

In and Out of Tags

SESSION

ORGANIZER: Claus Huitfeldt

AUTHORS:

Peter Cripps
Claus Huitfeldt
Maria Sollohub

The Wittgenstein Archives at the University of Bergen, Harald Haarfagres gt 31, N-5007 Bergen, Norway

KEYWORDS: text encoding, SGML, MECS

AFFILIATION: University of Bergen

E-MAIL: Claus.Huitfeldt@hd.uib.no
FAX NUMBER: +47-55-58 94 70
PHONE NUMBER: +47-55-58 29 50

The Wittgenstein Archives at the University of Bergen organises a session (90 minutes) to present some experience and new results of its work which it believes will be of broad interest for the text encoding community in general. The session will consist of three papers:

- «Quality Assurance In Between Tags»; Maria Sollohub
- «Attributes: A Problem», Claus Huitfeldt
- «Attributes: A Solution», Peter Cripps

The first paper starts by pointing out that whereas in discussions of text encoding much attention tends to be placed on encoding techniques, code structures and syntax, the task of ensuring the quality of text material contained between code tags is of equal importance for projects making use of text encoding. For example there is a need to spell-check multi-lingual encoded documents while remaining in primary format, to be able to check that particular elements satisfy requirements concerning format or use of specific terminology etc. Using the experience of The Wittgenstein Archives over the last five years as a point of reference, typical problems are described and solutions are suggested. The solutions presented are implemented in the Archives' own encoding scheme, MECS, but need not be MECS-specific. The issues raised are of relevance to all text encoding projects.

The second paper first sets out to understand text structures in terms of relationships between textual objects and their properties. A decision to encode certain properties with generic identifiers and

others with attributes implies a distinction between primary and secondary properties which is often useful, but which may also sometimes have undesirable consequences. Finally, certain kinds of properties are identified which seem to require attributes, yet cannot be represented properly with SGML's attribute mechanisms.

The third paper sketches a new and more flexible syntax for the handling of attributes. This mechanism allows for attributes to qualify not only generic identifiers but also other attributes. It distinguishes between two types of attributes, open and closed, and two kinds of attribution, weak and strong. The open/closed distinction belongs to an attribute's definition, whereas the strong/weak distinction refers to two ways in which an attribute may be applied.

Quality Assurance In Between Tags

Maria Sollohub

The Wittgenstein Archives at the University of Bergen, Harald Haarfagres gt 31, N-5007 Bergen, Norway

KEYWORDS: SGML, MECS, quality assurance

AFFILIATION: University of Bergen

E-MAIL: Maria.Sollohub@hd.uib.no
FAX NUMBER: +47-55-58 94 70
PHONE NUMBER: +47-55-58 29 50

Introduction

In discussions on text encoding and projects employing text encoding techniques, much emphasis tends to be placed on code structures and syntax. But what about the text in between the tags? How can we check the quality of the encoded text? This paper will attempt to identify some of the problems raised by this question, describing solutions and suggesting extensions to these solutions.

The solutions presented here are implemented in MECS (Multi-Element Code System), the encoding scheme developed at and employed in the work of The Wittgenstein Archives at the University of Bergen. The ISO standard, SGML (Standard Generalized Markup Language), is compatible with MECS. In terms of the software solutions illustrated here, however, it is important to be aware that they do not have to be MECS-specific. By following the same principles, similar solutions could be and have partly been found in other encoding systems.

Quality problems in between the tags

The Wittgenstein Archives at the University of Bergen was established in 1990 and thus has over five years of text encoding experience. This has emphasised for us the importance not only of adopting an appropriate encoding scheme, but also of just how essential it is to be able to control the quality of the text between the code tags.

The material being transcribed by The Wittgenstein Archives consists of 20 000 pages of manuscripts and typescripts. Wittgenstein's habit of continuously revising and rearranging his manuscripts means that they are characterised by different types of deletions, insertions, remarks in the margins, and cross-references, often creating several alternative formulations of a single expression. Neither is it always clear which of these formulations he has finally decided upon. In terms of transcription, The Wittgenstein Archives' basic minimal requirement is to be able to produce both a strictly diplomatic and a normalised/simplified reading version of each and every individual manuscript. We thus need methods of checking that this requirement is being fulfilled.

Of course, all text encoding projects deal with different source material and will be faced with different problems and needs. Our point here is to emphasise that one should not overlook the importance of being able to control the quality of the material in between the tags, in addition to being able to check the syntax and structure of the encoding.

