Technical Report

XML 文書に対する アクセシビリティ・ガイドライン適合性検証

高田喜朗[†]　　中村岳志　　関 浩之[‡]

[†]y-takata@is.naist.jp
[‡]seki@is.naist.jp

2004 年 10 月 5 日

Graduate School of Information Science
Nara Institute of Science and Technology

あらまし

身体能力や利用環境等の差異に関わらずより多くの人が World Wide Web（ウェブ）上の情報を利用できるようにすること（ウェブ・アクセシビリティ）の重要性が広く認識されつつあり，アクセシブルなウェブコンテンツを作成するための要件を述べたガイドライン，及び，作成したウェブコンテンツがガイドラインを満たすか否かを検証するツールが提供されている．本研究では，既存の検証ツールに比べて以下のような特長を持つ新しい検証ツールの実現を目的として，ガイドライン記述言語の提案及び検証ツールの実装を行う．(1) ガイドライン記述を検証系の外部から入力として与えるので，ガイドラインの変更が容易である．(2) 任意の XML 文書を検証対象にできる．(3) 比較的複雑な構文チェック等，ガイドラインの検証以外にも利用できる．本報告ではまず，ガイドライン記述言語 SGSL を提案し，その表現能力について考察する．そして，SGSL から XSLT への変換に基づく検証ツールの実装について述べる．最後に，日米の主要機関 40 ウェブサイト約 3,000 ページに対して本検証ツールを適用した例のほか，いくつかの文書形式に対して本ツールを適用した例について述べる．

1. まえがき

World Wide Web（ウェブ）は現在，一部の専門家だけでなく一般の人々にとっても主要な情報入手手段の一つとなっている．それに伴い，身体能力やウェブ利用環境の差異に関わらず，ウェブサイト上の情報を困難なく読める（または聴ける）ようにすることがサイト制作者に求められている [16]．このように普遍的に利用可能なウェブサイトのことをアクセシビリティの高い（またはアクセシブルな）ウェブサイトという．例えば，ほとんどの情報が画像だけで提供されているウェブページは，モバイル機器等の文字ベースの計算機を使っている利用者や，目が不自由なため読み上げソフトウェア等を使用している利用者にとっては利用が困難である．即ちアクセシビリティが低い．これに対し，ウェブページ作成者が各画像に対して代替テキスト（画像を利用できない環境において代わりに使用される文章）を与えた場合には，アクセシビリティが改善される．他にも，ウェブサイトをアクセシブルにするためのさまざまな要件が知られている [23]．

ウェブサイトのアクセシビリティを改善するために，さまざまな面での活動が行われている．例えば，アクセシブルにすることを義務付ける法律の制定，制作者の教育，新しい技術の研究開発，アクセシビリティガイドラインの策定などがある．アクセシビリティガイドライン（以降，単にガイドラインという）とは，ウェブページをアクセシブルにするために必要なさまざまな項目を述べたものである．最もよく知られているものとして，World Wide Web Consortium (W3C) が制定した Web Content Accessibility Guidelines (WCAG)[23] がある．他にも，企業や団体が発行した同様のガイドラインが多数存在する [16]．これらのガイドラインが提供する情報は非常に有益だが，多数のさまざまな項目が扱われているため，ガイドラインを参照しながらウェブページを作成することは制作者にとって労力を要する．また，アクセシビリティに関する問題は，制作者自身の環境では問題を生じないことが多く，見落とされやすい．その結果，依然としてアクセシビリティの低いウェブサイトは多く存在する．例えば，米国及び日本の主要な政府機関・民間団体のウェブサイトを対象とした調査 [4] では，WCAG 中の優先度 1（2.4 節を参照）のチェックポイントを満たしているページの数は調査対象全体の 35%だったと報告されている．

そこで，制作者の労力を軽減するため，与えられたウェブページがガイドラインに

適合しているか否かを自動検証するツールがいくつか提供されている．Bobby[27, 14]はそのようなツールの一つであり，与えられた HTML 文書が WCAG に適合しているか否かを検証する．Weblint[2], WebSAT[12], LIFT[20] などの HTML 文書解析ツールも，ガイドライン適合性検証のために用いることができる．

これらのツールは有用だが，特定のガイドライン集合（例えば WCAG）及び特定の文書形式（例えば HTML）を想定しているため，それら以外の任意の文書形式，任意のガイドラインに適用することは容易ではない．即ち，検証対象である文書形式及び検証したいガイドラインのいずれを変更する場合でも，ツール自体の修正が必要である．例えば，WCAG 中のチェックポイント「すべての画像に代替テキストを提供せよ」は画像を掲載できる任意の文書形式に適用可能であるが，既存のツールは，文書形式として HTML を想定して，「すべての `img` 要素に `alt` 属性が指定されているか否か」を調べるように，即ち HTML 専用にプログラムされている．したがって，プログラムを変更しない限り，他の文書形式には適用できない．また，自組織が独自に定義したガイドラインについて検証するといった用途のために，容易にガイドラインを変更・追加できることが望ましい．

本報告では，ガイドラインを記述するための簡明な言語 SGSL (Simple Guideline Specification Language) を提案し，与えられた文書が SGSL で書かれたガイドライン記述に適合しているか否かを検証するツールの実装について述べる [11]．本稿で述べる検証ツールは，既存の検証ツールに比べ以下のような特長を持つ．

- ガイドライン記述は検証ツールと分離されており，ガイドラインを変更する場合にツール自体を変更する必要がない．また，SGSL によってガイドラインが形式的に記述されるため，どのような検証が行われるのか明示的でわかりやすくなると考えられる．

- 任意の XML 文書を検証対象にできる．XML は HTML を含むさまざまな文書形式を表現できるメタ言語であるため，提案手法は十分広い範囲の文書に適用可能である．また，SGSL は，XML 文書に関するガイドラインを簡潔に記述できるよう設計されている．

- アクセシビリティ・ガイドライン適合性検証に限らず，任意の記入様式への適合性を検証するために使用できる．例えば，公的手続きなどに XML 文書が用いら

れる際に，ある項目への記入が必須であることや，ある 2 つの項目の内容が同じでなければならないといった複雑な記入様式への適合性 [9] を検証することができる．

以降の本稿の構成は以下のとおりである．2. では，SGSL の構文と意味を述べ，SGSL の表現能力について議論する．3. では実装した検証ツールについて説明する．4. では，検証実行例として，日米の主要機関 40 ウェブサイト約 3,000 ページに対して本検証ツールを適用した例について述べる．また，HTML 文書以外の文書形式への適用例として，SVG (Scalable Vector Graphics)[22] に関するアクセシビリティ検証，及び XBRL (eXtensible Business Reporting Language)[29] に関する構文規則適合性の検証を行った例について述べる．最後に 5. でまとめを述べる．

2. ガイドライン記述言語 SGSL

2.1 言語の設計方針

本稿で提案するガイドライン記述言語 SGSL の設計方針について述べる．本研究では，既存の検証ツールと同様に，文書の構文に基づくチェックポイント（例えば「すべての `img` 要素に `alt` 属性を指定せよ」）を記述するための言語として SGSL を設計し，「サイトの内容に合わせて最も明瞭で簡潔な文章にせよ」などのようなチェックポイントの検証はユーザに委ねるものとする．XML 文書の構文を定義するための言語として，文書型 (schema) 定義言語がいくつか存在する．代表的な文書型定義言語として，XML DTD, W3C XML Schema, RELAX NG などがあり，これらは正則木文法に基づいている [10]．しかし，これらの言語の表現能力は，ガイドラインを記述するのには十分でない．例えば，WCAG のチェックポイント「異なる URL を指すリンクに同じリンクフレーズを使わないこと」はこれらの言語では表現できない．そこで本研究では，SGSL を，XPath (XML Path Language)[24] の拡張として定義する．XPath は，XML 文書が表す木構造中の 1 頂点または頂点集合を指定するために用いられる言語である．さらに，頂点及び頂点集合に関する論理式も XPath で書くことができる．XPath を用いることにより，文書型定義言語に比べて複雑な条件を記述することができる．しかし，前述のチェックポイントを表現するためには，「与えられた頂点集合の中のすべての頂点が与えられた論理式を満たす」という言明を記述できる必要があるが，これは XPath では書くことができない．SGSL は，このような記述ができるよう XPath を拡張したものである．

2.2 XPath

ここでは XPath について簡単に述べる．詳細は [24] などを参照されたい．

一つの XML 文書は，要素及び属性を頂点とする木構造を表すものととらえられる．XPath は，主に，この木構造中の頂点集合を指定するための言語である．即ち（頂点集合型の）XPath 式 p を XML 文書 d に適用すると，d 中の頂点の部分集合が返される．

頂点集合型 XPath 式（経路式ともいう）は，UNIX のファイルパス名に類似した記

法で記述される．例えば /html/body/h1 という式を XML 文書 d に適用すると，d 中の根要素である html の子要素 body の子である h1 要素すべてからなる集合を返す（そのような要素がなければ空集合を返す）．属性を指定するときは，属性名の前に@を付ける（例えば /html/@lang）．1 頂点を表すデータ型はなく，1 要素からなる頂点集合で代用する．

