Characters are not simply names, nor documents trees

John Plaice[1)] and Chris Rowley[2)]

Abstract

In this position paper, we outline a model for documents in which the concepts of character, text and structure are not fixed, as in the current standard XML/Unicode model.

We begin with an analysis of the current model and show how the atomist and nominalist bases for this model can be situated within broader aspects of current theoretical computing. We then contrast these bases with developments in practical computing, and demonstrate that an alternative model is appropriate.

Our new model moves away from the view that characters are simply names with no meaning, atoms with no content: a text may be encoded at various levels of atomicity, as needed, and its structure may be much more complex than a tree.

1 Introduction

In this paper, we outline our first draft of a model for text and for documents that allows the latter to be perceived as dynamic, evolving entities that are sensitive to a changing environment. This model is much more general than the current standard model, based on Unicode [12] and XML [15], that s documents to be fixed trees whose leaves are linear sequences of mere names.

In our new model, we consider that the primary purpose of using computers to work with texts is not simply to create massive (often write-only) textual databases, but to be able to use computers to process these texts in innovative ways, either fully automatically, or semi-automatically with manual intervention. By changing the focus from storage in files to arbitrarily complex manipulation with a computer or network of computers, we can devise methods of working with text in which content and form, copy and markup, and logical and visual markup are not permanently kept separate.

The processing we are considering includes typesetting of complex documents, sophisticated grammar and spell-checking, and audio reproduction. We wish to develop a common text model that will support all of these, and more. The difficulty lies, of course, in that each of these can be arbitrarily complex, in different ways.

The need for this new model is elaborated through an analysis of the current model in Sections 2–4. The Unicode and XML standards have been jointly designed so that documents can be "understood" without making any reference to any context. Although XML was first presented some years ago with the announcements that finally we would be able to insert semantics in our documents, in practice only syntactic issues have been clarified.

The problems are not merely technical, in which a patch here or there might rectify the situation. Rather, both Unicode and XML have been designed with philosophical — ideological? — preconceptions of what a document should or must be, with significant repercussions on text processing.

By ignoring the context, every aspect of a document becomes an island, an isolated entity with no links or ties with the other isolated entities. This problem applies at all levels, from the entire document itself right down to the individual character in a text stream. This *separateness* is not an accident, it is the conscious basis for both standards.

It is ironic that as these standards become ubiquitous, even our smallest devices, such as mobile phones, are becoming powerful computers capable of sophisticated processing, Any document can be continually processed and reprocessed, on-demand, as needs change, and the document is viewed, presented, or analyzed in a different context.

Our new model, outlined in Sections 5–7, generalizes the current model through the systematic incorporation of context and the use of more flexible data structures.

[1)] School of Computer Science and Engineering
The University of New South Wales
UNSW SYDNEY NSW 2052
Australia

[2)] Faculty of Mathematics and Computing
Open University
1-11 Hawley Crescent
London, UK

2 Unicode/XML as Standard Model

We present the key principles of the two standards, starting with the XML design goals [15, §1.1]:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

To summarize, it was important for the XML designers to have a quickly prepared standard for documents that are easy to parse and to create, with no worries about efficiency, utility or compactness. Thus no special focus was given to text.

As for the documents themselves, their structure is given below [15, §2]:

> Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup. The logical and physical structures must nest properly....

Because of the nesting of logical and physical structures, an XML document is a tree with a single root.

For text, the XML standard adds [15, §2.1–2]:

> This specification, together with associated standards (Unicode and ISO/IEC 10646 for characters, Internet RFC 1766 for language identification tags, ISO 639 for language name codes, and ISO 3166 for country name codes), provides all the information necessary to understand XML Version 1.0 and construct computer programs to process it....
>
> Definition: A parsed entity contains text, a sequence of characters, which may represent markup or character data.
>
> Definition: A character is an atomic unit of text as specified by ISO/IEC 10646....
>
> Legal characters are tab, carriage return, line feed, and the legal characters of Unicode and ISO/IEC 10646.

