相関ルールを利用したCSS自動生成手法の提案

— HTMLからスタイル情報の分離 —

曽山　裕† 松本　章代†† Martin J.Dürst††

† 青山学院大学大学院 理工学研究科　〒 229–8558 神奈川県相模原市淵野辺 5–10–1
†† 青山学院大学理工学部　〒 229-8558 神奈川県相模原市淵野辺 5-10-1
E-mail: †soyama@sw.it.aoyama.ac.jp, ††{akiyo,duerst}@it.aoyama.ac.jp

あらまし Web ページにスタイルを指定する CSS は，ページ作成者，閲覧者の双方にとって有益である．しかし，HTML に付けられたスタイルを手動で CSS 化することは非常に手間がかかるため，CSS を利用していない Web ページも数多く存在する．本稿ではこの書き直しの支援のため， Web ページを構成するスタイル・構造・コンテンツのうち，スタイル情報にセレクタを付与したものを外部 CSS に自動分割する手法を提案する．規則集合の抽出は二つの段階で行う．第一段階ではセレクタとスタイルにおける相関ルールを発見する．第二段階ではスタイルに対するセレクタの最大有効範囲を探索する．この手法を適用することで，style 属性で乱雑に指定されていたスタイルを外部 CSS にまとめることを実現した．

キーワード CSS, Cascading Style Sheets, 相関ルール, Web ページ分析

Proposing a Method for Autogenerating CSS by Using Association Rules

— Isolating Style Information from HTML —

Yutaka SOYAMA†, Akiyo MATSUMOTO††, and Martin J. DÜRST††

† Graduate school of Science and Engineering, Aoyama Gakuin University
Fuchinobe 5–10–1, Sagamihara, Kanagawa, 229–8558 Japan
†† College of Science and Engineering, Aoyama Gakuin University
Fuchinobe 5–10–1, Sagamihara, Kanagawa, 229–8558 Japan
E-mail: †soyama@sw.it.aoyama.ac.jp, ††{akiyo,duerst}@it.aoyama.ac.jp

Abstract CSS allows to specify the styling of Web pages, which is highly beneficial for both page creators and viewers. However, separating style from Web page content and structure by hand requires a lot of effort. Therefore, many Web pages still contain embedded style information. In this paper, we propose a method to extract CSS style information with appropriate selectors from Web pages. The extraction of the rule set is performed in two steps. The first step is the discovery of the association rules between selectors and styles. The second step is the search of the largest effective range of the selector for the style. Using this method, it becomes possible to automatically extract style information scattered in style attributes in a Web page into an external CSS style sheet.

Key words CSS, Cascading Style Sheets, Association Rule, Web Page Analysis

1. はじめに

Web ページは，構造，コンテンツ，スタイルという 3 つの情報から構成される．構造は HTML の要素による Web ページの文書構造であり，コンテンツはイメージや文章などの閲覧者が実際に見ることとなる内容である．スタイルは Web ページの装飾についての情報であるが，これを HTML 中に記述していくとコードが複雑化し，編集や運用が困難となる．

そこで W3C によって提案されたのが CSS (Cascading Style Sheets) である [1]．CSS はスタイルシートの一種であり，現在 Web 上で非常に広く実装され利用されている．スタイルシートによって，スタイル情報を外部ファイルに記述することが可能となり，スタイルを効率的に編集することができる．また，複数の HTML に同じスタイルシートを参照させることでスタイ

ルの再利用が可能となっている．さらに CSS は，スタイルの出所，詳細度，位置の 3 点からスタイルに優先度を付けることで，スタイルが重複した場合に最終的に適用すべきスタイルを決定する．

このように CSS を導入することはさまざまな利点があるが，ブラウザの機能が充実していない時代には利用されていなかった．HTML から CSS への書き直しにかかってしまう手間や，必ずしも必須な技術でないことから，CSS が適用されないまま放置されてしまった Web ページも存在する．

本稿では， Web ページ分析やデータマイニングの手法を用い，CSS 導入におけるこの問題を解決するための手法を提案する．ここで提案する手法は， HTML に付与されたスタイル情報に， CSS の利点を十分に活かせるようなセレクタを自動付与し，CSS と修正された HTML を作成するというものである．

