Document Object Models

- Define some quasi-formal notion of “document” and objects therein.
- Define operations (e.g., formatting, presentation) on documents in terms of operations on these elements.
- Multiple functions:
  - type versus composition
  - structure versus presentation

Document Object Distinctions

- Abstract/logical/structural/content vs. concrete/format/presentational objects
  - Logical: A type or instance of some abstraction.

    E.g., word (type), *paragraph* (type), “hello” (word instance), “When in the course of ...” (paragraph instance)
  - Presentational: 2-D rendering of a logical instance:

    “hello” (“hello” in 18 point Times New Roman, etc.)
- Older approaches failed to make these distinctions.

Document Structure

- More straightforward, formalized
- Use hierarchies, grammars for representation (for type, composition)
- Pragmatic rough edges

Document type hierarchy (genre)

Document => *Letter* | *Thesis* | *Paper* | *...*

Paper => *Journal-paper* | *Conference-paper* | *...*

Journal-paper => *CACM-paper* | *Computing-Surveys-paper* |*...*

Computing-Surveys-paper =>* <cs1> | <cs2> | ...

- <...> are document instances.

Straightforward but not extensively exploited.

Don’t quite need a grammar for this.

Document Composition

CACM-paper => *Header Body References*

Header => <*Title*, *Auth-stuff*, *Abstract*, *Keywords*>

Auth-stuff => *Auth-Affl*$^{+}$

Auth-Affl => *Auth-name* [*Affl*]

Body => [*Introduction*] *Section*$^{+}$ [*Ack*] *Refs*

Section => *Heading Parag*$^{+}$

Parag => *Sentence*$^{+}$

Document Composition Models

- Are CFGs and trees sufficient?
  - Want support for ordered and unordered elements
    - » E.g., Figures might “float” to convenient locations.
  - Some cross-dependencies are arguably not context-free, e.g.:
    - » assuring all references are in the bib.
    - » Grammar for *Auth-Affl*: Can (must?) omit affiliation if same as that following author.
  - Graphs rather than trees
    - » Lots of non-tree cross-references in documents (refs to section headings, bibs, figures)

Document Composition Models (con't)

- In the past, finessed because hardly anyone specified their markup language syntax formally.
  - and did little checking for well-formedness.
- Extensions like free-order aren't formally non-CF.
- Use permissive CFGs; depend on semantics for well-formedness.
  - Paragraphs are ASCII strings rather than sequences of sentences.
  - Line up references with referents semantically; same for variable *Auth-Affl* syntax.
- Can have "re-entrant" graphs within this framework.

Document Presentation

- Some typical presentational choices:
  - font (size, type, face); page, line, word (e.g., hyphenation) breaking
- “Concrete” is abstract wrt device characteristics (e.g., resolution, color), and hence requires a separate representation.
- Less obvious how to define, represent or formalize.
- Proposed definition: Given a content, “presentation” is all the choices we have to make to determine the “geometry” of a document.

Structure/Presentation Distinction

- is relative to the author's intention.
  - Might want to explicitly specify a color (or color class) or font.
- seems somewhat arbitrary at times.
  - E.g., "abstract quoting" versus *abstract quoting*
    - » I.e., we commonly use italics and quotes for the same purpose, but the former is generally classified as presentational and the latter as content.

Levels of Presentation Structure

- Presentations comprise
  - 2-D area sequences (page space)
  - Font specification
  - Layout within areas (filling)
  - Determining presentation-dependent references (numbering pages, sections)
- Neutral wrt pixels on a device.
- Window resizing requires some re-presenting; window scrolling doesn’t.
  - Perhaps there should be a level that is neutral wrt things like resizing.

Document Transformation

- Computing explicit references
  - values of variables, macro expansion
- Generating specific document elements latent in the original description
  - TOC, index
- Determining presentation-dependent references
  - numbering pages, sections
  - Note: These are abstract elements that aren’t determined until presentation time.

Document Formatting

- Takes an abstract document instance and produces a concrete document presentation.
- Functions:
  - terminal node target object specification
    - » e.g., character selection and font determination
  - area and sequence determination
  - placement of constituent objects
    - » e.g., filling subject to constraints

Early Systems

- 1960's: First text processing systems, essentially formatters
- Line-oriented, output to teletype
- ASCII in, ASCII out
  - Input likely to be cards or paper tape
  - Generally device-specific output
- E.g., RUNOFF, FORMAT
  - format and layout specifications mixed with text (which is still common)

RUNOFF

- 1960s, Saltzer, MIT
- Limited set of objects and low-level formatting operations (although enhanced over time)
  - operations mixed in with text
- Abstract objects:
  - characters, words, word sequences
- Concrete Objects:
  - lines
    - » center, break, no-format
  - blocks (of lines)
    - » set-length, fill, justify, indent, spacing
  - pages
    - » headers, set-length, begin-new, number
- Very influential initial condition!