The following are typical problems encountered by text encoding projects:

- the need to spell-check encoded documents while remaining in primary format (thereby retaining references in the encoded source transcription)
- the need to be able to check that particular elements satisfy particular conditions (e.g. language requirements, dating formats, use of specific terminology etc.)

Examples and Solutions

Consider the following example:

```
<s>Ich war mit <name>Heinz</name> im
Theater</s>
<s><name><k>H</k>einz</name> hat gesagt:
<q><c>I</c>ch war es nicht, ich <inc>hab
</inc>...</q></s>
<s><q>Was hast Du?</q></s>1
```

Running this piece of text through a normal spell-checker will not give sensible results. At The Wittgenstein Archives we use a profile which can filter an encoded transcription to produce a complete list of graphwords together with line and column references (LRef. and CRef.). The above

example will thus produce the following list:

LRef.	CRef.	ich
LRef.	CRef.	war
LRef.	CRef.	mit
LRef.	CRef.	Heinz
LRef.	CRef.	im
LRef.	CRef.	Theater
LRef.	CRef.	Heinz
LRef.	CRef.	hat
LRef.	CRef.	gesagt
LRef.	CRef.	ich
LRef.	CRef.	war
LRef.	CRef.	es
LRef.	CRef.	nicht
LRef.	CRef.	ich
LRef.	CRef.	was
LRef.	CRef.	hast
LRef.	CRef.	Du

The profile works by enforcing a specific behaviour for the contents of each and every code in the code scheme. It will be apparent, for example, that “hab” from <inc>hab</inc> does not feature in the list of graphwords. This is because the contents of an <inc>..*</inc>* code is defined as incomplete and therefore will not be counted as a graphword. It is also interesting to note what happens to the contents of the <c>..*</c>* and <k>..*</k>* codes in this filtering process. In order to fully understand what happens here, it is necessary to be aware of the fact that the filtering process automatically changes the case of the first character of every sentence, in order to counter capitalisation of words whose standard form is in lower case². Thus, for each occurrence of <c>..*</c>* and <k>..*</k>* the case of the letter concerned is either changed (<c>) or kept (<k>) in order to send an appropriately normalised version of the resulting word to the word list.

We may, however, have a more specialised interest in the contents of a particular code and its occurrences in the course of a transcription. By altering the filtering profile very slightly, we can define a filter which will produce a word list of the contents of all occurrences of a single code. In the above example, we could extract the contents of the <name> code in this way, with the following result:

LRef.	CRef.	Heinz
LRef.	CRef.	Heinz

Lists such as the two above can be checked for errors in a wide variety of ways – manually, automatically or both. A commercial spell-checker which ignored numerals (i.e. would not be confused by the line and column references) could be used in the case of the first list.

The triviality of the above example should not mask the potential of these tools, e.g. if we imagine more complicated examples – more complex encoding, a larger volume of source material, several transcribers, transcription work continuing over a long period of time (years). In our work at The Wittgenstein Archives it has been proven to us again and again just how vital it is to be able to check the individual graphwords of a transcription. Many graphwords are “corrupted” by inserted code tags and it is surprising how easy it is for the existence of such code tags to make it difficult to recognise misspellings of words. There is also a need to check the standard application of encoding conventions with respect to specific codes. Here too, different transcribers (especially over a long period of time) can easily deviate from project-specific encoding conventions. Below are some specific examples of “quality” control processes in use at The Wittgenstein Archives in Bergen:

[These examples will be supplemented with extracts from transcription work at the Archives in the form of visual aids during a presentation of this paper.]

I. Vocabulary control and spell checking

Wittgenstein uses the languages German, English, French and Latin, and has a wide repertoire of Wittgenstein-specific vocabulary. The checking process involves the production of separate lists for each of these languages. Rather than rely on a commercial spell-checker, we have built up master lists in each of the four languages against which we check our language lists from each individual transcription, adding to them any new acceptable graphwords that result along the way. The checking process (in the form of a single piece of software) functions as follows:

- produce list of German graphwords
- compare list against master list of German words
- produce list of “new” words
- ask the transcriber to check new words (references provided), adjusting transcription for genuine mistakes
- produce new list of “new” words
- ask transcriber to repeat checking until all “new” words are acceptable

- repeat process for English words
- repeat process for French words
- repeat process for Latin words

- send all lists of “new” words to “word list manager” who does a final acceptability control and adds new words to master lists.

