SGML & XML Content Models

Pekka Kilpeläinen
Department of Computer Science
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki
Finland
E-mail: `Pekka.Kilpelainen@cs.Helsinki.FI`

May 19, 1998

Abstract

The SGML and XML standards use a variation of regular expressions called *content models* for modeling the markup structures of document elements. SGML content models may include so called *and* groups, which are excluded from XML. An *and* group, which is a sequence of subexpressions separated by an &-operator, denotes the sequential catenation of its subexpressions in any possible order. If one wants to shift from SGML to XML in document production, one has to translate SGML content models to corresponding XML content models.

The allowed content models in both SGML and XML are restricted by a requirement of *determinism*, which means that a parser recognizing document element contents has to be able to decide without lookahead, which content model token to match with the current input token, while processing the document from left to right. It is known that not all SGML content models can be expressed as an equivalent XML content model. It is also known that transforming an SGML content model into an equivalent XML content model may cause an exponential growth in the length of the content model. We discuss methods of eliminating *and* groups and analyze the circumstances where they can be applied. We derive a tight bound of $en!$ on the number of symbols in the result of eliminating an *and* group of n symbols, where $e = 2.71828\ldots$ is the base of natural logarithms. We present the analysis in a pedagogical manner, emphasizing mathematical methods which are typical to the analysis of algorithms. We also show that minimal deterministic automata for recognizing an *and* group of n distinct element names contain 2^n states and $n2^{n-1}$ transitions, excluding the failure state and transitions leading to it.

Keywords: SGML, XML, content model, regular expression, determinism, unambiguity, deterministic automaton, analysis

1 Introduction

The Standard Generalized Markup Language (SGML) [13, 11] allows documents to be modeled using document type definitions (DTDs). A DTD establishes a set of named document element types and describes the allowed contents of their instances using content models, which are a variation of ordinary regular expressions. As an extension to regular expressions, content models may include so called *and* groups. An *and* group, syntactically represented by an expression $E_1\& \cdots \&E_n$, denotes the sequential catenation of its subexpressions $E_1, \ldots, E_n$ in any possible order. For example, the DTD fragment shown in Figure 1 specifies that `message` elements consist of `from`, `to` and `subject` elements, which can occur in any order, and a `body` element appearing after them. The second row is a combined element type definition for three element types. It specifies that elements of types `from`, `to` and `subject` consist of unstructured strings, expressed by the SGML keyword #PCDATA.

```
<!ELEMENT message                    - -   ((from & to & subject), body)>
<!ELEMENT (from | to | subject)      - -   (#PCDATA)>
```

Figure 1: A fragment of an SGML DTD.

Extensible Markup Language (XML) [3] is a simple dialect of SGML, defined by removing some features and options from the specification of SGML. The goal of XML is to allow SGML documents to be served, received and processed on the Web. A major distinction between SGML and XML is that the latter does not require a DTD to escort the document. XML only requires documents to be *well-formed.* The well-formedness means, roughly, that the start and the end of all document elements have to be marked with properly nesting pairs of matching markup tags. Indeed, a DTD is seldom needed for network delivery and subsequent document formatting. A DTD is needed, though, while creating XML documents, e.g., to validate their syntactic correctness or to guide authors in a structured editing environment.

Many individuals and organizations consider XML as a replacement of SGML. Moving from SGML to XML requires modifications of document type definitions and their content models. Especially, *and* groups must be replaced by corresponding XML expressions. It is known that the elimination of *and* groups from a content model is not always possible according to the rules of SGML and XML. If the elimination can be done, it is known to lengthen the content model by an exponential factor. We derive a tight upper bound to this lengthening: The number of symbol occurrences needed in an XML equivalent of any *and* group built of n symbols and $n-1$ operators is $O(en!)$, where $e = 2.71828\ldots$ is the base of natural logarithms.

The rest of the paper is organized as follows. We describe SGML and XML content models formally in Section 2, and discuss their relation. In Section 3 we discuss methods of eliminating *and* operators from content models. We present sufficient conditions for a wide class of SGML content models, which can be translated to their XML counterparts using the presented methods. The lengthening effect of *and* group elimination is then analyzed in Section 4. We show in Section 5 that the size of deterministic automata for an *and* group of distinct element names is smaller than the length of the corresponding XML model group, but still exponential. Finally, Section 6 is a conclusion.

2 Deterministic content models

The DTD formalism used in SGML and XML to define allowed document structures is essentially a variation of extended context-free grammars [14]. The right-hand sides of the productions are called *content models.* An SGML content model consists of a *model group*, which is similar to a regular expression, and optional *exceptions.* Exceptions modify the content model by excluding optional symbols allowed by the model group and allowing optional symbols not mentioned in the model group. The precise semantics of exceptions and their elimination from content models has been considered in [14].

Model groups are similar to regular expressions over an alphabet V, which consists of *generic identifiers* (i.e. names of element types) of the DTD, and #PCDATA. We refer to the members of V as *symbols.* The language $L(E)$ defined by an SGML model group E is defined inductively as follows.[1]

$L(a) = \{a\}$ for $a \in V$. $\quad L(F,G) = \{vw \mid v \in L(F), w \in L(G)\}$.
$L(F|G) = L(F) \cup L(G)$. $\quad L(F^*) = \{v_1 \dots v_n \mid v_1, \dots, v_n \in L(F), n \geq 0\}$.
$L(F?) = L(F) \cup \{\lambda\}$. $\quad L(F^+) = \{v_1 \dots v_n \mid v_1, \dots, v_n \in L(F), n \geq 1\}$.
$L(F_1\& \cdots \& F_n) = \{v_1 \dots v_n \mid v_i \in L(F_{\phi(i)}), i = 1, \dots, n, \phi$ is a permutation of $\{1, \dots, n\}\}$.

Notice that & is not an associative binary operator. For example, $L(a\&b\&c) = \{abc,\ acb,\ bac,\ bca,\ cab,\ cba\}$, whereas $L(a\&(b\&c)) = \{abc,\ acb,\ bca,\ cba\}$ and $L((a\&b)\&c) = \{abc,\ bac,\ cab,\ cba\}$. SGML considers also the *seq* and *or* operators ',' and '|' multinary rather than binary, but the distinction is essential only for the & operator.

One should be aware that the above equations are actually a simplification of the informal rules of the SGML standard. The standard imposes priority rules on operators, whose exact semantics are not captured by the above declarative simplification. SGML standard gives a higher priority to re-satisfying the most recently satisfied content model token with an operator $^+$ or *, which has effect on content models with *and* groups. For example, $abba \notin L((a\&b^+)^*)$, even though both $ab \in L(a\&b^+)$ and $ba \in L(a\&b^+)$.