2. 規則集合の抽出手法

本稿で提案する手法では，style 属性によってスタイルを指定した HTML を入力として，CSS と不要なスタイル情報を除いた HTML を出力する．CSS の特徴であるカスケーディングは今後の課題とし，スタイル情報自動分割の第一歩として，”スタイルシートの規則集合は，{ セレクタ } ⇒ { スタイル } が 100% 成り立つ相関ルールとして，抽出可能である” という仮定の元で規則集合を抽出することとした．規則集合抽出までの処理の流れは以下の通りである．

（1） パス・スタイルの頻度木の生成
（2） パスとスタイルの間の相関ルールの発見
（3） 子セレクタの連結最小化処理
（4） 子孫セレクタ判定処理
（5） 抽出された結果を CSS に出力

2.1 対応セレクタ

今回の手法ではタイプセレクタ，子セレクタ，子孫セレクタ，id セレクタによる規則集合を抽出する．タイプセレクタでは指定した要素に， id セレクタは指定した id 属性の値を持つ要素に対してスタイル付けを行う時に用いる．タイプセレクタや id セレクタのように単一の記述のセレクタを基本セレクタという．基本セレクタを組み合わせることで子セレクタや子孫セレクタを記述する．例えば，子セレクタは“ `body>p` ”と記述することで body 要素の直接の子要素である p 要素に，子孫セレクタは“ `#main h2` ”という記述によって id 属性の値が main であるノードの子孫となる h2 要素にマッチする．

2.2 パス・スタイルの頻度木の生成

HTML を解析し，図 1 に示す各要素のパスと，各要素のスタイルについての頻度木を生成する．この頻度木は要素ノード，スタイルノード，id ノードの 3 種類のノードを持ち，それぞれが出現頻度を保持している．以下にそれぞれのノードについて詳しい特徴を述べる．

要素ノードは要素を表し，要素までのパスをラベルとして保持する．スタイルノードはスタイルの種類一つにつきそれぞれ一つのノードを生成し，スタイルが指定されている要素ノードまでのパスにスタイル情報を加えたものをラベル付けする．また，各スタイルノードは装飾の対象となる要素ノードを親ノードとする．スタイルノードが規則集合によって指定された場合，適用元ルールを示す規則集合を登録し，それによって新しく発見された規則集合が他の規則集合と重複しているかを判定することができる．

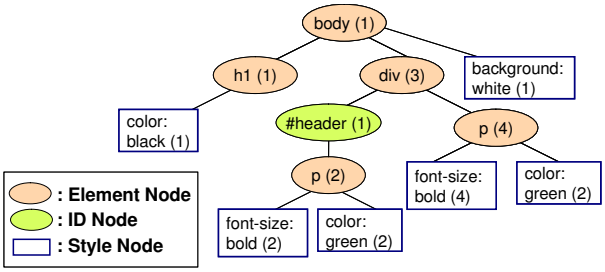

図 1 パスとスタイルの頻度木の例

id ノードは id 属性が指定されている要素が含まれている要素ノードの子ノードとする．また， id 属性が指定されている要素が終了するまで，id を含まないパスと，id を含むパスの，計 $1 + id\_count$ 通りのラベルをもったノードを生成する．ここの id_count は，同時に出現している id の数である．また， id ノードの子孫ノードには id ノードを生成しない．例えば，`body>div` のパスを持つ要素ノードに，一つが `id="header"` が指定されている要素が含まれていた場合，`body>div` と `body>div#header` の 2 通りのラベルのノードが生成される．さらに， `body>div#header` の子ノードである ul 要素に `id="list_of_contents"` が指定されていた場合には，新規に `body>div>ul#list_of_contents` が生成され，同時に `body>div>ul` ， `body>div#header>ul` という計 3 通りのラベルのノードが生成されるという仕組みである．これによって，id によってノードを限定することによって得られる規則集合と，要素のみのセレクタで発見できる規則集合のどちらにも対応することができる．

