cases, which are handled via special-case programming in the parser. The grammar for our marriage records is no exception. In essence it is a very simple, non-recursive, regular grammar but with just a few context-sensitive features. A simplified skeleton version of the grammar, in a metasyntax close to that used by YACC, might be:

```
%token DAYNO  /* validated integer from tokenizer
                    in range 1-31 */
%token WORD  /* single word i.e. arbitrary text
                  string delimited by spaces */
wedrec: ginfo '&' binfo suppinfo date
ginfo: gfname* gsname place county occup marstat
binfo: bfname* bsname place county occup marstat
gsname|gfname|bfname|bsname : WORD
place: ',' 'p.' WORD+ | ',' 'of' WORD+ | EMPTY
suppinfo: ',' 'lic.' | ',' 'both of' WORD+ county
                     | EMPTY
county: ',' 'co.' WORD+ | EMPTY
occup: ',' WORD+ | EMPTY
marstat: ',' 'b.' | ',' 'sp.' | ',' 'w,' | EMPTY
/* the above denote bachelor, spinster and
  widow/widower respectively */
date: DAYNO month year
month: 'Jan.' | 'Feb.' | 'Mar.'  …  |'Dec.'
year: '1' '[5-9]' '[0-9]' '[0-9]' | ',,' |
```

Figure 5. Simplified grammar for a marriage record

An example of a typeset entry in accordance with the above grammar might look like:

Richard Garrett, p. Norton, joiner, w. &
Anne Reid Banks, of Mansfield,
co. Nottm., seamstress, sp., *lic.* … 12 Oct. 1812

A parse tree for this sample marriage record, parsed with the above grammar, is shown in Figure 6.

Inspection of Figures 5 and 6 shows the following principal features: each record must terminate with the date of the marriage, which is a simple sub-grammar; the major sub-tree linking groom and bride information is rooted on the `&` character and it is this sub-tree which forms the focus of the next section when indexing is discussed; forenames and surnames for groom and bride are space delimited but when sub-clauses of extra information are given they are comma delimited. Any, or all, of the groom or bride forenames may be absent, but surnames must be present. Furthermore, parsing is greatly helped by the fact that, for both bride and groom, any ancillary information must be given strictly in the following order: place of residence, county of residence, occupation and current marital status. However, any or all of the sub-clauses can be absent (denoted by `EMPTY` in Figure 5) and this fact complicates the look-ahead to determine, for each sub-clause, which of the non-terminals on the right-hand side of the `ginfo`, or `binfo`, grammar rules it should belong to. Fortunately, extra help is at hand in that non-terminals such as `place`, `county`, and `marstat` all start with a particular finite set of fixed tokens such as `p.`, `of`, `co.` and `w.`. Indeed, the only remaining sub-clause lacking such an indicative token is that for occupation (`occup`) but its very token-free uniqueness, and its position in the sub-clause sequence, yield sufficient clues to enable it to be identified.

It is clear from the foregoing analysis that, with some effort, a deterministic parser could be built for these marriage records, either by adapting the grammar of Figure 5 manually so that it is deterministic (an SGML or XML parser approach would generally insist on this) or by using a tool such as YACC to transform the grammar automatically to a deterministic form. In this latter case great care has to be taken to investigate any reported shift-reduce and/or reduce-reduce conflicts to ensure that the parse tree produced will truly be the desired one.

Although it might be possible to develop a fully-deterministic, hand-generated or YACC-generated, parser for these marriage records it remains the case that parsers made in this way tend to be better suited to the long-range syntax of a conventional programming language, where nested structures can stretch over several pages. For the Phillimore material the structure is very short range; all marriage records are independent of one another and none of them spans more than five lines.