RUNOFF Example

```
.center
RUNOFF EXAMPLE
.sp 2
The above command skips some
lines.  This text will be
filled and justified.
.nojust
We've turned off justification.
.indent 5
This will be indented.
So will this one
.indent 0
.adjust
We're back to normal.
```

Second Phase

- General programming language conveniences
  - conditionals, macros, variables (counters), expressions
  - Note that we are getting more procedural!
- Lots more functionality of same sort.
- New functionality, such as automatic section numbers, TOC and index generation
- More output device flexibility
  - but still little device independence
- Introduction of logical objects
  - but lower-level aspects still visible
- Still editing the markup language.
  - I.e., no user interface, WYSIWYG editor, but rather, “document compilers”

PUB

- Developed 1971 at Stanford by Tesler.
- Some higher-level objects
  - Structural: footnotes, sections, subsections
    - » automatic numbering, TOC generation
  - Presentational: columns
- Presentation model: Pages made of *areas*
  - *Text areas* flow from page to page.
  - Other types (e.g., *titles*) are fixed size.
  - Normal page is heading title area, text area, footing title area
- Various programming constructs
  - *blocks* - Nested and sequential contexts served as backdoor for structure representation.
  - *responses* -- macros triggered by various events
  - *portions* - can *send* text, commands to portions; these can be used when portions are reached.

NROFF, et al.

- Bell Labs, mid-70s, still in use.
- Like PUB, weak structure support, reliance on programming features:
  - *environments* (like blocks, but more limited)
  - *macros*, *traps* and *diversions* used to implement headers, etc.
- Very low level, generally used in conjunction with macro packages, preprocessors.
  - These tend to carry semantic weight.
- Extensibility probably key to longevity

XROFF -ms

- Macro packages like Lesk's *-ms* provide more structured, abstract objects.
  - footnotes, section headings, paragraphs, columns
- Multiple fonts, sizing
- Some abstract document structuring. E.g.:
  - Using .TL, .AU, etc., one could specify title, author, etc., independent of where or how they actually appeared.
- Still highly procedural, interplay of high-level and low-level constructs makes semantics obscure.

EQN, TBL, etc., add types

- High-level, declarative languages specific to particular types in addition to textual objects. E.g., EQN adds math objects, TBL table objects, PIC picture objects.
  - “Compiled” into troff/nroff.
- EQN allows description of equations textually, formatting done based on type and structure, e.g.:
  - Function names set in roman, other text in italics
  - Various components set in appropriate sizes.
- TBL basically does layout into rectangular boxes.
- EQN target objects are (objects inside) rectangles; expressions put together rectangles into bigger rectangles.

TBL Example

```
.TS
c c c
|l|l|l|.
level      function        location
_          _               _
top        presentation    users
level      services        workstation
_          _               _
next       application
level      logic
...
_          _               _
.TE
```

to produce...

level	function	location
top level	presentation logic	user workstation
next level	application logic	
bottom level	DBMS services	central server

Scribe

- Brian Reid, CMU, late 70s
- User describes *logical document structure*, formatting details determined by system.
  - based on document type, output device, etc.
  - no direct user lower-level positioning, etc.
  - highly portable, device independent, etc.
- Lots of tools developed a la NROFF.
- No facility for complex objects (tables, equations, etc.) and extension hard.

Scribe Model

- Nested *environments* have concrete *attributes*, enclose text.
  - Formatting is applying environmental attributes to enclosed text.
  - Unspecified attributes are inherited from the surrounding environment.
  - A document type specifies a global environment.
    - » A local data base maintains definitions of attributes for available types.
  - User can define and modify environments.
  - The attribute set is fixed.
- There are a fixed set of (generally) procedural commands.
  - Have local rather than regional effects.

