Language Information in Structured Documents: A Model for Mark-up and Rendering

Frank Mittelbach and Chris Rowley
LaTeX3 project
Frank.Mittelbach@eds.com, C.A.Rowley@open.ac.uk

Abstract

In this paper* we discuss the structure and processing of multi-lingual documents, both at a general level and in relation to a proposed extension to the (no longer so new) standard LaTeX. Both in general and in the particular case of this proposal, our work would be impossible without the enormous support, both practical and moral, we get from our fellow members of the LaTeX3 project team† (who maintain and enhance LaTeX) and from people all over the world who contribute to the development of LaTeX with their suggestions and comments.

Introduction

The paper starts by examining the language structure of documents and from this a language tag model for LaTeX is developed. It then discusses the relationship between language and document formatting and the types of actions needed at a change of language. This will lead to a model that supports the specification of these actions and of their association with the tag structure in the abstract document.

The model is then extended to provide the necessary support for regions that have their own visual context or that receive content from other parts of the document, thus breaking the basic tree structure of an abstract document — this is in the section entitled "Special Regions".

Finally a high level summary of the required interfaces is given. A full formal specification, to be used for a prototype implementation in LaTeX, is currently under development—a first public test implementation is expected to exist for the 1997/12/01 release of LaTeX.

If you are interested in the issues raised in this paper or in other aspects of our work to enhance LaTeX, please join the project's electronic discussion list. To do this, please send a message to:

`listserv@relay.urz.uni-heidelberg.de`

Containing this line:

`subscribe LATEX-L` *`your name`*

Language Structure of Documents

Structured documents can be understood as being explicitly or implicitly labeled with "language tags" denoting that a portion of the document contains data written in a certain "language".

These tags have the following properties:

- They impose on the document a hierarchical tree structure that may not be compatible with that document's other logical structure, e.g., there might be a language change in the middle of a logical element such as a list item.[1]
- At any one point in the document the "current language" can be determined.

The term "language" in this context is somewhat vague and might need further qualification; but for the purpose of the following discussion it is sufficient to define it as a 'label' whose value affects certain aspects of formatting.

Hierarchy of language tags

The structure created by attaching such language tags to the text can be considered to be of varying complexity. The simplest case would be to regard this as a flat structure: for each point in the document only a "current" language is defined, disregarding the fact that certain language segments can be considered to be embedded within others. This model of language within documents is, for example, employed within the current Babel system where, by default, all language changes are in this sense global.

* This paper was originally given at the Multilingual Information Processing symposium, March 1997, Tsukuba, Japan.

† Current LaTeX3 project team members are Johannes Braams (NL), David Carlisle (UK), Michael Downes (USA), Alan Jeffrey (UK) and Rainer Schöpf (DE).

[1] However, for practical purposes it is normally possible and acceptable to artificially force the structure imposed by the language tags into the logical hierarchy imposed by other tags.

In a more complex model each area has a "current" language but may be embedded within a nest of larger areas, each in its own language. In such a model, a change of language has a different quality, and therefore may invoke different formatting changes, depending on the level in the hierarchy at which it occurs.

Our investigations lead us to conclude that, to properly render a document, one needs a combination of both models:

- the concept of a base language for very large portions of a text (for most documents this will in fact be only one such language for the full text): this has a flat structure, there is only one base language at any point in the text;
- the concept of imbedded language segments: these are nestable (to any number of levels) and are used for relatively small-scale insertions within a base language, such as quotations or names.

Language tag (visual) structure

In addition to the nesting structure of language tags, there is a more visual component that influences rendering of a document: the paragraph structure. To properly model this typographical treatment it is necessary to classify the language tags according to whether a language segment contains only complete paragraphs or is part of the running text of a single paragraph. A begin/end pair of tags is called a "block-level" tag if its body consists of complete paragraphs and a "paragraph-level" tag otherwise. As later examples will show, the typographical treatment for these two types is often different.

