UNIVERSITY
OF TRENTO

**DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY**

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

# COMPLEXITY OF CONTEXT-FREE GRAMMARS WITH EXCEPTIONS, AND THE INADEQUACY OF GRAMMARS AS MODELS FOR XML AND SGML

Romeo Rizzi

April 2002

Technical Report # DIT-02-0058

Also: *Markup Languages: Theory and Practice* 3 (1) 2001.

# *Complexity of Context-free Grammars with Exceptions*
## *and the inadequacy of grammars as models for XML and SGML*

Romeo Rizzi

April 26, 2002

*ITC-Irst*
*Via Sommarive, 18*
*I-38050 Trento-Povo, Italy*
rizzi@irst.itc.it

**Abstract**

The Standard Generalized Markup Language (SGML) and the Extensible Markup Language (XML) allow authors to better transmit the semantics in their documents by explicitly specifying the relevant structures in a document or class of documents by means of document type definitions (DTDs). Several authors have proposed to regard DTDs as extended context-free grammars expressed in a notation similar to extended Backus–Naur form. In addition, the SGML standard allows the semantics of content models (the right-hand side of productions) to be modified by exceptions. Inclusion exceptions allow named elements to appear anywhere within the content of a content model, and exclusion exceptions preclude named elements from appearing in the content of a content model. Since XML does not allow exceptions, the problem of exception removal has received much interest recently. Motivated by this, Kilpeläinen and Wood have proved that exceptions do not increase the expressive power of extended context-free grammars and that for each DTD with exceptions, we can obtain a structurally equivalent extended context-free grammar. Since their argument was based on an exponential simulation, they also conjectured that an exponential blow-up in the size of the grammar is a necessary devil when purging exceptions away. We prove their conjecture under the most realistic assumption that NP-complete problems do not admit non-uniform polynomial-time algorithms. Kilpeläinen and Wood also asked whether the parsing problem for extended context-free grammars with exceptions admits efficient algorithmic solution. We show the NP-completeness of the very basic problem: given a string $w$ and a context-free grammar $G$ (not even extended) with exclusion exceptions (no inclusion exceptions needed), decide whether $w$ belongs to the language generated by $G$. Our results and arguments point up the limitations of using extended context-free grammars as a model of SGML, especially when one is interested in understanding issues related to exceptions.

**Key words**: exceptions, context-free grammars, computational complexity, exponential blow-up, XML, SGML.

## 1 Introduction

The Standard Generalized Markup Language (SGML) [7, 8] is an international standard (ISO 8879) for document definition and interchange. SGML has been proposed to promote