II. Extraction of contents of individual codes such as “person” and “dating” codes

This is particularly relevant for a large volume of material transcribed over a long period of time, where it is necessary to check the consistency of transcription conventions. For example, The Wittgenstein Archives’ encoding system expects every occurrence of a person name to be encoded in two parts, where the first part contains the name as it appears in the source material, and the second part contains a standardised form of the same name. But who decides what the standardised form of the name is? “Skolem, Thoralf Albert” or “Skolem, Thoralf” or ? One of the points of having such a code is to facilitate indexing at a later date, and it is therefore essential that only one of the many alternatives is adopted as the standard. Extracting a complete list of person names from transcriptions will give a very good means of controlling such convention consistency.

The dating code presents a similar problem and is easy to transcribe incorrectly. Whereas wrongly coded names may also be checked against master lists in the vocabulary control process, dates are not so easily checked. The extraction process is a quick and effective method of checking consistency within specific code tags in a large volume of source material, giving reference indications directly linked to the encoded document.

The language division into German, English, French and Latin is particularly useful for vocabulary control at The Wittgenstein Archives, but other divisions are just as plausible – and for other projects perhaps more useful. A need that has been brought to our attention by The Norwegian Term Bank in their work on terminology is that of being able to check the use of specific vocabulary within a particular SGML code. E.g., given that there is a code <definition>...</definition>, used for every entry in an extensive reference work (perhaps more than one person responsible for encoding), how can we control the use of terminology within that code? The processes described above could provide a solution to this problem.

One step further – some possible extensions

The Wittgenstein Archives employs a number of tools and methods in order to achieve quality “in between tags”, but there are still areas where we would like even more control. Could we find a way of defining acceptable formats for particular codes, such that a transcriber trying to use a diffe-

rent format will receive an error message? This would be useful, for example, for defining codes that should contain *solely* numerical data, *no* numerical data, or dates in only *one acceptable format*.

[*More specific examples of what we would like to achieve in this direction will be presented at the conference.*]

Conclusion

In the essential document analysis stage prior to a new text encoding project, it is important not to think merely in terms of which encoding scheme and what kind of code tags are required, but to spend some time considering what type of control mechanisms will be necessary in order to check the quality of the encoded documents. What problems will the material pose in terms of quality control? Are there tools to deal with them? Can the necessary tools be purchased or developed? At The Wittgenstein Archives we have learned from experience. In order to ensure quality transcriptions of 20,000 pages, we use an array of control tools, checking both syntax and code contents (vocabulary and encoding conventions). The MECS tools described here can also be used on SGML encoded documents, but there is nothing to prevent similar "SGML" tools being developed along the same principles. The main concern should be that such tools exist and continue to develop in order to cater for the needs of a growing number of text encoding projects.

Notes

- 1 The codes used in this example are hypothetical, but are based on those used at The Wittgenstein Archives: <s> = sentence, <name> = name, <k> = keep case, <q> = quote, <c> = change case, <inc> = incomplete.
- 2 In German, all nouns (not just proper nouns) are capitalised in their standard form. It is also acceptable to capitalise the personal pronoun "Du" (you). Thus, some care has to be taken in the use and application of the "change case" and "keep case" rules.

References

- ISO: "Information Processing – Text and Office Systems Standard Generalized Markup Language (SGML)", International Organization for Standardization, ISO 8879-1986, Geneva 1986.
- Claus Huitfeldt: "MECS – A Multi-Element Code System", forthcoming in *Working Papers from The Wittgenstein Archives at the University of Bergen*, 1995.

Claus Huitfeldt and Ole Letnes: "Encoding Wittgenstein". Paper read at the joint ACH-ALLC Conference in Washington D.C., June 1993. Printed in *Conference Abstracts*, The Center for Text & Technology of the Academic Computer Center, Georgetown University, June 1993, pp 83-85.

Claus Huitfeldt: "Manuscript Encoding: Alphatexts and Betatexts". Paper read at the joint ACH-ALLC Conference in Washington D.C., June 1993. Printed in *Conference Abstracts*, The Center for Text & Technology of the Academic Computer Center, Georgetown University, June 1993, pp 85-88.

