連載

XML 誕生秘話

村田　真

日本 IBM（株）東京基礎研究所
国際大学研究所
mmurata@trl.ibm.com

4　形式言語理論と XML

□■はじめに

XML の誕生にまつわる事情を伝えてきたこの連載も，今回で終わりになる．今回は，形式言語理論と XML のかかわりについて述べてみたい．

形式言語理論は，コンパイラや自然言語処理のための基礎理論として知られているが，XML の基礎理論でもある．XML 文書とは何かは，文法を与えることによって厳密に定式化される．しかし，XML 1.0 における文法の役割は，コンパイラや自然言語処理における役割とは大きく違うと私は考えている．そして，この違いは XML の成功に貢献しただけではなく，XML の発展すべき方向を示していると考えている．

□■ XML の 2 つの文法

XML 1.0 には 2 種類の文法が存在する．1 つは，XML 1.0 仕様書に示されている BNF である．もう 1 つは，ユーザが作成する DTD である．

(1) BNF

XML 1.0 仕様書では BNF 記法を用いて，XML 文書とはどんな文字列であるかを規定している．たとえば，

```
[42] ETag   ::= '</' Name S? '>'
```

という規則は，終了タグ（非終端記号 ETag）がどんな文字列であるかを定めている．右辺に現れる Name は，名前を表す文字列が何であるかを示す非終端記号である．S? は空白文字列があってもなくてもよいことを示す．これらの規則からなる BNF が，XML 1.0 仕様書の核心である．

XML 文書の構文解析とは，第一義的には，文字列をこの BNF と照合し，成功したかどうかを報告することである．しかし，成功・失敗を表す 1 ビットだけでは実用にはならない．アプリケーションプログラムは，構文解析の結果として，要素・属性・テキストからなる木構造を受け取る．別の言い方をすれば，BNF は，XML 文書に木構造を与えていることになる☆1．

(2) DTD

特定の用途に XML を用いるには，どんな名前の要素・属性によって何を表すかをあらかじめ約束しておく．たとえば，ベクタ図形を表すために開発された SVG1.0[2] では，線を表すために line という名前の要素を用い，長方形を表すために rect という名前の要素を用いるという約束がある．要素や属性がどのような木階層を形作るか（たとえば rect は line の子要素になれるかどうか）もあらかじめ約束しておく．このような約束の集まりを XML ではスキーマという．

スキーマを表現するために，XML 1.0 は DTD という機構を提供している．ある用途のための XML 文書がどんなものであるかは，DTD を作成することによって規定することができる．たとえば，SVG 1.0 仕様書にある DTD には，以下のような記述がある．この記述は，glyphRef という名前の要素が利用できること，この要素が子要素を持たないことを表している．

```
<!ELEMENT glyphRef EMPTY>
```

DTD が扱うのは，要素・属性・テキストからなる木構造であり，文字列ではない．たとえば，glyphRef 要素を文字列“<glyphRef/>”によって表現するか，文字

☆1 残念ながら，XML 文書の表す木構造にどのような情報が含まれるかが完全に合意されているわけではない．その結果，XML Information Set[1] は玉虫色の仕様になっている

列“<glyphRef></glyphRef>”によって表現するかは SVG 用の DTD は規定しない.

XML 1.0 における妥当性検証とは，BNF に基づく構文解析で得られた木構造を DTD と照合し，成功したかどうかを報告することである．BNF に基づく構文解析は木構造を生成したが，妥当性検証が新たな構造を生成することはない.

(3) 他技術との比較

コンピュータ技術において，2 段階の文法または解析処理が現れる例は多い．プログラミング言語における字句解析と構文解析，自然言語処理における形態素解析と構文解析，ASN.1 におけるユーザ定義ファイル（抽象構文）と基本符号化規則（転送構文），データベースにおける論理スキーマと物理スキーマなどがある．2 つの解析処理が適用されるという点で，XML はこれらの技術と似ているといえる.

しかし，XML においては 2 つの解析処理の役割分担が，これらの技術とは異なる．XML では，下位の文法によって得られるデータ構造を重視する．実際，XML のための汎用 API（DOM や SAX など）が扱うのは，BNF に基づく構文解析によって得られる木構造であり，DTD に基づく解析結果ではない．一方，他技術では，上位の文法によって得られるものが重要であり，下位の文法はそのための道具に過ぎない（形態素解析はその唯一の例外であろう）．字句解析や形態素解析によって得られるのはシンボル列でしかない.

以上をまとめると，XML には次の原則が成り立つという特徴がある．これを generic XML の原則と呼ぶ.