A Tag Model for LaTeX

To support the above model, including both nesting of language tags and the differentiation between block- and paragraph-level tags, the following tag structure for a system like LaTeX is proposed:

- A document language tag (implicit). This tag can be used to attach language-related typographical actions that should not change even if the document contains more than one base language.
- Base-language tags: used only at top-level, no nesting. These tags denote the major language(s) within a document. In the case of essentially mono-lingual documents the base language would be the same as the document language.
- Language-block tags: contain complete paragraphs, nestable. These denote larger imbeddings either directly within the base language or further down in the nesting hierarchy.
- Language-fragment tags: only within paragraphs, nestable. These denote smaller imbeddings but are otherwise identical to language block tags.

Note that since, at least in the logical structure of a document, paragraphs can occur within paragraphs, block tags can be nested within fragment tags.

Document interfaces

As LaTeX 2$_\varepsilon$ does not have built in support for named attributes, its support for language changes is best implemented by introducing additional language tags (commands and environments). A concrete syntax for these tags could include the following:

- A preamble declaration for the document language (this is also the base language in monolingual documents) with the language-label as argument.
- A base-language change command with the language-label as argument. This command is declarative to highlight the flat structure of base languages.
- A language-environment with the language-label as argument and text as body. Such an environment starts a new paragraph so as to enforce the block-level nature of the tag.
- A language-command with the language-label and text both as arguments. In contrast to the environment, this command applies language-related actions to its second argument, which cannot directly contain full paragraphs.

For LaTeX3 we shall probably normalize this interface by supporting a language attribute on appropriate tags. This would allow, for example, a trivial translation of the language features currently being proposed for HTML into LaTeX for rendering purposes. However, even in that case generic tags for changing language are necessary as typical documents contain language changes that do not coincide with the tag boundaries of other logical tags.[2]

Language-dependent Processing

Setting up the tags tells us only how to encode a multi-lingual document. We now need to specify how these tags affect the processing of the document; how do we attach actions to them? Before answering this question we shall first discuss a

[2] It is proposed that HTML 3.2 supports a `<span>` tag for this purpose.

number of representative examples of the effects of language on this processing, classified according to the categories input, transformation and formatting.

The actions shown below are all commonly related to a change of language within a document. Nevertheless, it is not the case that each of them should necessarily be implemented by attaching them firmly to language changes. For some it might be more appropriate to freeze them for the whole document or to attach them to areas within the document that do not coincide with language boundaries.

Input

Input encodings Entering text in a certain language often requires special input methods (this is especially true for languages with complex scripts) but even in cases where direct keyboard entry is possible it might be necessary to add information about the keyboard codepage that is to be used, so as to interpret the source characters correctly. At present LaTeX supports variable interpretation of the upper half of the 8-bit plane, thus allowing source text to be 8-bit encoded in one of the many keyboard encodings used world wide.

Short-refs With the development of language packages and the subsequent development of the Babel system, it became common practice to extend the mark-up language of LaTeX using so called "short-refs" as a compact method for inputting certain commands. Short-refs are character sequences that do not start with TeX's escape character, i.e., usually '\', but nevertheless act like commands. That is, they do not represent the equivalent glyph sequence but have either additional effects (e.g., the punctuation marks in French typography, which produce additional space) or even denote completely different actions (e.g., `""` for a break point without a hyphen).

In addition to the above short-refs, some TeX fonts implement short-refs by using (or misusing) the ligature mechanism to implement arbitrary input syntax, e.g., ` `` ` generating " or `---` generating an em-dash.

Short-refs can be used for different purposes:

- providing a compact input notation for commonly used textual commands such as characters with diacritical marks;
- providing a compact and readable input notation for special applications, e.g., `==>` for `\Longrightarrow`;
- providing typographical features not otherwise supported (e.g., extra space in front of punctuation characters).

The first two items are related to input syntax and not directly linked to the language of the current text although historically they have been provided by language packages, e.g., `"a` as a short-ref for `\"{a}` was implemented by `german.sty` and within Babel its meaning gets deactivated within regions marked up as belonging to other languages.

The third item is directly related to language since short-refs of this type are used to implement a typographic style that is characteristic of a language in such a way that the user is not forced to use explicit mark-up in the document.

