The background of the page features a large, faint, golden seal of the University of Bologna. The seal is circular and contains a central shield with a cross, surrounded by various figures and architectural elements. The Latin text 'UNIVERSITAS BOLOGNENSIS' is visible at the top, and 'SIGILLUM BOLOGNENSIS' is at the bottom. The seal is partially obscured by the text.

Patterns for descriptive documents: a formal analysis

**Antonina Dattolo Angelo Di Iorio Silvia Duca
Antonio Angelo Feliziani Fabio Vitali**

Technical Report UBLCS-2007-13

April 2007

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)

The University of Bologna Department of Computer Science Research Technical Reports are available in PDF and gzipped PostScript formats via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` or via WWW at URL `http://www.cs.unibo.it/`. Plain-text abstracts organized by year are available in the directory ABSTRACTS.

Recent Titles from the UBLCS Technical Report Series

- 2006-22 *Broadcasting at the Critical Threshold*, Arteconi, S., Hales, D., October 2006.
- 2006-23 *Emergent Social Rationality in a Peer-to-Peer System*, Marcozzi, A., Hales, D., October 2006.
- 2006-24 *Reconstruction of the Protein Structures from Contact Maps*, Margara, L., Vassura, M., di Lena, P., Medri, F., Fariselli, P., Casadio, R., October 2006.
- 2006-25 *Lambda Types on the Lambda Calculus with Abbreviations*, Guidi, F., November 2006.
- 2006-26 *FirmNet: The Scope of Firms and the Allocation of Task in a Knowledge-Based Economy*, Mollona, E., Marcozzi, A. November 2006.
- 2006-27 *Behavioral Coalition Structure Generation*, Rossi, G., November 2006.
- 2006-28 *On the Solution of Cooperative Games*, Rossi, G., December 2006.
- 2006-29 *Motifs in Evolving Cooperative Networks Look Like Protein Structure Networks*, Hales, D., Arteconi, S., December 2006.
- 2007-01 *Extending the Choquet Integral*, Rossi, G., January 2007.
- 2007-02 *Towards Cooperative, Self-Organised Replica Management*, Hales, D., Marcozzi, A., Cortese, G., February 2007.
- 2007-03 *A Model and an Algebra for Semi-Structured and Full-Text Queries (PhD Thesis)*, Buratti, G., March 2007.
- 2007-04 *Data and Behavioral Contracts for Web Services (PhD Thesis)*, Carpineti, S., March 2007.
- 2007-05 *Pattern-Based Segmentation of Digital Documents: Model and Implementation (PhD Thesis)*, Di Iorio, A., March 2007.
- 2007-06 *A Communication Infrastructure to Support Knowledge Level Agents on the Web (PhD Thesis)*, Guidi, D., March 2007.
- 2007-07 *Formalizing Languages for Service Oriented Computing (PhD Thesis)*, Guidi, C., March 2007.
- 2007-08 *Secure Gossiping Techniques and Components (PhD Thesis)*, Jesi, G., March 2007.
- 2007-09 *Rich Media Content Adaptation in E-Learning Systems (PhD Thesis)*, Mirri, S., March 2007.
- 2007-10 *User Interaction Widgets for Interactive Theorem Proving (PhD Thesis)*, Zacchiroli, S., March 2007.
- 2007-11 *An Ontology-based Approach to Define and Manage B2B Interoperability (PhD Thesis)*, Gessa, N., March 2007.
- 2007-12 *Decidable and Computational Properties of Cellular Automata (PhD Thesis)*, Di Lena, P., March 2007.

Patterns for descriptive documents: a formal analysis

Antonina Dattolo¹

Angelo Di Iorio²

Silvia Duca²

Antonio Angelo Feliziani²

Fabio Vitali²

Technical Report UBLCS-2007-13

April 2007

Abstract

Combining expressiveness and plainness in the design of web documents is a difficult task. Validation languages are very powerful and designers are tempted to over-design specifications. This paper discusses an offbeat approach: describing any structured content of any document by only using a very small set of patterns, regardless of the format and layout of that document. A segmentation model, called Pentaformat, underpins our ideas and is presented in the first part of the paper. The core of this work is rather a formal analysis of some structural patterns, based on grammars and language theory. The study has been performed on XML languages and DTDs and has a twofold goal: coding empirical patterns in a formal representation, and proving their completeness.

1. Department of Mathematics and Applications R. Caccioppoli, University of Napoli Federico II, Italy

2. Department of Computer Science, University of Bologna, Italy

Contents

1	Introduction	3
2	Related works	3
3	Pentaformat: a segmentation model for digital documents	4
4	A generic format to describe structures	5
5	Patterns for descriptive documents	7
6	Formal representation of Patterns	8
6.1	<i>The general grammar G</i>	8
6.2	<i>Our grammar P</i>	8
7	Patterns Completeness	9
7.1	<i>Elements types detection</i>	10
7.2	<i>Refinement</i>	12
8	Conclusions	13

1 Introduction

The World Wide Web has become the greatest repository of information ever existed. The more the centrality of the WWW increases, the more web documents become heterogeneous and complex. New languages for specific domains are continuously proposed and standardized, and the existing ones are upgraded, merged together or developed separately in order to accommodate new requirements. The consequence is that it is quite hard to import or convert documents written in different languages and by different applications. Yet, the prospect of a Semantic Web[BLHL01] where both humans and machines can process information without any ambiguity is extremely attractive, but today it is still very possible to found messy, obscure and application-dependent documents.