> アプリケーションに渡される木構造は，上位の文法を用いることなしに，下位の文法と XML 文書だけから決定できる.

この違いは，XML に利点と欠点をもたらした．欠点として，XML 文書が冗長であること[☆2]，汎用 API が低レベルで使いにくいこと[☆3]などがある．利点としては，上位の文法の有無に関係なく，汎用 API ひいては汎用 XML ツールが利用可能なことが挙げられる．言い換えると, DTD が何であっても Xerces（http://xml.apache.org/）などのパーサは必ず使えるし，汎用の XML エディタやブラウザも必ず動く．XML が普及した理由の 1 つは，generic XML の原則にあると私は考えている．冗長で低レベルであっても，「上位の文法に依存しない操作が常に可能」という簡単さ・相互運用性がものを言ったのである.

□■ SGML から XML 1.0 へ

原理的な説明から XML 誕生秘話に戻ろう．BNF に従った解析と DTD に従った解析が分離したのは，XML 1.0 になってからである．SGML においては，両者はまったく分離していなかった．SGML 文書は必ず DTD を持つことになっており，DTD を持たない SGML 文書は許されていなかった.

SGML の設計にはそれなりの理由がある．SGML が設計された当時，タグを入力するためのコストは大きな問題であると思われていた．そこで，DTD を見れば分かるタグを省略することによって，このコストを下げようとした．例として，段落のたびに開始タグ と終了タグ を入力するのを避けることを考える．要素 p の終了タグが省略可能であると DTD で指定し，p は入れ子にならないように DTD を作成すれば，SGML では終了タグ を省略することができる．具体的には，2 つの終了タグ を持つ以下の文書

 これは段落． これは段落．

を，終了タグ を省略して

 これは段落． これは段落．

のように短縮することができる.

しかし，タグの省略はいくつかの問題を引き起こした．SGML 文書の解析が難しくなったという問題点はよく指摘される．より本質的な問題点は，DTD に合わない SGML 文書を扱えなくなったことである．たとえば，

 これは段落． これは段落．

という文書を正しく解析することはできない．2 つ目の を見た時点で，終了タグが勝手に補われてしまうためである.

DTD に合った SGML 文書しか扱えないという問題は，深刻な結果をもたらした．第 1 に，SGML 文書を作成するには，よく考え抜いた DTD をあらかじめ作成しておくことが必須となった．これは SGML を使い始める上で最初の難関となった．第 2 に，どんなに考え抜かれた DTD であっても，想定されていない事態は必ず発生した．SGML 文書を入力しているときに想定外の事態が発生すれば，文書作成そのものを諦めるか，DTD にその場しのぎの変更を加えることを余儀なくされた.

XML 1.0 の設計は，SGML についての反省に基づいている．XML が成功するとすれば，DTD を持たない XML 文書のほうが多くなったときであり，それが可能なよう

☆2 スキーマを見れば分かることでも文字列によって繰り返し表現するため.
☆3 スキーマに書かれている情報を汎用 API は扱えないため.

に XML を設計しなければならない．W3C XML WG の電話会議で，そういう主張が繰り返された．

その結果，XML では，DTD を持たない文書や DTD に合わない文書は積極的に認められた．BNF だけに従った解析が必ず可能なように，タグの省略は禁止された．XML パーサとしては，BNF と DTD の両方を用いるもの以外に，BNF だけを用いるものが認められた．これらの変更が，XML にある多くの改良点のうち最も本質的なものだと私は思う．

□■原則を破る機構

実は，XML 1.0 には generic XML の原則を破る機構がいくつかある．SGML 陣営を XML に引き入れるためには必要な機構だともいえるが，悔いが残る点でもある．それらの機構のうち 2 つをここでは取り上げる．

(1) 解析対象実体

解析対象実体は一種のマクロであり，DTD で宣言される．XML 文書で解析対象実体が参照されると，マクロが展開される．したがって，解析対象実体を参照する XML 文書については，DTD を読み込まない限り，木構造を正しく作成することができない．DTD を読み込まずに解析すると，未展開の参照が残ってしまう．

(2) 属性のデフォルト値と正規化

DTD には，属性のデフォルト値を指定することができる．この属性を持たない要素が現れたとき，DTD に指定されたデフォルト値が自動的に補われる．当然，DTD を読まなければ，木構造からデフォルト値は欠落する．

XML 文書に指定された属性値を，DTD での指定に従って正規化（空白文字の削除など）する機構もある．DTD を読まなければ正規化は行われないので，DTD を処理したかどうかによって，得られる木構造が変わってしまう．