子を表す区切り記号 / の代わりに，子孫を表す記号 // を使うことができる．例えば /html//h1 は，根要素 html の子孫である h1 要素からなる集合を返す．親，祖先，文書中で先に出現する要素などの中から要素を指定することもできる．例えば //a/ancestor::img は，任意の a 要素の祖先であるような img 要素からなる集合を返す．この ancestor のように探索範囲を指定する記述を軸 (axis) という．

経路式中，各要素名の後に [ ] で囲んだブール型 XPath 式を書くことができる．これを述語という．述語は，その式を真にする要素のみ選択するという意味を表す．例えば //table[@summary]/caption は，summary 属性をもつ任意の table 要素の子である caption 要素からなる集合を返す．ここでは頂点集合型の式@summary をブール型の式として用いている．この場合，頂点集合が空なら偽，空でなければ真を表す．

ブール型 XPath 式では，要素や属性のテキスト値に関する比較演算子（等しい (=)，等しくない (!=) など）を使うことができる．例えば //img[@alt="zzz"] は，値が zzz である alt 属性をもつ img 要素からなる集合を返す．また，//img[@alt=//table/@summary] のように比較演算子の両辺が頂点集合の場合も，比較はテキスト値に関して行われる．頂点集合 S_1, S_2 について，式 $S_1 = S_2$（または S_1 != S_2）は，テキスト値の等しい（または等しくない）要素の組 $e_1 \in S_1, e_2 \in S_2$ が存在するとき，かつそのときのみ，真となる．

2.3 SGSL

SGSL の構文を図 1 に示す．SGSL の中心的要素は，検証対象文書が満たすべき，または満たしてはいけない条件を表す論理式である．SGSL 論理式は，ブール型 XPath 式に，全称及び存在限量子を表す every 及び some 式，及び変数定義を表す let 式を加えたものであり，XPath の比較的単純な拡張といえる．論理演算子 and, or, not

チェックポイント ::= 論理式
| if 論理式 then warn (警告文)

論理式 ::= ブール型 XPath 式
| every 変数 in 頂点集合 satisfies 論理式
| some 変数 in 頂点集合 satisfies 論理式
| let 変数 := XPath 式 in 論理式
| 論理式 Bop 論理式
| not 論理式

頂点集合 ::= 頂点集合型 XPath 式

Bop ::= implies | or | and

図 1　SGSL の構文

は XPath にも存在するが，XPath からの拡張部分に適用できるよう，SGSL でも改めて定義している．

SGSL において，一つのガイドライン記述はチェックポイントの有限集合である．以降，d を検証対象文書，$f, f_1, \ldots$ を任意の論理式とする．図 1 で示すとおり，チェックポイントは 1 個の論理式 f_1，または，if f_2 then warn (s) の形式の式である．前者の場合，もし d が f_1 を満たさなければ，警告とチェックポイント名が出力される．後者の場合，d が f_2 を満たすとき，文字列 s が警告文として出力される．即ち f_1 は検証対象文書が満たすべき条件を表しており，f_2 は指定された警告文を出力すべき条件を表している．論理式 every \$$v$ in s satisfies f は，XPath 式 s が表す頂点集合の中のすべての頂点 v について論理式 f が満たされるならば真となる．同様に，some \$$v$ in s satisfies f は，頂点集合 s 中の少なくとも一つの頂点 v について f が満たされるならば真となる．let \$$v$:= p in f は，論理式 f の中で用いられる変数 v を XPath 式 p の評価結果に束縛することを表す．なお，SGSL 論理式中の XPath 式は，文脈頂点 (context node) に依存しない式，即ち相対経路式を (他の経路式の部分式である場合を除いて) 含まない式でなければならない．implies，or，and，not は通常のブール演算子と同じである．

表 1　WCAG チェックポイントの分類

	Full	Partial	Partial Once	Ask Once	Total
Priority 1	7	13	3	6	29
Priority 2	13	12	8	7	40
Priority 3	5	3	7	7	22
Total	25	28	18	20	91

2.4　ガイドライン記述例

多くのガイドラインは SGSL を使って簡潔に記述することができる．例えば，WCAG に関して Bobby が行っている検証内容（後述の Full, Partial, Partial Once に該当するチェックポイント 71 個）のうち，文書型指定（`<!DOCTYPE ...>`）に関するもの 1 個を除いてすべて SGSL で記述可能であった[1]．本節では，SGSL による WCAG チェックポイントの記述例をいくつか紹介する．以下，チェックポイント番号は，WCAG 及び Bobby で用いられているものである．

WCAG のチェックポイントの中には「最も明瞭で簡潔な文章にせよ（チェックポイント 14.1）」のように自動的に検証するのは困難なものも含まれる．Bobby で行われている分類を参考に，WCAG の各チェックポイントを表 1 のように 4 つのカテゴリ (Full, Partial, ...) に分類した．Full は自動検証可能なものを表す．Partial 及び Partial Once は，チェックポイントを満たしていない可能性がある場合に警告を出力するが，実際に満たしていないか否かの判断は利用者に任せるものを表す（例えば「色を再現できない環境でも同じ情報が得られるようにせよ（チェックポイント 2.1）」）．Partial はその箇所を行番号で示せるもの，Partial Once は文書全体に関する条件を表す．Ask Once は，検証対象文書の内容に関わらず常に利用者への注意として提示するものを表す（上記のチェックポイント 14.1 など）．なお，表 1 中の優先度 (priority) は，WCAG におけるチェックポイントの分類である．これについては 4.2 節の検証実行例で述べる．

[1] XPath のデータモデルは文書型指定を扱わないので，文書型指定が正当かどうかの判断は XML パーザに任せることにし，SGSL では扱わないことにした．

(例 1) 「すべての `img` 要素に `alt` 属性を指定せよ (チェックポイント 1.1-1, Full) 」

```
every $x in //img satisfies $x/@alt
```

チェックポイントとして論理式のみが書かれているので，この論理式は，検証対象文書 d が満たすべき条件を表している．即ち，d 中のすべての `img` 要素 `$x` が論理式 `$x/@alt` を満たしているならば d は適合，そうでなければ不適合となる．式 `$x/@alt` は，変数 `x` に束縛された `img` 要素が `alt` 属性を持つか否かを表すブール型 XPath 式である．

(例 2) 「代替テキストが 150 字を超える場合，別に説明を提供することを考慮せよ (チェックポイント 1.1-6, Partial) 」

```
if some $x in //@alt satisfies
      string-length($x)>150
   then warn("consider providing a
      separate description")
```

チェックポイントが `if` f_2 `then warn(` s `)` の形なので，f_2 が成り立つときかつそのときのみ警告文 s が出力される．即ち，検証対象文書中に文字列長が 150 を超える `alt` 属性が存在するとき，「consider providing a separate description 」という警告文が出力される．関数 `string-length` は XPath 1.0 の組み込み関数であり，引数として与えられた文字列の長さを返す．なお，この例の `if` 文の条件部は，XPath だけを使って `//@alt[string-length(.)>150]` と書くこともできる．

(例 3) 「異なる URL を指すリンクに同じリンクフレーズを使わないこと (チェックポイント 13.1-4, Full) 」

```
every $x in //a satisfies
   every $y in //a satisfies
      ($x/@href != $y/@href implies
         $x != $y)
```

検証対象文書 d 中のすべての `a` 要素の対 `$x`, `$y` が式 `$x/@href != $y/@href implies $x != $y` $\cdots$(*) を満たすときかつそのときのみ，d はこのチェックポイン

トに適合する．式 (*) は，変数 x, y に束縛された a 要素の href 属性の値が異なるならば，a 要素自身の値（リンクフレーズ）も異ならなければならないことを意味している．

(例 4) 「見出し（要素 h1, ..., h6）を正しく入れ子にすること（チェックポイント 3.5-1, Full）.」

```
every $x in //h1|//h2|...|//h6 satisfies
  let $y := $x/preceding*[self::h1|...|self::h6][1]
    in (($x/self::h6 implies $y[self::h5|self::h6])
    and ($x/self::h5 implies $y[self::h4|...|self::h6])
    and ($x/self::h4 implies $y[self::h3|...|self::h6])
    and ($x/self::h3 implies $y[self::h2|...|self::h6])
    and ($x/self::h2 implies $y))
```

上の式では，検証対象文書 d 中のすべての見出し（h1, ..., h6）要素 $x について，まず，$d$ 中において $x の直前に現れる見出し要素を変数 y に代入する．そして，見出し要素 $x と $y が正しい関係にあるかどうか調べる．例えば部分式 $x/self::h6 implies $y[self::h5|self::h6] は，$x が h6 要素であれば $y は h5 または h6 要素であることを要求している．$x が h3, ..., h5 の場合についても同様の部分式が書かれている．部分式 $x/self::h2 implies $y は，$x が h2 要素であれば $y が空でないこと，即ち $x が d 中の先頭の見出しではないことを要求している．