Transformations

Here, 'transformations' include only manipulations of the source text that are independent of formatting information (i.e., those that act entirely on the logical document). Usually such transformations enrich the document content in one way or the other by using knowledge stored outside the document source.

Generated text This is text that is not directly encoded in the source document but is produced from tags therein. Generated text can be classified into two categories: content-related and structure-related. Here content-related text is that generated by tags that can appear anywhere in the source text (a typical LaTeX example would be the `\today` command) while structure-related text refers to text that is associated with a high level logical structure (e.g., the heading produced for a bibliography or the fixed text used in a figure caption).

While it is imaginable to keep structure-related text in one language even though the surrounding language changes, content-related text most likely will have to change at every language tag.

Hyphenation The finding and marking of possible hyphenation points is, perhaps, the most obvious language-related transformation. Indeed, it is often considered to be the defining characteristic of a 'language'.

When using TeX this relationship is unfortunately obscured by some technical details of the implementation of hyphenation. One of these is that TeX's hyphenation does not depend only on the 'language' but also on the current font encoding (which can differ within a single language). Another is TeX's restriction that one can properly hyphenate a whole multi-lingual paragraph only if the font encodings used therein share a single lower-case table

(and this is likely not to be the case if more than one script is present).

Upper- and lower-case transformations The mapping between upper- and lower-case characters (for those writing systems that make such a distinction) is language-dependent (and not just script-dependent): for example, in Turkish ı→I and i→İ in contrast to the usual mapping i→I used in most other languages. There can also be a one-to-many mapping as for the German ß that maps to SS.

Formatting

Although each of the examples listed here has been documented as characteristic of the typography associated with a particular language, they are all also aspects of the design over which a document designer may wish to have control that is independent of the language of the text.

Direction The direction of the text and, more generally, the writing system used are very strongly associated with the language in use.

Micro-rendering This covers the details of rendering at the level of individual glyphs and the relationships, often complex, between the characters which form the textual part of the logical document and the glyphs used to render this text, especially when aiming for the highest levels of typographic quality. These details often depend on what glyphs are provided by the available fonts. Also, when using TEX, this level of formatting is typically controlled entirely by the choice of font, whereas it should be possible to specify such details independent of the font since they also depend on the language in use.

Some examples:

- The precise positioning of diacritics depends on the language; e.g., a language such as German with many umlauts puts them closer to the top of the basic letter than is typically done with the diaeresis in English or French typography.
- The use of aesthetic ligatures varies from language to language, e.g., the ffl-ligature is traditionally not used in Portuguese and Turkish typography (implementing this is difficult in TEX since these transformations are normally controlled entirely by the font and there is no simple way to 'turn them off').

Macro-rendering More global aspects of typography can also be language-dependent, for example:

- the formatting of in-line quotes (i.e., what 'quotation marks' to use);
- rendering of enumerations;
- aspects of page layout (e.g., float placement).

As with most language-related actions they usually have a wide range of formatting possibilities and can be considered to depend, at least partially, on house style or other factors.

Attaching Actions to Change of Language

Having described some typical changes that need to be made at a language tag, we now look at how to tie particular actions to a particular tag, noting that it is not sensible, for example, to change every aspect of the formatting if only an in-line fragment of a few words is to be in a different language.

Attaching actions to tags

First we note the following facts.

- The type of actions that are required at language tags can be modeled by setting the values of a collection of parameters to those appropriate for the new language.
- Some actions may not make sense at certain levels of the hierarchies. For example, while one wants to use the correct hyphenation algorithm at any level of the hierarchies changing of micro-rendering, such as the positioning of diacritics, might be applied only to language changes for whole paragraphs but not for fragments.
- However, for most actions it is not possible to specify one place in the hierarchies that will produce the correct location of that action for *all* documents. The correct place might, for example, depend on document type or on a particular house style.

There are two (at least) possibilities for specifying, for a particular document, where in the tag hierarchy an action should be 'attached' (see Figure 1). These are by the nesting-level in the hierarchy of language tags or by the visual type of the language tags as described in the section entitled "Language tag (visual) structure". These visual tag-types implicitly define a partial hierarchy, from the top: document, base, block, fragment.