各ノードの頻度について図 1 の例に基づいて説明する．body の子要素の div のノードの頻度が 3 であるというのは，“ HTML 中に `body>div` というパスが 3 回出現した ”という意味合いで頻度をつけている．例えば，この要素ノードの子要素に p 要素であり，`body>div>p` というパスが初めて出現した場合，ラベルが `body>div` のノードの頻度は変わらず，`body>div>p` というラベルの要素ノードが新規に生成され，このノードの頻度のみが +1 される．同じようにさらにその子要素の p に関しては“ body>div>p というパスに対応する要素は 4 つである ”という意味となる．スタイルに関しても，各要素ノードごとに指定されたスタイルの種類と，各頻度も保持する．

2.3 パスとスタイルの間の相関ルールの発見

スタイルとセレクタの間の相関ルールを発見するにあたって，まずは一番簡単なところから規則集合を見つける．このステップではパス・スタイルの頻度木をもとに，すべてのスタイルノードと，その親となるセレクタノードの頻度を比較する．もしそれらの頻度が同じであった場合，パスとスタイルの間に確信度 1 の相関ルール，すなわち規則集合が成り立つ．規則集合

が発見されるたび，その規則集合に対して 2.4, 2.5 項の処理を行う．また，もしスタイルノードがすでに何かしらの規則集合によって指定されているスタイルであるならば，そのスタイルノードについてはこれらの一連の処理は行わずに次のスタイルノードに処理を移行する．すべてのスタイルノードに関して処理が完了した場合，すべての抽出可能な規則集合を発見したこととなる．

2.4 子セレクタの連結最小化処理

要素のパスは子セレクタの連結と考えることができる．よってこの項では，得られた子セレクタの連結の先頭から不必要な箇所を削除する処理を行う．

まず，先頭の要素から一つ要素を除外した部分セレクタの生成を行う．ここで，もしセレクタの先頭以外に id セレクタが含まれていた場合，自動的に最後に出現した id 以前のセレクタをすべて除外したものを部分セレクタとして生成する．これは id セレクタが一つ含まれていれば， HTML 中の箇所は一意に決定できるためである．この部分セレクタに関しても，同じスタイルとの間でも確信度 1 の相関ルールが成り立つか判定する．成り立つ場合，このセレクタを現在までの中の最適セレクタとし，この条件が成り立たなくなる，あるいはタイプセレクタの形式になるまで，同じ処理を再帰的に行う．

もし相関ルールが成り立たない場合，現在の最適セレクタとスタイルを規則集合として出力するか，あるいはセレクタに 3 つ以上の基本セレクタが含まれているならば 2.5 項の処理を行う．

この処理の開始から 2.5 項の処理に渡すまでのフロー図を図 2 に，セレクタに対する処理によるセレクタの変化を表したものを図 3 に示す．

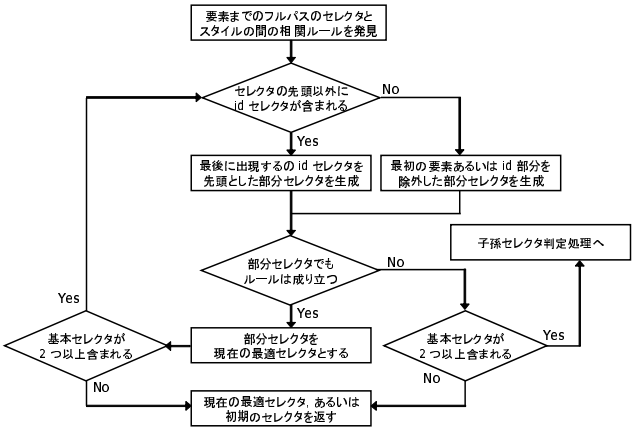

図 2　連結最小化処理のフロー図

2.5 子孫セレクタの判定処理

以上の処理によって，子セレクタの連結で表わされるセレクタを抽出することができる．そこで，その子セレクタの連結に含まれる基本セレクタの数を k とすると，$k > 2$ の場合に，セレクタの先頭と末尾以外の要素か id のうち，除外できる部分を探索し，子孫セレクタに変換する．