the interchangeability and application-independent management of electronic documents by providing a syntactic metalanguage for the definition of textual markup systems. SGML is widely used in government and industry, and it has received increased attention from academia since HTML evolved to a formal application of SGML. The Extensible Markup Language (XML) [3] is, essentially, a simplified and more restrictive version of SGML. The role of XML is to allow SGML documents to be served, received, and processed on the Web. XML is the proposed syntactic metalanguage for the specification of document grammars for W3 documents. A main goal and driving rationale behind the design of SGML and XML is to allow authors to better transmit the semantics in their documents by explicitly specifying the relevant structures in a document or class of documents by means of document type definitions (DTDs). In spite of a warning by Prescod (see the document "Formalizing XML and SGML Instances with Forest Automata Theory" [13], available at [17]), several authors [9, 11, 2, 5, 1] model DTDs as extended context-free grammars expressed in a notation that is similar to extended Backus–Naur form. In addition, the SGML standard allows the semantics of content models (the right-hand side of productions) to be modified by exceptions. In SGML there are two kinds of exceptions: *inclusion exceptions* allow named elements to appear anywhere within the content of a content model, and *exclusion exceptions* preclude named elements from appearing in the content of a content model. Exceptions provide a powerful shorthand notation for DTD authors and thus are used in most industry and government standard DTDs. Unlike SGML DTDs, XML DTDs do not allow exceptions but proposals to incorporate some exception mechanisms in XML also are still somewhat under debate as a means of reducing the difficulty of translating SGML DTDs into XML DTDs. Indeed, the problem of how to remove exceptions from a given DTD has until now defeated attempts to obtain a general solution, in spite of the strong interest and commitment involved on this front (see e.g. [9, 11, 10, 18]). Motivated by this, Kilpeläinen and Wood [1] proved that exceptions do not increase the expressive power of extended context-free grammars and that for each DTD with exceptions, we can obtain a structurally equivalent extended context-free grammar. Since their argument was based on an exponential simulation, they also conjectured that an exponential blow-up in the size of the grammar is a necessary devil when purging exceptions away. We prove their conjecture under the most realistic assumption that NP-complete problems do not admit non-uniform polynomial-time algorithms. In [1], Kilpeläinen and Wood also posed the following question: does an extended context-free grammar with exceptions allow efficient algorithmic solutions of the most common language recognition problems associated with it? We give a strong negative answer to this question by showing the NP-completeness of the very basic problem: given a string $w$ and a context-free grammar $G$ (not even extended) with exclusion exceptions (no inclusion exceptions needed), decide whether $w$ belongs to the language generated by $G$. Our results and arguments point up the limitations of taking extended context-free grammars as a model of SGML, especially when one is interested in understanding issues related to exceptions. In the document "Formalizing XML and SGML Instances with Forest Automata Theory" [13], available at [17]), Prescod spent a word of warning on the fact that extended context-free grammars do not provide a faithful formalization of the way SGML and XML parsing actually works in practice. Our negative results show that this dichotomy is not just a matter of current technology, since there would be intrinsic drawbacks in treating DTDs as grammars when parsing.

## 2 Background and notation

A context-free grammar is a rewriting system in which the left-hand side of each rule must be a single symbol, so that symbols are rewritten "context-freely". More formally, a *context-free grammar* $G = (N, \Sigma, P, S)$ is made of two disjoint finite alphabets $N$ (the set of *nonterminal symbols*) and $\Sigma$ (the set of *terminal symbols*), of a *sentence symbol* $S$ and of a finite set $P$ of production schemas. Every *production schema* $\pi$ in $P$ has the form $A \mapsto \omega$, where $A \in N$ and $\omega \in V^*$ is a string over the alphabet $V := N \cup \Sigma$. The string $\omega$ is called the *content model* of $\pi$ and denoted by $w(\pi)$. As usual, capital letters denote nonterminals, lowercase letters denote terminals, Greek letters are for strings, $\epsilon$ for the empty string, $|w|$ stands for the length of string $w$ and $G := \sum_{\pi \in P}(|w(\pi)| + 1)$ expresses the length of a reasonable encoding of $G$. The language $\mathcal{L}_G$ *generated* by $G$ is the set of those strings $\omega \in \Sigma^*$ which can be derived from $S$ through a sequence of applications of production schemas in $P$. The following result [14] is fundamental in parsing theory.

**Theorem 2.1** *Given a context-free grammar $G = (N, \Sigma, P, S)$ and a string $w$, deciding whether $w \in \mathcal{L}_G$ can be done in $O(|G| \times |w|^3)$ deterministic time and $O(|G| \times |w|^2)$ space.*

Extended context-free grammars are context-free grammars in which the right-hand sides of productions, also called *content models*, are regular expressions. A *regular expression over* $V$ is a special string on $V \cup \{\emptyset, \epsilon, \cup, ^*, (, )\}$ which is used to describe a *language on* $V$, i.e. a subset of $V^*$. Nothing but what we list here below is a *regular expression*:

- $\emptyset$ is a regular expression and describes the empty language $\mathcal{L}_\emptyset = \emptyset$;
- $\epsilon$ is a regular expression and describes the void string language $\mathcal{L}_\epsilon = \{\epsilon\}$;
- for each $\nu \in V$, $\nu$ is a regular expression and describes the language $\mathcal{L}_\nu = \{\nu\}$;
- when $F$ and $G$ are regular expressions, then $F \cup G$ is a regular expression and describes the language $\mathcal{L}_{F \cup G} = \mathcal{L}_F \cup \mathcal{L}_G$;
- when $F$ and $G$ are regular expressions, then $FG$ is a regular expression and describes the language $\mathcal{L}_{FG} = \{\omega_F \omega_G \,|\omega_F \in \mathcal{L}_F, \omega_\in \mathcal{L}_G\}$;
- when $F$ is a regular expression, then $F^*$ is a regular expression and describes the language $\mathcal{L}_{F^*} = \mathcal{L}_F{}^*$, where the star denotes the Kleene operator, that is, language $\mathcal{L}^*$ is made of those strings which are concatenations of any number of strings in $\mathcal{L}$;
- parenthesis are used just for grouping: when $F$ is a regular expressions, then $(F)$ is a regular expression and describes the language $\mathcal{L}_{(F)} = \mathcal{L}_F$.

An *extended context-free grammar* $G = (N, \Sigma, P, S)$ is defined the same as a context-free grammar except that each production schema $\pi \in P$ has the form $A \mapsto exp$, where $exp$ is a regular expression over $V$. When $\omega = \omega_1 A \omega_2 \in V^*$, $\pi = A \mapsto exp \in P$ and $\alpha \in \mathcal{L}_{exp}$, then the string $\omega_1 \alpha \omega_2$ can be derived from the string $\omega$.

Every context-free language (one for which there exists a context-free grammar which generates it) is clearly an extended context-free language. To see the contrary, first note

that we can always assume that $\emptyset$ does not appear in any production (unless $\mathcal{L}_G = \emptyset$), and then replace all productions $A \mapsto exp_1 \cup exp_2$ with productions $A \mapsto exp_1$ and $A \mapsto exp_2$, all productions $A \mapsto exp_1 exp_2$ with productions $A \mapsto EXP_1 EXP_2$, $EXP_1 \mapsto exp_1$ and $EXP_2 \mapsto exp_2$, all productions $A \mapsto exp^*$ with productions $A \mapsto EXP*$, $EXP* \mapsto \epsilon$ and $EXP* \mapsto expEXP*$, and, for the sake of precision, all productions $A \mapsto (exp)$ with productions $A \mapsto exp$. This construction is actually polynomial (and in fact linear), hence Theorem 2.1 above leads to the following well known result (see [16]).

**Theorem 2.2** *Given an extended context-free grammar $G = (N, \Sigma, P, S)$ and a string $w$, deciding whether $w \in \mathcal{L}_G$ can be done in $O(|G| \times |w|^3)$ deterministic time and $O(|G| \times |w|^2)$ space.*

An *extended context-free grammar $G = (N, \Sigma, P, S)$ with exceptions* is similar to an extended context-free grammar except that the production schemas in $P$ have the form $A \mapsto exp\,[+I][-X]$, where $A$ is in $N$, $exp$ is a regular expression over $V$, and $I$ and $X$ are subsets of $N$. The intuitive idea is that a derivation of a string $\omega$ from the nonterminal $A$ and started using the production schema $A \mapsto exp\,[+I][-X]$ must not involve any nonterminal in $X$, yet $\omega$ may contain, in any position, strings that are derivable from nonterminals in $I$. When a nonterminal is both included and excluded, its exclusion overrides its inclusion.