Two final factors must now be considered. Even when a grammar can be reduced to LR(1) form, with parser speed in mind, the set of grammar rules is often made much simpler if expressed in a form implicitly requiring a k symbol lookahead—exactly as in Fig. 5. Secondly, and related to this first point, is that in real life the LR(k) ideality of a grammar is often spoiled by just a few context-sensitive issues. Just one example will show that the grammar of Figure 5 is no exception to this. Within the `suppinfo` grammar rule there is a 'both of' alternative which allows a clause to be added to the effect that groom and bride, prior to the wedding, were both resident at the same place, or in the same parish. Of course, this 'both of' clause, within `suppinfo`, is illegal if either the groom, or the bride, has already had a `place` entry in their own part of the marriage record.

The above analysis leads us to the view that a parser based on string processing of a complete marriage record (with the implicit look-ahead that this implies) will offer much greater flexibility and adaptability even if it does suffer just a little in terms of speed. In the next section we describe an *awk*-based parser for making the surname index for a marriage register. However, the very fact that the records are fully parsable along the lines suggested in

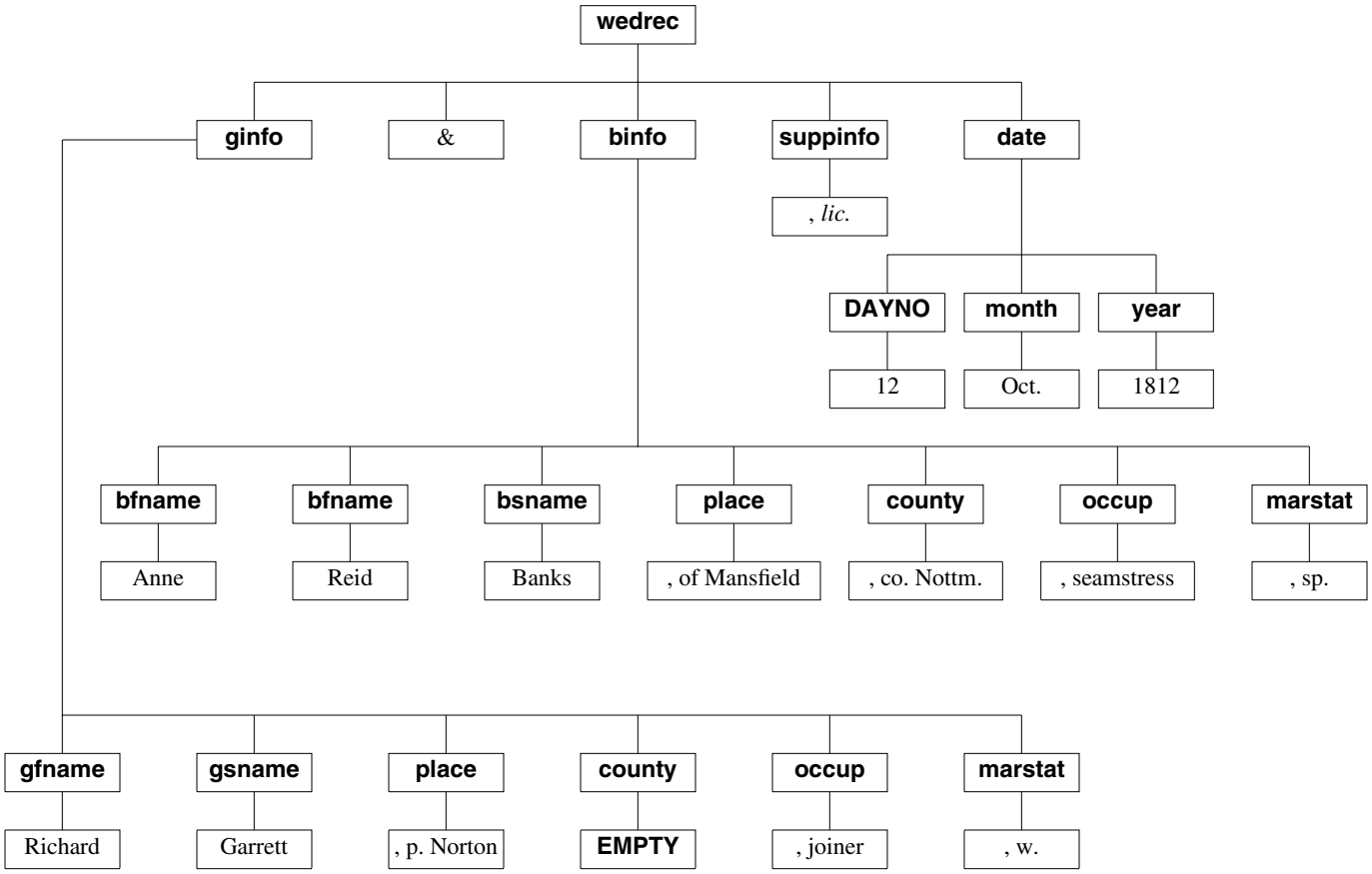


Figure 6. Parse tree for a typical marriage register record

Figs. 5 and 6 shows that it would be perfectly feasible to reprocess the records into GEDCOM or XML-based notations as discuseed in Section 6.

4. Making an index

The Preface to almost all of the original Phillimore volumes includes a statement to the effect that indexing would begin once all the parish marriage registers in the country had been transcribed. In the event, these indexes never materialised. However, the discussion in the previous section indicates that scripting languages such as *awk* would be ideally suited to the creation of a parser for indexing.

In addition to its other capabilities *awk* has already been much used for making indexes [12, 8], and the second of these references makes the point that indexing commands can be silently embedded within the typesetting code itself. However, the grammatical regularity of Phillimore records means that the job of creating an index keyed on the surnames of bridegrooms and brides will be much simpler than a general indexing task. Furthermore, Figure 4 shows that the coding for the tables is already detailed and cluttered enough, which led to a general reluctance to embed extra indexing code.