We do not perceive *heterogeneity* as a problem, to be solved by flattening all those languages into a plain, unspecific and incomplete one. Neither we want to define *'The'* universal exhaustive language able to describe any domain and any application. What we are rather looking for is a general model to design languages and documents, in order to make them simple, clear, processable. An abstract model could help designers in defining new languages, users in interpreting and reading texts and implementers in coding advanced applications of content management. In this paper we describe our model, called Pentaformat, according to which *any* document, regardless of its actual format and storage, can be segmented into a set of few components (dimensions) independent but connected each other. The separation of logical subcomponents of a text document is certainly not an innovation. Our proposal steps a bit forward: we claim that it is possible to define a generic format able to fully describe the most relevant bits of each dimension. In particular, any *structure* can be always expressed as a pattern-based projection of a very small set of objects and composition rules. We already presented in [DIGV05] patterns to capture the most common structures of digital documents. What we want to do here is resuming that analysis and proposing a theoretical framework to discuss some properties of those patterns.

In particular we present a grammar-based proof of their completeness. Our analysis is performed over XML-based languages and DTDs, for sake of simplicity, but could be extended to other languages with appropriate modifications. The final result is that any DTD, by applying a very small set of reductions, can be transformed into a simpler one based on few patterns but able to *describe* the same fundamental document's structures.

The paper is organized as follows: section 2 discusses related works, section 3 introduces the Pentaformat, while section 4 discusses descriptive nature of our patterns, presented in section 5 and formalized in sections 6 and 7.

2 Related works

The analysis and segmentation of documents into subcomponents, at least content and format, is so embedded and well-accepted by the community that providing a complete list of references is practically impossible (canonical references are [CRD87] and [SMB00]). On the opposite side, we found very interesting ideas about the impossibility of separating content and presentation and actually segmenting documents reusable subcomponents. In [Hil02], Hillesund argued that there is no way of separating (not only from a process perspective, but even from a conceptual one) content and presentation but they are strictly interconnected and mutually dependent. He considers the paradigm of XML "one input - many outputs" basically wrong and claims that it can be only substituted by a weaker "many inputs - many outputs". Indeed it is practically impossible to reuse content fragments and merge them from different sources into a good composite one. [Wal02] answered that position by stating that separation is possible either from a logical or practical perspective and by holding DocBook [W.99] as example of the success of such distinction. Unlike the global critique of Hillesund, in [Pie05] the author argued that in some context it could be useful and profitable to write documents taking in mind both content and presentation, and managing them as a whole unit instead of separated sub-components.

More abstract and formal analyses have raised interest among researchers too. For instance, the hypertext community proposed many formal models to study structural properties of the

(hyper)text. [Par98] modeled a hypertext as a language composed by a limited set of basic components (pages and anchors) and an infinite combination of complex structures created by surfing links among those components. By using grammars to describe that language, he studied and proved some hypertext structural properties. The Dexter Reference Model[HS94] was an important effort to capture the most relevant abstractions of hypertexts, and provide a theoretical framework to compare existing systems to that model. Abstract models are also used to recognize content/layout patterns. For instance, [BZI97] modeled the most relevant entities of a tree-based document, and processed input documents to express them according to that model. By applying statistical inferences and learning algorithms over the abstract model, the system extracts the logical tree that minimizes recognition errors.

3 Pentaformat: a segmentation model for digital documents

The model introduced in this paper, called Pentaformat, refines the canonical content/presentation distinction, by identifying five components that can be extracted from any document, regardless of its actual layout and storage formats:

- *Content*: the plain information made of text and images (we mainly focus on these elements, and leave out audio and video for the moment).
- *Structure*: the labels used to make the meaning of the content explicit. Structure is meant to indicate the role of text elements and their relations, and to make the content interpretable and processable. Both structure and content constitute the basic information written and organized by the author.
- *Presentation*: the set of visual features added to maximize the impact of the document on human readers. Presentation is built over the structures and aims at strengthening what is inherently expressed by structured content. Note that we do not consider presentation as a useless or avoidable layer, but rather as one of the possible expressions of the original information, interpretable and appealing for human readers.
- *Behavior*: the set of dynamic actions triggered by events on a document. The increasing importance of interactivity and dynamic content is testified, for instance, by the last trend of the WWW, where Ajax applications, advanced toolbars, DHTML pages are being more and more widespread.
- *Metadata*: the set of information about the document. They are meant to make resources searchable, indexable and manageable in wider contexts.