ここでは，相関ルールにおける頻出アイテムの組み合わせを

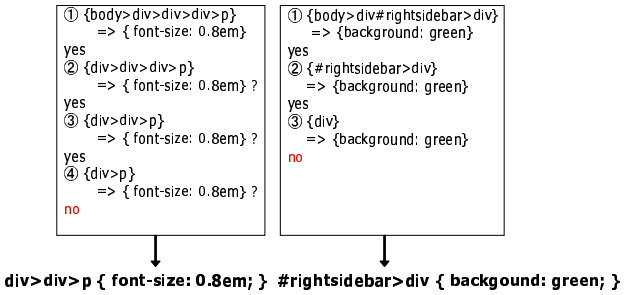

図 3　処理中のセレクタの変化

効率的に探索する apriori アルゴリズム [2] を応用する．C_n は，子セレクタの記号である $>$ で区切られた要素や id をアイテムとした，セレクタから除外可能のアイテムの集合 (除外候補アイテム集合) である．L_n は，実際にセレクタから除外可能であったアイテムの組み合わせ集合 (除外可能アイテム集合) である．n は，組み合わせたアイテム数を表す．処理手順は以下の通りである．ただし， $1 \leqq i \leqq k-2$ である．

（1） C_1 を生成 $(i = 1)$
（2） C_1 から L_1 を生成し，$i = 2$ とする
（3） L_{i-1} の要素を組み合わせて C_i を生成
（4） C_i から L_i を生成し，i に 1 加算
（5） C_i, L_i が空，あるいは $i = k-2$ になるまで (3), (4), (5) の繰り返し

この手法で最後にルールが成り立ったセレクタを最適セレクタとする．具体的に処理の例を示す．まず，先頭と末尾以外の各基本セレクタが，除外可能部分についての候補集合 C_1 となる．規則集合 `body>div>div>ul>li>span {color:red;}` の場合，C_1 = `div(1)`, `div(2)`, `ul`, `li` となる．

次に，C_1 の各要素について，該当部分をセレクタから除外した部分セレクタを順に生成し，部分セレクタとスタイルの間に規則集合が成り立つならば，その要素を L_1 に加える．成り立たないならば，その要素は除外不可能部分となり，以降の処理ではその要素の該当部分を除外したセレクタの生成はされない．例えば， `div(1)` を除外した `body div>ul>li>span` というセレクタに対し，`body div>ul>li>span {color:red;}` が成り立つかどうかを判定し，成り立つのであれば， `div(1)` を L_1 の要素として追加する．これを `span` まで同様に繰り返し， L_1 を生成する．さらに，L_1 から，二つの基本セレクタを除外する候補集合 C_2 を生成する．これは apriori アルゴリズムと同じ考え方で L_i から C_{i+1} を生成する．すなわち，C_{i+1} に含まれる各 $i+1$ アイテム集合の部分集合である， i アイテム集合はすべて L_i に含まれていなければならない，というものである．仮に L_1 ={`div(1)`, `div(2)`, `ul`} であった場合， C_2 ={{`div(1)`, `div(2)`}, {`div(1)`, `ul`}, {`div(2)`, `ul`}} となる．さらに， C_2 の基本セレクタを除外したセレクタすべてでもルールが成り立つ場合，L_2 ={{`div(1)`, `div(2)`}, {`div(1)`, `ul`}, {`div(2)`, `ul`}} となり，そこから導き出される 次の候補集合 C_3 は C_3 ={{`div(1)`, `div(2)`, `ul`}} である．さらに，この場合でもルールが成り立つ場合，最終的に抽

出される規則集合は `body li>span {color:red;}` となる．

2.6 スタイル適用箇所の重複

同じスタイルノードに対して，以前発見された規則集合よりも，さらによい規則集合が新しく抽出されることもあり得る．そのような問題への対策として，“ ある規則集合 A が発見された後， A と同じスタイルに関して， A の適用範囲を全て含む規則集合 B が発見された場合，A を破棄し，B をより最適な規則集合として抽出する ”というルールを用いてスタイル箇所の重複に対処した．基本的に後に抽出された規則集合を優先するという手法であり，からなず id なしのラベルのスタイルノードを先に探索対象とし，抽出できなかったスタイルを id セレクタによって抽出可能である，あるいは，抽出された規則集合の対象のスタイルノードの頻度が id ありのラベルのスタイルノードと頻度が等しい場合にのみ，id セレクタを用いた規則集合を行うこととなる．

3. id 属性の自動付与処理

id セレクタの有効性を活用するためのオプションとして，役割の明らかな要素に対し，あらかじめ id を割り振る処理を行う．本手法では， Web ページ 上の位置・サイズから該当要素を上位レベルコンテンツブロックへ分類することでこの機能を実現する．