Instead, the approach taken was to write yet another *awk* script (`indexclean`) that processes the *troff* coding for a given register and ignores everything except for the groom and bride part of the marriage record, rooted around the `&` character. In other words, only the text in column 1 of the table needs to be parsed (this is where the surnames must reside). Routine processing such as stripping out the embedded formatting characters (e.g. # and ^) is very straightforward in *awk*, as also is the joining of multi-line records into a single line. Once each record has been reduced to a clean, space-separated, format the parsing rules for picking out surnames can be applied.

Parsing begins by removing any sub-strings corresponding to personal titles such as `Mr.`, `Mrs.` and `Rev.`. Next, the record is split into two parts around the `&` character, which links the groom's information to the bride's information. For each of these two parts any comma-separated sub-clauses, detailing occupation, home parish etc., are also removed via *awk*'s `split` and `sub` commands. What remains is a simpler record, consisting of the groom's full name, an `&` character, and the bride's full name. The word at the right-hand end of this simplified record is the bride's surname; the word immediately preceding the `&` is the groom's surname. The

`indexclean` script puts out each name in the form of a surname, followed by a comma and two spaces, followed by a first name and any middle names, then two more spaces followed by the page number where that marriage is recorded. Updating of page numbers is made easy for `indexclean` by the fact that the *troff* coding for the to update the pagenumber register uses the explicit `.bp`, break page, command between pages.

Once `indexclean` has done its job, the output it produces is sorted on a primary key of surname, a secondary key of first name and a tertiary key of page number. Another simple *awk* script inserts typesetting and hyperlinking codes (see next sub-section) into the sorted list.

Quite apart from the advantages of having a proper index to a Phillimore volume, the very act of processing the *tbl* records for the index proved to be an excellent way of detecting further errors and inconsistencies in the re-set material. The `indexclean` script is a parser for the records and the appearance of rejected records signalled a variety of problems. Perhaps the most intriguing of these was that no fewer than 6 records, out of the 750 or so in the Brailsford register, failed to parse simply because they included no `&` character — the word `and` had been typeset instead. It may say something deeply significant about the human cognitive system that all these instances were missed in my careful proofreading largely, I suspect, because the error was made by the original compositors and did not show up as a difference between original and reset material.