XML content models are similar to SGML model groups except that they do not use the &-operator.

Both SGML and XML require content models to be *deterministic* in the sense that each nonempty prefix of an input word determines uniquely which symbol of the content model matches each of its symbols. SGML standard calls determinism *unambiguity*. Brüggemann-Klein and Wood [9] call determinism more precisely 1-unambiguity. We use "determinism" according to the XML specification, but follow Brüggemann-Klein and Wood [6, 8, 9] in the technical treatment of determinism.

Let E be an expression over alphabet V. We denote the set of symbols occurring in E by $sym(E)$. We use $sym(w)$ also to denote the set of symbols occurring in a word w. We call different occurrences of symbols in E *positions* of expression E. We refer to them by considering *markings* of expressions. An expression E' over the alphabet $\Pi = \{a_i \mid a \in V, i \in \{1, 2, \dots\}\}$ is a *marking of* E if E' is obtained from E by attaching a different subscript i to each occurrence of a symbol of E. We use characters E, F, G and H for referring to expressions, characters a, b, $\dots$ near the beginning of the alphabet for symbols of V and characters x, y, $\dots$ near the end of the alphabet for subscripted symbols of Π. Finally, we use $r, s, \dots, w$ for words over V or Π. For a subscribed symbol $x = a_i$ we denote its underlying character a by $\chi(x)$. For a marked expression E' we use $\chi(E')$ to denote the corresponding unmarked expression, and for a string w over a marked alphabet Π we denote by $\chi(w)$ the string consisting of the underlying symbols of the positions of w.

Let E be a content model and let E' be a marking of E. Content model E is *non-deterministic* if it has two *competing positions* x and $y \in sym(E')$ with the same underlying symbol. By this we mean that there are words u, v and w over Π and two different marked symbols x and y of Π for which both uxv and uyw are words of $L(E')$ and $\chi(x) = \chi(y)$. Content model E is *deterministic* if it is not non-deterministic. It is easy to see that the definition of determinism does not depend on the marking chosen for E. Intuitively, a word $w \in L(E')$ for a marking E' of a content model E corresponds to a unique path through E. For example, the content model $E = ((a,b)|(a,c))$ is non-deterministic. This can be seen from its marking $E' = ((a_1, b_2)|(a_3, c_4))$, for which $a_1 b_2 \in L(E')$, $a_3 c_4 \in L(E')$ and the underlying symbol of the competing positions a_1 and a_3 is the same.

We define subexpressions of a content model in a standard manner, in such a way that the content model is also its own subexpression. It is rather easy to see that a content model is deterministic if and only if all of its subexpressions are deterministic. Following the SGML standard we will call subexpressions of content models also *model groups*.

[1] We use symbol λ to denote the empty word.

The determinism or non-determinism of a given model group can be checked algorithmically in linear time [6].

The informal priority rules of SGML mentioned above have not been studied formally. If we adopt the simplified formal semantics of content models, it is clear that the languages definable by SGML and XML content models form a subclass of regular languages. It is known that the languages of deterministic XML content models form a proper subclass of the languages of deterministic SGML content models. This was shown by Brüggemann-Klein using automata-theoretic arguments to demonstrate that, e.g., the language $L((a\&b?\&c?)^*)$ cannot be expressed by a deterministic XML content model [4, 8]. Whether the languages definable by deterministic SGML content models form a proper subclass of regular languages is an open problem [7, 9].

We discuss methods and possibilities of translating SGML model groups into equivalent XML model groups by the elimination of *and* groups in the next section.

3 Eliminating the and operator

The difference between SGML model groups and XML content models is that the former may contain *and* groups. In this section we discuss the methods and possibilities of eliminating *and* groups.

A first attempt to eliminate *and* groups could be to replace an *and* group $E_1\&\cdots\&E_n$ by an *or* group, which explicitly enumerates all the $n!$ sequential permutations of its subexpressions, as follows:

$$(E_1,\ldots,E_n)|(E_1,\ldots,E_{n-2},E_n,E_{n-1})|\cdots|(E_n,\ldots,E_1)\ .$$

A drawback with this transformation, in addition to the exponential length of the result, is that it produces inherently non-deterministic model groups. This is because for each subexpression E_i, $i=1,\ldots,n$, the *or* group has $(n-1)!$ subexpressions beginning with an identical copy of subexpression E_i.

A more careful way of expanding *and* groups into equivalent *or* groups appears to belong to SGML folklore [16]. This transformation can be defined as follows. For an *and* group $E = E_1\&\cdots\&E_n$ we define its expansion $\mathcal{E}(E)$ by

$$\begin{array}{rcl} \mathcal{E}(E) & = & (E_1,\mathcal{E}(E_2\&E_3\&\cdots\&E_n)) \\ & & |(E_2,\mathcal{E}(E_1\&E_3\&E_4\&\cdots\&E_n)) \\ & & \vdots \\ & & |(E_n,\mathcal{E}(E_1\&E_2\&\cdots\&E_{n-1})),\ \text{when}\ n>1; \\ \mathcal{E}(E) & = & E,\ \text{when}\ n=1. \end{array}$$

Notice that even though $\mathcal{E}(E)$ contains multiple copies of expressions E_i, $i=1,\ldots,n$, each top-level subexpression of $\mathcal{E}(E)$ begins with a different subexpression E_i. For example,

$$\mathcal{E}(a\&b\&c) = (a,((b,c)|(c,b)))\ |\ (b,((a,c)|(c,a)))\ |\ (c,((a,b)|(b,a)))\ .$$

There are some simple situations where also the above transformation fails to preserve determinism. For example $E = a\&b?$ is deterministic, but $\mathcal{E}(E) = (a,b?)|(b?,a)$ is non-deterministic. For example, its marking $(a_1,b_2?)|(b_3?,a_4)$ accepts both words a_1 and a_4. We will return to this situation shortly at the end of this section. Now we will present a wide class of content models, for which the $\mathcal{E}$ transformation leads to correct results, i.e., to equivalent and deterministic content models.

An expression E is *nullable* if $\lambda \in L(E)$. We say that an *and* group $E = E_1 \& \cdots \& E_n$ is *complete* if none of its immediate subexpressions is nullable, i.e., $\lambda \notin L(E_i)$ for each $i = 1, \ldots, n$. The nullable subexpressions of a content model can be computed in linear time by applying the below rules:

1. $\lambda \in L(a)$, where $a \in V$, if and only if $a = \#\text{PCDATA}$;
2. $\lambda \in L(F, G)$ if and only if $\lambda \in L(F)$ and $\lambda \in L(G)$;
3. $\lambda \in L(F|G)$ if and only if $\lambda \in L(F)$ or $\lambda \in L(G)$;
4. $\lambda \in L(F^*)$;
5. $\lambda \in L(F^+)$ if and only if $\lambda \in L(F)$;
6. $\lambda \in L(F?)$;
7. $\lambda \in L(F_1 \& \cdots \& F_n)$ if and only if $\lambda \in L(F_i)$ for each $i = 1, \ldots, n$.