More formally, where $I, X \subseteq V$, a language $\mathcal{L}$ with inclusions $I$, denoted by $\mathcal{L} + (I)$, is the language that consists of the strings in $\mathcal{L}$ with arbitrary strings from $I^*$ inserted into them. The language $\mathcal{L}$ with exclusions $X$, denoted by $\mathcal{L} - (X)$, is the language of those strings in $\mathcal{L}$ which do not contain any symbol in $X$. Notice that $(\mathcal{L} + (I)) - (X) \subseteq (\mathcal{L} - (X)) + (I)$, but the converse does not hold in general. In the sequel, $\mathcal{L} + (I) - (X)$ stands for $(\mathcal{L} + (I)) - (X)$. Following Kilpeläinen and Wood [1], we formally describe the global effect of exceptions by attaching exceptions to nonterminals and by defining derivations from nonterminals with exceptions. We denote a nonterminal $A$ with inclusions $I$ and exclusions $X$ by $A_{[+I][-X]}$. When $\omega$ is a string of a regular expression over $V$, we denote by $\omega_{(I;X)}$ the string obtained from $\omega$ by replacing every appearance of every nonterminals $A$ in $\omega$ with $A_{[+I][-X]}$. Let $\omega = \omega_1 A_{[+I][-X]} \omega_2$ be a string over terminal symbols and nonterminal symbols with exceptions. Then the string $\omega_1 \alpha' \omega_2$ can be derived from $\omega$ whenever the following two conditions hold:

1. $A \mapsto exp\,[+I_A][-X_A]$ is a production schema in $P$;
2. $\alpha' = \alpha_{(I \cup I_A; X \cup X_A)}$ for some string $\alpha \in \mathcal{L}_{exp} + (I \cup I_A) - (X \cup X_A)$.

Observe that the second condition reflects the idea that exceptions are propagated and cumulated by derivations. Finally, the language $\mathcal{L}_G$ of an extended context-free grammar $G$ with exceptions consists of the strings in $\Sigma^*$ derivable from the sentence symbol with empty inclusions and exclusions.

Although exceptions seem to be a context-dependent feature in that legal expansions of a nonterminal depend on the context in which the nonterminal appears, Kilpeläinen and Wood [1] showed that exceptions do not extend the descriptive power of extended context-free grammars (and hence of context-free grammars); they did so by giving a transformation that produces an extended context-free grammar which is structurally equivalent to an extended context-free grammar with exceptions. The transformation propagates exceptions to

production schemas and modifies their associated regular expressions to capture the effect of exceptions. We refer to their paper [1] for more details but recall that, as they observed, their transformation may increase the number of productions by a factor which is exponential in the number of the exceptions. They conjectured that this exponential blow-up is unavoidable. We show this to be the case, unless the whole of problems in NP can be solved non-uniformly in polynomial time, contrary to the common belief. They also explicitly posed the practical question whether the recognition problem for a generic extended context-free grammar with exceptions is efficiently solvable by other means, like parsing on the fly. (Indeed, existing SGML parsers like the Amsterdam SGML parser [15] handle exceptions in an interpretive manner. The names of excluded elements are kept in a stack, which is consulted whenever the parser encounters a new element.)

We show that a quite restricted form (context-free grammars with only exclusion exceptions in input) of this parsing problem is already NP-complete. This is sufficient to conclude (under the weaker and more popular assumption that P$\neq$ NP) that no efficient procedure can translate a context-free grammar with (only exclusion) exceptions $G_{\mathrm{EXC}}$ into an equivalent extended context-free grammar $G_{\mathrm{NO\ EXC}}$, since otherwise, by combining such a procedure with the one whose existence is stated in Theorem 2.1, we would get an efficient algorithm for a problem which captures the whole complexity of NP (in jargon, NP-complete). However, a compact grammar $G_{\mathrm{NO\ EXC}}$ could still exist, even if we are not provided effective means to derive a description of $G_{\mathrm{NO\ EXC}}$ from a description of $G_{\mathrm{EXC}}$. To deny this possibility we have to rely on the somewhat stronger assumptions as mentioned above and as expressed more precisely in Theorem 3.3.

# 3 Two negative results

Let $B$ be a boolean formula in conjunctive normal form. Let $X = \{x_1, \ldots, x_n\}$ be the set of variables and $C = \{c_1, \ldots, c_m\}$ be the set of clauses in $B$. The following problem is perhaps the most famous among the NP-complete [6] ones.