Scribe Example

```
@Make(article)
@Heading (Scribe Example)
@Center (By RW
for CS294-5)

@Section(The Scribe system)
Here is a paragraph of text.

A blank line indicates a new paragraph.

@SubSection(Environments)

The concept of an @Begin(i)environment@End(i)
@Cite(ReidPaper) is central to Scribe.
```

TeX

- Knuth, late 70's
- Focus on
  - high-quality typesetting, esp. math.
  - expressive power versus ease of use. (I.e., another "markup assembly language"
- Introduced lots of new (and often still best) formatting algorithms.

TeX Presentation Model

- Concrete document is modeled as “bounding boxes” and “glue”.
  - boxes might contain characters, words, lines, paragraphs, pages, equations
  - aligning two boxes produces an enclosing box, a la EQN
- Glue holds boxes together
  - A piece of glue has a normal size.
  - Can be compressed or stretched according to various constraints.
  - Lots of operations performed with glue:
    - » justification: adjust glue so ends of words at margins
    - » centering: put infinitely stretchable glue at ends of phrase.

TeX Markup

- Specifications can be scoped to groups.
- Support for some abstract objects: characters, words, paragraphs, equations; lines, pages
- Separate math mode, similar to EQN.
- Additional of semantic weight carried by macro packages:
  - LaTeX
  - Lots of document type style sheets created.

E.g.: Table in LaTeX

- Note: No tables in TeX as such.

```
\begin{tabular}{|l|l|l|}
\multicolumn{1}{c}{level} &
\multicolumn{1}{c}{function} &
\multicolumn{1}{c}{location} \\ \hline
top & presentation & users \\
level & services & workstation \\ \hline
next & application & \\
level & logic & \\ \hline
bottom & DBMS & central \\
level & services & server \\ \hline
\end{tabular}
```

TeX Algorithms

- hyphenation
- page breaking
- breaking paragraphs into lines
  - use dynamic programming to find arrangement of lines minimizing hyphenation and glue “badness” (stretching or compressing)
  - Penalties/rewards can be specified for a break at a point.
  - In effect, searches lots more possible ways to break than other algorithms.
- TeX algorithms
  - aim at best “batch” solutions; need to consider incremental, fast, perhaps “anytime” solutions for on-line documents.
  - Still are hard to understand and require trial and error to get a reasonable set of constraints.

WYSIWYG Document Editors

- Bravo (Lampson, ‘78) on the Alto (Thacker, ‘79).
  - First modern WYSIWYG editor/formatter.
  - Modern editors all derive from this.
- Limited number of simple text objects
  - Other types -- drawings, figures -- can be created with other tools and included.
- *Forms* (templates) are partially specified document types (e.g., letter, memo, tech report)
- And have same drawback:
  - Hard to enforce structure instead of appearance.
  - E.g., user specifies bigger, bolder numbered line to designate section heading; no easy way to change characteristics globally.

SGML

- Began with GML at IBM in ‘70s (Goldfarb)
- SGML is a meta-language for purely structural document type descriptions.
  - I.e., no specific systems involved.
- Attempts complete separation of structure from presentation (which is supposed to be specified elsewhere).

Issues - Presentation

- Presentation models are rather weak (e.g., areas with glue)
- Would be nice to specify:
  - Arbitrary distance/alignment constraints between objects. (Even TeX allows only constraints between distances of arbitrary *adjacent* objects.)
  - Relative size of objects (Graphics systems sometimes support a version of this.)
  - Control over areas/page spaces.
    - » Non-rectangular layout areas. E.g., two columns with box cut out in middle for picture.
    - » PostScript-like clipping?

Issues: Markup Languages

- Hopeful trend toward declarative languages, with well-defined semantics.
  - Documents are more or less declarative, so this should be adequate. (But what about *active documents* and the like?)
  - Conducive to WYSIWYG editing; only way non-programmers will produce/edit a document.
  - Portability (because of semantic well-formedness)
  - Presumably, behavior more transparent.

Separation into structure, presentation

- Trend from undifferentiation to complete separation.
- Might want separate but mixable notations.
- Relations between layers:
  - Earlier in this document is

    “hello” (“hello” in 18 point Times New Roman, etc.)

    Would like to have said:

    hello<f1:font-size>” (“hello” in $f1 point Times New ...)
- How do we facilitate WYSIWYG editing on document structure?
- Other intermediate levels of representation?

Environments

- Support tools facilitate document creation, but inhibit portability.
  - Ability to define new types (content types, document types, fonts) may make it difficult to transport documents.

For next time

- SGML
- I'll post readings, assignment, on web page.
- Should get up to speed for our two visiting speakers.