4.1. Hyperlinking the Index

A draft PDF version of the re-set Brailsford marriage register is available for download [13]. It contains the re-set first few pages of preliminary material, including Phillimore's signature and the logo of his company (these were inserted using the *psfig* pre-processor [14] for *troff*), followed by 20 pages of marriage records, followed by the surname index. On the second page of this index it can be seen, at once, that precisely two people surnamed Brailsford actually got married at Brailsford church over the 150-year period covered by the register. If one then moves the Acrobat cursor over these, or any other, index entries it will be seen that each entry is a hyperlink. Clicking on the link will jump the user back to the page where that name occurs.

The automated hyperlinking technique was able to draw on experience gained from the CAJUN project [15], which was one of the very earliest projects in automated PDF hyperlinking for electronic journals. Over the years there has been a notable increase in the number of possible PDF `pdfmark` [16] commands, which can add many forms of annotation 'hyperstructure' into PDF files. This is done by placing the appropriate `pdfmark` commands into a PostScript output stream, prior to distilling it to PDF. Most of the extra `pdfmark` sophistication proved unnecessary because the surname index entries have the great virtue of being backwards-going links and so the page to jump to is already known. Thus, the very simplest form of `/LNK pdfmark` could be used, having the general syntax:

```
[ /Rect [ button coords. ] /Border [0 0 0]
/Page pageno /View [ dest. view ] /LNK pdfmark
```

in which the `[0 0 0]` after `/Border` indicates that link buttons are not to have a visible border.

Now, arbitrary PostScript strings can be inserted into the *ditroff* output stream using the `\X` command [14]. A *troff* macro, named `LK`, was defined in order to encapsulate all the `\X` commands that were needed to build up a `/LNK pdfmark` of the type outlined above. The macro has just a single argument, corresponding to the destination page number. By placing a call of `.LK` between each sorted index entry the macro can establish the PostScript `currentpoint` and then adjust it by the known line spacing and type size so as to work out the `Rect` co-ordinates appropriate for an index entry on the subsequent line.

5. Problems encountered

The accumulation of circumstantial evidence, for the first *sed* script, about OCR mis-scans was tedious but posed no great intellectual difficulties. Equally, emitting the basic *tbl* coding from `ascript` was straightforward. The major difficulty encountered in developing and testing the automated re-setting process was the emergence of inconsistent table widths on successive pages of the output. It is an unfortunate fact of *tbl*'s code generation for *troff* that a requested column width (such as `w(2.8i)`, seen at the top of Figure 4) is treated as a *minimum*, which will be silently expanded if the material in the first tabular column goes over-length.

Thus the major work of this project, so far, has been in refining `dotscript` to make the character widths as accurate as possible so that, in turn, the calculated padding and leader patterns are exactly correct. Many frustrating hours were spent in detecting the (now obvious) fact that *troff*'s `\p` and 'field' facilities cannot insert packing spaces any smaller than the width of the font's word-space character. Therefore, for very tightly set lines, the field mechanism has to be dropped in favour of the insertion of multiple thin spaces (see Figure 4).

The 2.8 inch width of the first table column amounts to some 4145 internal *troff* units and the summation of character widths, for the groom and bride details, is compared to this upper limit. But the whole decision of switching from one set of padding and leader patterns to another is quite extraordinarily sensitive to a variation of just a few units in this total string width. As the fine-tuning of these adjustments continued it was difficult not to envy the original compositors, who could achieve the desired effect with just a few thin pieces of lead.

Surprisingly, the creation of a surname index, which would have required hours of index-card sorting and careful typesetting if it had been done in 1907, was simplicity itself. The `indexclean` script and the `LK` macro, which creates the hyperlinked index from the `index.trf` file, were created in less than a day.

