A TEX User's Guide to ISO's Document Style Semantics and Specification Language (DSSSL)

Martin Bryan
The SGML Centre
29 Oldbury Orchard,
Gloucestershire, U.K.
Internet: mtbryan@cheltenham-he.ac.uk

Abstract

The importance of capturing structure information in documents that are likely to be re-used is increasingly being recognized. In both academic and commercial circles the role of ISO's Standard Generalized Markup Language (SGML) in capturing and controlling document structure is becoming more widely acknowledged.

While TEX itself does not utilize document structure information, many of its macro facilities, such as LATEX, provide some, albeit high-level, structure control. LATEX3 seeks to increase the level of structure recognition by adding recognition of attributes within macro calls to allow more than one interpretation of a structure-controlling markup tag.

ISO's approach to the problem of linking document structure to document formatting engines such as provided by TEX is to develop a language that can be used to add suitable sets of formatting properties to SGML-coded and other structured documents. The Document Style Semantics and Specification Language (DSSSL) has two main components: (1) a General Language Transformation Process (GLTP) that can take a document with a predefined input tree and transform that tree into the form required for subsequent processing, and (2) a set of Semantic Specific Processes (SSPs) that specify how specific operations, such as document formatting, shall be specified at the input of the formatter.

This paper explains these two processes, and shows how they can be used in conjunction with TEX.

Introduction

The advantages of adopting a structured approach to document markup have been known to TEX users for many years. Most of the macro languages developed for use with TEX use the names of the structural elements that make up a document to identify the way in which the document is to be presented to users. Notice that I did not use the word 'formatted' here. Today TEX is used to present documents to users on screen almost as often as it is used to prepare documents for printing, and this fact has to be a key factor in the development of any new language for describing how to present text. The characteristics of screens differ from those of paper, so the rules used for presenting information on screens have to differ from those used to present the same information on paper.

Within the Open Systems Interconnection (OSI) program of standards development at the International Organization for Standardization (ISO), there are two main standards for the creation and presentation of structured text. For data whose structure is controlled by the presentation process the Open (originally Office) Document Architecture (ODA) can be used to define the logical structure of typical office-related documents. For more general purpose applications, the Standard Generalized Markup Language (SGML) can be used to describe the logical structure of captured data, and the Standard Page Description Language (SPDL) can be used to describe the formatted result. ISO's Document Style Semantics and Specification Language (DSSSL), which acts as the link between these two forms, has been designed to allow systems to interchange information that can be used to convert logically structured SGML files into physically structured, presentable, SPDL files, or into the form required for processing by existing text formatters, such as TEX.

About DSSSL

As TEX users you will understand that there is a difference between the way in which information is presented on a page or screen and the way in which it is created and used. One of the key problems that has faced the SGML community is that the logical structure that is needed to guide users during data capture or data retrieval is not necessarily the best structure for text formatting. If users are constrained to a model that reflects the way in which the document is to be formatted they are likely to object to the need to capture data in a structured format (and, given human nature, are likely to go to the opposite extreme and insist on creating totally unstructured, "What You See Is All You'll Get" (WYSIAYG), documents).

What is needed is a coherent means of describing the relationship between elements used to navigate through an information set and the objects used to present specific parts of the information to users. The transformation that is required to achieve this forms the first part of the DSSSL standard, which defines a General Language Transformation Process (GLTP) that can take objects in an SGML-encoded data tree and associate them with objects that can be used by a formatter. This transformation uses an advanced, SGML-knowledgeable, query language to identify the relationships between objects making up the source document and those making up the output of the transformation process. The relationship between SGML elements, their attributes, the file storage entities that contain them and the entities and elements they contain can all be identified and mapped, as appropriate, for output. This means that attributes in the source document can create new elements or entities in the output document, and that elements or storage entities can be used to control attribute (property) setting in the structure passed to the text formatter.