The next theorem states that the *and* group elimination is correct for a content model consisting of a single complete *and* group.

Theorem 3.1 *Let $E = E_1 \& \cdots \& E_n$ be an* and *group. Then $L(\mathcal{E}(E)) = L(E)$. If E is complete and deterministic, $\mathcal{E}(E)$ is deterministic.*

Proof. We proceed by induction on n. The base case $n = 1$ is clear, since then $\mathcal{E}(E) = E$. Assume that $n > 1$ and that the claim holds for $n - 1$.

We first show that $L(\mathcal{E}(E)) = L(E)$. Assume that $w \in L(\mathcal{E}(E))$. This means that $w = vu$, where $v \in L(E_i)$ and

$$u \in L(\mathcal{E}(E_1 \& \cdots \& E_{i-1} \& E_{i+1} \& \cdots \& E_n))$$

for some $i \in \{1, \ldots, n\}$. Without loss of generality we can assume that $i = 1$. By the inductive assumption $u \in L(E_2 \& \cdots \& E_n)$, i.e., there is a permutation ϕ of $\{2, \ldots, n\}$ such that $u = u_2 \ldots u_n$ for $u_j \in L(E_{\phi(j)})$, $j \in \{2, \ldots, n\}$. Then $\phi' = \phi \cup \{(1, 1)\}$ is a permutation for which $u_j \in L(E_{\phi'(j)})$, $j \in \{1, \ldots, n\}$, when we let $u_1 = v$. Thus $w = u_1 \ldots u_n \in L(E)$. The converse inclusion $L(E) \subseteq L(\mathcal{E}(E))$ can be shown in a similar manner.

Next assume that E is a complete and deterministic *and* group. Let $uxv, uyw \in L(\mathcal{E}(E)')$, where x and y are two different positions of the marked content model $\mathcal{E}(E)'$. Let the prime $'$ denote a fixed marking of $\mathcal{E}(E)$. Its restriction to any subexpression F of $\mathcal{E}(E)$ is obviously a marking of F. Let ${E_i}'$, $i = 1, \ldots, n$ be the first subexpression of each of the n branches of the top-level *or* group in $\mathcal{E}(E)'$. If x and y are positions of ${E_i}'$ and ${E_j}'$ for some $i, j \in \{1, \ldots, n\}$, we must have $\chi(x) \neq \chi(y)$, since otherwise the positions x and y of ${E_1}' \& \cdots \& {E_n}'$ would establish the non-determinism of E. Then assume that at least one of the positions x and y belongs to $sym(\mathcal{E}(E')) - \bigcup_{i=1}^{n} sym(E_i')$. This means that $u \neq \lambda$, since $\lambda \notin L({E_i}')$, $i = 1, \ldots, n$. We can assume without loss of generality that $sym(u) \cap sym({E_1}') \neq \emptyset$. If $x, y \in sym(\mathcal{E}(E_2 \& \cdots \& E_n)')$, we must have by the inductive assumption that $\chi(x) \neq \chi(y)$. Finally, assume (again without loss of generality) that $x \in sym({E_1}')$ and $y \in sym(\mathcal{E}(E_2 \& \cdots \& E_n)')$. Let F' be the subexpression of $\mathcal{E}(E_2 \& \cdots \& E_n)'$ which contains position y and which is a marking of some E_j, where $j \in \{2, \ldots, n\}$. Then

$${E_1}' \& \cdots \& E'_{j-1} \& F' \& E'_{j+1} \& \cdots \& {E_n}'$$

is a marking of E in which x and y are competing positions. Since E is deterministic, $\chi(x) \neq \chi(y)$. □

The above *and* group elimination is not necessarily valid, when the *and* group is a subexpression of a larger content model. For example, the content model $(a \& b^+)^*$ is deterministic, whereas

$(\mathcal{E}(a\&b^+))^* = ((a, b^+)|(b^+, a))^*$ is not. This can be seen from the words accepted by its marking $((a_1, {b_2}^+)|({b_3}^+, a_4))^*$, where any occurrence of symbol b_2 can be followed by both b_2 and b_3.

Next we present a sufficient condition for the validity of *and* group replacement in subexpressions. Let E' be a marking of E and let F' be a subexpression of E'. We denote by $E'[F'/G']$ the result of replacing subexpression F' in E' by an expression G'. For example, when $E' = ((b_1\&c_1), (b_2\&c_2))^*$ and $F' = (b_2\&c_2)$, we have

$$E'[F'/((b_2, c_2)|(c_3, b_3))] = ((b_1\&c_1), ((b_2, c_2)|(c_3, b_3)))^* \ .$$

Let E be a model group and F a subexpression of E. We say that F is a *sublanguage expression* of E, if $L(F) \subseteq L(E)$. We say that F is a *sublanguage expression of an iteration* if it is a sublanguage expression of the model group G^* or G^+ for some expression G. Since being a sublanguage expression is a transitive relation, we can compute in linear time those subexpressions of a marked content model E', which are sublanguage expressions of an iteration. The computation can be performed by a preorder traversal of the expression tree of E', applying the following simple rules:

1. A symbol $a \in \Pi$ has no further sublanguage expressions.
2. Subexpressions F' and G' are sublanguage expressions of an expression $F'|G'$.
3. Subexpression ${F_i}'$, $i \in \{0, 1\}$, is a sublanguage expression of an expression $({F_0}', {F_1}')$ if and only if $\lambda \in L(F'_{1-i})$.
4. Subexpression F' is a sublanguage expression of expressions ${F'}^*$, $F'?$ and ${F'}^+$.
5. Subexpression ${F_i}'$, $i \in \{1, \ldots, n\}$, is a sublanguage expression of an expression ${F_1}'\& \cdots \&{F_n}'$ if and only if $\lambda \in L({F_j}')$ for each $j \in \{1, \ldots, n\} - \{i\}$.

We present two lemmas about sublanguage expressions and their languages. The first lemma states that a single non-empty word accepted by both a marked model group and its subexpression witnesses that the subexpression is also a sublanguage expression of the model group.

Lemma 3.2 *Let E' be a marked model group and let F' be a subexpression of E'. Then F' is a sublanguage expression of E' if and only if there is a non-empty word $w \in L(F') \cap L(E')$.*

Proof. If $L(F') \subseteq L(E')$, there is a non-empty word $w \in L(F') \cap L(E')$, since languages of model groups always contain some non-empty word.