6. Future work

Work is now complete for all of the Phillimore Volumes 1–4 (Derbyshire), including indexing, and a start has been made on volume 5. Volumes 1 and 2 are now in print, and available, thanks to the print-on-demand facilities of Lightning Source UK. Up-to-date news of printing progress and the availability of Vols. 1 and 2 from Amazon (UK) can be found via reference [13]

Other volumes in the Phillimore Derbyshire series can now be attempted, as time permits, both as an end in themselves and also as an aid to refining still further the *sed* and *awk* transformation scripts. If the author has the fortitude to complete all 15 Phillimore (Derbyshire) volumes then a comprehensive cross-volume surname index could readily be built.

For the longer-term goal of processing Phillimore material into more abstract and interchangeable formats it is interesting to note that the genealogical community has had the hierarchical and structured GEDCOM [1] data format available to it for many years. GEDCOM was developed by the Church of Christ of Latter Day Saints (LDS) and in favourable circumstances it can be used as a data exchange format between the various genealogical software packages that support it. It provides a multitude of tags for defining family and marital relationships and it seems likely that most of the optional subordinate clauses in a Phillimore register entry could be represented via some GEDCOM tag or other.

GEDCOM denotes its hierarchies by tagging its elements with the hierarchical level at which they occur (the tree root is at level zero). and Figure 7 shows an example of a marriage between two individuals, annotated according to the GEDCOM 5.5 standard.

The `HEAD` element contains details of the source (`SOUR`) of the information and also an indication that the character set in use is American Standard Extended Latin (`ANSEL`). (GEDCOM 5.5, the default *de facto* standard in the genealogical community, is still not Unicode compatible). Thereafter, the notation very much centres around individuals and family groupings. Each individual is given a unique ID (e.g. `0 @I01@ INDI`) within the GEDCOM file and a subsidiary tag records the 'event' of a birth (`BIRT`), with second-level tags for the details of place and date. In the example of Figure 7 the record for each individual finishes with a pointer (`1 FAMS @F1@`) to a family grouping where that individual belongs. This grouping is encoded further down the file (`0 @F1@ FAM`) and the immediately following tags are back pointers to the individuals that are participating in this family group, with the roles of husband and wife. The marriage 'event' (`1 MARR`) has subsidiary tags to record date and place of marriage. Notice that the lack of hierarchical containment in GEDCOM requires a final tag of `0 TRLR` to show where the tree structure ends.

```
0 HEAD
1 SOUR FTW
1 DEST PAF
1 DATE 21 Oct 1997
1 CHAR ANSEL
0 @I01@ INDI
1 NAME Michael Howard /KING/
1 SEX M
1 BIRT
2 DATE 11 May 1953
2 PLAC Hannover, Germany
1 FAMS @F1@
0 @I02@ INDI
1 NAME Penelope Mary /PHELAN/
1 SEX F
1 BIRT
2 DATE 24 Sep 1956
2 PLAC Cheltenham, Glos, England
0 @F1@ FAM
1 HUSB @I01@
1 WIFE @I02@
1 MARR
2 DATE 10 Apr 1982
2 PLAC Cheltenham, Glos, England
0 TRLR
```

Figure 7. A GEDCOM 5.5 example

```
<GED>
<HEAD>
<SOUR>FTW</SOUR>
<DEST>PAF</DEST>
<DATE>21 Oct 1997</DATE>
</HEAD>
<INDI ID="I01">
<NAME>Michael Howard <S>KING</S></NAME>
<SEX>M</SEX>
<EVEN EV='BIRT'>
<DATE>11 May 1953</DATE>
<PLAC>Hannover, Germany</PLAC> </EVEN>
<FAMS REF="F1"/>
</INDI>
<INDI ID="I02">
<NAME>Penelope Mary <S>PHELAN</S></NAME>
<SEX>F</SEX>
<EVEN EV='BIRT'>
<DATE>24 Sep 1956</DATE>
<PLAC>Cheltenham, Glos, England</PLAC> </EVEN>
<FAMS REF="F1"/>
</INDI>
<FAM ID="F1">
<HUSB REF="I01"/>
<WIFE REF="I02"/>
<EVEN EV='MARR'>
<DATE>10 Apr 1982</DATE>
<PLAC>Cheltenham, Glos, England</PLAC>
</EVEN>
</FAM>
</GED>
```