Attributes: A Problem

Claus Huitfeldt

The Wittgenstein Archives at the University of Bergen, Harald Haarfagres gt 31, N-5007 Bergen, Norway

KEYWORDS: text encoding, SGML, MECS

AFFILIATION: University of Bergen

E-MAIL: Claus.Huitfeldt@hd.uib.no

FAX NUMBER: +47-55-58 94 70

PHONE NUMBER: +47-55-58 29 50

1. Text Objects and Their Properties

Letters of the alphabet, numbers, punctuation marks and a few other conventional signs are basic constituents of any text written in an alphabetical writing system. Any computing system which is capable of representing strings of alphanumeric characters is therefore also capable of representing at least the most basic linguistic contents of texts written in such writing systems.

However, looking back on a long tradition of manuscripts and printed books, we are definitely not prepared to admit that there is nothing more to texts than that. Not only can written texts contain graphical elements such as drawings etc., but the page layout, typography and graphical design may also play a crucial role in identifying, emphasising and increasing readability of parts of a text and conveying structural relationships between them. Since a computer text file *is* in a certain sense simply a long string of characters, it is by marking them up with reserved character combinations that text processing systems let us represent such properties and structures. Text encoding systems such as SGML are an attempt to systematize, generalize and standardize such markup.

The marks or *tags* serve to identify specific parts

or *elements* of the text and to ascribe specific properties to these elements. In a text encoded in accordance with the TEI's Guidelines for SGML encoding, e.g., we will frequently find structures such as:

```
... <emph> ... </emph> ...
```

The start tag and the end tag (i.e., the reserved character combinations '<...>' and '</...>') indicate the start and end of an element and ascribe to this element a property indicated by the *generic identifier* (the GI, in this case 'emph'). The TEI Guidelines tell us that this particular GI indicates the property 'emphasized', which 'marks words or phrases which are stressed or emphasized for linguistic or rhetorical effect' (TEI P3, p 955).

Broadly speaking, SGML-encoded documents consist entirely of such elements, which may be nested within other elements (cf. the OHCO-model of texts in DeRose et al). We could therefore also say that an SGML document is an ordered sequence of characters and markup, the markup ascribing certain properties to parts of the sequence and indicating certain relationships between these parts.

This seems to invite a rather clear-cut conception of texts as collections of *objects with properties*: The characters are so to speak the basic building blocks or elementary particles which cannot be decomposed any further, they are the smallest possible objects out of which higher-level objects, elements, are built. An object is either a character string or an element, and properties can be ascribed to such objects by GIs.

2. Attributes

It may be that one and the same object has more than one property, or that we want to classify or qualify an ascribed property further. SGML allows us to express such features by means of *attributes*:

```
... <foreign lang=fr> ... </foreign> ...
```

In this case, the GI 'foreign' ascribes the property of 'belonging to some language other than that of the surrounding text' (TEI P3, p 981) to the element, while the attribute 'lang' identifies this language more specifically as being French (indicated by the attribute value 'fr'). In other cases, the attribute may supply further information about the same element:

```
... <foreign lang=fr rend=italics> ... </foreign> ...
```

The value 'italics' of the attribute 'rend' (for 'rendition') does not provide a further classification or qualification of the language in question, but indi-

cates that the element was or should be printed in italics.

Attributes can be useful since they allow us to express complex structures in a regular way, allowing for various sorts of processing depending on the purpose at hand:

```
<foreign lang=fr rend=italics> ... </foreign>
<emph rend=italics> ... </emph>
<name type=person reg='Smith, John'
rend=bold> ... </name>
```

SGML also lets us enforce rules e.g. to the effect that the attribute 'reg' is required on the GI 'name' but not allowed on the GIs 'emph' and 'foreign', that the attribute 'rend' is allowed on all GIs and required on 'emph', etc. The SGML attribute mechanism thus gives us a very strong tool to describe textual structures.

In practice one will usually design an SGML encoding system so that what are perceived as *primary* properties are represented by GIs, whereas attributes either *qualify* or *classify* these primary properties or add *secondary* attributes to those ascribed by the GI.

What is considered primary and secondary will vary from context to context. What for certain purposes may be encoded like this:

```
<foreign rend=italics> ... </foreign>
<emph rend=italics> ... </emph>
<name rend=bold> ... </name>
```

may for other purposes more suitably be encoded like this:

```
<italics type=foreign> ... </italics>
<italics type=emph> ... </italics>
<bold type=name> ... </bold>
```

It is sometimes said that the choice of whether to represent a certain property as a GI or as an attribute is a matter of taste and style. But while an element can have several attributes it can only have one GI. This may lead to problems in cases where one and the same element has properties which are both indicated by GIs.