The DSSSL GLTP performs a role that TEX does not address. It will allow you to take an SGML-coded document and turn it into a format that is suitable for processing by a known set of TEX macros. By allowing, for example, structurally related cross-references (e.g., see Chapter 4) to be resolved prior to formatting, with the appropriate formatting properties being generated in response to the type of reference, DSSSL should be able to reduce the amount of work that needs to be done during formatting significantly.

The DSSSL document formatting Semantic Specific Process (SSP) will:

- provide a set of formatting properties that have internationally acceptable formal definitions of their meaning,
- provide a model for describing areas into which data is to be positioned or flowed, and
- provide a method for describing which area, or type of area, each object in the structured document should be placed into.

The first stage of the document formatting SSP consists of describing the relationships between the various areas that make up a page, and the properties of each area, in an area definition. The second stage consists of describing how the elements that make up the GLTP output tree are to be 'flowed' into the areas described in the area definition. Figure 1 shows the relationship between these processes.

Rather than invent a completely new language to define the relationships between objects in the various transformations, DSSSL has been developed as an extension to IEEE's Scheme variant of LISP, produced by MIT. The adoption of this advanced AI language offers a number of important advantages. In particular LISP is an object-oriented language that is ideally suited to querying data structures and trees. Any location in a document tree can be described in Scheme as a list of the objects that make up the tree, e.g.:

```
(document body (chapter 4)
(section 5) (subsection 2) (para 3)).
```

Scheme provides a compact, but clearly defined, set of functions that can be used to manipulate and transform object lists. DSSSL provides the additional functions needed to describe the relationships between Scheme processes and SGML constructs, e.g.: `(query-tree *root* '(EL "p")`.

Another key consideration in the choice of Scheme was the simplicity of its powerful recursive processing features. Formatting is largely a recursive process, and Scheme can simplify the expression of recursive processes.

While the exact form of the Scheme constructs is still under discussion the following examples will give you some idea of the form the final language will probably take. A typical area definition for the running head of a left-hand page might be:

```
(area-def left-header
   '(rep glyph)
    (set-properties
     (area-type line)
     (origin (point 7pi 0.5in))
     (x-extent 39pi)
     (y-extent 1pi)
```