The design principles for Unicode are given below [12, pp.15–6]:

> The Unicode Standard draws a distinction between characters and glyphs. Characters are the abstract representations of the smallest components of written language that have semantic value. They represent primarily, but not exclusively, the letters, punctuation, and other signs that constitute natural language text and technical notation. Characters are represented by code points that reside only in a memory representation, as strings in memory, or on disk. The Unicode Standard deals only with character codes.
>
> Glyphs represent the shapes that characters can have when they are rendered or displayed. In contrast to characters, glyphs appear on the screen or paper as particular representations of one or more characters. A repertoire of glyphs makes up a font. Glyph shape and methods of identifying and selecting glyphs are the responsibility of individual font vendors and of appropriate standards and are not part of the Unicode Standard.

> Various relationships may exist between character and glyph: a single glyph may correspond to a single character, or to a number of characters, or multiple glyphs may result in a single character.

Therefore, if we put together the text principles from XML and the Unicode principles, we read that the XML standard essentially hands all responsibility for text to Unicode; but Unicode deals only with character codes, with only a few ad hoc extensions to specify properties of these characters. It seems that there is as yet no standard to deal with text as anything more than a sequence of bytes, notwithstanding the hype.

3 Atomism: Emptying the Context

The whole XML exercise, and here we refer not simply to the overtly stated principles, but also to the actual practice, is that a document is created to be stand-alone. It might refer to external resources, using URIs, but the document itself has fixed structure and content. A document designer must ensure that a document includes all the information necessary for its use, regardless of the context in which it might be used. The document is self-contained.

The philosophical principle behind this approach is called *atomism*. The atomist view of the universe, commonly associated with the names Leucippus and Democritus (5th century B.C.E.) is that the cosmos is composed of indivisible atoms, *moving through empty space*. The opposite philosophical standpoint, called *plenism*, assumes that there is no empty space and the atoms, if they exist, are not indivisible.

This debate is best known in physics with respect to the nature of light. The atomist view, upheld by Newton, Einstein, Schrödinger, maintains that light is made up of particles. Throughout most of the twentieth century, this has been the standard model, where the particles are called photons. The plenist view is that there is a luminiferous aether and that light is a wave, a vibration of the aether. It was the standard model in the nineteenth century, upheld by Huyghens, Fresnel, Maxwell and Planck.

In computing, the atomist view corresponds to object-oriented programming, while plenism corresponds to intensional programming, in which a program is assumed to be adapt to an evolving multidimensional context [7].

With respect to documents, the atomist point of view is not new. It has been the dominant position since the development of the printing press under Gutenberg and the subsequent production of books sold as commodities. Peter Ramus (1515–1572) played a significant role in developing the view that a book should be self-contained, defining the basic precepts at the beginning, followed by the text and the subsequent conclusions. Both Marshall McLuhan and Walter Ong have written at length about the transformation from the age of manuscripts to the age of books.

That the atomist point of view prevailed with the development of printed books was perhaps an inevitable process. Certainly this is the point of view put forward by McLuhan in his *Gutenberg Galaxy*. If a book is passed from one reader to the next, it is the same book each time it is read (unless of course someone annotates it or tears the pages).

However, for a digital text, the process is different, because it is actively rendered *every* time that it is to be viewed. Since this is the case, one can possibly have the choice of having it rendered *differently* each time that it is to be read. In other words, we can imagine having the document presented differently as the context changes.

The work initiated by William Wadge in intensional documents [13] begins with this idea and goes much further. Here the very *structure* of the document can become context-sensitive. This idea has been applied experimentally by Manolis Gergatsoulis and Panos Rondogiannis to XML to produce Multidimensional XML [10]. More recently, the authors have contributed to adding these ideas to the Omega Typesetting System [6].

4 Nominalism: The Focus on Syntax

Related to atomism is a philosophical doctrine called *nominalism*, in which universals are rejected, leaving only names. The Unicode standard is clearly nominalist, as it states that "The Unicode Standard deals only with character codes", which are identified by specific names such as LATIN CAPITAL LETTER G or the even less useful CJK UNIFIED IDEOGRAPH 4E32. Although the Unicode book does give substantial information about the history of and relationships between many of these characters, this information is not part of the standard.

The Unicode approach has its utility, giving us a single standard in which one can type most of the world's characters. However, the nominalist approach *freezes* the world's writing systems at a particular point in time *from a single vantage point*.

As Yannis Haralambous [5], among others, has been pointing out, what exactly constitutes a glyph or a character is not a clear cut issue, and may well vary through time, language and culture, i.e., even characters may be context-dependent. And in cultures using Chinese characters, the actual set of characters is continually growing, as old characters are discovered and new ones are created.