2.5 SGSL の表現能力

既に述べたように，2.4 節のチェックポイント例 2 は，if 文の条件部を XPath 式だけで書くことができる．同様に例 1 は，XPath 1.0 の組み込み関数 not を使って，not(//img[not(@alt)]) という XPath 式で表すことができる．例 4 はより複雑だが，付録 A. で述べた方法で XPath 式に変換することができる（ただし記述長は長くなる）．しかし，例 3 はこの方法では変換できない．

この節では，SGSL の表現能力が XPath より真に強いか否かについて考察する．2.4 節例 3 は XPath では記述できない（即ち SGSL の表現能力は XPath より真に強い）と著者らは予想しているが，現時点では明らかではない．簡単のため，XPath における not 以外の組み込み関数，及び数に関する演算子（大小比較と四則演算）を無視することにする．XPath からこれらを除いた部分クラスを $\mathcal{XP}$ と呼ぶ．また，XPath の任意の部分クラス $\mathcal{X}$ について，SGSL 中の XPath 式を $\mathcal{X}$ の式に限ったクラスを $\mathrm{SGSL}(\mathcal{X})$ とする．論理式のクラス間の表現能力の強弱を $\leq, \lneqq$ で表す．以下では，$\mathcal{XP}$ のいくつかの部分クラス $\mathcal{X}$ について，$\mathcal{X}$ と $\mathrm{SGSL}(\mathcal{X})$ の表現能力に差があるか調べる．$\mathcal{X} \leq \mathrm{SGSL}(\mathcal{X})$ は明らかなので，$\mathcal{X} = \mathrm{SGSL}(\mathcal{X})$ かどうかが問題となる．

近年，XPath 等の木構造データ向き問合せ言語の表現能力について活発に研究されている（[1, 6, 13, 17] など）．これらの研究では，一階述語論理 (FO) やその拡張との比較により，問合せ言語の表現能力を解析することがよく行われる．任意の $\mathcal{XP}$ 式はそれと等価な FO 式に変換できることが容易にいえる（[1, 13] など）．$\mathrm{SGSL}(\mathcal{XP})$ は $\mathcal{XP}$ に限量子を加えたものであるので，$\mathrm{SGSL}(\mathcal{XP})$ 式もまた等価な FO 式に変換できる．即ち $\mathcal{XP} \leq \mathrm{SGSL}(\mathcal{XP}) \leq \mathrm{FO}$ である．

Benedikt ら [1] は，XPath のいくつかの部分クラスについて表現能力を調べ，特に XPath の部分クラス $\mathcal{X}^{\uparrow}_{r,[]}$ と FO の部分クラス $\exists^{+}(child, desc)(c, s)$ が等価であることを示した（[1] の定理 1）．$\mathcal{X}^{\uparrow}_{r,[]}$ は，$\mathcal{XP}$ から，組み込み関数 `not`，属性値の比較演算子，及び兄弟間の順序関係に基づく軸（2.4 節例 4 に現れる `preceding` など）を除いたものに相当する．$\exists^{+}(child, desc)(c, s)$ は，FO から，兄弟間の順序関係，各頂点の属性値を表す関数，否定演算子及び全称限量子を除いたクラスである．このクラスでは，2.4 節例 1 を表現できないことが容易にわかる．仮に例 1 と等価な $\exists^{+}(child, desc)(c, s)$ 式 f が存在したとすると，f を満たす任意の XML 文書 t_1 について，`alt` 属性を持たない `img` 頂点を t_1 に新しく加えて得られる文書 t_2 も（f には全称限量子も否定演算子もないので）f を満たすこととなり，矛盾が生じる．2.4 節例 1 で示した SGSL 式は $\mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[]})$ に属すので，$\mathcal{X}^{\uparrow}_{r,[]} \lneqq \mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[]})$ がいえる．なお，[1] では，$\mathcal{X}^{\uparrow}_{r,[]}$ より広いクラスは扱われていない．

実際の XPath は，否定演算子に相当する組み込み関数 `not` を持つので，表現能力は $\mathcal{X}^{\uparrow}_{r,[]}$ より真に強い．実際，前述のように，2.4 節例 1 は $\mathcal{XP}$ 式 `not(//img[not(`

表 2　表現能力の比較

$$
\begin{array}{ccccc}
\mathcal{X}^{\uparrow}_{r,[]} & \lneqq & \mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[]}) & & \\
\mathcal{X}^{\uparrow}_{r,[],\neg} & = & \mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[],\neg}) & & \\
\mathcal{XP} & \leq & \mathrm{SGSL}(\mathcal{XP}) & \leq & \mathrm{FO}
\end{array}
$$

`@alt)])` で表せる．$\mathcal{X}^{\uparrow}_{r,[]}$ に関数 `not` を加えたクラスを $\mathcal{X}^{\uparrow}_{r,[],\neg}$ と呼ぶことにすると，$\mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[],\neg})$ の論理式は付録 A. の方法で等価な $\mathcal{X}^{\uparrow}_{r,[],\neg}$ 式に変換できる．即ち，$\mathcal{X}^{\uparrow}_{r,[],\neg} = \mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[],\neg})$ である．

以上をまとめると表 2 のようになる．$\mathcal{XP} \lneqq \mathrm{SGSL}(\mathcal{XP})$ であると予想しているが，現在は明らかでない．XPath の 2 つの部分クラス $\mathcal{X}^{\uparrow}_{r,[]}, \mathcal{X}^{\uparrow}_{r,[],\neg}$ については，SGSL との差があるかどうか解決している．$\mathcal{X}^{\uparrow}_{r,[],\neg}$ より広い XPath の部分クラス $\mathcal{X}$ について $\mathcal{X} \lneqq \mathrm{SGSL}(\mathcal{X})$（あるいは $\mathcal{X} = \mathrm{SGSL}(\mathcal{X})$）であることを証明することは今後の課題である．

なお，本節では表現可能な論理式のクラスに関して考察を行ったが，実用上は，同じ論理式を記述する際の記述の長さについても考慮が必要である．例えば，$\mathcal{X}^{\uparrow}_{r,[],\neg} = \mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[],\neg})$ であることを上で述べたが，付録 A. の方法で $\mathrm{SGSL}(\mathcal{X}^{\uparrow}_{r,[],\neg})$ 式を等価な $\mathcal{X}^{\uparrow}_{r,[],\neg}$ 式に変換した場合，記述長が非常に長くなる場合がある．XPath と SGSL の論理式の記述長に関する比較も今後の課題である．

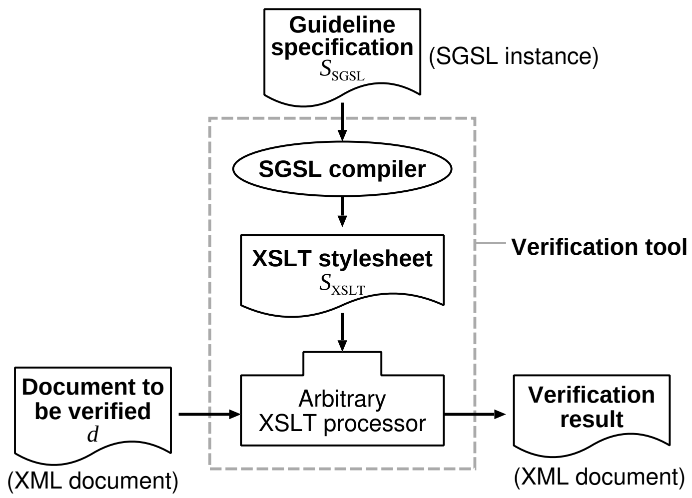

図 2　検証ツールの構成

3.　検証ツール

3.1　概要

実装した検証ツールの構成を図 2 に示す．検証ツールは，SGSL で書かれたガイドライン記述及び検証対象の XML 文書を入力とし，検証結果（レポート）を出力する．

実装には，XML 文書から別の XML 文書への変換規則を記述するための言語である XSLT 1.0 (XSL Transformations)[26] を用いた．XSLT は，XPath を部分仕様として含み，さらに変数への値の束縛や反復動作等，SGSL の実装に必要な機能を持つ．

本検証ツールは SGSL コンパイラと XSLT 処理器から構成される．SGSL コンパイラは，SGSL で書かれたガイドライン記述 S_{SGSL} を XSLT スタイルシート S_{XSLT} に変換する．S_{XSLT} は，検証対象である XML 文書 d から d に対する検証結果を表す XML 文書への変換規則を定義している．XSLT 処理器が S_{XSLT} を d に適用することで，この変換，即ち検証が行われる．SGSL コンパイラは Java プログラムとして独自に実装した．XSLT 処理器には MSXSL[5] 及び JAXP (Java API for XML Processing)[18] を用いた．

SGSL から XSLT スタイルシートへの変換方法（3.2 節で述べる）は，[8] で提案

されている，問合せ言語 XQuery 1.0 (XML Query Language)[25] を XSLT に変換する手法を参考にしている．SGSL は XQuery に比べて言語仕様が小さいため，本稿で用いる手法は [8] の手法に比べて単純になっている．なお，別の方法として，XQuery の実装を使用して SGSL のための検証ツールを実装することも可能と考えられるが，XQuery は現在仕様策定中であり，今のところ十分な実装が提供されていない．