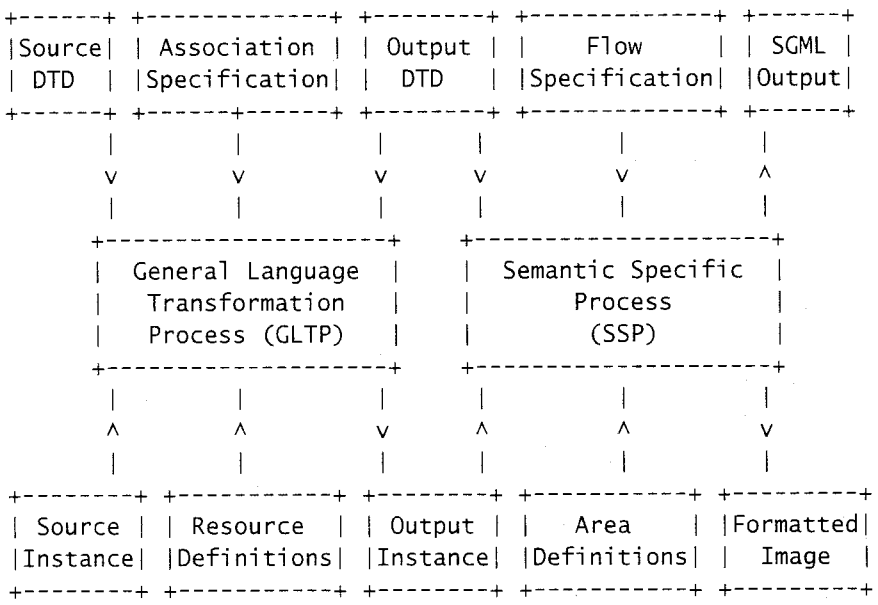

Figure 1: The Modules of DSSSL.

```
    (placement-path-start
    (point 0 baseline-offset))
    (placement-path-end
    (point 39pi baseline-offset))
    (align centre)
    (word-spaces (18 15 18))
    ; there are 54 units to 1 em
    (letterspace (6 6 6))
    (hyphenation-allowed #f)
    (font-name "/Monotype/Helvetica/Medium")
    (point-size 10pt)
    (set-width 10pt)
    ((case usc)
    ; Uppercase for capitals
       - small caps for lowercase
   ))
```

while a typical flow specification could have the form:

```
((title title-p)
     ((wide-text single-column-text-area)
       (set-properties
        (prespace 1in)
        (postspace 3pi)
        (font-name heading-font)
        (font-size 36pt)
        (path-separation 42pt)
        (align centre)
        (last-line centre)
        (hyphenation-allowed #f)
        (cond (count
               (children (line title title-p))
               < 1)
               (generated-text
                "Unnamed Report")
        )) )
        (left-header left-page)
```

In this example the title element on the title page flows into both an area for holding wide text lines in a single colum text area and into the running header for left-hand pages. For the title page the formatting properties are applied as part of the flow definition because, in this case, the area being used to contain the text is a general-purpose one whose default typesetting properties are not suitable for the type of text about to be poured into it. DSSSL's ability to qualify the way in which areas are formatted dependent on the contents that are flowed into it provide a type of functionality provided by few formatters. Note particularly the inclusion of a standard Scheme condition expression that acts as a trap for cases where no data has been entered into the title field of the report. In this case the left-hand running head will be blank, but the title page will at least have a heading (Unnamed Report).

One of the interesting points of discussion among DSSSL developers is the relationship between the various sets of properties that can be used to control processing. The way in which text should be presented to users can be defined as:

- using attributes defined in the source instance;
- using attributes defined as part of the output GLTP;
- through SSP-defined properties assigned to areas in the area definition; or
- through SSP-defined properties attached to the flow specification.

Which of these should take precedence, and in what order should they be applied? Personally I feel that attributes in the source, or those created as part of the transformation process, should be able to override properties defined as part of an area definition, but should they also override any properties specified while defining how output objects should be flowed into areas? There seems to be no clear-cut answer to this dilemma, and it looks as if the DSSSL team will need to define a precedence order as part of the standard based on gut feeling, unless someone can come up with a convincing case for a particular approach. (Any offers?)

How Can DSSSL Be Used in Conjunction with TEX?

The DSSSL GLTP process can be used to transform SGML-encoded files into forms suitable for processing by existing TEX macro sets, such as those provided by LATEX. In such cases there is little need to associate area definitions with the GLTP transformation as this is the function of the TEX macros. However, in the longer term, it would be advantageous if we could map the way in which DSSSL describes areas, and the properties used to describe the required output, directly into TEX. Hopefully this will not prove too difficult a task, especially given the transformational power provided by Scheme.

The work currently being done on LATEX3 should make it easier to use TEX as the output process of the DSSSL process. As I understand it, one of the aims of LATEX3 is to allow attributes/properties to be used to control the way in which macros process the associated text. If we can find a way to map the properties defined in DSSSL to equivalent functions in TEX it should be possible to provide simple transformation algorithms that will turn the output of the DSSSL GLTP process into appropriate LATEX3 macro calls. While we are still a number of years from being able to do this, now is the time to plan how this should be achieved, before DSSSL or LATEX3 are completed. For this reason the DSSSL team is keeping a close eye on what is happening in the TEX world.

As DSSSL is designed to be used with a wide range of formatters, including those based on TEX, it is important that the way in which DSSSL properties can be interpreted in a TEX environment be carefully studied prior to publication of the final standard. For this reason it is important that the TEX community track the changes that will be published in the second draft of the international standard when it is published later this year.