This focus on names is consistent with the obsession of much of theoretical computer science with what are called *syntactic theories*, in which all of computing is limited to "following pointers". As an example, Robin Milner, Turing Award winner and inventor of the π-calculus, expresses himself in an interview with Martin Berger [2]:

> BERGER: The concept of names and naming is very important in your work. When did it occur to you that this is a fundamental notion in computing? That you have these points where you can interact, that you can hide...?
>
> MILNER: ...I think it was when we found out that you could encode data as processes with name passing.... you can get all the data, not by means of other kinds of objects, but as processes. And the way you access the data is via interaction and the interaction is via names.

The implications of such statements are that some of the most sophisticated programming found anywhere, such as the kernel of an operating system, where one must continually deal with race conditions, is mere detail, since it simply becomes interaction via names.

Even more extreme statements can be found in the world of mathematics, as exemplified by Martin Gardner, the longtime mathematics editor for *Scientific American* [4]:

> Mathematics is not only real, but it is the only reality. That is that entire universe is made of matter, obviously. And matter is made of particles. It's made of electrons and neutrons and protons. So the entire universe is made out of particles. Now what are the particles made out of? They're not made out of anything. The only thing you can say about the reality of an electron is to cite its mathematical properties. So there's a sense in which matter has completely dissolved and what is left is just a mathematical structure.

Even the material nature of the universe, with its limitless complexity, can be put into question!

The problem lies not in the use of abstractions, nor in the use of names, but in the supposition that abstractions are the only reality or are simple "zoom-outs" from more complex interactions happening at a lower level. Rather, they should be viewed as abbreviations, useful in some but not necessarily all situations.

5 Computers are for Computing

Implicit to all of these systems is the idea that computers do not actually *do* anything: they are just used to shuffle things around. The astonishing processing power of even the smallest computers of today seems to exist only to serve documents preprocessed elsewhere and downloaded from the network.

Yet, first-year courses in programming all focus on iteration and recursion, each of which is used to repeatedly manipulate a data structure where certain parameters have changed values.

It is this point that we consider to be the most important when trying to develop a new model for text. We should be concerned not simply about the format for files, but, more importantly, about the kinds of data structures and the kinds of manipulations that we wish to undertake on these data structures. As Niklaus Wirth wrote almost 20 years ago. "Algorithms + Data Structures = Programs" [14].

A new model for text should assume that a computerized text is a multi-faceted, dynamic data structure in which, through automatic processing and manual intervention, including editing, it can be transformed many times over into something new, annotated, or added to, without having to force everything into a fixed, atomized tree.

6 Incorporating the Context

Since this is a position paper, we will only outline the proposed model. In particular, we will not focus on syntactic issues, and will follow the advice given by Ashcroft and Wadge in [1] and give the *prescriptive semantics* that we wish the new model to follow. Exact operational and syntactic issues will be dealt with in future papers.

The simplified version of the model we are presenting in this section is a direct generalization of the existing XML/Unicode model. The important extra ingredient is that context has been incorporated at all levels. In other words, we can assume that we are still dealing with structures resembling trees whose leaves are sequences of "characters".

Thus we begin with the idea of a multidimensional context (see [7]). We then consider a document to be an *intension*, i.e., a mapping from contexts to *extensions*; for now, these extensions closely resemble current XML/Unicode documents. This practice, used in intensional programming, has its origins in the logical work of Carnap [3]. In logic, these contexts are called *possible worlds* and they correspond to a complete description of a system. In our model, the idea is that each distinct context specifies a complete document tree.

Since the set of possible contexts is unboundedly large, all of these possible document trees cannot be stored separately. Instead, it is the intension that is encoded, rather than the separate extensions. Thus this intension must, for each entity, encode how the extension of that entity varies with the context.

In practice this encoding of an intension would be done in either of two ways, or a mixture of the two. First, the extension-specification of an entity can vary parametrically according to the values of one or more dimensions of the context; this is the simple case. Second, a finite collection of possible specifications can be given and, for each specification, information is added to determine that the subset of contexts for which that specification should be used. We call each of these separate extension-specifications a *version* of that entity.

