and removed (or renamed) at will — the assumption being that you are going to use an external parser/validator from time to time.

Other tabs allow the entry of entity declarations and notations. The whole DTD under development can be stored in binary form or exported to the standard plaintext file for inspection and saving. The program is at an early stage of development, so there are some rough edges, but the concept of a simple graphical interface to schematic-style entry is an attractive one.

2.6.6. NormDTD (Richard Light)

MS-DOS
`ftp://ota.ox.ac.uk/pub/ota/TEI/software/`

This is a public domain DOS program written by Richard Light (author of the *SGML Tagger*, see section 3.4.4) to handle those occasions where an SGML system cannot accept the complexity of large DTDs with deeply nested marked sections and parameter entity references to external files.

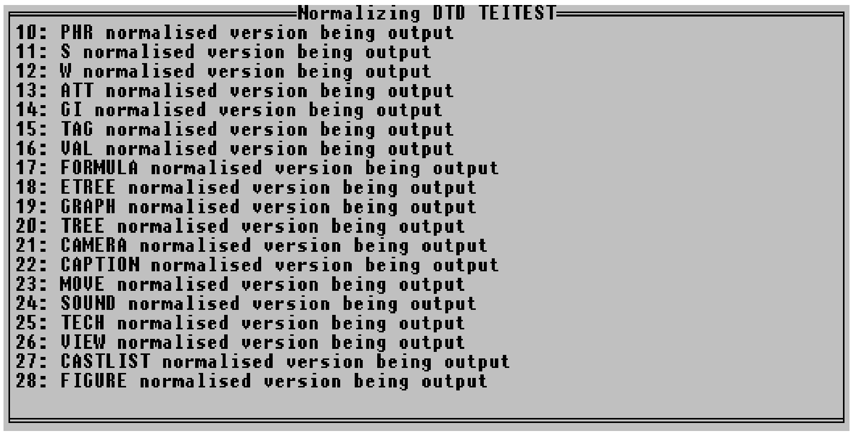
It ‘flattens’ the DTD to a single file, duplicating where necessary all the references that were previously handled by parameter entities. The element content models in this normalized DTD will not contain any references to elements that are not declared, and so it can be used by highly-strung packages such as *RulesBuilder* (see section 3.3.2.3) that refuse to process such applications (the TEI in particular: see section 2.3.6) for this reason.

2.6.6.1. Installation

NormDTD is supplied for download as a self-extracting executable, `normdtd1.exe`, from the Oxford Text Archive’s public FTP server, in the TEI software directory. This program expands in the same directory where it is run to the application itself, `normdtd.exe`, a low-memory version `normlow.exe`, and a subsidiary executable `taggerrd.exe` with an overlay for use where memory constraints require the process to be run in separate stages.

2.6.6.2. Operation

The command line for execution is `normdtd` *`dtdfile outputfile`* If the output file exists, the program stops, so any output from a previous run of the same name must be deleted manually first. If you don’t give an output file name, one will be created with the same name but the filetype `.dtn`



```
Normalizing DTD TEITEST
10: PHR normalised version being output
11: S normalised version being output
12: W normalised version being output
13: ATT normalised version being output
14: GI normalised version being output
15: TAG normalised version being output
16: VAL normalised version being output
17: FORMULA normalised version being output
18: ETREE normalised version being output
19: GRAPH normalised version being output
20: TREE normalised version being output
21: CAMERA normalised version being output
22: CAPTION normalised version being output
23: MOVE normalised version being output
24: SOUND normalised version being output
25: TECH normalised version being output
26: VIEW normalised version being output
27: CASTLIST normalised version being output
28: FIGURE normalised version being output
```

Figure 61. NormDTD normalizing the TEI DTD

There are some uneven patches in *NormDTD*, so the author suggests 'you should run a parser over the resulting normalized DTD to check for hanging separators and ambiguous context models'. In practise this means checking that there are no content models ending

```
<!ELEMENT LG - O
((HEAD)*, (L | LG)+, )
>
```

where the extra comma is left 'hanging' after the plus sign, and needs removing. This takes only a few seconds with the repeat-replace function of any text editor (and for the whole TEI it occurred just twice).

The software has been reported to hang when presented with an invalid DTD, so it is wise to parse and validate your application and check that it is 'clean' before trying to normalize the DTD.