Figure 1 shows our segmentation model, emphasizing the role of each abstract constituent. Our claim is not only that any document can be considered as the integration of those five dimensions, but even that they are clearly distinguishable from each other, and can be interchanged and reformulated to obtain different documents. In order to better explain the nature and impact of our segmentation model, some properties of these dimensions are discussed below:

- *Logical separation*: we consider each dimension as a partial perspective onto the same document. Each dimension provides specific information (orthogonal to all others), is created through the help of specific competences and has a specific role for the overall meaning of the document itself. Note that talking about logical separation does not mean these components are always created separately and by different users; rather, it means they can be *abstracted* and separated *a posteriori* to express different kinds of information about the same source document.
- *Mutual connection*: from a different point of view, these dimensions are strictly connected. They are built on the top of each other, and they "work together" for the overall meaning of the document. No dimension makes much sense when examined in isolation.

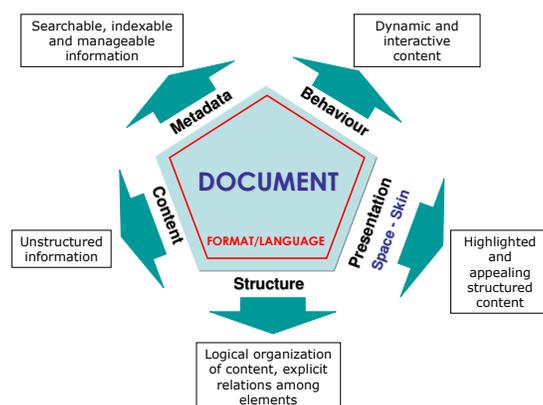


Figure 1. The Pentaformat

- *Context-based relevance*: no hierarchy is imposed *a priori* over these dimensions, but they are equally important from a theoretical point of view, although the content can be probably granted some more relevance since it is the basic information upon which everything else is built. It is the context that determines the relevance and replaceability of the other dimensions.
- *Context-based interchangeability*: depending on the context of use of the document, these components can be replaced with new ones. For instance, we can use the structure to fit the content into a completely different presentation, or express metadata into a completely different vocabulary, etc.
- *Language independence*: information captured by each dimension can be expressed in different languages. However, the actual instantiation into a specific format does not influence the meaning of that information. For instance, structural elements can be translated either into HTML or TEI, while presentational information can be translated either into SVG or XLS-FO, and so on. In conclusion, a cross-dimensional property is necessary to complete our model: the *language* each dimension is expressed in.

So the real point of our work is to be able to separate and extract all constituents of a document so as to reformulate a few of them, or to reuse some of them in different contexts. To this end we look at producing generic formats that describe the specific constituent elements of each document, so as to facilitate understanding and reuse. A generic format is therefore a set of elements describing the relevant bits of the documents in terms of content, structure, presentation, behavior and metadata, although in this paper we will only refer to the first two dimensions.

4 A generic format to describe structures

Defining a generic format for logical structures means finding a model to capture the most common objects of a class of documents. The issue is then understanding which constructs and rules provided by validation languages are really needed.

A point is very important: a generic format *describes* structures *a posteriori*, rather than imposing rules over them. The distinction between prescriptive and descriptive languages have been widely studied in the literature [Qui96][Ren00]. What these approaches change is the role itself of the validation, as outlined in [Pie01]. Traditional way of conceiving validation is *strict*, because

validation is used as a “go/non-go” gauge to verify *in advance* whether or not a data set conforms to a set of requirements. A *loose* validation is rather used to capture abstract and structural information about a text. Both strict and loose validations are useful. What is important is designing languages and schemas by keeping in mind their features and differences, and applying them in right contexts. A fully descriptive schema, for instance, cannot be used as a means to verify minor imperfections or to impose structures to new documents; on the other hand, a schema for strict validation is not suitable to express common features of documents, discovered from a large set of document instances.

Segmentation and extraction of structural elements are by definition something that happens after the creation of a document, something *fully descriptive*. The point is then understanding how validation languages can (and should) be used for writing *descriptive schemas*. However, different *levels of descriptiveness* exist, depending on what designers reckon as important, what can be relaxed, what can be omitted, what can be expressed in a different way. To discuss these levels, we will use DTD-based examples since they are shorter and more direct but we could have used XML-Schema [TDMM01], RelaxNG [Mur00] or any other language, in exactly the same way. We identify six relevant levels of descriptiveness:

- **Prescriptive (P)**: a prescriptive DTD imposes a set of rules which all matching documents must follow. Prevent errors in a production chain, based on strict validation.
- **Descriptive No Alternatives (DNA)**: a descriptive DTD without alternatives do not allow users to force a choice between two (or more elements). The basic idea is that alternatives are meant to inhibit incorrect structures, but they are not required when all documents already exist and the DTD is used to describe all those documents (including variations and exceptions otherwise unpredicted by a strict/prescriptive DTD).
- **Descriptive No Cardinality (DNC)**: a descriptive DTD without alternative can be further generalized by relaxing constraints over the cardinality of each single element. The idea is that by forcing cardinalities some documents could be considered invalid, even if they belong to the same class. Validation is not meant to prevent errors, but to describe existing resources.
- **Descriptive No Order (DNO)**: constraints over the order can be relaxed as well. Imposing an order is something extremely useful when invalid documents obstruct a complex process, but it makes much less sense when the goal is identifying subcomponents. A descriptive document is not meant to say where each object is located (a presentation layer can change that property), but which are the objects contained in the document itself.
- **Super Descriptive (SD)**: relaxing both constraints over cardinality and order, besides alternatives, designers can create abstract DTDa which consider any object as a sequence of repeatable and optional elements (as in the example). Apparently vague, these DTDs are meant to only define the set of objects of the documents.
- **(Un)Descriptive (UD)**: relaxing any constraint designers could say that anything includes anything. Not useful in practice, those DTDs are only mentioned to complete our spectrum.