Assume e.g. that one has chosen to represent emphasized phrases and names as GIs with attributes as illustrated above, and we encounter an emphasised name printed in bold italics. Either one must add a new attribute to the system, indicating one of the properties which would normally have been represented as a GI, e.g. like this:

```
<name type=person reg='Smith, John'
rend='bold italics' mode=emph>
... </name>
```

(The sole purpose of the attribute 'mode' is to carry the value of what would otherwise have been a GI.) Or one must nest two elements with the relevant GIs in question inside each other, and either duplicate common attributes or decide which of the two elements should carry them, e.g. like this:

```
<name type=person reg='Smith, John'  
  rend=bold>  
<emph rend=italics>  
  ... </emph></name>
```

Both cases seem to leave room for some slack or even inconsistency in encoding practice, and they mean that the same phenomena will be encoded by different mechanisms or in different ways from case to case.

The latter case also raises the question which should be the outer and which the inner of the two elements, leaving additional room for slack and inconsistency.

3. Encoding Without Attributes

Among the aims of the Wittgenstein Archives at the University of Bergen is to transcribe the (mostly unpublished) 20,000-page manuscript Nachlass of the Austrian philosopher Ludwig Wittgenstein. The encoding system used in this project is based on MECS (cf. Huitfeldt 1993 and 1995), which in all respects relevant for the present discussion is identical to SGML.

In this project, we decided not to make use of attributes at all.

Instead, a separate GI was introduced for every possible combination of properties, i.e. for what would otherwise have been represented as a combination of GIs and attributes. One of the reasons for this decision was that it was rather difficult to decide which were to count as primary and which as secondary properties of the texts.

E.g., Wittgenstein frequently marks parts of his texts with underlining. There are several different types of underlining, – such as straight, wavy, dotted and broken lines, underlinings with one, two or several lines. We know that Wittgenstein had his personal conventions for such markings in the manuscripts, and that the different kinds of underlining have different meanings. We know e.g. that a straight line means emphasis and that wavy lines in general indicate dissatisfaction with content or formulation, but we do not know the exact meaning of all these conventions. And although we do know that Wittgenstein indicated emphasis and dissatisfaction also by other means, a lot of uncertainty of interpretation usually pertains to these other occurrences.

Therefore, we limit our interpretation of the text

to identifying the convention used, – we do not take the further step of interpreting what the convention in each individual case stands for, – i.e. we indicate the underlinings, not their meaning.

In SGML, we might have encoded all these properties with one GI and two attributes, e.g.

```
<u shape=s number=1> for 1 straight line,  
<u shape=s number=2> for 2 straight lines,  
<u shape=w number=1> for 1 wavy line,  
etc.
```

Instead, we encode like this:

```
<us1> for 1 straight line,  
<us2> for 2 straight lines,  
<uw1> for 1 wavy line,  
etc.
```

The number of possible combinations of such properties is considerable but limited, – the number of GIs we have to handle becomes larger than the number we would have had to deal with had we used a system with attributes – but it is manageable. The Wittgenstein Archives encoded texts for several years in this manner, and everything seemed to function well.

4. New problems

However, after a while we proceeded to a part of the Wittgenstein Nachlass which *did* cause us problems. Some texts have been edited by several different individuals or by Wittgenstein himself at different times, i.e. they are written in different "hands". Text originally written in one hand has sometimes been subject to cancellation, modification or addition by a later hand, which may in turn have been subject to alteration by a yet later hand, etc.

E.g. a word originally written in one hand (by Wittgenstein himself) may be underlined by a later hand (e.g. his colleague Russell), the underlining may have been cancelled by a third hand (e.g. Ramsey) and the cancellation cancelled by a fourth hand (e.g. Wittgenstein himself, thus dismissing Ramsey and agreeing with Russell). The second hand, which cancelled the underlining may also have deleted (i.e. cancelled) the word itself, – and again this cancellation may have been lifted by a later hand, and so on.

The introduction of new GIs in order to cover all combinations of these parameters throughout 20,000 pages became a rather hopeless business in the long run.

It is worth noticing that the complexity which threatened to break our system down was *not* the number of properties involved (in fact, only two properties, 'hand' and 'cancelled', were involved). Neither was it the number of values that these

properties could have ('cancelled' has only two values, and the number of hands in Wittgenstein's manuscripts is not very large). Nor was the number of different GIs which could have these properties overwhelming.

What was special about these properties was that they could not only be properties of a text element, but also properties of properties, properties of properties of properties, and so on: A word can be cancelled (deleted), the cancellation can be cancelled, the cancellation of the cancellation can be cancelled – and on each of these levels the cancellation in question can have the property of being made in a different hand.