For the converse assume that there is a non-empty word $w \in L(F') \cap L(E')$ and show $L(F') \subseteq L(E')$ by a structural induction on E'. The base case $E' = F'$ is clear. We sketch the two non-obvious cases of the inductive step. Notice that when ${E_i}'$ and ${E_j}'$ are such subexpressions of a marked model group that none of them is a subexpression of the other, then $sym({E_i}') \cap sym({E_j}') = \emptyset$. This means in the case $E' = ({E_1}', {E_2}')$ that the assumption $w \in L(F') \cap L(E')$ implies that $w \in L({E_i}')$ and $\lambda \in L(E'_{1-i})$ with $i \in \{0, 1\}$. This implies $L({E_i}') \subseteq L(E')$, and the inductive assumption $L(F') \subseteq L({E_i}')$ gives that $L(F') \subseteq L(E')$. The case $E' = {E_1}'\& \cdots \&{E_n}'$ is handled similarly. □

The next lemma says that a non-nullable subexpression F' of a marked content model E' is a sublanguage expression of an iteration if and only if words of $L(F')$ can occur in words of $L(E')$ as repetitive subwords with no intervening symbols.

Lemma 3.3 *Let E' be a marked model group and let F' be a non-nullable subexpression of E'. Then expression F' is a sublanguage expression of an iteration if and only if there is a word $w \in L(E')$ of the form $w = uv_1v_2t$ with $v_1, v_2 \in L(F')$.*

Proof. For a model group E', a word accepted by any of its subexpressions occurs as a subword of some $w \in L(E')$. If F' is a sublanguage expression of an iteration, it is easy to see that there is a word $w \in L(E')$ of the form $w = uv_1v_2t$, where $v_1, v_2 \in L(F')$. On the other hand, if F' is not a sublanguage expression of an iteration, then by Lemma 3.2 we have $v \notin L(G')$ for any $v \in L(F')$ and any iterative subexpression G' of E' of the form H^* or H^+. This means that a subword v_1v_2 of a word $uv_1v_2t \in L(E')$ with $v_1, v_2 \in L(F')$ can be generated only if F' is either of the form (E_1', E_2') or of the form $E_1' \& \cdots \& E_n'$. If $F' = ({E_1}', {E_2}')$, this requires that $\{v_1, \lambda\} \subseteq L({E_1}')$ and $\{v_2, \lambda\} \subseteq L({E_2}')$, which contradicts the non-nullability of F'. By a similar argument the word v_1v_2 cannot be generated by F' of the form ${E_1}' \& \cdots \& {E_n}'$ either. □

To argue about the languages of content models with replaced subexpressions we present the next two lemmas.

Lemma 3.4 *Let E' be a marking of a content model over V, let F' be a subexpression E', and let G be a model group over Π.*

1. *If $L(F') \subseteq L(G)$, then $L(E') \subseteq L(E'[F'/G])$.*
2. *If $L(G) \subseteq L(F')$, then $L(E'[F'/G]) \subseteq L(E')$.*
3. *If $L(G) = L(F')$, then $L(E'[F'/G]) = L(E')$.*

Proof. A straightforward induction of the structure of E'. Notice that the third claim immediately follows from the two other claims. □

The next fairly technical lemma can be intuitively understood by considering words $w \in L(E')$ for a marking E' of model group E as accepting paths trough symbol positions of E. The lemma states that when F' is a subexpression of E' satisfying our assumptions and G' is the replacement of F', then replacing the subwords of words in $L(E')$ which correspond to paths going through F' by any subwords corresponding to paths going through G' yields words of $L(E'[F'/G'])$, and vice versa.

Lemma 3.5 *Let E' be a marked model group and let F' be a non-nullable subexpression E', which is not a sublanguage expression of an iteration. Let G' be a non-nullable marked model group with $sym(G') \cap sym(E') = \emptyset$. Let $n \geq 0$, $v_1, \ldots, v_n \in L(F')$, $w_1, \ldots, w_n \in L(G')$, and $u_0, \ldots, u_n \in \Pi^*$ such that $sym(u_i) \subseteq (sym(E') - sym(F'))$ for $i = 0, \ldots, n$. Then*

$$u_o v_1 u_1 \ldots v_n u_n \in L(E')$$

if and only if

$$u_o w_1 u_1 \ldots w_n u_n \in L(E'[F'/G']) \ .$$

Proof. An induction of the structure of E'. Notice that it is sufficient to consider one direction of "if and only if", since $(E'[F'/G'])[G'/F'] = E'$ and the assumptions about F' and G' are identical. □

The above lemmas allow us to prove our second main theorem, which states that the elimination of complete *and* groups which do not occur as a sublanguage expression of an iteration can be done without violating determinism.

Theorem 3.6 *Let E' be a marking of a deterministic content model E and let F' be an* and *group in E'. Then $L(\chi(E'[F'/\mathcal{E}(F')])) = L(E)$. If F' is complete and not a sublanguage expression of an iteration, $E'[F'/\mathcal{E}(F')]$ is deterministic.*

Proof. Let $F = \chi(F')$. By Theorem 3.1, $L(\mathcal{E}(F)) = L(F)$. Thus $L(\chi(E'[F'/\mathcal{E}(F')])) = L(\chi(E'[F'/\chi(\mathcal{E}(F'))])) = L(\chi(E'[F'/F])) = L(E)$ by Lemma 3.4.

Next consider the determinism of $E'[F'/\mathcal{E}(F')]$. Assume that F' is a complete *and* group and not a sublanguage expression of an iteration. Let G' be such a marking of $\mathcal{E}(F')$ that $sym(E') \cap sym(G') = \emptyset$. Then $E'[F'/G']$ is a marking of content model $E'[F'/\mathcal{E}(F')]$. Let $rxs, ryt \in L(E'[F'/G'])$, where x and y are two competing positions of $E'[F'/G']$. There are three possibilities. First, consider the case $\{x, y\} \subseteq sym(E')$, i.e., both x and y are positions of the original content model. Then we can show by Lemma 3.5 that $uxv, uyw \in L(E')$ for some $u, v, w \in \Pi^*$. This means by the determinism of E that $\chi(x) \neq \chi(y)$.

Next assume that at least one of the positions, say x, is a position of G'. Let $r = r_1 r_2$ and $s = s_1 s_2$ so that r_2 is the longest suffix of r consisting of symbols of G', and s_1 is the longest prefix of s consisting of symbols of G'.

If $y \in sym(G')$, let $t = t_1 t_2$ so that t_1 is the longest prefix of t consisting of symbols of G'. Since G' occurs in E' at the place of F', it is not a sublanguage expression of an iteration. By Lemma 3.3 both words $r_2 x s_1$ and $r_2 y t_1$ are therefore complete words of $L(G')$. Then $\chi(x) \neq \chi(y)$, since $\mathcal{E}(F')$ is deterministic by Theorem 3.1.