Table 1 shows a very simple DTD declaration, transformed according to all these models.

On the basis of our experience and previous analysis [DIGV05] we identify the DNO paradigm (with some important variations we will discuss) as a good solution to design generic formats for document structures. Actually we did not cite directly DNO but we studied situations where such descriptiveness (relaxing alternatives and order, but maintaining cardinality) is enough to express everything users need. Moreover we proposed and discussed some patterns, concluding that by adopting these and only these patterns, all those descriptive situations could be covered.

The goal of this paper is providing a formal analysis of those patterns, in order to investigate relations and reductions between prescriptive and descriptive schemas and proving the completeness of our approach.

Descriptiveness level	Content Model
Prescriptive	<!ELEMENT X (A, (B C), D*)>
Descriptive No Alternativess	<!ELEMENTX (A, (B?, C?), D*)>
Descriptive No Cardinality	<!ELEMENT X (A*, (B*, C*), D*)>
Descriptive No Order	<!ELEMENT X (A & (B? & C?) & D*)>
Super Descriptive	<!ELEMENT X (A (B C) D)* >
General	Any

Table 1. Different levels of descriptiveness

5 Patterns for descriptive documents

Patterns we proposed in [DIGV05] are strictly related with the Pentaformat. They do not aim at describing all the dimensions of a document (presentation, metadata and behavior are neglected), but at *normalizing* the existing structures into new ones that express the same logical organization and basic content. That is why few elements are surprisingly enough:

- **Marker:** an empty element, in case enriched with attributes, whose meaning primarily depends on its position within the context. A marker is not meant to provide characterization of the text content, but to identify special rules for a given position of the text.
- **Atom:** a unit of unstructured information. An atom contains only plain text and is meant to indicate a specific role or semantics for that information
- **Block and Inline:** a block of text mixed with unordered and repeatable inline elements that, in turn, have the same content model. They are used to model any objects which ultimately carry the text written by the author.
- **Record:** a set of heterogeneous, unordered, optional and *non-repeatable* elements. Records are first used to group simple units of information in more complex structures, or to organize data in hierarchical subsets.
- **Container:** a set of heterogeneous, unordered, optional and *repeatable* elements. The name itself emphasizes the generality of this pattern, used to model all those circumstances where diversified objects are repeated and collected together.
- **Table:** a sequence of homogeneous elements. Tables are used to group similar objects into the same structure and, also, to represent repeating tabular data.

A deeper discussion of each pattern is out of the scope of this paper (further explanations can be found in our previous work) but some properties deserve some more space.

First of all, patterns are orthogonal: each of them has a specific role and covers a specific situation, and no content model is repeated.

Furthermore, specific rules are imposed over the class of objects allowed in the content-model of each pattern. For instance, an inline element can be contained only within a block, a container cannot directly contain plain text, a record or a table cannot be contained in a block, and so on. Table 2 shows these constraints (each row indicates elements allowed in the content model of each pattern).

	EMPTY	Text	Marker	Atom	Block	Inline	Record	Container	Table
Marker	X								
Atom		X							
Block		X	X	X		X			
Inline		X	X	X		X			
Record			X	X	X		X	X	X
Container			X	X	X		X	X	X
Table			X	X	X		X	X	X

Table 2. Composition rules over patterns

Such strictness is meant to widen the expressiveness and the applicability of patterns. By limiting the possible choices, in fact, the role played by each pattern is highly specialized and it is possible to associate a single pattern to the users' needs. Moreover pattern-based documents are clear, unambiguous and easy to be processed.

Wrappers. To transform a generic DTD in a pattern-based one we use the property of *homogeneity* in content model declarations. This means that, every time a content model contains a mixed presence of repeated elements and single ones (or alternatives), a new element is created. This element, called *wrapper*, will substitute that 'wrong' declaration fragment, inheriting the content model.

For example, consider an element declaration of type `<!ELEMENT X (A, (B|C))>`; what we don't want to have is the presence of a sequence and a choice at the same time. For this reason, we create a new element `W (<!ELEMENT W (B|C)>)` and, substituting it in the previous declaration, we obtain a homogenous declaration `<!ELEMENT X (A, W)>`.