Figure 8. The material of Figure 7 in GedML form

In 1999 Michael Kay proposed GedML [17] as a contribution towards moving GEDCOM into XML-compatible notation. Figures 7 and 8 are adapted from this reference and Figure 8 shows the material of Figure 7 re-encoded as GedML. Note that the availability of attributes in XML metasyntax means that births, deaths and marriages can all be classified as attributed types of a single element called `EVENT`. Notice also that GEDCOM's `TRLR` element is not needed in GedML, which can use XML's implicit containment notation, `<GED/>`, to denote the end of the tree structure.

Since 1999 others have developed Kay's proposal, and there is even a very early draft of an XML-based GEDCOM 6.0. However, this standard is not "recommended for implementation" and, at the moment, none of the popular genealogical packages supports it. The reason for lack of progress seems to be that GEDCOM 5.5 is already in difficulty as a result of differing opinions as to what each tag should actually *mean* and what it should imply. Already there are a number of vendor-specific additions to GEDCOM 5.5 which compromise its use as a data-exchange format.

It is clear from section 3 that transforming Phillimore material to GEDCOM 5.5, or GedML should pose few problems but the next section makes clear the advantage of using an explicitly XML-based intermediate notation to bring this about.

7. Conclusions

It is worth emphasizing that there is nothing unique to *sed*, *awk*, *troff* and *tbl* in being suitable for the sort of work in this paper—it could equally well have been done using other low-level software tools such as *Perl* and TEX. Nevertheless, I am not aware of any previous work that has attempted to re-set, and enhance, already published work using a similar set of transformational and typesetting tools to those described here.

The real value of this exercise has been to re-visit tried and tested techniques from computer science and to realise, yet again, that the field of document engineering is an excellent one for illustrating computer science fundamentals in novel and illuminating ways. In terms of applicability and general interest, the availability of a limited print run for each of the volumes shows that is more than just an academic exercise. Indeed, the re-setting of century-old marriage registers is not at all the niche interest it might seem. Within the UK, at least, genealogy is a hugely popular hobby, due in part to the successful BBC television series *Who Do You Think You Are?*, and all forms of published and indexed genealogical material find a ready market.

Although the Phillimore Registers for Derbyshire run to just 15 volumes the total number of available Phillimore marriage volumes is well over two hundred. The consistent standards for transcription and typesetting, across the entire series, seem to show that the techniques developed here would be widely applicable. More generally, there must be a wealth of printed tabular material, from the Victorian era and earlier, that might be amenable to the sort of treatment described in this paper (court records, ships manifests, sales ledgers, catalogues and so on). The twin requirements are that the tabular displays should be of regular size, and repeated on dozens or hundreds of pages to make the task worthwhile. Secondly, the entries in each table should conform to a sufficiently rigid set of rules that some form of grammar can be devised. If this is the case then automated indexing and XML-based tagging of content both become feasible.

The project so far has shown clearly, for regular and structured material of the sort under discussion, that the efforts of one keen programmer, armed with suitable programmer-friendly software tools and typesetting programs, can indeed substitute for the efforts of a small army of workers attempting to re-set the material from scratch with the aid of WYSIWYG software. Crucial to success is the need to take great care with the quality of the page scans so as to minimise both the proofreading burden and the number of corrections that have to be made.

The idea of imposing the discipline of re-setting, and indexing, an already-published marriage register, in order to correct and validate the original material, was far more successful than I had ever imagined. As the processing scripts have matured much of the routine work becomes less onerous but one cannot stress too strongly the continued need for accurate proof-reading.