This approach has already been demonstrated to be useful in the development of Multidimensional XML. However, here we go further, in that we consider that the context affects not just the structure of a document (the XML part), but also the text itself (the "Unicode characters" and their interpretation). Already, in our other article in this volume [9], we have described the utility of this idea.

Our initial model of text is a still as a sequence of "characters", but now the context determines, for example, their atomicity so such aspects are now variable. For example, an English text can be encoded as Latin letters, or as English words — avoiding problems of national spelling variants — or some other structure encoding grammatical components. To keep track of what is being encoded, the text *must* also be part of the intensional paradigm.

The model goes even further, because even the properties of individual "characters" can depend on contextual information of arbitrary complexity. Such extensions of the text model allow the document itself to keep track of subtleties of character and glyph variants when visual appearance is crucial, as in encoding historical texts.

¿From a technical point of view, we assume that this part of the model will not be difficult to implement, as we will be able to take advantage of the infrastructure created by Paul Swoboda [11].

Note that the context for a document can be changed by many factors, including the actions of applications, such as formatters. The context can therefore change during the processing of a document and this is where much of the computational flexibility and power of this model will become apparent.

7 Moving Beyond the Tree

In this section we extend the model of document structure into something very different from that of XML. In this model, multidimensionality is crucial not just for the context, but for the document structure itself and all the associated data structures.

Our fundamental observation is that all but the most trivial documents really consist of multiple streams; these are encoded linearly in existing systems only for historical reasons. For example, a footnote does not "belong" to the text in which it is placed and it is certainly not a subset or substructure of that text. It is only for reasons of convenience when typing that it the footnote text may appear at a particular position in the source file; as far as the document is concerned, a footnote is a separate stream, together with a synchronization structure linking it to a specific position in the main text. This notion of multiple synchronized structures is natural for encoding a wide range of documents, such as commentaries, and it is amenable to building document structures in which one can annotate documents at will.

It is this intuition that we wish to retain for our general model: a document should consist of such synchronized parallel multidimensional structures and thus cannot be understood from a simplistic representation as branches of a tree.

But our model goes even further than this vision, because we observe further that the natural way to represent many elements of a complex document is based on the multidimensional array, as appears in spreadsheets. This structure allows us to encode tables of arbitrary complexity, as well as create an infrastructure for editing and formatting without neglecting their fundamental nature or their relationships to the rest of the document.

The full model also incorporates nested and ordered structures (i.e. the classical tree structures) but integrates them closely with the multidimensional arrays and the synchronized parallel structures. The types of these structures are very general but are definitely not arbitrary, nor are they computationally intractable. Mixed hierarchical and tabular structures (not arbitrary, but very well structured) are also becoming more common in data-based applications. A simple example: each of the rows in a table is decomposed into nested sub-rows or even nested sub-tables.

To summarize, such structures are a long way from the classical structures of computing, such as the relational database and the tree, but they are also far from being 'arbitrary relations'. They are needed because they appear quite naturally in documents, particularly as follows.

- In generalized tabular data, which is often naturally multi-dimensional with no preferred dimension.
- In documents with multiple, unrelated hierarchies.
- As multiple streams with 'synchronisation points'.

8 The Utility of Our Model

In this section we briefly describe a number of the many tasks in document processing that will take advantage of the new model. We are of course aware that good, innovative systems that are based on ideas developed to solve an apparently specialized problem often turn out to be productive in more main-stream applications. Thus it is likely that when our model is implemented and is shown to be well suited to text processing, then it will be equally useful for many other applications.

We begin with some simple examples, at the word or phrase level:

- Japanese text is known to need pronunciation information readily available. This should be integral to the text.
- For teaching applications this is probably even more important for English (which also has multiple pronunciations).
- Asynchronous textual communication needs a lot of enrichment: the equivalent of the non-textual bits of synchronous communication.
- 'Information matching' is not 'string matching' as the former needs information about: atomicity, equivalences and order relations on substrings and, for non-exact matching, topological relations on the 'information content' of the text. A very simple example: comparing a US text and a UK text.
- Many commonly occurring text strings have a 'meaning' that is more formal than that derived from natural language analysis. Some more obvious examples are times, dates and other measurements; less obvious are addresses, and names of people, places, etc. The formal structure and abstract meaning of these often needs to be explicitly encoded as part of the text. Neither of the two extreme current methods of doing this is sufficient: these are additional mark-up or run-time matching to regular expressions.
- All useful text could usefully be linked to dictionary or glossary entries; in some cases this

should be extended, for example, to concordance data and multiple dictionaries (e.g. learners', etymological, historical, technical).