Moreover, by our descriptive perspective, it is necessary to satisfy all constraints expressed in Table 2; given a generic DTD, the introduction of wrappers permits to "by-pass" all those situations where a constraint is violated. For example if a container element is declared as `<!ELEMENT C (A|B)*>` and `B` is an inline element, then a constraint is violated because a container cannot contain inline (see Table 2). In this case, we create a new block element, the wrapper `W (<!ELEMENT W (#PCDATA|B)*>)` and we change `C` definition with `<!ELEMENT C (A|W)*>` (that do not violate constraints).

All changes introduced by patterns are targeted to "clean" (or homogenize) structures and to make documents more descriptive.

6 Formal representation of Patterns

In order to deeply analyze patterns we performed a formal analysis, based on language theory. In formal language theory, a language is defined as a set of words built over a set of terminal symbols, and grammars define rules to combine together terminals through productions. Our idea is then to derive properties of validation languages (whether pattern-based or not) by analyzing the grammars which produce these languages.

We chose DTDs because they are more direct, but similar considerations could be extended to other languages like XML-Schemas or RelaxNG. Although these languages are more powerful, in fact, creating and even reading the corresponding grammars would be much more difficult and time-consuming. More important, the vast majority of existing schemas proved to be structurally equivalent to DTDs [BMNS05].

In the following of this section, we introduce the general grammar `G` for DTDs and we propose our grammar `P` able to produce all the DTDs which use only our patterns, postponing the comparison between related languages to next Section 7.

6.1 The general grammar `G`

The general grammar `G`, provided by the W3C [BPSMM00], produces all the possible DTDs. To make it easier, we extracted some rules and worked only on them. In particular, we are interested in the element type declarations (summarized in Table 3) since they define the overall structure of a document.

6.2 Our grammar `P`

The grammar `P` aims at expressing in a formal way constraints and composition rules over pattern-based documents. The production rules of that grammar are summarized in Table 4. Productions [p01-p08] are used to declare the seven different patterns, while the remaining ones are introduced to specify their content models.

We perform some initial simplifications to make simpler and clearer the analysis: for instance, we omit attributes declarations, and we do not consider some unusual declarations as

[45]	elementdecl	::=	'<!ELEMENT' S Name S contentspec S? '>'
[46]	contentspec	::=	'EMPTY' 'ANY' Mixed children
[51]	Mixed	::=	'(' S? '#PCDATA' (S? ' ' S? Name)* S? ')**' '(' S? '#PCDATA' S? ')'
[47]	children	::=	(choice seq) ('?' '**' '+')?
[48]	cp	::=	(Name choice seq) ('?' '**' '+')?
[49]	choice	::=	'(' S? cp (S? ' ' S? cp)+ S? ')'
[50]	seq	::=	'(' S? cp (S? ' ' S? cp)* S? ')'

Table 3. General grammar P

[p01]	elementdecl	::=	markerelementdecl atomelementdecl blockelementdecl inlineelementdecl recordelementdecl containerelementdecl tableelementdecl
[p02]	markerelementdecl	::=	'<!ELEMENT' S MarkerName S markercontentspec S? '>'
[p03]	atomelementdecl	::=	'<!ELEMENT' S AtomName S atomcontentspec S? '>'
[p04]	blockelementdecl	::=	'<!ELEMENT' S BlockName S blockcontentspec S? '>'
[p05]	inlineelementdecl	::=	'<!ELEMENT' S InlineName S inlinecontentspec S? '>'
[p06]	recordelementdecl	::=	'<!ELEMENT' S recordName S recordcontentspec S? '>'
[p07]	containerelementdecl	::=	'<!ELEMENT' S ContainerName S containercontentspec S? '>'
[p08]	tableelementdecl	::=	'<!ELEMENT' S TableName S tablecontentspec S? '>'
[p09]	markercontentspec	::=	'EMPTY'
[p10]	atomcontentspec	::=	'(' S? '#PCDATA' S? ')'
[p11]	blockcontentspec	::=	maicontentspec
[p12]	inlinecontentspec	::=	maicontentspec
[p13]	maicontentspec	::=	'(' S? '#PCDATA' (S? ' ' S? maiName)+ S? ')**'
[p14]	recordcontentspec	::=	'(' S? mabrctName '?'? (S? '&' S? mabrctName '?'?)* S? ')'
[p15]	containercontentspec	::=	'(' S? mabrctName (S? ' ' S? mabrctName)* ')**'
[p16]	tablecontentspec	::=	'(' S? mabrctName S? ')**'
[p17]	maiName	::=	MarkerName AtomName InlineName
[p18]	mabrctName	::=	MarkerName AtomName BlockName RecordName ContainerName TableName

Table 4. Our pattern-based grammar P

(#PCDATA) * (that can be substituted with the equivalent (#PCDATA)). Moreover, we do not consider the terminal symbol '+' both for shortness and both because it could be associated to the terminal '**' from a descriptive perspective.

Another point is worth being explained: we introduce the terminal symbol '&', that in SGML syntax means that all elements must occur in any order, in order to better formalize the DNO model.

7 Patterns Completeness

So far we discussed about generic descriptive formats, and proposed the *patterns* as a formal model for the description of content and structures. Now we compare the languages generated by the two grammars, G e P , introduce the completeness definition and state the completeness of our pattern-based language.