**Problem 3.1 (3SAT)** *Given a boolean formula $B$ in conjunctive normal form and with precisely three literals per clause, is there a satisfying truth assignment for $B$?*

To face a 3SAT instance on $n$ variables, consider the context-free grammar $G_n$ with exclusion exceptions, over alphabet $\Sigma = \{x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n, \vee, (, ), \wedge\}$ and with nonterminal symbols $S, L, X_1, \ldots, X_n, \overline{X}_1, \ldots, \overline{X}_n$. To specify $G_n$, we take $S$ as the sentence symbol of $G_n$, and, for $i = 1, \ldots, n$, provide a bunch of "clause-generating" productions

$$\begin{array}{ll} S \mapsto (L \vee L \vee X_i) \wedge S\ [-\overline{X}_i] & S \mapsto (L \vee L \vee \overline{X}_i) \wedge S\ [-X_i] \\ S \mapsto (L \vee X_i \vee L) \wedge S\ [-\overline{X}_i] & S \mapsto (L \vee \overline{X}_i \vee L) \wedge S\ [-X_i] \\ S \mapsto (X_i \vee L \vee L) \wedge S\ [-\overline{X}_i] & S \mapsto (\overline{X}_i \vee L \vee L) \wedge S\ [-X_i] \end{array}$$

a bunch of "last-clause-generating" productions

$$\begin{array}{ll} S \mapsto (L \vee L \vee X_i) & S \mapsto (L \vee L \vee \overline{X}_i) \\ S \mapsto (L \vee X_i \vee L) & S \mapsto (L \vee \overline{X}_i \vee L) \\ S \mapsto (X_i \vee L \vee L) & S \mapsto (\overline{X}_i \vee L \vee L) \end{array}$$

and a bunch of "literal-generating" productions

$$X_i \mapsto x_i \qquad \overline{X}_i \mapsto \overline{x}_i$$
$$L \mapsto x_i \qquad L \mapsto \overline{x}_i$$

Note that $G_n$ depends solely on $n$, the number of boolean variables occurring in the input boolean formula $B$.

**Theorem 3.2** *Given a context-free grammar $G$ with exclusion exceptions and a string $w$, deciding whether $w \in \mathcal{L}_G$ in NP-complete.*

*Proof:* Clearly, the problem is in NP, since one can always guess a derivation, and derivations of $w$ have length at most linear in $|w|$. Let $B = (\hat{x}^{1,1} \vee \hat{x}^{1,2} \vee \hat{x}^{1,3}) \wedge \ldots \wedge (\hat{x}^{m,1} \vee \hat{x}^{m,2} \vee \hat{x}^{m,3})$ be a given instance of 3SAT, where, for $h = 1, \ldots m$ and $k = 1, \ldots 3$, the $k$-th literal in clause $h$ (i.e. $\hat{x}^{h,k}$) stands either for $x_i$ or for $\overline{x}_i$, for some $i = 1, \ldots n$. For example, a given instance $B$ could be represented by the string $\sigma_B = (x_2 \vee x_3 \vee \overline{x}_1) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3)$. To prove the theorem, it suffices to prove the following claim.

***Claim:*** $B$ admits a satisfying truth assignment if and only if $\sigma_B \in \mathcal{L}_{G_n}$.

**only if.** If $B$ admits a satisfying truth assignment $f$, then, under $f$, each clause contains at least one true literal. Associate one such literal $\hat{x}_c$ to each clause $c$. Generate the clauses from left to right, using a production out of the second bunch of productions (the bunch of "last-clause-generating" productions) to generate the very last clause, after all the previous clauses have already been generated using productions out of the first bunch (the bunch of "clause-generating" productions). More precisely, if $\hat{x}_c$ is positive use one of the three productions on the left, whereas if $\hat{x}_c$ is negative use one of the three productions on the right. (Which one of the three productions to use is dictated by the position of $\hat{x}_c$ into $c$.) Note that, for $i = 1, \ldots, n$, the exclusion exception $[-X_i]$ ($[-\overline{X}_i]$) can be imposed somewhere in the proposed derivation only if $x_i$ is false (true) under $f$. But then the proposed derivation never uses the nonterminal $-\overline{X}_i$ ($X_i$), hence no conflict with an imposed exception occurs along the proposed derivation.