□■ DTD を持たないサブセット

DTD を持たない XML 仕様を作ろうという提案は，XML 1.0 制定途中から何度もあった．XML 本体からは DTD を分離しておき，DTD とは別のスキーマ言語をいつか作ろうという意図であった．この分離を行えば，前節の問題はすべて解消する．

この提案についてはさまざまの衝突があったが，いったん以下のように決着した．まず，SGML との互換性を保つため，XML 1.0 では DTD を残すことに決めた．そして，XML 1.0 の制定後に，DTD を持たないサブセット仕様を作ることになり，W3C XML Syntax WG が実作業を担当することになった．私もこの WG のメンバであった．

しかし，W3C XML Syntax WG はサブセットを制定しないまま解散した．今でも，W3C の Technical Architecture Group においてサブセットの必要性が議論されているが，早期に決着する見込みはない．サブセットができない理由は，機能をどこまで削除するかについて合意が取れないからである．DTD の削除だけではなく，属性を削除しようという意見，処理命令を削除しようという意見，Unicode 以外の文字コードを禁止しようという意見などがある．もちろん，削除すべきではないという意見もあり，議論は収束しない．

仕様制定に携わった経験のある人ならよく知っていることだが，仕様制定とはタイミングである．タイミングを逃せば，世間の注目を失ってしまったり，当事者が燃え尽きてしまったりする．元々 XML の設計は，SGML との互換性に最大限の配慮をして始まり，1997 年中の完成を予定していた．1997 年後半になって XML は注目を浴びるようになり，SGML との互換性はそれほど重要ではなくなったが，あのときに軌道修正して DTD を削除すれば議論は発散し，XML 勧告そのものが流れてしまったかもしれない．したがって，1998 年 2 月の時点において，generic XML の原則を破る機構を含んだままで XML 1.0 を出版することには必然性があったと思う．しかし，XML 1.0 を出版したことによって，XML サブセットの制定は困難になったともいえる．

□■ XML のためのスキーマ言語とホオートマトン

DTD に代わるスキーマ言語として，W3C XML Schema と RELAX NG が注目されている．generic XML の原則を破る方向で進んでいるのがスキーマ言語 W3C XML Schema であり，原則に従って発展したのがスキーマ言語 RELAX NG である．W3C XML Schema は理論なしに設計されたが，RELAX NG は，木オートマトン理論を根底に置いている．

(1) W3C XML Schema

DTD の後継として，W3C はスキーマ言語 W3C XML Schema[3] を開発した．制定にあたったのは W3C XML Schema WG であり，初期には私もメンバであった．

W3C XML Schema は，BNF によって与えられる木構造ではなく，スキーマに基づく検証によって得られる PSVI（Post-Schema-Validation Infoset）を中心として XML 技術を再構成することを目標としている．すなわち，generic XML の原則（および利点と欠点）は W3C XML Schema によって破壊される．

XML コミュニティには，この再構成を推進する人と，猛烈に反対する人がいる．両者の対立は，貴族とボヘミアンの対立と言われることがある．ボヘミアンは，RELAX NG を支持することが多い．むろん私はボヘミアン陣営に属しており，W3C XML Schema の最初のドラフト発行にメンバとしてただ 1 人反対したことがある．

本稿ではこれ以上は論じないが，日本語による参考文献として文献 4) を挙げておく．

なお，W3C が W3C XML Schema を必ず使うというわけではない．HTML 2.0 の仕様書 [5] や RDF 仕様書 [6] は，W3C XML Schema を採用せず，RELAX NG を採用している．

(2) 木オートマトン

BNF に基づく構文解析が木構造を返すのなら，スキーマは木構造を扱う文法とみなすことができる．木構造を扱うための形式言語理論としては，木オートマトンが 1960 年代から研究されてきた．現在は，木オートマトンがスキーマのための標準的な理論となっており，XML プログラミング言語やデータベース言語の設計に木オートマトンを使うことはほぼ常識となっている☆4．XML は理論が役立つ数少ない分野なのである．木オートマトンについての解説は多いが，日本語によるものとして文献 8）を挙げておく．

スキーマの理論として木オートマトンを用いることのメリットは大きい．第 1 に，DTD は局所木言語（local tree language）という木言語クラスしか扱えないが，木オートマトンをスキーマとすることによって正規木言語（regular tree language）という木言語クラスを扱うことができる．後者は，前者をその一部として含んでいる．現実的・実用的なスキーマであって，局所木言語では表現できず，正規木言語としてしか表現できないものが存在する☆5．第 2 に，正規木言語のクラスは，ブール演算などの重要な演算について閉じている．これは，XML 用のプログラミング言語や検索言語における型検査の基盤となる．