Let $L(P)$ and $L(G)$ be the languages generated respectively by our grammar P and the general grammar G . $L(P)$ is the set of all possible pattern based DTDs, while $L(G)$ is the set of all possible DTDs.

$$(1) \quad L(P) \subset L(G)$$

We want to demonstrate that for each DTD, producible from G , it exists a pattern-based DTD, producible from P , which is equally descriptive at DNO level. To do it, we present a reduction algorithm, which applied to a DTD, generates a pattern-based DTD, equally descriptive at DNO level. Formally: Given $L(P)$ and $L(G)$, let $r: L(G) \rightarrow L(P)$ be a function that implements our

reduction algorithm; we want to state that

$$(2) \quad \forall d \in L(G) \exists p \in P \ni d \xrightarrow{r} p$$

with p and d equally descriptive at DNO level. The symbol \xrightarrow{r} indicates that d is reduced to p applying the function r .

During reduction process, we relax some constraints, prescribed in grammar G ; in this way, the set of documents, accepted by the pattern-based DTDs, generated by P , is at least large as the set of documents generated by the original DTD.

The algorithm for the transformation of a generic DTD in a pattern-based DTD consists of two phases, called *Element types detection* and , described in details in next two subsections 7.1 - 7.2.

7.1 Elements types detection

The *elements types detection* phase shows how each element $X \in L(G)$ can be mapped in an element $X' \in L(P)$, and so associated at one of the seven possible patterns. The main effects of this phase are synthesized in the diagram of Figure 2. We distinguish 8 different cases and show one example, for each of them.

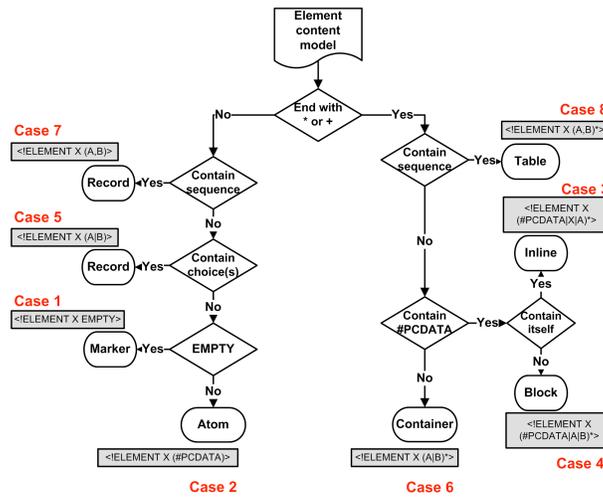


Figure 2. Patterns recognition

In the following, each detection is justified by the exhaustive analysis of $L(G)$: the productions [46-51] are applied and compared with productions [p01-p18] of our grammar P . Each case (Cases 1-8), present in diagram of Figure 2, is discussed in that follows.

Case 1 By applying production [46], if we derive
 $\text{contentspec} \Rightarrow \text{'EMPTY'}$
 the element X is associated to a **Marker** (cfr. [p02] and [p09] of our grammar P , Table 4).

Case 2 By applying productions [46] and [51], if we derive
 $\text{contentspec} \Rightarrow \text{Mixed} \Rightarrow (\#PCDATA)$
 the element X is associated to a **Atom** (cfr. [p03] and [p10]).

By applying productions [46] and [51], if we derive
 $\text{contentspec} \Rightarrow \text{Mixed} \Rightarrow (\#PCDATA \mid \text{Name} \mid \dots \mid \text{Name})^*$,
 then let $n = \text{Name} \mid \dots \mid \text{Name}$, or in more general form $n = N_1 \mid \dots \mid N_m$, with $m \geq 0$. If $n = \emptyset$, then
 $\text{contentspec} \Rightarrow^* (\#PCDATA)^*$

Relaxing constraints, we reduce $(\#PCDATA)^* \xrightarrow{r} (\#PCDATA)$ (see Case 2).

Case 3 If $n \neq \emptyset$, then

$$\text{contentspec} \Rightarrow \text{Mixed} \Rightarrow (\#\text{PCDATA} \mid N_1 \mid \dots \mid N_m)^*$$

In this situation, if $\exists i \ni X = N_i$, with $i = 1, \dots, m$, then X is associated to an **Inline** (cfr. [p05], [p12], [p13] and [p17]).

Case 4 While, if $n \neq \emptyset$ and $\exists i \ni X = N_i$, with $i = 1, \dots, m$, X is associated to a **Block** (cfr. [p04], [p11], [p13] and [p17]).

In these last two cases, in phase of refinement, only if $\forall i, i = 1, \dots, m, N_i \in \text{MAI} = \{\text{MarkerName}, \text{AtomName}, \text{InlineName}\}$, then no reduction rule must be applied.