**if.** Assume there exists a derivation of $\sigma_B$ in $G_n$. Follow this derivation from left to right, clause after clause. When rule $B \mapsto (L \vee L \vee X_i) \wedge B\ [-\overline{X}_i]$ is employed, then define $f(x_i) = true$. When rule $B \mapsto (L \vee L \vee \overline{X}_i) \wedge B\ [-X_i]$ is employed, then define $f(x_i) = false$. Act analogously whenever other "clause-generating" rules from the first bunch are encountered and also when a "last-clause-generating" rule is eventually employed. In the end, when each clause has been generated, complete the definition of assignment $f$ arbitrarily. Note that, thanks to the exclusion exceptions imposed, no inconsistency in the definition of $f$ can have occurred. Moreover, $f$ is a truth assignment which makes at least one literal true for every clause. □

We say that a language $\mathcal{L}$ *can be solved non-uniformly in polynomial-time* when there exist a polynomial $p(.)$, and a family $\{A_n \,|n \in \mathbb{N}\}$ of algorithms such that for every $n \in \mathbb{N}$

- $A_n$ recognizes the language of the strings in $\mathcal{L}$ of length $n$ in time at most $p(n)$;
- the length of the description of $A_n$ is at most $p(n)$.

A common belief (in [12], at page 269, see the general discussion about Conjecture B and follow the references given) quite close to the P$\neq$ NP one, is that NP-complete problems can not be solved non-uniformly in polynomial time.

**Theorem 3.3** *Unless all problems in NP can be solved non-uniformly in polynomial time, there exists a family* $\{G_n | n \in \mathbb{N}\}$ *of context-free grammars with exclusion exceptions such that for no polynomial* $p(.)$ *and for no family* $\{H_n | n \in \mathbb{N}\}$ *of context-free grammars we have both* $\mathcal{L}_{G_n} = \mathcal{L}_{H_n}$ *and* $|H_n| \leq p(n)$ *for all* $n \in \mathbb{N}$.

*Proof:* Combine the claim in the above proof with Theorem 2.1. □

Theorem 3.3 gives evidence that, for context-free grammars, exceptions are a powerful shorthand notation, in that eliminating them may cause exponential growth in the size of the grammar. Unfortunately, Theorem 3.3 does not actually address the motivation behind the conjecture of Kilpeläinen and Wood [1]: it does not provide final evidence that such exponential growth would also result from eliminating exceptions from SGML DTDs. Indeed, Theorem 3.2 clearly indicates that extended context-free grammars with exceptions offer only an imperfect match for SGML. (As pointed out by Prescod, SGML parsing is indeed efficient, while our results suggest that no efficient parsing procedure exists for the general family of extended context-free grammars with exceptions). In the following section, we list those key points of distinction between the two classes of languages that we believe are most responsible for the dramatic difference in complexity behavior. In this way, we hope to hint at some of those aspects which possibly limit the validity and/or suitability of grammars with exceptions as a model for SGML and XML.

# 4 SGML versus context free grammars with exceptions

In this section, we indicate what in our opinion are the two main reasons for the discrepancy between SGML and its extended context-free grammar with exceptions model. In the attempt to encode $G_n$ into SGML, one quickly realizes the following differences to be significant.

- whenever a production is applied, SGML gives trace of this event dropping a tag (to be true, there are some rules to allow some tag omission, but indeed, the rules are just there so that we can actually infer the missing tags);
- in SGML, for any non-terminal A there is a single production with A on the left-hand side. This point of distinction becomes relevant when exceptions are kept under consideration.

Of these elements of distinction, the first concerns both XML and SGML and had already been indicated by Prescod in the document "Formalizing XML and SGML Instances with Forest Automata Theory" [13], available at [17]. While we perceived as crucial also the second element of distinction in the case of SGML, we do not believe the same to hold for XML, since XML does not allow exceptions and hence a single legal rule $A \mapsto w_1|w_2$ can often express the rules $A \mapsto w_1$ and $A \mapsto w_2$.