- In specialized texts, such as teaching texts, engineering specifications or legal documents, words and phrases have precise meanings and relationships that need to be encoded as part of the text.

Even staying within visual presentation of text, a dynamic approach to atomicity and equivalences of substrings are needed to support the following:

- Glyph selection and positioning (typesetting words and punctuation).
- Relative positioning of glyphs (white-space, decorations, punctuation, italic corrections etc).
- Line- and page-breaking.
- Generated text and reuse of text in different contexts.

Any single extra facet of text, such as many of those above, probably can be modelled with an ad hoc extension of the current model: 'Unicode stream + XML structure + IDREFs'. But here we are trying to develop a common model to support at least *all* aspects of the automation of the following activities: production and typesetting of complex documents, sophisticated grammar and spell-checking, database publishing and audio reproduction. Whilst none of these need be individually complex, any of them may be.

Moreover, taking a glimpse into the future of documents, all of these textual and structural enrichments need a fortiori to be present in the universe of dynamic, interactive and multiply authored documents that may one day fulfil the vision of the original web philosophers.

9 Conclusion

The model we have presented, in which context-sensitivity is paramount, and in which parallel and multidimensional data structures are the norm, with multiple levels of atomicity for the "characters", is in our opinion a minimum for working with complex documents with text.

Key to the development of the model was the realization that the Unicode standard freezes what writing is to look like, just at a time when writing is no longer restricted to being fixed. When workers were chiseling texts into stone, they were not worried about subsequent automatic processing of their words since the marks on the tablets, and probably the information they contained, were understood to be eternal.

Eschewing nominalism, and reasserting the interconnectedness of our tasks, offers us many new vistas. We can consider, for example, that editing is a special kind of formatting, in which certain layout aspects are simplified. It is by making analyses of this kind that we will be able to transform this proposed general model into one that is implemented, and used, for a broad variety of tasks involving text.

Bibliography

[1] E. A. Ashcroft and W. W. Wadge. ℞ for Semantics. *ACM TOPLAS* 4(2):283–294, 1982.

[2] Martin Berger. An Interview with Robin Milner. `http://nick.dcs.qmul.ac.uk/~martinb/interviews/milner/`

[3] Rudolf Carnap. *Meaning and Necessity.* University of Chicago Press. Enlarged Edition, 1956.

[4] Martin Gardner. Gardner on Gardner: JPBM Communications Award Presentation. *Focus – The Newsletter of the Mathematical Association of America* 14(6), December 1994.

[5] Yannis Haralambous. Unicode et typographie: un amour impossible. *Document numérique* 6(3–4):105–137, 2002.

[6] John Plaice and Yannis Haralambous. The Omega Typesetting and Document Processing System. `http://omega.cse.unsw.edu.au`

[7] John Plaice and Joey Paquet. An Introduction to Intensional Programming. In *Intensional Programming I*, World-Scientific, Singapore, 1997.

[8] John Plaice and W. W. Wadge. A New Approach to Version Control. *IEEE-TSE* 19(3):268–276, 1993.

[9] Chris Rowley and John Plaice. New directions in document formatting: What is text? This volume.

[10] Yannis Stavrakas, Manolis Gergatsoulis and Panos Rondogiannis. Multidimensional XML. In *Distributed Communities on the Web*, *LNCS* 1830:100–109, Springer, 2000.

[11] Paul Swoboda. *A Formalization and Implementation of Distributed Intensional Programming*. PhD Thesis, The University of New South Wales, 2003.

[12] The Unicode Consortium, *et al. The Unicode Standard, Version 4.0.* Addison-Wesley, 2003.

[13] William W. Wadge, Gord Brown, Monica M. C. Schraefel, Taner Yildirim. Intensional HTML. In *Principles of Digital Document Processing*, *LNCS* 1481:128–139, Springer, 1998.

[14] Niklaus Wirth. *Algorithms + Data Structures = Programs.* Prentice-Hall, 1985.

[15] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen and Eve Maler, editors. *Extensible Markup Language (XML) 1.0 (Second Edition)*, 2000. `http://www.w3.org/TR/REC-xml`