Finally consider the case $x \in sym(G')$ and $y \in sym(E') - sym(F')$. Since $L(\chi(G')) = L(\chi(F'))$, there are a position $z \in sym(F')$ and words $u_2, v_1 \in \Pi^*$ for which $u_2 z v_1 \in L(F')$ and $\chi(u_2) = \chi(r_2)$, $\chi(z) = \chi(x)$, and $\chi(v_1) = \chi(s_1)$. Now we want to show by applying Lemma 3.5 that there are words $u, v, w \in \Pi^*$ for which $uzv, uyw \in L(E')$. This is straightforward if $r_2 = u_2 = \lambda$. Remember that r_2 is the longest suffix of r which consists of symbols of G'. If $r_2 \neq \lambda$ and since $y \notin sym(G')$, we must have by Lemma 3.3 that $r_2 \in L(G')$. Since F is deterministic, u_2 is the unique word of $L(F')$ with $\chi(u_2) = \chi(r_2)$. The existence of $u, v, w \in \Pi^*$ for which $uzv, uyw \in L(E')$ follows then from this by Lemma 3.5. Since E is deterministic, $\chi(y) \neq \chi(z) = \chi(x)$. □

There are also some further possibilities of removing *and* groups. One of them pertains to nullable *and* groups $E_1 \& \cdots \& E_n$, which occur as a sublanguage expression of an iteration. Such an *and* group can be replaced by the corresponding *or* group $E_1 | \cdots | E_n$. For example, $(a?, (b?\&c^*), d^*)^+ \equiv (a?, (b?|c^*), d^*)^+$. We establish a few lemmas to study the correctness of this transformation. The first one of them is a simple observation about the languages of iterative expressions.

Lemma 3.7 *Let E be an expression of the form F^* or F^+. If $u \in L(E)$ and $v \in L(E)$, then $uv \in L(E)$.*

The second lemma states that replacing a subexpression F' of a marked content model by a nullable subexpression does not effect those words accepted by the content model, which do not contain symbols of the original subexpression F'.

Lemma 3.8 *Let E' be a marked model group and let F' be a subexpression of E'. If there is a word $w \in L(E')$ such that $sym(w) \cap sym(F') = \emptyset$ and G is a nullable expression, then $w \in L(E'[F'/G])$.*

Proof. A structural induction on E'. □

The third lemma states, intuitively, that if a path through a content model visits a nullable sublanguage expression F', it could as well bypass F' or other subexpressions of the content model.

Lemma 3.9 *Let E' be a marked model group and let F' be a nullable sublanguage expression of E'. If there is a word $tuv \in L(E')$ such that $u \in L(F') - \{\lambda\}$ and $(sym(t) \cup sym(v)) \cap sym(F') = \emptyset$, then (i) $t \in L(E')$, (ii) $u \in L(E')$, and (iii) $v \in L(E')$.*

Proof. The second claim follows immediately from $L(F') \subseteq L(E')$. Claims (i) and (iii) can be verified by a structural induction on E'. $\square$

Finally, the last lemma states that replacing a nullable *and* group, which is a sublanguage expression of an iterative expression by the corresponding *or* group does not change the language accepted by the iterative expression.

Lemma 3.10 *Let E' be a marked model group of the form $(G')^*$ or $(G')^+$, and let $F' = ({F_1}' \& \cdots \&{F_n}')$ be a nullable sublanguage expression of E' so that F' is not a subexpression of any other iterative subexpression of E'. Then $L(E'[F'/({F_1}'|\cdots|{F_n}')]) = L(E')$.*

Proof. Denote $({F_1}'|\cdots|{F_n}')$ by H'. Since $\lambda \in L({F_i}')$ for all $i = 1,\ldots,n$, we have $L(H') \subseteq L(F')$. Thus $L(E'[F'/H']) \subseteq L(E')$ by Lemma 3.4.

Let then $w \in L(E')$. Let $w = w_1 \ldots w_m$, where $w_i \in L(G') - \{\lambda\}$. We show by induction on m that $w \in L(E'[F'/H'])$. When $m = 0$, the claim holds by Lemma 3.8. Then assume that $m > 1$ and that the claim holds for $m - 1$. It suffices to show that $w_m \in L(E'[F'/H'])$, since then $w \in L(E'[F'/H'])$ by Lemma 3.7. If $sym(w_m) \cap sym(F') = \emptyset$, then $w_m \in L(G'[F'/H'])$ by Lemma 3.8, and $L(G'[F'/H']) \subseteq L(E'[F'/H'])$. Otherwise let $w_m = tuv$, where $u \in L(F') - \{\lambda\}$ and $(sym(t) \cup sym(v)) \cap sym(F') = \emptyset$. Then $t, v \in L(G')$ by Lemma 3.9, and $t, v \in L(G'[F'/H'])$ by Lemma 3.8. Now $u = u_1 \ldots u_n$, where $u_i \in L(F'_{\phi(i)})$ for each $i = 1,\ldots,n$ and some permutation ϕ of $\{1,\ldots,n\}$. This means that $u_i \in L(H')$ for each $i = 1,\ldots,n$. Since H' is a sublanguage expression of $G'[F'/H']$, we have $u_i \in L(G'[F'/H'])$ for each $i = 1,\ldots,n$. Then $u = u_1,\ldots,u_n \in L(E'[F'/H'])$, and $tuv \in L(E'[F'/H'])$ by Lemma 3.7. $\square$

Now we are ready to give our third theorem, which states that a nullable *and* group occurring as a sublanguage expression of an iteration can be replaced by the corresponding *or* group.

Theorem 3.11 *Let E' be a marking of a content model E and let $F' = ({F_1}'\&\cdots\&{F_n}')$ be a subexpression of E'. If F' is a nullable sublanguage expression of an iteration, then*

1. $L(E'[F'/({F_1}'|\cdots|{F_n}')]) = L(E')$ *and*
2. $\chi(E'[F'/({F_1}'|\cdots|{F_n}')])$ *is deterministic if and only if E is deterministic.*

Proof. Denote $({F_1}'|\cdots|{F_n}')$ by H'. Let G' be the smallest iterative subexpression of E' for which $L(F') \subseteq L(G')$. Then $L(G'[F'/H']) = L(G')$ by Lemma 3.10, and $L(E'[F'/H']) = L(E')$ by Lemma 3.4. Then, since the languages are equal, there are words $uxv, uyr \in L(E')$ with $x \neq y$ and $\chi(x) = \chi(y)$ if and only if $uxv, uyr \in L(E'[F'/H'])$. $\square$

We believe that the $\mathcal{E}(E)$ transformation could be extended to handle also some incomplete *and* groups. As a simple example, consider $E = a\&b?$, for which we saw $\mathcal{E}(E) = (a, b?)|(b?, a)$ to be non-deterministic. A possible strategy is to prevent the first subexpression in each *or* branch from accepting the empty string, in this case as follows: $(a, b?)|(b, a)$. We will not elaborate these possibilities further in this paper.