3.1 上位レベルコンテンツブロックへの分類

本稿では構造理解のしやすい CSS セレクタの抽出のためのオプションとして，Yu Chen ら [3] の上位レベルコンテンツブロックへの分類手法をベースにした手法を用いる．まずは 3.2 項で id セレクタの有用性について述べる．

3.2 id セレクタの有用性

CSS で Web ページのスタイルを変更するためには，セレクタが Web ページ上のどの位置について適用されているのか，理解が容易であることが望ましい．しかし，ブログなどのサイトに頻繁に用いられている Web ページ構造として，div 要素を多用し， Web ページを header や footer, 本文などの領域に分割していく方式がある．このようなサイトの場合，構造が複雑化してしまい， CSS におけるセレクタも複雑化する危険性をはらんでいる．この問題のために， id セレクタを使用することが推奨されている．id によって，セレクタの適用箇所の特定や構造上の意味理解が容易となり，セレクタ自体もシンプルにすることができる．また， id などによって一意の要素が指定できることによって，スタイルとの関係をより細かく探索することも可能となる．その例を示す図 4 によって， id セレクタの方が適用箇所や役割がわかりやすく，かつセレクタがシンプルになっていることがわかる．赤字で示した箇所のように，id によって要素を特定させることができるため新しい規則集合の抽出も可能となる．

3.2.1 上位レベルコンテンツブロックへの分類ルール

header, footer, leftsidebar, rightsidebar の判定には，Yu Chen らの手法による条件を適用する．header, footer は要素の height, width の値から動的に閾値 N を決定し， Web ページの上から N ピクセル以内に収まる要素ならば header, 下から N ピクセル以内に収まるならば footer と判定する．また，閾値 N は以下の式によって決定する．

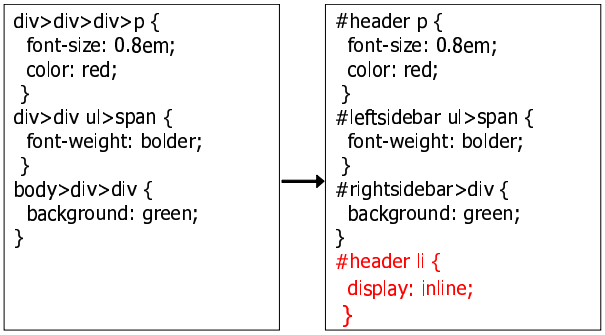

図 4 id セレクタの有効性の例

$$N = base\_threshold + F(height/width) \tag{1}$$

$$F(x) = \frac{a}{(b * x + c)} \tag{2}$$

さらに Yu Chen らは実験に基づき，それぞれの値は $base\_threshold = 160, a = 40, b = 20, c = 1$ が最も適切あることを確認している．よって，本稿でもこの値を用いる．sidebar は Web ページの大きさによって，閾値を動的に変更させるため， Web ページの左右それぞれ 25% に収まるような要素を sidebar とする．

Yu Chen らは DOM 木の上位の出現順に探索し，header, footer, leftsidebar, rightsidebar の順番に条件を比較し，最初に当てはまった要素を上位レベルコンテンツブロックに分類していたが，本稿ではより厳密に判定するため，条件に当てはまる要素のうち面積が最大であったものを上位レベルコンテンツブロックに分類することとした．また，本稿では新たに，各上位レベルコンテンツブロックを分類した後，全体の 30% の面積を占める要素を main を判定する条件を設定した．これは他の上位レベルコンテンツブロックが最大面積で存在した場合における，残りの領域すべてをまとめた要素がある場合は，それを main とすべきであるという考えである．

HTML 中のすべてのブロック要素について，以上のような条件を header, footer, leftsidebar, rightsidebar, main の順番で条件が成り立つかどうかを比較していく．また，いずれかの上位レベルコンテンツブロックの子，子孫要素となったものは対象外とする．ここで条件が成り立ったものを各上位レベルコンテンツブロックの候補とする．