Another job for the future is to convert all the re-set material into the most general XML-based tag notation that can be devised, firstly so that extra consistency checks can be applied and secondly so that, with the aid of XSLT scripts, conversions can be undertaken, not only to standard genealogical formats such as GEDCOM but also into any other desired format.

Acknowledgements In writing this paper I have drawn heavily on the experience of others in the following areas: *awk*, Adobe Capture, GEDCOM, genealogy, OCR, `pdfmarks`, ReadIris Pro and typography. For help with one or more of the above topics thanks are due to Dennis Auter, Steve Bagley, Chuck Bigelow, Kerrie Brailsford, Matthew Hardy, Duff Johnson, Michael Kay, Brian Kernighan, Dennis Nicholson and Mike Parker.

The idea for this project was sparked by the complete sets of original Phillimore Marriage Registers, for both Derbyshire and Nottinghamshire, held by Bromley House Library in Nottingham. The librarians there have kindly let me borrow a succession of the original volumes as this work has progressed.

Special additional thanks are due to Chuck Bigelow and Kris Holmes for permitting me to alter some of the stems on five lower-case letters of their elegant Lucida Blackletter font, as shown in Figure 2(b), so as to better

match the original material of Figure 2(a). That the revised font has already been nicknamed "Bâtarde" bears eloquent testimony both to the typeface family it sprang from and also to my own well-intended, but essentially amateur, efforts.

References

1. *GEDCOM*. http://en.wikipedia.org/wiki/GEDCOM
2. *Phillimore Parish Registers*. uk-genealogy.org.uk/Registers/index.html
3. S & N Genealogy Supplies, *Phillimore Parish Registers*. http://www.genealogysupplies.com/index.htm
4. Myfonts.com, *WhatTheFont: font identification software*. http://new.myfonts.com/WhatTheFont
5. B. W. Kernighan, ''A Typesetter Independent TROFF,'' Computing Science Technical Report No. 97, Bell Laboratories, Murray Hill, New Jersey 07974, March 1982.
6. D F Brailsford, ''In-house production of Examination Papers using *troff*, *eqn* and *tbl*,'' in *Proceedings PROTEXT I Workshop, Dublin*, p. 21–28, Boole Press, October 1984.
7. J. F. Ossanna, ''NROFF/TROFF User's Manual,'' Computing Science Technical Report No. 54, Bell Laboratories, Murray Hill, New Jersey 07974, 11th October, 1976.
8. Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger, *The AWK Programming Language,* Addison-Wesley, 1988.
9. N. Chomsky, ''On certain formal properties of grammars,'' *Information and Control*, vol. 2, p. 137–167, 1959.
10. S. C. Johnson, ''YACC—Yet Another Compiler Compiler,'' Computer Science Technical Report 32, Bell Laboratories, Murray Hill, New Jersey, 1974.
11. A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translating and Compiling, Vol. I: Parsing,* Prentice Hall, Englewood Cliffs, New Jersey, 1972.
12. Jon L. Bentley and Brian W. Kernighan, ''Tools for Printing Indexes,'' *Electronic Publishing—Origination, Dissemination and Design*, vol. 1, no. 1, p. 3–17, April 1988.
13. *Brailsford Marriage Register (draft version)*. http://www.cs.nott.ac.uk/~dfb/bmr-README
14. N. Batchelder and Trevor Darrell, *Psfig—A Ditroff Preprocessor for PostScript files,* Computer and Information Science Dept., University of Pennsylvania, 1988. Internal Report.
15. Philip N. Smith, David F. Brailsford, David R. Evans, Leon Harrison, Steve G. Probets, and Peter E. Sutton, ''Journal Publishing with Acrobat: the CAJUN project,'' *Electronic Publishing—Origination, Dissemination and Design*, vol. 6, no. 4, p. 481–493, December 1993.
16. Adobe Systems Incorporated, *pdfmark Reference Manual (version 8)*, November 2006.
17. *GedML: Genealogical data in XML.* http://users.breathe.com/mhkay/gedml/