In both cases an action is defined to be executed down to a prescribed level in the hierarchy. As noted above, different actions might be executed down to different levels so there needs to be a mechanism to specify this level for each action. To limit the complexity of the model we think it is advisable to assume that this stopping level depends on the action but not on the language. It was pointed out in Tsukuba that this is probably an oversimplification, i.e., that there exist cases where it would be better to model the formatting of language-related

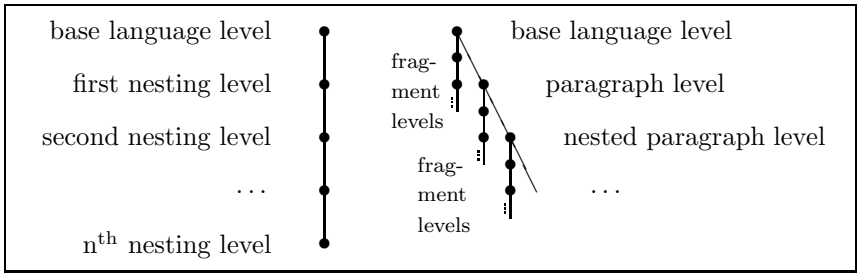

Figure 1: The two hierarchies

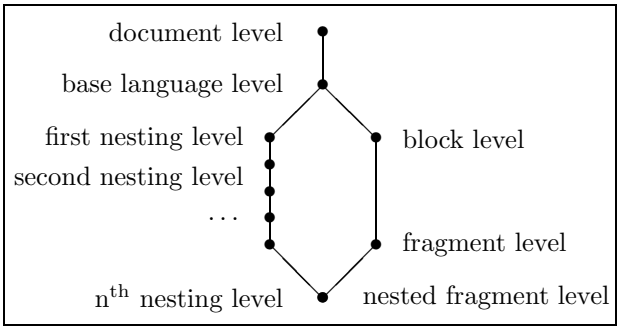

Figure 2: Tag hierarchy diagram (THD)

items by attaching of language/action pairs to levels. However, we think that these cases are sufficiently rare that they can be handled by the action itself.[3]

It is also possible to combine these two hierarchies and allow the attachment of actions to tags via either hierarchy (see Figure 2). In this case, for each action it is necessary to define:

- on which of the two hierarchies the stopping of the action depends;
- down to what level the action is carried out in that hierarchy.

Data structures for this model

For this model of language tags/actions, the system needs to specify the contents of the following three data structures.

Tag hierarchy diagram (THD) While combining the two hierarchies we have simplified their structure (compare figures 1 and 2), i.e., multiple nestings of paragraphs are collapsed into a single node. At the same time a new root node (document-level) was added. This node serves as an anchor point for typographic requirements that should stay fixed throughout the document even if the base language changes.

The required number of significant nestings in the hierarchy of nesting-levels is an open question but probably $n = 3$ is sufficient to specify typical formatting requirements.

The two end points of the hierarchies (n^{th} nesting-level and nested-fragment-level) are combined as they essentially mean to carry out attached actions in all cases, thus it does not matter on which hierarchy they are specified.

Another interesting point is that the two base-language-levels,one from each hierarchy, are combined.[4]

Nevertheless, it should be noted that the "level" of a tag within the THD is logically described by a pair of nodes (one on each hierarchy) even though in some cases these nodes collapse into one.

Language actions table (LAT) This two-dimensional table (indexed by parameter-group and language-label) stores the effect of each action (i.e., the value for a parameter-group) for each language (possibly only a default value if no value has been explicitly defined for that language). Each entry is an expression that returns a set of values appropriate to the parameter-group.

[3] An action that depends both on language and level could be specified in the model by executing it on all levels with an additional conditional within the action body testing for the current language.

[4] From this it follows that in this model a base language change is only allowed between paragraphs.

It may be possible[5] to also allow special actions to be specified, such as:

- leave unchanged;
- use some default (e.g. the value for the document language).

Parameter assignment map (PAM) This one-dimensional table maps each each action (modeled by a parameter-group) to a single node in the THD.

Such an assignment means that this parameter group changes its value (using the method specified in the LAT) at all levels down to (and including) the node to which it is mapped.