4. 予 備 実 験

4.1 実 行 例

今回提案したアルゴリズムが正しく動作するか確認するために予備実験を行った．入力のデータとして，スタイルをすべて style 属性で指定し，header や footer などを用いた典型的な形式の自作の HTML ファイルを用いた．入力の HTML をブラウザで表示したものが図 5 であり，この HTML には 35 個の要素，63 個のスタイルが存在した．

上位レベルコンテンツブロックの判定においては rightsidebar

図 5　入力ファイルを Opera 9.52 で表示した結果

以外の上位レベルコンテンツブロックが発見することができ，それらをもとに規則集合の抽出を行ったところ，スタイル情報を分離した結果としての CSS と，修正された HTML を得た．出力された CSS を図 6，入力 HTML のソースの一部を図 7，それに対応する箇所の出力 HTML のソースを図 8 に示す．

```
#footer {
  float: left;
  margin-top: 20px;
  margin-left: 5%;
  width: 90%;
  border-width: 2px 0 0 0;
  border-style: solid;
  border-color: grey;
}
#footer li {
  padding: 5px 3px;
  display: inline;
  width: 25%;
}
#footer>p {
  font-size: 0.8em;
  text-align: right;
}
#footer>ul {
  text-align: center;
}
#header {
  border-width: 2px 4px;
  border-style: solid;
  border-color: green;
  background-color: #ffb9b9;
  margin-left: 10%;
  width: 80%;
}
#header>p {
  margin: -8px 0 3px 30px;
}
#leftsidebar {
  width: 17%;
  margin-left: -2%;
  position: relative;
  top: 20px;
}
#leftsidebar>ul {
  color: green;
}
#main {
  float: right;
  width: 81%;
}
#main p {
  font-size: 0.95em;
}
h1 {
  width: 75%;
  border-width: 0 0 3px 0;
  border-style: solid;
  border-color: purple;
  margin-top: 2px;
}
h2 {
  width: 80%;
  border-width: 0 0 3px 0;
  border-style: solid;
  border-color: brown;
  margin-bottom: -8px;
}
span {
  font-style: italic;
}
```

図 6　出 力 CSS

```
<div style="border-width:2px 4px; border-style: solid; border-color:
 green;background-color: #ffb9b9; margin-left: 10%; width: 80%;">
 <h1 style="width: 75%; border-width:0 0 3px 0; border-style: solid;
  border-color: purple; margin-top: 2px;">
 スタイル情報の自動分離<span style="font-style: italic;">
 (Auto-Discerp Styles)</span></h1>
 <p style="margin:-8px 0 3px 30px;">HTML からCSS へのスタイル情報の
 自動分離手法についての提案</p>
</div>
```

図 7　入力 HTML ソースの一部

```
<div id="header">
  <h1>スタイル情報の自動分離<span>(Auto-Discerp
Styles)</span></h1>
  <p>HTML からCSS へのスタイル情報の自動分離手法についての提案</p>
</div>
```

図 8　出力 HTML のソースの一部

4.2　考　　察

図 6 から，id セレクタを使用したことでセレクタが非常にわかりやすくなり，かつほぼすべてのスタイル情報を CSS に分離することができたことがわかる．

また，この実験において 61 個のスタイル指定を CSS に抽出することが可能であったため，ほぼすべてのスタイルを指定できる CSS を生成することができたといえる．しかし，id や構造では一意に指定できない要素も存在したため，その要素にのみ指定されたスタイルを CSS に抽出することができなかったという問題点も存在する．今回抽出することができなかったスタイル情報への対策として，隣接セレクタなどを用いたセレクタの抽出，パターンやスタイルのまとまりから class 属性を定義することによる class セレクタの抽出など，さらなるスタイル情報の抽出方法を検討する必要がある．また，スタイル適用箇所の重複やカスケードをより厳密に考慮することや，セレクタ自体のわかりやすさ，セレクタによるスタイルのまとまり具合などを評価することによって，より人間が作成した CSS に近づけることも考えられる．また，実際に用いるための諸問題として，font 要素や bgcolor 属性などの推奨されていないスタイルの指定方法への対策も必要である．

5.　CSS 作成支援と関連研究

5.1　CSS 作成支援ツール

CSS についての知識がない者に対する CSS 作成支援ツールや，経験者の HTML や CSS のコーディングを支援するために開発されたツールは数多く存在する．これらは新しく HTML を作成するときに CSS を作成することを目的としたものが多い．本稿で提案するような既存の HTML からスタイル情報を抜き出すものとは異なる部分が多いが，今までにどのような CSS に関する取り組みがされているのか紹介する．