5. Multilevel Attribution

Although the question as to whether properties should be represented in the form of GIs or attributes is mostly a practical one, we have now found a criterion to identify certain properties that *cannot* be represented with GIs only.

What is characteristic about these properties is that they can occur at any level, that they can be used recursively, and that there is in principle no limit to the number of levels at which they can apply. Since the difference between MECS and SGML is that whereas MECS has no attributes SGML has, one would think that for the Wittgenstein Archives the above was a strong argument in favour of SGML. However, it turns out to be just as difficult to take the above factors into account in SGML as in MECS. (The TEI's encoding of certainty and responsibility (TEI P3, p 521-528) and its use of feature structure mechanisms (TEI P3, p 475-519) are examples of solutions to similar problems in SGML.)

As mentioned earlier, an SGML attribute qualifies an element or its GI, not the other attributes of the same element. I.e., we can of course design special attributes for each of the levels involved on an ad hoc basis, but these will be entirely dependent upon some specific application for their correct interpretation.

6. Conclusion

Multi-level properties, i.e. properties which can be properties of properties, and which can occur at any level of attribution, cannot be represented by GIs and must be represented by some kind of attribute mechanism.

However, such an attribute mechanism must have capabilities we do not find in SGML, i.e certain attributes must be attributable of elements as well as of other attributes, and this attribution must be able to occur unrestricted at any level of recursion.

References

- ISO: "Information Processing – Text and Office Systems Standard Generalized Markup Language (SGML)", International Organization for Standardization, ISO 8879-1986, Geneva 1986.
- C.M. Sperberg-McQueen and Lou Burnard (eds.): "Guidelines for the Encoding and Interchange of Machine-Readable Texts (TEI P3)", Chicago and Oxford April 1994.
- Claus Huitfeldt: "MECS – A MultiElement Code System", forthcoming in Working Papers from the Wittgenstein Archives at the University of Bergen, 1995.
- Claus Huitfeldt: "MECS-WIT – A Registration Standard for the Wittgenstein Archives at the University of Bergen", forthcoming in Working Papers from the Wittgenstein Archives at the University of Bergen, 1995.
- DeRose, Durand, Mylonas, and Renear: "What is Text, Really?" in *Journal of Computing in Higher Education*, Winter 1990, Vol I (2), p 3-26.

Attributes: A Solution

Peter Cripps

The Wittgenstein Archives at the University of Bergen, Harald Haarfagresgt. 31, N-5007 Bergen, Norway

KEYWORDS: text encoding, SGML, MECS

AFFILIATION: The Wittgenstein Archives at the University of Bergen

E-MAIL: Peter.Cripps@hd.uib.no

FAX NUMBER: +47-55 58 94 70

PHONE NUMBER: +47-55 58 94 73

1. The problem restated

The principle objective of The Wittgenstein Archives at Bergen University is to publish the literary estate – the Nachlass – of the Austrian philosopher Ludwig Wittgenstein in machine readable form. This Nachlass comprises some 20,000 pages of which roughly 65% are manuscripts and the remainder typescripts. The greater part of the handwritten material, as well as much of the typed, is replete with later alterations, deletions, insertions, rearrangements, cross-references and the like, the sum of which constitutes a formidable challenge to text encoding.

Some of the textual features we wish to record in our transcription work invite the use of a markup language which can describe properties on more than one level.

For instance, a word or words constituting a later addition to a text might be inserted either above or below the original line of writing; such an insertion may or may not include a marking to indicate that it is confirmed as an addition to the main text; such a confirmation marking, if present, might in turn be modified by means of a wavy underlining to indicate that the addition has subsequently been called into doubt, and this wavy underlining might itself be crossed through, thereby effectively reaffirming what was formerly doubted.

And still this isn't the whole story. In addition to the different markings we can often distinguish that each step in the process makes use of a different ink, and this information needs to be recorded independently of what that ink was used for. I.e. we can never assume a consistent correlation between, say, confirmation markings and blue ink, or doubt markings and red ink.

What we have here is a text feature with one essential property – its being a later addition to the base text – and a number of secondary properties, such as position (above or below the line) and degree of authorial approval (“marked” = confirmed, “marked with added wavy underlining” = doubted, “marked with added wavy underlining, wavy underlining cancelled” = reaffirmed).