4 The length of expanded and groups

In this section we analyze the length of an expanded content model, where the *and* operators have been eliminated according to the method presented in the previous section. We develop the analysis in a slow and pedagogical fashion.

Marcy Thompson formulated the problem as follows [16]:

> *Find a convenient and correct way to indicate the length/complexity of an unambiguous expansion of an unambiguous & group of length n. Assume that the & group consists only of element names.*

So, let us consider the expansion of a model group $E = a_1 \& a_2 \& \cdots \& a_n$, where $n \geq 1$, and $a_1, \ldots, a_n$ are distinct element names. As we saw in Section 3, its expansion $\mathcal{E}(E)$ is of the form

$$(a_1, \mathcal{E}(a_2 \& a_3 \& \cdots \& a_n)) \mid (a_2, \mathcal{E}(a_1 \& a_3 \& \cdots \& a_n)) \mid \ldots \mid (a_n, \mathcal{E}(a_1 \& a_2 \& \cdots \& a_{n-1})) .$$

That is, $\mathcal{E}(E)$ consists recursively of n expansions of an *and* group of $n-1$ elements, each preceded by one of the element names. Let us denote the length of the expansion of an *and* group of n elements by $L(n)$. When $n = 1$, the group is just a single element, which serves as its own expansion of length 1. We get the following recurrence:

$$\begin{aligned} L(1) &= 1 \\ L(n) &= nL(n-1) + n, \text{ when } n > 1. \end{aligned}$$

To get a closed-form approximation of $L(n)$ for $n > 1$, we start by repeatedly expanding $L(n)$ according to the recurrence:

$$\begin{aligned} L(n) &= nL(n-1) + n \\ &= n\big((n-1)L(n-2) + n - 1\big) + n \\ &= n(n-1)L(n-2) + n(n-1) + n \end{aligned}$$

We notice that the expansion starts to involve a generalization of factorial known as *falling factorial powers.* We adopt from [12] the convenient notation $n^{\underline{k}}$, which is defined for $n \geq k \geq 1$ by

$$\underbrace{n(n-1)\cdots(n-k+1)}_{k \text{ factors}} .$$

Notice that $n^{\underline{1}} = n$ and $n^{\underline{n}} = n^{\underline{n-1}} = n!$.

Using the factorial power notation, we continue working on the recurrence:

$$\begin{aligned} L(n) &= n^{\underline{2}} L(n-2) + n^{\underline{2}} + n \\ &= n^{\underline{2}} \big((n-2)L(n-3) + (n-2)\big) + n^{\underline{2}} + n^{\underline{1}} \\ &= n^{\underline{3}} L(n-3) + n^{\underline{3}} + n^{\underline{2}} + n^{\underline{1}} \\ &\vdots \\ &= n^{\underline{i}} L(n-i) + \sum_{k=1}^{i} n^{\underline{k}} \\ &\vdots \\ &= n^{\underline{n-1}} L(1) + \sum_{k=1}^{n-1} n^{\underline{k}} = n! + \sum_{k=1}^{n-1} n^{\underline{k}} \end{aligned}$$

We seem to be getting towards a closed form for $L(n)$. It remains to find is a closed form approximation for the sum $\sum_{k=1}^{n-1} n^{\underline{k}}$.

There are some excellent textbooks, like [12], presenting mathematical tools for approximating expressions arising in the analysis of algorithms. Cormen, Leiserson and Rivest suggest in their textbook [10, page 47]:

Sometimes, a good upper bound on a series can be obtained by bounding each term of the series, and it often suffices to use the largest term to bound the others.

Since each of the $n-1$ terms of our sum are at most $n!$, following the above suggestion gives us that

$$L(n) \leq n! + (n-1)n! = nn!$$

The above estimate feels too rough, though, since $\mathcal{E}(E)$ is clearly a more compact presentation than a trivial listing of $n!$ alternative *seq* groups of length n. We can pursue further by choosing another representation for $n^{\underline{k}}$. Since

$$n^{\underline{k}} = \frac{n!}{(n-k)!} \ ,$$

we next try bounding

$$\sum_{k=1}^{n-1} \frac{n!}{(n-k)!} = n! \sum_{k=1}^{n-1} \frac{1}{(n-k)!} = n! \sum_{k=1}^{n-1} \frac{1}{k!}$$

Cormen, Leiserson and Rivest continue:

The technique of bounding each term in a series by the largest term is a weak method when the series can in fact be bounded by a geometric series.

Let us follow their second suggestion. Since $k! \geq 2^{k-1}$ for each natural number k, we have

$$n! \sum_{k=1}^{n-1} \frac{1}{k!} \leq n! \sum_{k=1}^{n-1} \frac{1}{2^{k-1}} = n! \sum_{k=0}^{n-2} \frac{1}{2^k} = n! \frac{1-(\frac{1}{2})^{n-2}}{\frac{1}{2}} = 2n!(1 - \frac{1}{2^{n-2}}) \leq 2n!$$

The above estimate gives us

$$L(n) \leq n! + 2n! = 3n!$$

We can still get a slightly tighter approximation. Since $n! + n! \sum_{k=1}^{n-1} \frac{1}{k!} = n! \sum_{k=0}^{n-1} \frac{1}{k!}$, this is obtained, surprisingly, by bounding the sum $\sum_{k=0}^{n-1} \frac{1}{k!}$ from above by the corresponding *infinite* sum $\sum_{k=0}^{\infty} \frac{1}{k!}$. From the calculus we know that it is the Taylor's series for e^x at $x = 1$, where $e = 2.71828\ldots$ is the base of natural logarithms.[2] This gives us that

$$\sum_{k=0}^{n-1} \frac{1}{k!} < \sum_{k=0}^{\infty} \frac{1}{k!} = e$$

and

$$L(n) < en! \tag{1}$$

The last approximation is also a tight bound, since by Taylor's theorem the difference between the sum and its approximation is

$$R_n = e - \sum_{k=0}^{n-1} \frac{1}{k!} = \frac{1}{(n-1)!} \int_0^1 e^t (1-t)^{n-1} dt \ . \tag{2}$$

The second mean-value theorem for integrals applied to (2) gives that

$$R_n = \frac{e^p}{(n-1)!} \int_0^1 (1-t)^{n-1} dt = \frac{e^p}{(n-1)!} \Big/_0^1 -\frac{1}{n}(1-t)^n = \frac{e^p}{n!} \ \text{ for some } 0 \leq p \leq 1.$$

[2] Actually I had to write a small program for computing values of $\sum_{k=1}^{n-1} \frac{1}{k!}$ before retaining this fact.