## 5 Final Remarks

The role of XML is to allow documents to be served, received, and processed on the Web. Even though it is now clear that context-free grammars with exceptions have their limits in modeling SGML, the ideas in the proof of Theorem 3.2 can possibly help in understanding which aspects or ingredients of the exception mechanism should definitely not be included, for efficiency reasons, into the XML standard. (Or, at least, we do not know of other formal steps or partial results on this front.) Indeed, the use of exceptions begins to appear controversial even for authors, in that, although exceptions are useful and even handy at first, they add significantly to the complexity of authoring DTDs as their size and complexity grows. Even if this phenomenon had been sometimes denied, or attributed to poor style in the use of exceptions, it is shown in [9], on the basis of an empirical analysis, that the complexity of some DTDs is approaching (or has exceeded) manageable limits given existing tools for designing and understanding them, and it is nowadays commonly believed that only partial solutions (see [9, 10, 11]) to this problem can be attempted. Our results now provide a formal justification of the occurrence of these problems which are known to imply high costs for DTD design and corresponding problems with quality.

Some links pointing to relevant material about SGML exceptions can be found in the Web [18]. Forest-Regular Languages are the (partial) model proposed by Murata Makoto and Paul Prescod to describe SGML. Their publications on this topic are collected by Robin Cover in a web page [17].

## Acknowledgments

Thanks are due to the anonymous referees. In particular, one of the two referees worked out XML and SGML DTDs for 3SAT on 3 variables on the lines of the reduction given in Section 3. I have made part of his original material available on the World Wide Web [19] since it could help people from a different background in getting a grasp on the general behavior of the reduction and some of its subtleties, but also in appreciating the vague points of distinction suggested in Section 4.

## References

[1] P. Kilpeläinen and D. Wood, SGML and XML Document Grammars and Exceptions, Report HKUST-TCSC-99-01, January 25, 1999. To appear in Information and Computation, 2001.

[2] J. Albert, D. Giammarresi, and D. Wood. Extended context-free grammars and normal form transformations. In Automata Implementation: Third International Workshop on Implementing Automata, WIA '98, Heidelberg, Germany, 1998. Springer-Verlag.

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

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

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

[6] M.R. Garey and D.S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-completeness*, Freeman, San Francisco (1979).

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

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

[9] R.W. Matzen, A New Generation of Tools for SGML. In Markup Languages: Theory and Practice 1/1 (Winter 1999) 47-74.

[10] R.W. Matzen and G.E. Hedrick, A New Tool for SGML with Applications for the World Wide Web. Paper presented at SAC '98 - 1998 ACM Symposium on Applied Computing. February 27 - March 1, 1998, Marriott Marquis, Atlanta, Georgia, U. S. A.

[11] R.W. Matzen and G.E. Hedrick, Unraveling Exceptions, Conference Proceedings: SGML/XML 97, Washington D.C., December, 1997.

[12] C.H. Papadimitriou, *Computational complexity* EATCS Monographs on Theoretical Computer Science. Addison-Wesley Publishing Company, Reading, MA, (1994).

[13] P. Prescod, Formalizing SGML and XML Instances and Schemata with Forest Automata Theory. http://www.prescod.net/forest/shorttut/ 1998

[14] S. Sippu and E. Soisalon-Soininen, *Parsing Theory* Vol. 1: Languages and Parsing, EATCS Monographs on Theoretical Computer Science. Springer-Verlang, Berlin, New York, Tokyo, (1988).

[15] J. Warmer and S. van Egmond. The implementation of the Amsterdam SGML parser. Electronic Publishing, 2(2):65–90, July 1989.

[16] D. Wood. Theory of Computation. John Wiley and Sons, Inc., 1987.

[17] SGML/XML and Forest/Hedge Automata Theory
http://www.oasis-open.org/cover/hedgeAutomata.html

[18] Some relevant URL's concerning exceptions in SGML and XML
http://www.oasis-open.org/cover/topics.html

[19] URL to the examples provided by referee 2 and a comment which was common to both referees
http://www-math.science.unitn.it/ rrizzi/refereeSGMLexc/