3.2 XSLT による SGSL の実装

図 3 は，2.4 節のチェックポイント例 1 に対して SGSL コンパイラが生成した XSLT スタイルシート（XSLT で記述された変換仕様）である．XSLT スタイルシートは変換規則の集合であり，個々の規則をテンプレートという．以降，2.4 節例 1 で示したガイドライン記述を $S_{1,\mathrm{SGSL}}$，図 3 の XSLT スタイルシートを $S_{1,\mathrm{XSLT}}$ とし，これらを使って，XSLT によるガイドライン記述の実装方法を説明する．

$S_{1,\mathrm{XSLT}}$ は 4 つのテンプレートからなる．最初のテンプレート（3–7 行目）は実行開始点であり，$S_{1,\mathrm{XSLT}}$ が検証対象文書に適用されると最初に実行される．このテンプレートは，単に `GL1-f0`（8–22 行目）という別のテンプレートを呼び出し，`GL1-f0` の出力を `report` 要素中に出力する．`GL1-f0` は，`GL1-f1`（23–39 行目）及び `GL1-f2`（40–47 行目）を使って $S_{1,\mathrm{SGSL}}$ 中の論理式を評価し，`passed` または `warn` 要素のいずれか一方を結果 XML 文書中に出力する．$S_{1,\mathrm{SGSL}}$ 中の部分論理式は，$f_1 =$ `every $x in //img satisfies` f_2 及び $f_2 =$ `$x/@alt` の 2 つである．`GL1-f1` 及び `GL1-f2` はそれぞれ f_1 及び f_2 を評価する．

各部分式 f について，f を評価するテンプレートは，f が真のとき文字列 `1` を，f が偽のとき空文字列を出力する．例えばテンプレート `GL1-f2` は，部分式 $f_2 =$ `$x/@alt` を評価し，真ならば文字列 `1` を出力する．`GL1-f1` は，`xsl:for-each` 要素を使い，`//img` で表される頂点集合中の各 x について `GL1-f2` を呼び出す．このとき，`GL1-f2` を呼び出す `xsl:call-template` 要素（27–29 行目）が `xsl:variable` 要素（26–30 行目）中に置かれているので，`GL1-f2` の出力は変数 `return` に格納され，結果 XML 文書中には出力されない．そして，`return` $\neq$ `1` のとき，即ち現在の x に対して f_2 が偽のとき，変数 `result` に文字列 `0` が追加される．`xsl:for-each` 要素の実行終了後，`result` が空ならば，即ち f_2 が偽であるような x が存在しないなら

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="..." xmlns:html="...">
<xsl:template match="/">
  <report>
    <xsl:call-template name="GL1-f0"/>
  </report>
</xsl:template>
<xsl:template name="GL1-f0">
  <xsl:variable name="return">
    <xsl:call-template name="GL1-f1"/>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test="string($return)=1">
      <passed guideline="1"/>
    </xsl:when>
    <xsl:otherwise>
      <warn guideline="1">
        Provide alternative text for all images.
      </warn>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="GL1-f1">
  <xsl:variable name="result">
    <xsl:for-each select="//html:img">
      <xsl:variable name="return">
        <xsl:call-template name="GL1-f2">
          <xsl:with-param name="x" select="."/>
        </xsl:call-template>
      </xsl:variable>
      <xsl:if test="string($return)!=1">
        <xsl:text>0</xsl:text>
      </xsl:if>
    </xsl:for-each>
  </xsl:variable>
  <xsl:if test="not(string($result))">
    <xsl:text>1</xsl:text>
  </xsl:if>
</xsl:template>
<xsl:template name="GL1-f2">
  <xsl:param name="x"/>
  <xsl:variable name="result" select="$x/@alt"/>
  <xsl:if test="$result">
    <xsl:text>1</xsl:text>
  </xsl:if>
</xsl:template>
</xsl:stylesheet>
```

図 3　コンパイル結果

```
#xmlns:html=http://www.w3.org/1999/xhtml

#id 1.1-1
#title Provide alternative text for all images.
every $x in {//html:img} satisfies {$x/@alt}

#id 1.1-6
#title If ALT text >150 characters, consider providing
#title a separate description.
if some $x in {//@alt} satisfies
                       {string-length($x)>150}
then warn("consider providing a separate description")
```

図 4　コンパイラへの入力となるガイドライン記述例

ば，`GL1-f1` は文字列 1 を出力して終了する．`some-in-satisfies` 式や `let-in` 式の評価も同様のテンプレートで行える．

3.3　SGSL コンパイラ

3.2 節の方針に基づくコンパイラを Java プログラムとして実装した．コンパイラへの入力となるガイドライン記述例を図 4 に示す．コンパイラの実装を容易にするため，図 4 のように，ガイドライン記述中の XPath 式は { } で囲んで与えることとした．コンパイラは { } 中の XPath 式を解釈せず，そのまま出力である XSLT スタイルシートに書き込む．字句解析部及び構文解析部の作成には，字句解析器生成ツール JFlex[7] 及び構文解析器生成ツール CUP[3] をそれぞれ用いた．また，XSLT スタイルシートの出力に JAXP[18] を利用している．今回作成したソースプログラムのサイズは，JFlex 及び CUP への入力である構文規則と意味動作の記述が約 200 行，それ以外の部分が約 1,000 行である．

3.4　検証結果の形式

本検証ツールによって出力される検証結果の例を図 5 に示す．各チェックポイントについて，検証対象文書がそのチェックポイントを満たしている場合は `passed` 要素が出力され，満たしていない場合は `warn` 要素が出力される．これらの要素の `guideline`