This means that the error of approximating $L(n)$ by $en!$ is at most e, and the relative error approaches to zero quickly as n increases. This is seen also in Table 1, which presents a comparison between the first few computed values of $L(n)$ and $en!$. The table shows that the elimination of *and* groups of length more than three already starts yielding content models which are too complicated for human readers.

n	$L(n)$	$en!$
1	1	2.718...
2	4	5.436...
3	15	16.309...
4	64	65.238...
5	325	326.193...
6	1,956	1,957.162...
7	13,699	13,700.140...
8	109,600	109,601.122...
9	986,409	986,410.099...
10	9,864,100	9,864,100.995...

Table 1: Comparison of $L(n)$ and $en!$ for $n \leq 10$.

A question that remains is whether the lengthening of the content model caused by the elimination of the *and* operators could be even worse, if the content model contained further sub-groups. Since the lengthening is caused by the & operator alone, it is sufficient to consider nested *and* groups. Let E be an *and* group consisting of n element names arranged in k subordinate *and* groups for some $1 < k < n$:

$$\underbrace{(a_1 \& \cdots \& a_{n_1})}_{n_1 \text{ elements}} \& \cdots \& \underbrace{(a_{k_1} \& \cdots \& a_{k_{n_k}})}_{n_k \text{ elements}}$$

Now we are asking, whether E could produce a longer expansion than a flat *and* group of n symbols, which we analyzed above.

Let n_i be the *length* of the ith subexpression, i.e., the number of element names in it, for $i = 1, \ldots, k$. We can assume that the length of at least one of the subexpressions is greater than one, since otherwise E would be essentially identical to a flat *and* group.

Notice that the expansion of a flat *and* group of n element names contains original element names only, and they all have a symmetrical role. Therefore each original element name appears in the expansion exactly $L(n)/n$ times. The k sub-groups of the top-level *and* group E have a similar role than element names in a flat *and* group. This gives the following expression for the length of the expansion of E:

$$\frac{L(k)}{k} \sum_{i=1}^{k} L(n_i)$$

Let's see how this behaves on some small values of n. The shortest *and* group that we need to consider is of length three. It can be grouped only in two subordinate & groups, e.g., as follows:

$$(a_1 \& a_2) \& a_3$$

Eliminating the & operators from the above expression gives the model group

$$(((a_1, a_2)|(a_2, a_1)), a_3)|(a_3, ((a_1, a_2)|(a_2, a_1))) \ ,$$

whose length 10 is given correctly by

$$\frac{L(2)}{2}(L(2) + L(1)) = 2 \cdot (4 + 1) \ .$$

We notice that the result is shorter than the expansion of a corresponding flat *and* group, whose length is $L(3) = 15$. Similarly, $a_1 \& a_2 \& a_3 \& a_4$ can be arranged in either two subexpressions of lengths 3 and 1, or 2 and 2, or in three subexpressions of lengths 2, 1 and 1. The lengths of the corresponding expansions of these model groups are

$$\begin{aligned}
\frac{L(2)}{2}(L(3)+L(1)) &= 2\cdot(15+1) = 32, \\
\frac{L(2)}{2}(L(2)+L(2)) &= 2\cdot(4+4) = 16, \text{ and} \\
\frac{L(3)}{3}(L(2)+L(1)+L(1)) &= 5\cdot(4+1+1) = 30.
\end{aligned}$$

Again, each of these is less than $L(4) = 64$. Further, the different ways of partitioning an *and* group of length 5 in subordinate & groups gives for the expansion of each of the groups a length smaller than $L(5) = 325$:

$$\begin{aligned}
\frac{L(2)}{2}(L(4)+L(1)) &= 2\cdot(64+1) = 70 \\
\frac{L(2)}{2}(L(3)+L(2)) &= 2\cdot(15+4) = 38 \\
\frac{L(3)}{3}(L(3)+L(1)+L(1)) &= 5\cdot(15+1+1) = 85 \\
\frac{L(3)}{3}(L(2)+L(2)+L(1)) &= 5\cdot(4+4+1) = 45 \\
\frac{L(4)}{4}(L(2)+L(1)+L(1)+L(1)) &= 16\cdot(4+1+1+1) = 112
\end{aligned}$$

So, arranging short *and* groups differently does not lead to longer expansions. What about *and* groups containing more than five element names?

Since the length of the longest subordinate *and* group can be at most $n-k+1$, we have that

$$\frac{L(k)}{k}\sum_{i=1}^{k} L(n_i) \le \frac{L(k)}{k} k L(n-k+1) = L(k)L(n-k+1) \ .$$

By equation (1) on page 11 we have that

$$L(k)L(n-k+1) < ek!\, e(n-k+1)! = \frac{ek!\, en!}{n^{\underline{k-1}}} \ .$$

When $n \ge 6$, the above value is less than $en!$, since then[3] $n^{\underline{k-1}} \ge (n/2)k^{\underline{k-1}} \ge 3k! > ek!$. This completes our analysis, and allows us to safely state that transforming an SGML content model containing n occurrences of element names into an XML content model can give a content model containing at most $en!$ element names. Our earlier considerations showed that this bound is indeed tight.

5 Automata for and groups

In this section we consider the size of an automaton for recognizing the language $L(E_n)$ of an *and* group $E_n = a_1 \& \cdots \& a_n$ of length n. We saw in Section 4 that the length of the deterministic regular expression $\mathcal{E}(E_n)$ is $O(en!)$. A possible question is whether $L(E_n)$ could have a description of polynomial length as a an automaton.

[3] To tell the truth, I had checked beforehand that this approximation requires n to be larger than 5.

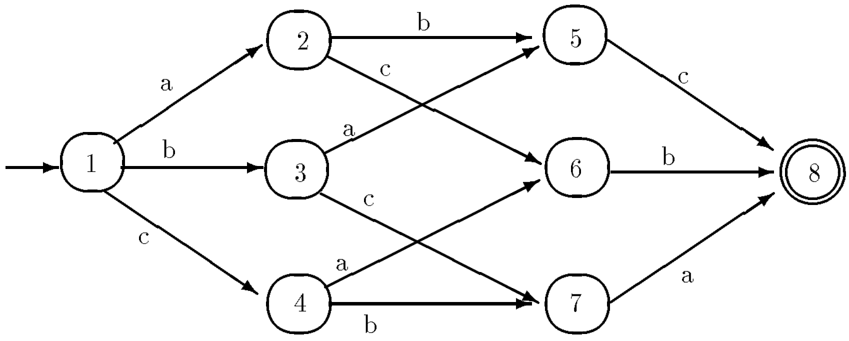

Figure 2: Deterministic automaton M_3 for $L(a\&b\&c)$.