In Standard Generalized Markup Language (SGML) secondary properties such as an insertion's position, its degree of authorial approval (status) and the type of ink it was written with would commonly be handled by means of attributes. But when we consider the problem of Wittgenstein's insertions closely it becomes clear that SGML attributes are not capable of describing such features in all their complexity.

Suppose in the above case that we have a single SGML type generic identifier (GI) to describe the primary property of a string's being inserted and that all the secondary properties are to be accounted for by means of attributes. The insertion's position is easy enough to deal with, since for any one insertion this will be invariable. Thus we might have a tag which looks like this:

```
<insertion position="above line"> ... </insertion>
```

Neither should the confirmation marking be a problem so long as it isn't modified by anything else:

```
<insertion position="above line"
  marking=confirmation> ... </insertion>
```

But what should we do if the confirmation marking has subsequently been modified by a wavy underlining to indicate doubt? The only way to quote an attribute in SGML is to list it in the open-tag of an element along with any others that

might be relevant. Both “marking=confirmation” and “marking=doubt” appear to be relevant in our example. Yet the insertion as a whole cannot be both confirmed *and* doubted!

And how should we represent the use of different inks? We cannot simply add an attribute “ink=red” if only the confirmation marking is in red ink whereas the text of the insertion is in black.

What we seem to be dealing with is properties of properties: the confirmation marking represents a property of the insertion as a whole, whereas the doubt marking annuls the confirmation; the fact of being written in black ink is a property of the insertion as a whole, whereas that of being written in red describes the confirmation marking alone. And so on. What we need here is a system whereby attributes can qualify each other as well as the GIs that describe the element's primary property.

2. MECSA – a more flexible attribute syntax

MECSA is an attribute syntax currently being developed as an adjunct to the Multi-Element Code System (MECS), the markup language used at the Wittgenstein Archives in Bergen. Well formed MECS texts are convertible to and derivable from SGML texts, and it is envisaged that MECSA attributes will likewise be translatable to and from SGML attributes.

MECSA incorporates a number of innovations, the most significant of which is that it permits attributes to qualify not only GIs but also other attributes. This it does simply by allowing for bracketing. For example, if an attribute “att2” describes a property of an attribute “att1” (rather than an immediate property of the feature described by the GI) then one can express the relationship in MECSA by quoting “att2” in brackets¹ *immediately after* “att1”, thus:

```
<GI att1=val1(att2=val2)> ... </GI>
```

Suppose further that “att2” requires the qualification of an attribute “att3”. This we would express by appending “att3” as a parenthesis to “att2”, thus:

```
<GI att1=val1(att2=val(att3=val3))> ... </GI>
```

And so on.

When two or more attributes apply to the same object (either the GI or another attribute) they are simply listed concurrently within the relevant bracket:

```
<GI att1=val1 att2=val2(att3=val3(att4=val4)
att5=val5)> ... </GI>
```

In this case “att1” and “att2” apply directly to

“GI”, “att3” and “att5” apply to “att2”, and “att4” applies to “att3”.

MECSA makes it appropriate to talk of attributes occurring on different levels. In the above example “att1” and “att2” are on the highest level and could be called primary – or first-level – attributes, while “att3”, “att4” and “att5” are sub-attributes (on lower levels), whereof “att3” and “att5” are second-level attributes and “att4” is on a third-level. In this way we could say that SGML provides a single level attribute syntax, whereas that of MECSA is multi-level.

The possible uses for such a system are numerous. In a particular application one might choose to ignore everything but the primary attributes, or alternatively to “work out” the brackets, beginning with the most deeply nested and progressing outwards, before putting the primary attribute(s) to their task(s). This will be clearer if we consider a practical example.

Let us imagine some MECSA attributes to describe the different properties associated with Wittgenstein’s insertions as these are outlined above. The different kinds of markings, which serve to confirm or doubt an insertion, can be accounted for by an attribute called “marking” which takes the values “confirmation” (for the initial insertion marking) or “doubt” (for the wavy underlining indicative of doubt). The attribute “ink” takes one of the values “black”, “blue” or “red”. Allowing that the “marking” attribute can be applied to itself and the “ink” attribute to both the GI and the “marking” attribute, we might then tag a particular insertion thus:

```
<insertion ink=black marking=confirmation
  (ink=blue marking=doubt(ink=red))> ...
</insertion>
```

Suppose now that we are interested in Wittgenstein’s text as it looked after he had reworked it in blue ink. At that stage he was evidently in approval of the inserted material, and consequently it should be included in the text we retrieve. In other contexts the attribute “marking=doubt” might well be used to suppress the effect of “marking=confirmation”. But by defining “ink=red” such that it first suppresses the effect of “marking=doubt”, we can leave the attribute “marking=confirmation” to function uninhibited. In this way we suppress what was in itself a suppressor.