Case 5 By applying productions [46], [47] and [49], if we derive:

$$\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{choice} \Rightarrow (\text{cp} \mid \text{cp} \mid \dots \mid \text{cp})$$

we relax constraints and we associated to X a **Record**, reducing

$$(\text{cp} \mid \text{cp} \mid \dots \mid \text{cp}) \xrightarrow{r} (\text{cp}? \ \& \ \text{cp}? \ \& \ \dots \ \& \ \text{cp}?)$$

(cfr. [p06], [p14] and [p18]). In this case, in phase of refinement, only if $\forall \text{cp}, \text{cp} \in \text{MABRCT} = \{\text{MarkerName}, \text{AtomName}, \text{BlockName}, \text{RecordName}, \text{ContainerName}, \text{TableName}\}$, then no reduction rule must be applied.

Case 6 By applying productions [46], [47] and [49], if we derive

$$\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{choice}^* \Rightarrow (\text{cp} \mid \text{cp} \mid \dots \mid \text{cp})^*$$

X is associated to a **Container** (cfr. [p07], [p15] and [p18]). In this case, in phase of refinement, only if $\forall \text{cp}, \text{cp} \in \text{MABRCT} = \{\text{MarkerName}, \text{AtomName}, \text{BlockName}, \text{RecordName}, \text{ContainerName}, \text{TableName}\}$, then no reduction rule must be applied.

By applying productions [46], [47] and [49], if we derive:

- $\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{choice}^+$,

we relax constraints, reducing $\text{choice}^+ \xrightarrow{r} \text{choice}^*$ (see Case 6).

- $\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{choice}?$,

we relax constraints, reducing $\text{choice}? \xrightarrow{r} (\text{cp}? \ \& \ \text{cp}? \ \& \ \dots \ \& \ \text{cp}?)$ (see Case 5).

Case 7 By applying productions [46], [47] and [50], if we derive

$$\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{seq} \Rightarrow (\text{cp}, \dots, \text{cp})$$

then we relax constraints, reducing $(\text{cp}, \text{cp}, \dots, \text{cp})$ to $(\text{cp} \ \& \ \text{cp} \ \& \ \dots \ \& \ \text{cp})$.

In this situation, X is associated to a **Record** (cfr. [p06], [p14] and [p18]).

Case 8 By applying productions [46], [47] and [50], if we derive

$$\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{seq}^* \Rightarrow (\text{cp}, \dots, \text{cp})^*$$

X is associated to a **Table** and a record wrapper is introduced for $\text{cp}, \dots, \text{cp}$ (cfr. [p08], [p16] and [p18]).

By applying productions [46], [47] and [50], if we derive:

- $\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{seq}? \Rightarrow (\text{cp}, \dots, \text{cp})?$,

we relax constraints, reducing $(\text{cp}, \dots, \text{cp})? \xrightarrow{r} (\text{cp}? \ \& \ \dots \ \& \ \text{cp}?)$ (see Case 5).

- $\text{contentspec} \Rightarrow \text{children} \Rightarrow \text{seq}^+ \Rightarrow (\text{cp}, \dots, \text{cp})^+$,

we relax constraints, reducing $(\text{cp}, \dots, \text{cp})^+ \xrightarrow{r} (\text{cp}, \dots, \text{cp})^*$ (see Case 8).

In previous analysis, we have leaved out the derivation, obtained by applying production [46]: $\text{contentspec} \Rightarrow \text{'ANY'}$; the reason is that, in practice, this approach is rarely used because it allows too much freedom, and therefore undermines the benefits that derive from defining document structures. Only for the sake of completeness, we mention it and we associate to it a Container wrapper, that can contain, directly or indirectly (by means other wrappers), any other element. During this phase, the presence of 'cp' claims, in many situations, the introduction of appropriate wrappers (see for example case 8).

The result of this phase is the reduction of each element to one (and only one) of the seven patterns, or, in alternative, is the introduction of a wrapper, that assumes the type of a given pattern. As consequence of this consideration, in order to complete our reduction, we need of checking that all elements respect inclusion constraints synthesized in Table 2. This is performed by next refinement phase.

7.2 Refinement

In the second phase it has to be done a cross check between elements to assure that all constraints are observed. Table 5 marks with an X only permitted combinations in elements' declarations, while marks with (x), $x = 0, \dots, 4$, the situations in which a reduction rule must be applied. For

	<i>EMPTY</i>	<i>Text</i>	<i>Marker</i>	<i>Atom</i>	<i>Block</i>	<i>Inline</i>	<i>Record</i>	<i>Container</i>	<i>Table</i>
<i>Marker</i>	X	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)
<i>Atom</i>	(0)	X	(0)	(0)	(0)	(0)	(0)	(0)	(0)
<i>Block</i>	(0)	X	X	X	(1)	X	(3)	(3)	(3)
<i>Inline</i>	(0)	X	X	X	(1)	X	(4)	(4)	(4)
<i>Record</i>	(0)	(0)	X	X	X	(2)	X	X	X
<i>Container</i>	(0)	(0)	X	X	X	(2)	X	X	X
<i>Table</i>	(0)	(0)	X	X	X	(2)	X	X	X

Table 5. Composition rules over patterns and reductions

each element declaration it has to be checked that all contained element in the content model don't violate the constraints expressed in Table 2. Every time an element content model brakes a rule (i.e. an inline declaration contains a block in its content model) a specified reduction rule has to be applied according to the previous table (in the previous example the (1) reduction rule has to be applied). Some times a reduction rule can change the type classification of an element implying a complete recheck of all element declarations, making this algorithm iterative. In the following subsections each reduction rule will be specified.