```
<?xml version="1.0" encoding="UTF-8"?>
<report xmlns:html="...">
  <warn guideline="GL1.1-1">
    Provide alternative text for all images.
  </warn>
  <passed guideline="GL1.2"/>
  <warn guideline="GL2.2">
    Check that the foreground and background colors
    contrast sufficiently with each other.
  </warn>
  ...
</report>
```

図 5　検証結果の例

属性は，対応するチェックポイント番号を表している．また，警告文は `warn` 要素の内容として出力される．図 5 の例では，検証対象文書はチェックポイント 1.2 を満たし，1.1-1, 2.2 を満たしていない．

4.　検証実行例

本節では，いくつかの文書形式に関して本検証ツールを適用した例を示す．まず，1つのウェブページ（HTML 文書）に適用した例を使って本ツールの実行速度等を述べた後，約 3,000 件のウェブページに適用した例について述べる．ここでは Bobby が行っている検証内容と等しくなるよう記述したガイドラインを使った．これらの適用例の結果から，ウェブページのアクセシビリティ検証については，本検証ツールの検証能力は既存ツールと同等，実行速度は既存ツールに劣るが実用的な範囲内といえる．

続いて，提案手法がさまざまな文書形式や検証内容に適用できることを示すため，2次元画像記述言語である SVG (Scalable Vector Graphics)[22] に関してアクセシビリティ検証を行った例，及び，財務諸表等の記述言語である XBRL (eXtensible Business Reporting Language)[29] に関して既存の文書型記述言語では表現できない構文上の規則を満たすかどうか検証を行った例を示す．

4.1　ウェブページのアクセシビリティ検証

2.4 節で述べたように，Bobby が行っている検証内容と等価な 70 個のチェックポイントを SGSL で記述した．ここでは，米国ホワイトハウスのトップページ（2003 年 1 月 15 日更新）が，これらの各チェックポイントを満たしているか否かの検証を行う．ただし，このウェブページは HTML 形式であるため，これを XHTML 文書（約 35K バイト）に変換してツールへの入力とする．この変換は JTidy[19] を用いて行った．まず，SGSL で書かれた WCAG のチェックポイント（約 22K バイト）を，SGSL コンパイラを使って，XSLT スタイルシート（約 156K バイト）に変換した．この変換に必要な時間は約 3 秒だった．XSLT 処理器として MSXSL を使用し，検証対象文書に XSLT スタイルシートを適用したところ，約 5 秒でレポートが生成された．以上の実行時間はフリーウェイ社製 PC (Pentium-4 2GHz, 1GB RAM) を使って測定した．既存の検証ツールの多くは 1 ~ 2 秒以内に結果を出力するので，本検証ツールの実行速度はそれより劣るが，実用的な範囲内にあると考える．

この検証対象文書は，70 個のチェックポイントのうち 30 個を満たし，40 個を満たさなかった．Bobby が行っている検証内容に合わせて記述したガイドラインを用

表 3　優先度 1 かつ自動検証可能な WCAG チェックポイント [23]

	Title
Guideline 1. Provide equivalent alternatives to auditory and visual content	
1.1-1	Provide alternative text for all images.
1.1-2	Provide alternative text for each APPLET.
1.1-3	Provide alternative content for each OBJECT.
1.1-4	Provide alternative text for all image-type buttons in forms.
1.1-10	Provide alternative text for all image map hot-spots (AREAs).
Guideline 6. Ensure that pages featuring new technologies transform gracefully	
6.2	Each FRAME must reference an HTML file.
Guideline 12. Provide context and orientation information	
12.1	Give each frame a title.

いているので，検証結果は Bobby で行った場合と同等である．つまり，提案手法の検証能力は既存ツールと同等といえる．なお，同じ WCAG に関する検証でも，検証内容は既存ツールごとに多少異なる．例えば WCAG のチェックポイント 6.3 では<a href="javascript:...">という形式のリンクは望ましくないと述べられているが，Bobby, WebSAT はこれを検出せず，LIFT は検出する．本検証ツールでこれを検出するようにしたい場合は not //a[starts-with(@href,"javascript:")] というチェックポイント記述を追加すればよく，既存ツールのようにツール本体を変更する必要がない．このように，検証内容の追加や変更が必要な場合，提案手法は既存ツールより容易に行える．

4.2　日米ウェブサイトのアクセシビリティ検証

本検証ツールが実用上問題なく使用できるかどうかを確認し，また石川のアクセシビリティ調査 [4] と比べてウェブサイトのアクセシビリティがどのように変化したかを調査することを目的として，ウェブページ約 3,000 件に対して本検証ツールを適用

表 4　表 3 のチェックポイントの SGSL 記述

<table>
<tr><td>1.1-1</td><td><code>(every $x in //img satisfies $x/@alt)</code>
<code>and every $x in //a/descendant::img</code>
<code>   satisfies string-length($x/@alt) != 0</code></td></tr>
<tr><td>1.1-2</td><td><code>every $x in //applet satisfies $x/@alt</code></td></tr>
<tr><td>1.1-3</td><td><code>every $x in //object satisfies</code>
<code> $x/applet or $x/embed or $x/img</code>
<code> or $x/input or $x/li</code>
<code> or $x/option or $x/textarea</code>
<code> or $x/button or $x/table</code>
<code> or $x/tr or $x/td or $x/th</code></td></tr>
<tr><td>1.1-4</td><td><code>every $x in //input[@type='image']</code>
<code>satisfies $x/@alt</code></td></tr>
<tr><td>1.1-10</td><td><code>every $x in //area satisfies $x/@alt</code></td></tr>
<tr><td>6.2</td><td><code>every $x in</code>
<code> //*[self::frame|self::iframe][@src]</code>
<code>satisfies</code>
<code>   contains($x/@src,'.html')</code>
<code> or contains($x/@src,'.htm')</code>
<code> or contains($x/@src,'.shtml')</code>
<code> or contains($x/@src,'.shtm')</code>
<code> or contains($x/@src,'.cfml')</code>
<code> or contains($x/@src,'.cfm')</code>
<code> or contains($x/@src,'.asp')</code>
<code> or contains($x/@src,'.cgi')</code>
<code> or contains($x/@src,'.pl')</code></td></tr>
<tr><td>12.1</td><td><code>every $x in //*[self::frame|self::iframe]</code>
<code>satisfies $x/@title</code></td></tr>
</table>

した．調査対象としたウェブサイトは [4] と同一で，米国：政府機関，米国：民間，日本：政府機関，日本：民間の 4 つのカテゴリから代表的なものを各 10 サイト，計 40 サイトを調査した．日米で，ほぼ同じ業種のサイトが選択されている．各サイトのトップページ，およびトップページから直接リンクされているページに対して検証を行った．ページの収集は 2003 年 1 月 13 日から 1 月 20 日の間に行い，HTML 形式のページは JTidy を用いて XHTML 文書に変換した．

[4] と同様に，WCAG のチェックポイントのうち，優先度が 1（最も重要）かつ自動検証が可能 (Full) である 7 個（表 1）に関して検証を行った．使用したチェックポイントの表題を表 3 に，SGSL によるそれらの記述を表 4 に示す．

検証対象である各 XHTML 文書について，これらの 7 個のチェックポイントを満たすかどうか検証した．7 個のチェックポイントすべてを満たす文書を合格 (Passed)，少なくとも一つ満たさないチェックポイントがある文書を不合格 (Not passed) とする．

実験結果と考察　実験結果を表 5–7 に示す．以降，調査対象文書数に対する合格文書数の比を合格率 (Rate) と呼ぶ．民間組織については，日米とも合格率が 20%前後で，政府機関に比べて低く，民間組織の方がアクセシビリティに対する配慮が少ないことをうかがわせる．また，政府機関についても，日米ともに合格率が 10%未満のサイトがいくつかある．調査対象全体での合格率は約 30%（合格 894 件，不合格 2,039 件）だった．

チェックポイント別の不適合文書数を表 8 に示す．小計 (Subtotal) 及び合計 (Total) の 2 行目は，全不合格文書数に対する比である．この表から，チェックポイント 1.1-1 を満たさなかった文書が他のチェックポイントに比べて多いことがわかる．

2001 年に行われた石川の調査 [4] では，全体の合格率は 35%と報告されている．このことから，有名ウェブサイトに限っても，ウェブ・アクセシビリティは依然不十分であることがうかがえる．なお，[4] では検証ツールとして当時の Bobby が使われたが，本実験では 2002 年当時の Bobby の検証内容を参照してチェックポイントを記述したため，検証内容，即ち WCAG チェックポイントの解釈が同一でない箇所がある．即ち，[4] の実験環境では合格となるが本実験では不合格となる文書が存在する．例えばチェックポイント 1.1-1（すべての画像に代替テキストを提供せよ）は，2001 年版 Bobby では「すべての `img` 要素 `x` が `alt` 属性を持つこと」と解釈されている．一方，

表 5　米国のウェブサイトのアクセシビリティ調査結果

Site	Passed	Not passed	Rate
United States Postal Service	33	1	97%
Internal Revenue Service	25	3	89%
Department of Justice	43	7	86%
Department of Education	55	14	79%
White House	35	19	65%
Library of Congress	20	14	58%
General Services Administration	22	37	37%
United States House of Representatives	6	22	21%
Department of Health & Human Services	2	31	6%
United States Environmental Protection Agency	1	34	3%
Subtotal	242	182	57%
Amtrak	31	14	69%
Yahoo!	127	113	53%
(Other eight sites)	23	714	3%
Subtotal	181	841	18%
Total	423	1023	29%

表 6　国内のウェブサイトのアクセシビリティ調査結果

サイト名	合格	不合格	合格率
厚生労働省	79	16	83%
首相官邸	63	14	81%
国税庁	37	12	76%
郵政事業庁	33	21	61%
総務省	16	42	28%
環境省	15	46	25%
法務省	3	13	19%
文化庁	3	35	8%
国立国会図書館	1	26	4%
参議院	1	26	4%
小計	251	251	50%
紀伊國屋書店 BookWeb	80	44	65%
JR おでかけネット	24	17	59%
（他 8 サイト）	116	704	14%
小計	220	765	22%
合計	471	1016	32%

表 7　ウェブサイトのアクセシビリティ調査結果（全体）

	Government	Companies	Total
U.S.A.	242 (57%) 182 (43%)	181 (18%) 841 (82%)	423 (29%) 1023 (71%)
Japan	251 (50%) 251 (50%)	220 (22%) 765 (78%)	471 (32%) 1016 (68%)
Total	493 (53%) 433 (47%)	401 (20%) 1606 (80%)	894 (30%) 2039 (70%)

Passed
Not passed

表 8　チェックポイント別不合格文書数

		Checkpoint number						
		1.1-1	1.1-2	1.1-3	1.1-4	1.1-10	6.2	12.1
U.S.A.	Government	205			6	3	2	1
	Companies	782	10	2	178	385	33	48
	Subtotal	988	10	2	184	388	35	49
		97%	1%	0%	18%	38%	3%	5%
Japan	Government	174				7	3	85
	Companies	552	1		7	85	132	188
	Subtotal	726	1		7	92	135	273
		71%	0%	0%	1%	9%	13%	27%
	Total	1714	11	2	191	480	170	322
		84%	1%	0%	9%	24%	8%	16%

表 9　SVG に関する調査対象ウェブサイト

	# of SVG documents	URL
Adobe	7	www.adobe.com/svg/
Sun	9	wwws.sun.com/software/xml/developers/svg/
W3C	128	www.w3.org/Graphics/SVG
Croczilla	293	www.croczilla.com/svg/
Total	437	

2002 年版 Bobby では，これに加え「リンクフレーズの一部である `img` 要素は，空でない `alt` 属性を持つこと」も要求している．

4.3　SVG 文書のアクセシビリティ検証

SVG[22] は，2 次元画像を XML 形式で記述するための言語である．SVG 文書は主に，円や線分，テキストなどの描画オブジェクトの構造と描画位置を記述したものであるが，このほかに文書全体や各オブジェクトの名前，説明などを記述することができる．画像を利用できない環境にある利用者に対し，ブラウザがこれらのテキストを抽出して提供することで，アクセシビリティを高めることができる [21]．

XHTML 以外への本検証ツールの適用例として，SVG 文書を提供しているウェブサイトを 4 つ選び，それらの SVG 文書（437 件）のアクセシビリティを調査した．調査対象としたサイトは表 9 の 4 つである．表 9 に挙げた各 URL から 3 個以下のリンクをたどって到達できる SVG 文書を収集し，検証対象とした．

ガイドラインとして，SVG 1.1 規格書 [22] の付録 H に書かれたガイドラインや規格書の本文，及び付属文書 [21] の内容を参考に，自動検証可能なチェックポイント 7 つを SGSL で記述した（表 10）．

実行結果　4.2 節と同様に，7 つのチェックポイントすべてを満たす文書を合格，少なくとも一つ満たさないチェックポイントがある文書を不合格とすると，検証対象とし

表 10　SVG 用チェックポイント

<table>
<tr><td rowspan="2">g1</td><td>最も外側の svg 要素は title 子要素をもつこと．</td></tr>
<tr><td>every $x in //svg[not(ancestor::svg)]
satisfies $x/title</td></tr>
<tr><td rowspan="3">g2-1

g2-2</td><td>同じ親をもつ title, desc 要素はそれぞれ高々一つ，かつ metadata 以外の他の兄弟より前．</td></tr>
<tr><td>(every $x in //title satisfies
  count($x/../title)=1) and
(every $x in //desc satisfies count($x/../desc) =1)</td></tr>
<tr><td>every $x in //title|//desc satisfies
 every $y in $x/preceding-sibling::* satisfies
  $y[self::title|self::desc|self::metadata]</td></tr>
<tr><td rowspan="2">g3</td><td>use 要素で参照されるオブジェクトは defs 要素内で定義すること．</td></tr>
<tr><td>every $u in //use satisfies
 let $href := string($u/@xlink:href) in
  (starts-with($href, "#") implies
   let $idref := substring-after($href, "#") in
    every $x in //*[@id = $idref]
    satisfies $x/ancestor::defs)</td></tr>
<tr><td rowspan="2">g4</td><td>テキストコンテンツで用いる自然言語を xml:lang 要素で明示すること．</td></tr>
<tr><td>every $x in //title|//desc|//text
  satisfies $x/ancestor::*/@xml:lang</td></tr>
<tr><td rowspan="3">g5-1

g5-2</td><td>説明的なテキストを含まないグラフィックには代替テキストを付けること．</td></tr>
<tr><td>every $x in //symbol[not(.//text)]
satisfies ($x/title or $x/desc)</td></tr>
<tr><td>every $x in //g[not(.//text)]
  [count(g|use|path|rect|circle|
        ellipse|line|polyline|polygon) > 1]
satisfies ($x/ancestor-or-self::g/title or
           $x/ancestor-or-self::g/desc)</td></tr>
</table>

表 11　SVG 文書のアクセシビリティ調査結果

	Checkpoint number						
	g1	g2-1	g2-2	g3	g4	g5-1	g5-2
Adobe	7				1		7
Sun	7				9		6
W3C	101		10	39	104		63
Croczilla	66		184	1	280	1	88
Total	181	0	194	40	394	1	164
	41%	0%	44%	9%	90%	0%	38%

た 437 文書はすべて不合格であった．チェックポイント別の不合格文書数を表 11 に示す．合計欄の 2 行目は，表 8 と同じく，全不合格文書数に対する比である．この表から，ほとんどの文書が満たしているチェックポイント（g2-1, g3, g5-1）や，逆にほとんどの文書が満たしていないチェックポイント（g4）が存在することがわかる．

4.4　XBRL 文書の構文規則適合性検証

XBRL[29] は，企業の財務諸表等を XML 形式で記述するための言語である．XBRL 文書の例を図 6 に示す．XBRL では「売上高」「売上原価」といった各項目を表す要素（図 6 では `Sales` 及び `CostSales`）が定義されており[2]，XBRL 文書は基本的にそれらの要素を一列に記述したものとなる．また，各項目のコンテキスト（どの期間についてのデータであるかや通貨単位など）は別の要素に分けて記述され，各項目からは名前で参照される．

XBRL 文書は，XML Schema で記述された文書型に従う以外に，以下のような制約を満たさなければならない．

- 項目及びコンテキストはそれぞれ数値型のものと非数値型のものがある．数値型項目が非数値型コンテキストを参照してはいけない．逆も同様．

[2] 正確には，各項目を表す要素は，各国 XBRL 協会などが制定した拡張規格で定義される．