2.6.7. SP (James Clark)

Unix, DOS/Windows
`http://www.jclark.com/sp/`

James Clark's *SP* package contains several programs which can be used in DTD development, but as the core of the package is concerned with parsing, the tools are dealt with more fully in section 4.3.5. I'm confining this section to the validation and processing of DTDs, rather than instances, although the program used is the same (*nsgmls*, the principal component of the *SP* suite).

Many DTDs are still written or maintained by hand in a plaintext editor, for a variety of reasons, not the least of which is the lack of software for large-scale DTD project development and code management. Parsing and validating a DTD frequently and rigorously during development is vital, in order to catch any errors before they become compounded.

To validate a DTD, you can just use *nsgmls* with the option `-p` (only parse the Prolog, and suppress any output), and `-f` (to redirect error messages into a file for examination):`nsgmls -pf`*`mydtd.err my.dtd`*The `-s` option, used when parsing instances to suppress the output, is implied by the `-p` option.

The options specifying the use of ancillary files require the filename *directly* after the letter (no space after the option). *Nsgmls* assumes the default catalog file is called `catalog` in the current directory, or in the location specified by the `-c` option:

```
nsgmls  -pfmydtd.err  -cmycat.cat  my.dtd
```

If you are using *Emacs* and *psgml*, you can press **Ctrl–C Ctrl–V**, which runs *nsgmls* on the current buffer (file) and traps the error messages to a window, but you'll have to edit the command line to put in the `-p` by hand, as the default is intended for validating whole documents. If there are errors, a keyclick on an error message will jump you to the file in question and place the cursor at the point of error. If you're using MS-Windows, you can install *RunSP*, which is a windowing shell for *nsgmls*: see section 4.3.5.1.

Because *Emacs* expects a DocType Declaration at the top of the file, if you get the following message when using *nsgmls* this way:

```
c:\sp\bin\nsgmls:my.dtd:20:2:E: "ENTITY" declaration not
allowed in prolog
```

it means you're probably trying to validate a DTD file. Instead, create a 1-line SGML file containing the DocType Declaration, referencing your DTD file with a System or Public Identifier (in effect, a document with a missing instance) and validate that instead.

XML parsing is slightly different. Quite apart from the additional constraints of XML syntax, a DTD may or may not be specified, and if it is, it may be retrievable from the Internet using a URL, rather than from a local file. The version of *nsgmls* distributed with the *Jade* converter/formatter (see section 5.2.4) has XML support, and this is on the CD-ROM. James Clark has also written *XP*, an XML parser in *Java* (see section 4.3.7.2)

The method of normalizing or 'flattening' a heavily parameterized DTD that I have referred to elsewhere for several DTDs can partly be done with the *spam* program using the following options:

```
spam  -mms  -ppxxfmyfile.err  myfile.dtd  >newfile.dtd
```

The ms value for the -m option removes all IGNOREd Marked Sections and unmarks all INCLUDEd Marked Sections; the -p option (twice) outputs the Prolog (SGML Declaration and DTD) and expands any entity references between declarations; and the -x option (also twice) expands all references to entities that contain tags. The error output (if any) is redirected into a file with -f, and the output (the flattened DTD) is written to the final filename (the > is a Unix and DOS technique to redirect output away from the terminal screen and into a file).

2.6.8. **Carthage** (Michael Sperberg-McQueen)

> Unix
> ftp://ftp-tei.uic.edu/pub/tei/sgml/grammar/carthage/

Michael Sperberg-McQueen wrote *Carthage* to overcome one of the problems encountered when compiling the full TEI DTD in systems which object to elements being referenced but not declared. The problem is explained in more detail in section 3.3.2.

This is a C program written for Unix, called carthago (for elements which are to be deleted; a subtle scholarly joke for those who know Latin: 'delenda est Carthago'). It reads a DTD and rewrites it, syntactically omitting from content models any elements which are not declared, where this is possible (on some occasions it cannot do all of them, for example if an element is marked as compulsory: further corrections have to be done manually). It can also delete IGNORE Marked Sections, detect entities declared more than once, and expand all parameter entities.

The software is offered 'as-is' (unsupported, and for regular SGML only at the moment) from the TEI FTP server at the University of Illinois at Chicago.

2.6.9. **Fred** (OCLC)

> Unix
> shafer@oclc.org

Before the days of easily obtainable (and especially, networked) information about SGML, misconceptions such as those explained in section 1.5.3 were even more widespread than they are today. Among the most pernicious was (is still, perhaps) the belief that SGML is just the act of making up some 'tag names' in pointy brackets and putting them in a file with your text. While XML goes a long way towards making