CSS 関連のツールとしては Google Code によって公開されたオープンソースである Jonathan Holst の HTML2CSS [4] が挙げられる．HTML2CSS はスタイル以外の情報を記述し終えた HTML の id, class タグの構造を解析し，あり得るセレクタのリストを CSS 形式で出力するものである．これによって，CSS を用いたスタイル情報の作成を支援することが可能である．さらに，コーディングしている途中の HTML に対しても，階層構造をセレクタで把握できるという長所も存在する．HTML2CSS は経験のあるコーダが新規に Web ページを作成する際には非常に便利なツールである．しかし，そもそもセレクタの意味がわからないレベルの初心者の場合は，ここから CSS を作成するのは困難と言えるだろう．また，既にスタイル情報を記述してしまった HTML から CSS の自動作成という点には着眼を置いていない．

5.2　関 連 研 究

本稿では CSS セレクタに HTML 構造を強く反映させるため，Web ページ分析を用い上位レベルコンテンツに分割する処理を行う．上位レベルコンテンツブロックとは，Web ページの header, footer, sidebar, 本文のことである．Web ページをいくらかのコンテンツの領域に分ける手法については，今ま

でにさまざまな手法が提案されている．

服部ら [5] の研究では，コンテンツ情報を抽出し，タグの深さの変化からコンテンツ間距離を計算し，コンテンツ間距離を基準にコンテンツを分割する手法を提案している．この手法は非常に効果的なアルゴリズムであるが，コンテンツ間距離について適切な閾値が必要であり，分割されたコンテンツ領域ごとに，どのような意味を持つのか認識することには主眼を置いていない．そのため適切な id 名を割り振ってセレクタに反映させるという今回の目的には適していない．

また，Yu Chen ら [3] が提案した手法では， Web ページの見た目やブロック要素の位置とサイズから， Web ページを複数の領域に分割している．最初のステップとして，位置とサイズから要素を header, footer, sidebar などの上位レベルコンテンツブロックに分類し，次のステップでは各上位レベルコンテンツブロックを，空白や境界線などのセパレータによって複数のコンテンツブロックに分割する処理を行う．最後に， Web ページをコンテンツごとの複数の小さなページに分けることで，携帯電話などの小さなスクリーンデバイスで表示することを実現している．

この研究で上位レベルコンテンツブロックへの分類を行っている目的は，あくまで Web ページ上のコンテンツの領域の分割にある．よって，上位レベルコンテンツブロックを一意に決定し，id 属性の値とすることを目的としている本稿で提案する手法では，さらに明確にこれらの分類を行う必要があるという点で異なる．

6. ま と め

本稿では，Web ページ分析と相関ルールを応用して，スタイル情報を含む HTML から CSS を自動作成する手法を提案した．セレクタとスタイルの間で相関ルール分析を行い，さらにセレクタを選別することで，理解や編集の容易な CSS の自動作成を行った．また， HTML の各要素の位置とサイズから上位レベルコンテンツブロックへの分類による id 属性を自動付与する手法を示した．

今後の課題としては，本手法では分離が不可能であったスタイル情報への対応や，より厳密な規則集合の重複処理などが挙げられる．

文　献

[1] Bos, B., Lie, H. W., Lilley, C. and Jacobs, I.: Cascading Style Sheets, level 2 CSS2 Specification, W3C Recommendation, `http://www.w3.org/Style/CSS/` (1998).

[2] Agrawal, R. and Srikant, R.: Fast algorithms for mining association rules in large databases, *Morgan Kaufmann Publishers Inc.*, pp. 487 – 499 (1994).

[3] Chen, Y., Ma, W.-Y. and Zhang, H.-J.: Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices, *Proc. World Wide Web Conference 2003* (2003).

[4] Holst, J.: HTML2CSS, `http://code.google.com/p/html2css/`.

[5] 服部元, 松本一則, 菅谷史昭：タグの深さを利用したコンテンツ間距離に基づく Web ページの自動分割方式, *DBSJ Letters*, Vol. 4, No. 1, pp. 149–152 (2005).