```
<?xml version="1.0"?>
<group xmlns="..." xmlns:t="...">
<t:Sales numericContext="c1">40000</t:Sales>
<t:CostSales numericContext="c1">7500</t:CostSales>
<numericContext id="c1">
  <period>
    <startDate>2003-01-01</startDate>
    <endDate>2003-12-31</endDate>
  </period>
  <unit>
    <measure>ISO4217:JPY</measure>
  </unit>
</numericContext>
</group>
```

図 6　XBRL 文書の例

- コンテキスト中に記述される期間の長さが 0 や負になってはいけない．
- 特に指定しない限り，同じコンテキストを持つ同じ項目が複数現れてはいけない．
- 「あるいくつかの項目の和が別のある項目に等しい」といった計算規則が別途定義されている．この計算規則に反する値を記述してはいけない．

これらは既存の文書型記述言語では記述できないため，これらを満たしているかどうかは，通常，XBRL 専用のソフトウェアを使って検証される．

ここでは，本検証ツールを使って，XBRL 文書がこれらの規則を満たしているかどうか検証を行った例を示す．検証対象文書として，ドイツ XBRL 協会 [28] から提供されているサンプル文書 3 件（それぞれ約 55KB）を用いた．これらはほぼ同内容で，2 件（文書 1 及び 2）は正当な文書，1 件（文書 3）は上記のような規則に関するエラーを含む文書である．文書 3 中の注釈を参考に，チェックポイント 8 つを SGSL で記述した（表 12–13）．これらを文書 1 ~ 3 に適用した結果，文書 1 ~ 2 はすべてのチェックポイントを満たし，文書 3 はすべてのチェックポイントを満たさなかった．

表 12　XBRL 用チェックポイント

A-1 非数値項目が数値コンテキストを参照してはいけない．

```
every $e in //*[@nonNumericContext] satisfies
  not(//numericContext/@id=$e/@nonNumericContext)
```

A-2 数値項目が非数値コンテキストを参照してはいけない．

```
every $e in //*[@numericContext] satisfies
  not(//nonNumericContext/@id=$e/@numericContext)
```

B 通貨名として ISO 準拠の省略名を使い，名前空間を指定すること．

```
every $m in //unit/measure satisfies
  (starts-with($m, "ISO4217:") and
   string-length(substring-after($m,"ISO4217:"))=3)
```

C-1 期間指定において長さが負であってはいけない．

```
every $p in //period[duration] satisfies
  not(starts-with($p/duration, "-"))
```

C-2 期間指定において長さが 0 であってはいけない．

```
every $p in //period[duration] satisfies
contains(translate($p/duration,"23456789","11111111"),"1")
```

C-3 endDate が startDate より前であってはいけない．