一方，木オートマトンをスキーマとすることは，従来とは異なる検証アルゴリズムが必要になることを意味する．従来，文字列を解析するために用いられてきた構文解析手法（LL, LR, チャートパーサ，Early のアルゴリズムなど）は文脈自由文法のためのものであり，これらを木オートマトンにそのまま適用することはできない．実際，近年開発された検証アルゴリズムは，非決定的トップダウン木オートマトンと非決定的ボトムアップ木オートマトンを組み合わせたもの [9]，derivative を巧妙に利用したもの [10] になっている．

(3) RELAX NG

スキーマ言語 RELAX NG[11] は，W3C XML Schema に対抗するスキーマ言語である．元々は標準化団体 OASIS で開発されたが，ISO/IEC JTC 1 でも規格直前の段階にあり，ほぼ自動的に JIS 規格になる．HTML 2.0 や RDF を始めとして，W3C でも利用が進んでいる．

RELAX NG は私も当事者の 1 人であり，いろいろと書きたいことはあるが，この連載の範囲外であろう．1 つだけ書いておきたいのは，形式言語理論なしに RELAX NG はあり得なかったということである．RELAX NG は generic XML の原則（すなわち，「スキーマによる妥当性検証はアプリケーションに渡す構造に影響を与えない」）に従っており，木オートマトンを基盤として設計されている．

スキーマ言語技術は RELAX NG によって大きく前進したが，汎用 API が低レベルで利用しにくいという欠点はまだ解消していない．解消するための技術として JAXB[12] や Relaxer[13] などのスキーマコンパイラ技術が広く知られているが，木オートマトンを型として導入した XML プログラミング言語・検索言語も盛んに研究されている．また，木オートマトンに基づく yacc というべき RelaxNGCC[14] も注目されるアプローチである．いずれも木オートマトンを用いているため，これらは RELAX NG とよく整合する．

□■おわりに

XML は広く普及した技術だが，まだ発展途上にある技術でもある．今後の発展のためには，XML をきちんと理解した上での斬新な試みが必要である．この記事によって，そのような試みがいささかでも増えれば，これにまさる喜びはない．

XML についての記事はこれまで何度も書いてきたが，この連載では他では書けなかったことを書くことができた．しかし，XML 技術の詳細，たとえば名前空間にまつわる事情など，書けなかったことも多い．これらの話題については，また別の機会もあるだろう．

参考文献

1) Cowan, J. and Tobin, R.（ed.）: XML Information Set, W3C Recommendation（2001）.
2) Ferraiolo, J.（ed.）: Scalable Vector Graphics（SVG）1.0, W3C Recommendation（2001）.
3) Fallside, D .（ed.）: XML Schema Part 0: Primer, W3C Recommendation,（2001）.
4) 川俣　晶 : XML における「ボヘミアンと貴族の階級闘争」を読み解く, http://www.atmarkit.co.jp/fxml/tanpatsu/24bohem/01.html（2003）
5) Axelsson et al.（ed.）: XHTML 2.0, W3C Working Draft（2003）.
6) Beckett, D.（ed.）: RDF/XML Syntax Specification（Revised）, W3C Working Draft（2002）.
7) Shin, K. : Some Equivalence between Multi-level Layout Processes and Single-level Layout Proceses, Principles of Document Processing（1991）.
8) 守屋悦朗 : 形式言語とオートマトン，サイエンス社（2001）.
9) Murata, M., Lee, D. and Mani, M.: Taxonomy of XML Schema Languages Using Formal Language Theory, Extreme Markup Language（2001）.
10) Clark, J. : An Algorithm for RELAX NG Validation, http://www.thaiopensource.com/relaxng/implement.html（2002）.
11) RELAX NG Specification, OASIS Committee Specification（2001）.
12) Fialli, J. and Vajjhala, S.（eds.）: Java Architecture for XML Binding（JAXB）, JSR-000031（2003）.
13) 浅海智晴 : Relaxer, http://www.relaxer.org（1999-2003）.
14) 岡嶋大介 : RelaxNGCC, http://relaxngcc.sourceforge.net/（2003）.

（平成 15 年 6 月 13 日受付）

☆4 私の知る限り，木オートマトンの構造化文書への最初の応用は，富士ゼロックス（株）の申吉浩の 1991 年の仕事 [7] である．
☆5 SGML にある exclusion/inclusion は，正規木言語のごく一部を実現するための ad-hoc な機構である．