Special Regions

The scheme we have outlined so far will work well for the main text of many documents but it needs to be supplemented in order to handle formatting of the following material (called special regions):

- regions that contain text which has moved from other parts of the document, e.g., table of contents, running heads;
- regions of text that are first formatted and then the whole block is moved, e.g., (from LaTeX) floating tables, footnotes;
- regions that can contain elements breaking the type hierarchy, e.g., paragraphs in table-cells.

There are several problems that arise when "moving things around" in a document: one of these, which arises only when logical (unformatted) text is being moved, is the need to move language information with the moving text. This is needed even if the text being moved is in the document language since this may not be the current language at the point to which it moves. Thus the data-type for 'logical stuff being moved' must be the text and a language-label (describing its language).

Formatting special regions

A problem that affects the formatting of all special regions is how to specify the language to be used and the effective level of language tags contained within the special region. It is not possible to simply extend the THD and PAM from the main part of the document since these assume that the nesting of the language tags in the logical document is faithfully represented in the formatted document. This is very clearly not the case with regions such as floats or end-notes which appear visually in totally unrelated parts of the document. It is also not true for paragraphs within tables since these can be, logically, paragraphs within paragraphs, and our classification of language tags into types does not allow for this.

One possible solution to this problem is to allow the specification of a local PAM for each type of special region. This requires:

- a method to set the starting-language for the region;
- the specification of a local PAM for the region.

The disadvantage of this solution is its inherent complexity: for each special region the designer of a document class needs to specify a full mapping of all language-related actions to the tag hierarchy (the local PAM). Since the numbers of both the special regions and the language-related actions are potentially unlimited, this would result in either a very complex set-up mechanism or the use of general defaults (e.g., the local PAM nearly always inherits from the global document PAM) in which case the solution is unnecessarily complicated.

A practical solution

A simpler solution is to use the PAM from the main document but to allow the specification, for each type of special region, of how the information from the PAM is used. This would be done by specifying the following:

- a method to set the starting-language for the region;
- the actual initialisation-level (init-level) for the change to this starting language;
- the effective level (inner-level), as far as imbedded tags are concerned, of this change to the starting-language for the region.

We now give an expanded description of these items.

Starting language In the case of special regions that receive unformatted text the starting-language will directly affect only the text generated by the region's tags themselves as each bit of received text will carry its own language label (see the section entitled "Special Regions"). In the case of regions that move after being formatted it defines the default language used when formatting this region.

Initialization At the start of the region, actions are executed as if the region started with a tag whose level (in the THD, i.e., a pair of nodes) is init-level using this starting-language. This results in setting parameters to values suitable for that starting-language whilst allowing for a region to move to a different visual context.

[5] Such details can have large effects on the efficiency of the implementation, thus we are being cautious here.

Inner processing Within the region, language tags are processed as if the region started with a tag whose level (in the THD) is inner-level (inner-level must be at least as deep[6] as init-level in the THD). This allows finer control over the subset of actions executed at imbedded language tags.

Interfaces for the Rendering Model

The following interfaces will be provided for use by writers of class and package files:

- specifying the THD (this will probably be fixed, at least in the first version);
- specifying entries in the PAM;
- specifying entries in the LAT;
- specifying explicitly that a language-command (i.e., parameter-group) will potentially be used by the current package or class;[7]
- specifying the starting-language and init/inner levels for special regions;
- handling language information for moving text.

In addition to the new commands and environments outlined in the section entitled "Document interfaces", the following interfaces will be provided for use in documents (the first two must be in the preamble):

- specifying the document-language;
- specifying all the languages used in a document;
- possibly an interface for overwriting the starting language of a particular special region

The second item above is not strictly necessary as the information can be obtained by processing the document; however, a large saving of time and space can be made if the full list of languages actually used is specified in the preamble.

[6] An alternative model would be to also allow inner-level to be one less than init-level. This would mean that language tags within the special region are acting as language changes on the same level as the starting language of the region.

[7] These declarations allow the local customizations for all language actions to be stored in one place (e.g., PAM or LAT modifications); the system can then select only those that are actually needed for the current document.