If one translates $\mathcal{E}(E_n)$ into a nondeterministic automaton using some of the standard techniques [5], like the one due to Thompson [15], the size of the resulting automaton is not smaller than expression $\mathcal{E}(E_n)$. On the other hand, deterministic automata are sometimes exponentially more succinct than deterministic regular expressions (see, e.g., [9]). We show next that the size of a minimal deterministic automaton M_n recognizing $L(a_1\&\cdots\&a_n)$ is not factorial but still exponential in n.

As an example, consider the language $L(a\&b\&c)$, which can be recognized by DFA M_3 shown in Figure 2. We ignore the failure states and failure transitions from the figure and from our analysis, since they can be thought to be built into the process which interprets an automaton. It is straightforward to check that M_3 is a minimal DFA. (See, e.g.,[2], [1], or [17].)

By generalizing from M_3 we see that a minimal DFA M_n accepting $L(E_n)$ consists of levels $0, \ldots, n$, which form a partition of the states of M. Level 0 consists of a single initial state and level n consists of a single final state. The states at levels $k = 0, \ldots, n-1$ have (non-failure) transitions to states at level $k+1$ only. Each state s at level k accepts suffixes of words in $L(a_1\&\cdots\&a_n)$ consisting of $n-k$ distinct symbols of $\{a_1, \ldots, a_n\}$, and it can be reached from the initial state by any word consisting of the k distinct symbols of $\{a_1, \ldots, a_n\}$ which do not occur in the prefixes accepted by s. For example, consider automaton M_3 in Figure 2. Level 0 of M_3 contains only the initial state 1, level 1 $= \{2, 3, 4\}$, level 2 $= \{5, 6, 7\}$, and level 3 contains only the final state 8. State 3 at level 1, for example, accepts words ac and ca, which are suffixes of length $3 - 1 = 2$ of words of $L(a\&b\&c)$, which do not contain symbol b occurring on a path leading to state 3.

Since each state s at level k is responsible for accepting suffixes of words of $L(a_1\&\cdots\&a_n)$ after k symbols of $\{a_1, \ldots, a_n\}$ have been seen, level k contains $\binom{n}{k}$ distinct states. By summing over the levels we get that the number of states of M_n is

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n .$$

Each state at level $k \in \{0, \ldots, n-1\}$ has $n-k$ non-failure transitions to states at the next level. Therefore we get that the number of non-failure transitions of M_n is

$$\begin{aligned} \sum_{i=0}^{n-1} \binom{n}{i}(n-i) &= n\sum_{i=0}^{n-1} \binom{n-1}{i} \\ &= n2^{n-1} . \end{aligned}$$

A complete specification of automaton M_n would also include a failure state and n transitions leaving from each state. Therefore it would simply have $2^n + 1$ states and $n(2^n + 1)$ transitions.

So, we have seen that the size of a minimal automaton that recognizes the language $L(a_1\&\cdots\&a_n)$ is smaller than the factorial length of the regular expression $\mathcal{E}(a_1\&\cdots\&a_n)$, but still exponential.

6 Conclusion

If one wants to move in document production from SGML to its simplified variant XML, one needs to translate SGML document type definitions to corresponding XML document type definitions. Especially one needs to eliminate the *and* operator from content models. It is known that the elimination of *and* operators is not always possible because of the requirement of deterministic content models. We have presented transformations, which allow a wide class of SGML content models to be translated to corresponding XML content models. We have also analyzed the lengthening effect of *and* group elimination and derived a tight upper bound of $O(en!)$ on the number of symbol occurrences in the result of eliminating an *and* group of n symbols. We also showed that a minimal deterministic automaton accepting $L(a_1\&\cdots\&a_n)$, where $a_1,\ldots,a_n$ are distinct symbols, has 2^n states and $n2^{n-1}$ transitions, excluding the failure state and the failure transitions.

We only mentioned possibilities of transforming *and* groups, which cannot be handled by the methods analyzed in this paper. These further transformation possibilities would deserve a more thorough inspection.

Brüggemann-Klein and Wood have studied the relation of model groups and regular expressions extensively [6, 8, 9]. It could be rewarding to apply their approach to study the issues considered in this paper. Especially, the notions of star freeness and sublanguage expressions of iterations are related and could possibly be combined to get a more unified treatment of the issues surrounding SGML and XML content models.

References

[1] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilers, principles, techniques, and tools.* Addison-Wesley, 1986.

[2] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation, and Compiling, Vol. I: Parsing.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972.

[3] T. Bray, J. Paoli, and C.M. Sperberg-McQueen, editors. *Extensible Markup Language (XML) 1.0.* W3C Recommendation 10-February-1998. The latest version is available at `http://www.w3.org/TR/REC-xml`.

[4] A. Brüggemann-Klein. Formal models in document processing. Habilitationsscrift, Faculty of Mathematics at the University of Freiburg, 1993.

[5] A. Brüggemann-Klein. Regular expressions into finite automata. *Theoretical Computer Science*, 120:197–213, 1993.

[6] A. Brüggemann-Klein. Unambiguity of extended regular expressions in SGML document grammars. In Th. Lengauer, editor, *Algorithms — ESA 93*, pages 73–84. Springer-Verlag, 1993.

[7] A. Brüggemann-Klein. Compiler-construction tools and techniques for SGML parsers: Difficulties and solutions. Universität Freiburg, Institut für Informatik, May 1994.

[8] A. Brüggemann-Klein and D. Wood. The validation of SGML content models. *Mathematical and Computer Modelling*, 25(4):73–84, February 1997.

[9] A. Brüggemann-Klein and D. Wood. One-unambiguous regular languages. *Information and Computation*, 142:182–206, 1998.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill / MIT Press, 1990.

[11] C. F. Goldfarb. *The SGML Handbook.* Clarendon Press, Oxford, 1990.

[12] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley, 1989.

[13] International Organization for Standardization. *ISO 8879: Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML)*, October 1986.

[14] P. Kilpeläinen and D. Wood. SGML and exceptions. In C. Nicholas and D. Wood, editors, *Principles of Document Processing, Proceedings of the Third International Workshop, PODP'96*, pages 39–49. Springer-Verlag, 1997. Preliminary version available as technical report HKUST-CS96-30 of The Hong Kong University of Science & Technology, Department of Computer Science, June 1996.

[15] K. Thompson. Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, June 1968.

[16] M. Thompson. Re: XML and SGML. A posting on Usenet newsgroup comp.text.sgml, December 1996.

[17] D. Wood. *Theory of Computation.* John Wiley & Sons, Inc., 1987.