On another occasion we might wish to view the text as it looked after the first pass. To achieve this we can use the “ink=blue” attribute to suppress any effect the “marking=confirmation” attribute might have. And so on.

It is not difficult to imagine further applications for such a system.

3. Parsing MECSA attributes

In MECSA, details about the legal combinations of attributes and GIs are recorded in an Attribute Definition Table (ADT), which in most respects serves the same purposes as the ATTLIST declarations in an SGML DTD.

One of the functions of the SGML ATTLIST, or – in the case of MECSA – the ADT, is to specify the value for a legally applicable attribute in the case that none is made explicit in a particular document. In SGML this is a straightforward process. If the GI in a particular tag lacks a legal attribute, then a parser can supply that attribute together with its default value in accordance with the appropriate ATTLIST. But since MECSA allows even one and the same attribute to apply to itself as well as to a GI an ADT cannot be allowed to handle the ascription of attributes to attributes in the same way as it handles the ascription of attributes to GIs. The danger we have to guard against is, of course, the possibility of an infinite regress. It should be obvious what would happen if a parser were asked to render explicit all the legal attributes in a system where “att1” can apply to “att1”!

MECSA avoids this danger by allowing for the supplementation (extension) only of first-level attributes. If the ADT specifies that “att1” can legally be applied to “GI1” then a parser can, on encountering a tag where “GI1” lacks “att1”, supply the attribute plus its default value. But although the ADT might specify that “att1” can legally be applied to “att1”, a MECSA parser will not supply a lower level occurrence of “att1” where one is already present. The ADT information that “att1” can be applied to “att1” is available for the purpose of checking whether the attributes which are already explicit in a document are legal, not for the purpose of supplying default values where they are absent.

In an SGML ATTLIST the legal values for each attribute are listed together with the legal attributes for the particular GI. But in the MECSA ADT the information about the values an attribute may take is handled separately from the information about the range of attributes which may legally be ascribed to a particular attribute object (GI or other higher level attribute). This separation is again a consequence of the fact that MECSA allows attributes to qualify attributes. (It would not be feasible to express this possibility using the structure of the SGML ATTLIST.)

It is worth noting, however, that a MECSA parser does not necessarily presuppose the existence of an ADT. In the absence of an ADT a MECSA parser may still check the attribute syntax. But it may also, if desired, compile a minimal-ADT from the document itself. In doing so it notes which attributes are applied to which GIs, which

to other attributes, and which values are already attached to those attributes. This information is then arranged in the format of a normal ADT, such that this record can, if desired, be used as a measure of the correctness of further document instances. The one thing such a syntax-checker cannot do is deduce an attribute's default value. Instead, in compiling a minimal-ADT, it supplies a System Default Value (SDV) in the place where an explicit default value would stand in a full-ADT. This SDV is a reserved character which ensures that attributes remain functionally meaningless when inserted into documents automatically on the basis of a minimal-ADT. If and when a document containing such attributes is checked against a full-ADT, these SDVs can then be replaced by meaningful values.

4. Conclusion

Due to the nature of certain textual phenomena encountered in Wittgenstein's Nachlass, the need arose for an attribute syntax of greater descriptive resolution than that offered by SGML. MECSA is a system which promises to satisfy this need by allowing attributes to qualify not only GIs but also one another. In designing this system such that documents using its syntax can be converted to and obtained from SGML documents, it is hoped that MECSA will be of use also in contexts other than the work currently being done at the Wittgenstein Archives in Bergen.

Notes

1. Technically speaking, MECSA provides for a "sub-attribute open delimiter" and a "sub-attribute close delimiter", the functions of which could be assigned to any suitable characters. It is not compulsory about the characters "(" and ")".

Bibliography

- Charles F. Goldfarb, "The SGML Handbook", Oxford 1990.
- Claus Huitfeldt, "MECS – A Multi-Element Code System", Bergen 1992; Working Papers from the Wittgenstein Archives at the University of Bergen, 1995.
- C.M. Sperberg-McQueen & Lou Burnard (eds.), "Guidelines for Electronic Text Encoding and Interchange – TEI P3", Chicago/Oxford, 1994.