```
every $p in //period[startDate and endDate]
satisfies
let $s :=string($p/startDate) in
let $e :=string($p/endDate) in
let $sy:=substring-before($s, "-") in
let $ey:=substring-before($e, "-") in
let $sm:=substring-before(substring-after($s,"-"),"-") in
let $em:=substring-before(substring-after($e,"-"),"-") in
let $sd:=substring-after (substring-after($s,"-"),"-") in
let $ed:=substring-after (substring-after($e,"-"),"-") in
($sy < $ey or ($sy = $ey and ($sm < $em or
                                ($sm = $em and $sd <= $ed))))
```

表 13　XBRL 用チェックポイント（続き）

<table>
<tr><td rowspan="2">D</td><td>同じコンテクストに属する同じ要素が重複してはいけない．ただし，タプル型要素の子である場合を除く．</td></tr>
<tr><td><pre>every $x in /group/*[@numericContext] satisfies
  let $n := name($x) in
  let $c := $x/@numericContext in
    not($x/following-sibling::*[name() = $n]
                          [@numericContext = $c])</pre></td></tr>
<tr><td rowspan="2">E</td><td>計算規則に反する要素があってはいけない．ただし，コンテクストの <code>cwa</code> 属性が <code>true</code> である場合，1 要素に限り省略できる．</td></tr>
<tr><td><pre>every $f in //bs.ass.currAss satisfies
let $c := $f/@numericContext in
let $s :=//bs.ass.currAss.inventory[@numericContext=$c]
         |//bs.ass.currAss.receiv[@numericContext=$c]
         |//bs.ass.currAss.securities[@numericContext=$c]
         |//bs.ass.currAss.cashEquiv[@numericContext=$c]
in (count($s)>=3
    and (count($s)=3 implies
          //numericContext[@id=$c]/@cwa="true")
    and (count($s)>3 implies $f=sum($s)))</pre></td></tr>
</table>

5. あとがき

本稿では，ウェブ文書のためのアクセシビリティ・ガイドラインを形式的に記述するための言語である SGSL を提案し，与えられた XML 文書が SGSL で書かれたガイドライン記述に適合しているか否かを自動検証するツールの実装について述べた．本検証ツールは，ガイドライン記述がツールと分離されているためガイドラインの変更が容易であり，また任意の XML 文書を検証対象にできることから，既存の検証ツールに比べ汎用性が高い．また，アクセシビリティ・ガイドラインに限らず，4.4 節で述べた XBRL 文書の検証のように，各種構文規則への適合性の検証にも利用できる．

国内でもアクセシビリティガイドラインを述べた JIS X8341-3[15] が制定されるなど，ウェブ・アクセシビリティへの関心が高まっている．また，WCAG や JIS X8341-3 に関してガイドライン適合性検証を行うツールも多数提供されるようになった．一般のウェブページ制作者にとっては，これらの特定のガイドライン及び HTML 形式に対して検証できれば十分な場合も多いと考えられ，その場合にはむしろ既存検証ツールの方が容易に使うことができ，有用性が高いと考えられる．提案手法は，XPath 式を理解できる程度のプログラミングの知識をもつ利用者を対象として，汎用プログラム言語でツールを開発するより容易に，アクセシビリティ検証のほかさまざまな文書形式に対するさまざまな検証を行えるようにすることを目的としている．これには，一人の技術者が，自組織の構成員のためにガイドラインを記述するという利用法も含まれる．この場合，自組織向けの検証ツールを容易に開発できるという利点がある．

今回実装した検証ツールでは，検証対象文書 d が各チェックポイントを満たすか否かを出力するのみで，どの箇所に問題があるかは出力しない．一方，既存の検証ツールは通常，d がガイドラインを満たさない場合に，d のどの箇所に問題があるかを（行番号等で）示す機能を持っている．文書作成作業の支援のためには，このような機能は必須といってよい．与えられたガイドライン記述に対しそれを満たさない箇所を抽出する方法について考察中であり，その概要を付録 B. で述べる．

今後の課題として，今回扱った構文的な性質の検証だけでなく，視覚的性質の検証や自然言語の使い方に関する検証等に検証法を拡張することが挙げられる．また，SGSL（及び XPath）の表現能力をより詳細に調べること，特に XPath の組み込み関数を増減したときに表現能力がどのように変化するか調べることも課題の一つである．

謝辞　検証実行例に関して貴重なご助言をいただきました大阪大学の松下誠助手，並びに検証システムの実装に関してご助言をいただきました本学の的野晃整さんに感謝いたします．

参考文献

[1] M. Benedikt, W. Fan and G. M. Kuper, “Structural properties of XPath fragments,” Proc. 9th International Conference on Database Theory (ICDT 2003), LNCS 2572, Jan. 2003.

[2] N. Bowers, “Weblint,” http://ejk.cso.uiuc.edu/cgi-bin/weblint, 1994.

[3] S. E. Hudson, “CUP parser generator for Java v0.10k,” http://www.cs.princeton.edu/˜appel/modern/java/CUP/, 1999.

[4] 石川准, “ウェブ・サイトのアクセシビリティ調査,” http://fuji.u-shizuoka-ken.ac.jp/˜ishikawa/webcheck.htm, 2001.

[5] A. Kimball, “Msxsl,” http://msdn.microsoft.com/library/default.asp?url=/library/en-us/Dnxslgen/html/msxsl.asp, 2001.

[6] N. Klarlund, T. Schwentick and D. Suciu, “XML: model, schemas, types, logics, and queries,” in Logics for Emerging Applications of Databases, pp.1–41, Springer, 2003.

[7] G. Klein, “JFlex 1.3.5,” http://www.jflex.de/, 2001.

[8] S. Lechner, G. Preuner and M. Schrefl, “Translating XQuery into XSLT,” ER 2001 Workshops, Revised Papers, LNCS 2465, pp.239–252, 2002.

[9] 今村誠, 鈴木克志, “電子申請における XML 文書内容検証方式,” 第 63 回情処学全大, 2X-4, pp.3–185–3–186, Sept. 2001.

[10] M. Murata, D. Lee and M. Mani, “Taxonomy of XML schema languages using formal language theory,” Extreme Markup Languages, 2001, http://www.idealliance.org/papers/extreme03/titles.html.

[11] 中村岳志, 高田喜朗, 関浩之, “XML 文書に対するアクセシビリティ・ガイドラインの自動検証,” 信学技報, SS2002-30, pp.19–24, Jan. 2003.

[12] National Institute of Standards and Technology (NIST), “The Web Static Analyzer Tool (WebSAT),” http://zing.ncsl.nist.gov/WebTools/WebSAT/overview.html, 1999.

[13] F. Neven, “On the power of walking for querying tree-structured data,” Proc. 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002), pp.77–84, June 2002.

[14] C. Nicolle and J. Abascal, *Inclusive design guidelines for HCI*, Taylor & Fransis, 2001.

[15] 日本工業規格, *JIS X8341-3:2004* 高齢者・障害者等配慮設計指針 — 情報通信における機器，ソフトウェア及びサービス — 第 *3* 部*:* ウェブコンテンツ, 2004.

[16] M. Paciello, *Web accessibility for people with disabilities*, CMP Books, 2000.

[17] T. Schwentick, “On diving in trees,” Proc. 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000), LNCS 1893, pp.660–669, Aug. 2000.

[18] Sun Microsystems, “Java API for XML Processing,” http://java.sun.com/xml/jaxp/.

[19] A. Tripp, G. Peskin, S. Lempinen and R. Gold, “JTidy,” http://lempinen.net/sami/jtidy/, 2000.

[20] UsableNet, “LIFT,” http://www.usablenet.com/, 1998.

[21] W3C, “Accessibility features of SVG,” http://www.w3.org/TR/SVG-access/, 2000.

[22] W3C, “Scalable Vector Graphics 1.0 Specification,” http://www.w3.org/TR/SVG11/, 2003.

[23] W3C, “Web Content Accessibility Guidelines 1.0,” http://www.w3.org/TR/WAI-WEBCONTENT/, 1999.

[24] W3C, “XML Path Language (XPath) Version 1.0,” http://www.w3.org/TR/xpath, 1999.

[25] W3C, “XQuery 1.0: An XML Query Language,” http://www.w3.org/TR/xquery/, 2002.

[26] W3C, “XSL Transformations (XSLT) Version 1.0,” http://www.w3.orgTR/xslt, 1999.

[27] Watchfire Corp., “Bobby,” http://bobby.watchfire.com/bobby/html/en/index.jsp, 2002.

[28] XBRL Deutschland, “XBRL sample files,” http://www.xbrl-deutschland.de/sg2/.

[29] XBRL Japan 監修, 坂上学, 白田佳子 編, *XBRL* による財務諸表作成マニュアル, 日本経済新聞社, 2003.

A. SGSL 論理式から XPath 式への変換