Reduction rule (0). All cases marked with this reduction rule will never happen because the definition of the element excludes those cases. For example if an element has been recognized as *Marker* it is impossibile that the declaration of the element contains other elements than the keyword 'EMPTY'.

Reduction rule (1). In the case that a block element is found in the declaration of other blocks or inline than a two-steps reduction has to be applied:

- the block element that is found inside other element will be no longer considered as a block but as an inline;
- having changed the type classification of an element all the element declarations have to be re-checked with the new type classification.

Reduction rule (2). In the case that an inline element compare inside a record, a container or a table element it has to be created a new block element that will substitute the inline element in all the wrong positions; the content model of the new block element will have to contain only the inline element. In this case it's not needed a re-check.

Reduction rule (3). In the case that a record, a container or a table element appear in content model of a block element, a three steps reduction has to be applied:

- remove the offending-element from the block-element containing it;
- create a new container-wrapper with content model of type (block-element | offending-element)*;

- substitute everywhere the `block-element` with the new container-wrapper.

In this case it's not needed a re-check.

Reduction rule (4). In the case that a record, a container or a table element appear in content model of an inline element than a three steps reduction has to be applied:

- remove the offending-element (s) from the inline-element containing it;
- push the removed offending-element (s) in all the parent block elements that contain as son or descendant the inline-element;
- having modified the content model of some elements, all the element declarations have to be re-checked.

8 Conclusions

In this paper we have first presented a general model for digital documents, called Pentaformat. This model states that *any* document, regardless of its actual format and storage, can be segmented into a set of few components independent but connected each other. Then we focused on structures and content.

We showed how any *structure* can be always expressed as a pattern-based projection of a very small set of objects and composition rules and we presented a grammar-based formalization of our patterns and a formal proof of their completeness. In the future, we plan to inspect, in the same formal way, other properties like correctness and minimality.

References

- [BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. *The Semantic Web*. Scientific American, 2001.
- [BMNS05] G. J. Bex, W. Martens, F. Neven, and T. Schwentick. Expressiveness of xsds: from practice to theory, there and back again. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 712–721, New York, NY, USA, 2005. ACM Press.
- [BPSMM00] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0. <http://www.w3.org/TR/REC-xml>, 2000.
- [BZI97] R. Brugger, A. Zramdini, and R. Ingold. Modeling documents for structure recognition using generalized n-grams. *Icdar*, 1997.
- [CRD87] J. H. Coombs, A. H. Renear, and S. J. DeRose. Markup systems and the future of scholarly text processing. *Commun. ACM*, 1987.
- [DIGV05] A. Di Iorio, D. Gubellini, and F. Vitali. Design Patterns for Descriptive Document Substructures. In *Proceedings of the Extreme Markup Conference*, Montreal, Canada, 2005.
- [Hil02] T. Hillesund. Many outputs many inputs: Xml for publishers and e-book designers. *Journal of Digital Information*, 2002.
- [HS94] F. Halasz and M. Schwartz. The dexter hypertext reference model. *Commun. ACM*, 1994.
- [Mur00] M. Murata. Relax (REgular LAnguage description for Xml). <http://www.xml.gr.jp/relax/>, 2000.

- [Par98] S. Park. Structural properties of hypertext. In *In Proceedings of the ninth ACM conference on Hypertext and Hypermedia*, Pittsburgh, Pennsylvania, United States, 1998.
- [Pie01] W. Piez. Beyond the 'descriptive vs. procedural' distinction. In *Proceedings of the Extreme Markup Conference*, Montreal, Canada, 2001.
- [Pie05] W. Piez. Format and content: Should they be separated? can they be?: With a counter-example. In *Proceedings of Extreme Markup Languages*, Montreal, Canada, 2005.
- [Qui96] L. Quin. Suggestive Markup: Explicit Relationships in Descriptive and Prescriptive DTDs. In *Proceedings of the SGML 96 Conference*, Boston, MA, USA, 1996.
- [Ren00] A. Renear. The Descriptive/Procedural Distinction is Flawed. *Markup Languages: Theory and Practice*, 2(4):411–420, 2000.
- [SMB00] C.M. Sperberg-McQueen and L. Burnard. A Gentle Introduction to XML. In *Guidelines for Electronic Text Encoding and Interchange*, 2000.
- [TDMM01] H. S. Thompson, Beech D., M. Maloney, and N. Mendelsohn. XML Schema Part 1: Structures. <http://www.w3.org/TR/xmlschema-1/>, 2001.
- [W.99] Norman W. oasis-open.org - docbook technical committee document repository. <http://www.oasis-open.org/docbook/>, 1999.
- [Wal02] N. Walsh. Xml: One input many outputs: a response to hillesund. *Journal of Digital Information*, 2002.