以下の (1)–(12) はそれぞれ，SGSL 論理式に対する等価変換規則となっている．SGSL 論理式 f にこれらの規則を任意回適用した結果 XPath 式が得られれば，f と等価な XPath 式が得られたことになる．少なくとも，f 中に属性値に関する演算（=, != や大小比較，四則演算など），及び文脈頂点集合に関する組み込み関数（`last` と `position`）が含まれない場合は，これらの変換によって f を等価な XPath 式に変換できる．また，これらの演算や関数を含む場合でも，f の形によっては XPath 式に変換できる．例えば，2.4 節の例 4 は組み込み関数 `position` を使っているが[3]，これらの変換によって XPath 式に変換できる．変換規則 (1)–(3) は，f 中の `every`, `some`, `let` 構文を消す規則である．これらによりすべての `every`, `some`, `let` が消去できれば，XPath 式への変換が完了する．特に，`some` の数を減らす規則 (3) が重要である．(3) は，式 `some` $\$x$ `in` p `satisfies` f において，f 中に現れる変数参照がすべて $\$x$ でかつ述語中には現れない（経路式の先頭にのみ現れる）ときに限り適用できる．(3) を適用できるようにするため，(4) は，$\$x$ を含まない部分式を `some` 式の外へ移動する．(5)–(12) は，変数参照を述語の外へ移動する規則である．正確には，(10), (12) が変数参照を述語の外へ移動する働きをし，それ以外の規則は (10), (12) を適用できるよう式を変形する働きをする．

以下では次のような記法を用いる．$f, f_1, \ldots$ を任意の論理式とし，$p, p_1, \ldots$ を任意の XPath 式とする．また，$q, q_1, \ldots$ を XPath 経路式の接尾部分式とする．x, y を，`every`, `some`, `let` 式で束縛される変数とする．

(1) `every` $\$x$ `in` p `satisfies` f を `not some` $\$x$ `in` p `satisfies not` f に書き換える．

(2) `let` $\$y$ `:=` p `in` f を，f 中に現れる $\$y$ をすべて式 (p) に（マクロ展開のように）置換して得られる式に書き換える．2.3 節で定義したように，p は文脈頂点に依存しない式なので，このように書き換えても式の意味は変わらない．

(3) `some` $\$x$ `in` p `satisfies` f において，f 中に現れる変数参照がすべて

[3] 述語 `[1]` は `[position()=1]` の略記である．

$\mathtt{\$}x$ でかつ述語中には現れないとき，$\mathtt{(}p\mathtt{)}\,\mathtt{[}f'\mathtt{]}$ に書き換える．ただし f' は，f 中に現れる $\mathtt{\$}x$ を．(文脈頂点を表す式) に置換して得られる式である．

(4)　$\mathtt{some}\ \mathtt{\$}x\ \mathtt{in}\ p\ \mathtt{satisfies}\ \mathtt{(}f_1\ bop\ f_2\mathtt{)}$ (ただし bop は and または or) において，f_2 に変数参照 $\mathtt{\$}x$ が現れない場合，$\mathtt{(some}\ \mathtt{\$}x\ \mathtt{in}\ p\ \mathtt{satisfies}\ f_1\mathtt{)}\ bop\ f_2$ に書き換える．

(5)　$f_1\ \mathtt{implies}\ f_2$ を $\mathtt{(not}\ f_1\mathtt{)}\ \mathtt{or}\ f_2$ に書き換える．同様に，$\mathtt{not}\ \mathtt{(}f_1\ \mathtt{and}\ f_2\mathtt{)}$ を $\mathtt{(not}\ f_1\mathtt{)}\ \mathtt{or}\ \mathtt{(not}\ f_2\mathtt{)}$ に，$\mathtt{not}\ \mathtt{(}f_1\ \mathtt{or}\ f_2\mathtt{)}$ を $\mathtt{(not}\ f_1\mathtt{)}\ \mathtt{and}\ \mathtt{(not}\ f_2\mathtt{)}$ に，$\mathtt{not}\ \mathtt{not}\ f$ を f に書き換える．

(6)　$p\,\mathtt{[}p_1\ \mathtt{and}\ p_2\mathtt{]}\,q$ を $p\,\mathtt{[}p_1\mathtt{]}\,\mathtt{[}p_2\mathtt{]}\,q$ に書き換える．

(7)　$p\,\mathtt{[}p_1\ \mathtt{or}\ p_2\mathtt{]}\,q$ を $p\,\mathtt{[}p_1\mathtt{]}\,q \mid p\,\mathtt{[}p_2\mathtt{]}\,q$ に書き換える．ただし，q 中に last または position が現れる場合を除く．

(8)　$\mathtt{(}p_1 \mid \ldots \mid p_k\mathtt{)}\,q$ を $p_1q \mid \ldots \mid p_kq$ に書き換える．ただし，q 中に last または position が現れる場合を除く．

(9)　XPath 式 $p_1 \mid \ldots \mid p_k$ がブール型の式として評価される場合 (述語中に現れる場合や SGSL 式中の論理式である場合)，$p_1\ \mathtt{or} \ldots \mathtt{or}\ p_k$ に書き換える．

(10)　XPath 式 h を，経路式 $\mathtt{\$}xq_1$ と，変数参照を含まない絶対経路式，定数，及び組み込み関数 (not を含む) からなる式とする．XPath 式 $p\,\mathtt{[}h\mathtt{]}\,q_2$ がブール型の式として評価される場合，$pq_2\ \mathtt{and}\ h$ に書き換える．$\mathtt{\$}x$ の値は文脈頂点に依存しないので，このように書き換えても式の意味は変わらない．

(11)　$\mathtt{some}\ \mathtt{\$}x\ \mathtt{in}\ p_1 \mid \ldots \mid p_k\ \mathtt{satisfies}\ f$ を $\mathtt{(some}\ \mathtt{\$}x\ \mathtt{in}\ p_1\ \mathtt{satisfies}\ f\mathtt{)}\ \mathtt{or} \ldots \mathtt{or}\ \mathtt{(some}\ \mathtt{\$}x\ \mathtt{in}\ p_k\ \mathtt{satisfies}\ f\mathtt{)}$ に書き換える．

(12)　XPath 式 h を，経路式 $\mathtt{\$}xq_1$ と，変数参照を含まない絶対経路式，定数，及び組み込み関数 (not を含む) からなる式とする．このとき，$\mathtt{some}\ \mathtt{\$}x\ \mathtt{in}\ p\,\mathtt{[}h\mathtt{]}\,q_2\ \mathtt{satisfies}\ f$ を $h\ \mathtt{and}\ \mathtt{(some}\ \mathtt{\$}x\ \mathtt{in}\ pq_2\ \mathtt{satisfies}\ f\mathtt{)}$ に書き換える．h の真偽は p に依存しないので，$p\,\mathtt{[}h\mathtt{]}\,q_2$ の評価値は，h が真のときは pq_2 と等しく，h が偽のときは空集合となる．したがって，このように書き換えても式の意味は変わらない．

B.　ガイドラインを満たさない箇所の指摘

2.3 節で述べたように，各チェックポイントは，1 個の論理式 f_1，または，`if` f_2 `then warn`（警告文）の形式の式である．ここで，前者は `if not` f_1 `then warn`（チェックポイント名）と等価であるので，全チェックポイントが後者であると仮定しても一般性を失わない．このとき，何らかの意味で f_2 を満たす箇所が，このチェックポイントに関して問題のある箇所となる．これを f_2 の形に応じて定義すればよい．このとき考慮すべき点として以下が挙げられる．

- 満たす箇所を指摘できない論理式もある．例えば，f_2 が `every-in-satisfies` 式であった場合，f_2 を満たすのは特定の箇所ではなく検証対象文書全体と考えられる．

- f_2 を満たす箇所は，個々の頂点とは限らず，頂点の組（有限系列）である場合もある．例えば

$$\begin{aligned} f_2 \;=\; & \texttt{some \$}x\ \texttt{in}\ p_1\ \texttt{satisfies} \\ & \quad \texttt{some \$}y\ \texttt{in}\ p_2\ \texttt{satisfies}\ f_3 \end{aligned}$$

  の場合，f_2 を満たす箇所（満たす例）は，f_3 を満たす x, y の組となる．

以上を踏まえ，検証対象文書 d 中において論理式 f を満たす箇所の集合 $\mathrm{sat}(f)$ を，例えば以下のように定義できる．ただし $\mathrm{sat}(f)$ の元は，d 中の頂点の有限系列である．以下，検証対象文書 d を固定する．初めに準備として，f 中の部分式 f_1 `implies` f_2 を $(\texttt{not}\ f_1)$ `or` f_2 に書き換えて，f 中に `implies` が現れないようにする．また，de Morgan の法則を適用して，f 中の `not` は XPath 式にのみ現れるように，即ち，f 中の `not` がすべて XPath 1.0 の組み込み関数 `not` となるように f を書き換える．以下では次のような記法を用いる．$f, f_1, \ldots$ を任意の論理式とし，p を任意の XPath 式とする．論理式 f 中の変数 x を値 v に束縛して得られる論理式を $f[x := v]$ と書く．頂点集合型 XPath 式 p に対し，p を d に適用した結果得られる頂点集合を $\mathrm{nodes}(p)$ と書く．頂点の有限系列に関して，空系列を ϵ と書き，頂点 n のみからなる系列を $\langle n \rangle$ と書く．系列 t_1, t_2 の連接を $t_1 : t_2$ で表す．論理式 f に対し，$\mathrm{sat}(f)$ は，f の構造に従って以下のように定義される．この定義では，f が偽のときかつそのときのみ $\mathrm{sat}(f)$ が

空集合となる．例えば $\mathrm{sat}(f_1\ \texttt{and}\ f_2)$ は，$\mathrm{sat}(f_1)$ と $\mathrm{sat}(f_2)$ の少なくとも一方が空集合のときかつそのときのみ空集合となる．

$$
\begin{aligned}
&\mathrm{sat}(p) = \begin{cases} \{\langle n\rangle \mid n \in \mathrm{nodes}(p)\} \\ \qquad p \text{ が頂点集合型}, \\ \{\epsilon\} \quad p \text{ がブール型かつ評価値が真}, \\ \emptyset \qquad \text{それ以外}, \end{cases} \\
&\mathrm{sat}(\texttt{every \$}x\ \texttt{in}\ p\ \texttt{satisfies}\ f_1) \\
&\qquad = \begin{cases} \{\epsilon\} \quad \text{すべての } n \in \mathrm{nodes}(p) \text{ について} \\ \qquad \mathrm{sat}(f_1[x := n]) \neq \emptyset, \\ \emptyset \qquad \text{それ以外}, \end{cases} \\
&\mathrm{sat}(\texttt{some \$}x\ \texttt{in}\ p\ \texttt{satisfies}\ f_1) \\
&\qquad = \{\langle n\rangle : t \mid n \in \mathrm{nodes}(p),\ t \in \mathrm{sat}(f_1[x := n])\}, \\
&\mathrm{sat}(\texttt{let \$}x\ \texttt{:=}\ p\ \texttt{in}\ f_1) = \mathrm{sat}(f_1[x := p]), \\
&\mathrm{sat}(f_1\ \texttt{or}\ f_2) = \mathrm{sat}(f_1) \cup \mathrm{sat}(f_2), \\
&\mathrm{sat}(f_1\ \texttt{and}\ f_2) = \{t_1 : t_2 \mid t_1 \in \mathrm{sat}(f_1),\ t_2 \in \mathrm{sat}(f_2)\}.
\end{aligned}
$$

実装の際は，3.2 節では各テンプレートが文字列 $\texttt{1}$ または空文字列を出力していたところを，各テンプレートが論理式 f に対して $\mathrm{sat}(f)$ を（適当な XML 形式に符号化して）出力するようにすればよい．