DTD に関して充足不能な CSS 規則の検出

筑波大学

図書館情報メディア研究科

2019 年 3 月

岡田　拓也

目次

第 1 章

はじめに

Web ページのスタイルを指定するために，一般的に用いられるスタイルシートに CSS[1][2]（Cascading Style Sheets）がある．また，CSS は HTML データだけでなく，XML データのスタイル記述言語としても広く用いられている（例えば，DcoBook[3] や MathML[4] など）．CSS は規則集合で構成され，規則を XML/HTML に適用することで，Web ページのスタイルを指定する．個々の規則はセレクタとプロパティで構成される．セレクタによりプロパティを適用する要素が指定される．また，セレクタでは要素間関係を記述したり，要素の持つ属性などが記述できる．CSS は記述の自由度が高いことがメリットとしてあるが，一方，CSS 作成の過程で，Web ページのスタイルに影響を与えない規則が記述されることがある．このような規則によって，CSS の可読性を低下させ，メンテナンスが困難になるなどの問題が生じる．

また，ある程度の規模を持つ Web サイトでは，複数の XML/HTML に対して共通の CSS を適用することが多い．このような場合において，ある XML/HTML のスタイルには影響を与えないが，一方，他の XML/HTML のスタイルに影響を与えるような規則が，CSS 作成者によって記述されることがある．さらに，XML/HTML データは時間の経過と共に更新されるのが常である．したがって，ある特定の XML/HTML データのどの要素にも適用されない CSS 規則が検出されたとしても，その規則が本当に不要なのか否かは自明でない．このとき，他の XML/HTML データや更新されたデータにおいても適用されることはなく，不要な規則を検出できれば有用であると考えられる．

そこで，本研究では XML/HTML のデータ構造を定義するスキーマの一つである DTD[5]（Document Type Definition）を参照することで，DTD に妥当などの XML/HTML のどの要素にも適用されることのない CSS 規則（以降，この規則を充足不能な規則と呼ぶ）を検出するアルゴリズムの提案・考察を行う．提案アルゴリズムでは，以下三つの条件に関して，単一の規則が充足不能か否かを判定できる．

- 規則のセレクタにおいて，要素の持つ属性が正しく記述されているか
- 規則のセレクタにおいて，要素間関係が正しく記述されているか
- 判定対象規則より優先度の高い規則集合の適用要素の集合に，判定対象規則の適用要素の集合が包含されていないか

これらの条件のうち一つでも満たさない条件があれば，規則を充足不能と判定する．提案アルゴリズムでは，DTD の定義するデータ構造と CSS セレクタの構造をそれぞれオートマトンで表現し，オートマトン

同士の演算により充足不能な規則を検出する．また，これら三つの条件を手動で確かめながら，充足不能な規則を発見することは，一般的に極めて難しい問題である．そこで，評価実験では，提案アルゴリズムを用いて充足不能な規則を検出する場合と，手動で充足不能な規則を検出する場合とで処理時間を比較した．この評価実験において，提案アルゴリズムを用いて充足不能な規則を検出する方が，処理効率が良く，提案アルゴリズムの有効性が認められる結果が得られた．

関連研究

CSS の静的解析について，Geneves らは CSS 規則を論理式に変換して解析するシステムを提案している [6]．Bosch らは CSS 規則のリファクタリングを行う手法を提案しており [7]，また，Mazinanian らは重複した CSS 規則を検出する手法を提案している [8][9]．CSS 規則を HTML データ上で解析するための手法は数多く提案されている．例えば，FireBug[10] や Chrome Developer Tools[11] では，HTML データのどの要素にも適用されない規則を検出できる．Hague らは，HTML5 アプリケーション上で冗長な CSS 規則を検出するための手法を提案している [12]．Mesbah らは，HTML データ上で冗長な CSS 規則を検出するための手法を提案している [13]．しかし，これらの手法は DTD を考慮していない．筆者の知る限り，DTD の下での規則の充足不能性について考察した研究は存在しない．

本論文の構成を以下に示す．2 章で DTD や CSS，充足不能な CSS 規則，オートマトンに関する定義を行い，3 章で充足不能な規則を検出する手法について説明する．4 章では評価実験について述べ，最後に 5 章でまとめと今後の課題を述べる．

第 2 章

諸定義

本章では，DTD や CSS，充足不能な CSS 規則，オートマトンに関する定義を行う．

2.1 DTD の概要

DTD[5] とは XML/HTML の構造を定義するためのスキーマである．DTD を参照することで要素間の親子関係や兄弟関係，属性の定義を確認できる．DTD を 3 次組 $D = (d, \alpha, s)$ と表す．ここで，d は Σ から Σ 上の正規表現集合への写像，α は Σ から属性集合への写像，$s \in \Sigma$ は開始ラベルである．ラベル $a \in \Sigma$ に対して，$d(a)$ を a の内容モデルという．例えば，図 2.1 の book を文書要素とする DTD を考える．

```
<!ELEMENT  book     (title, author+)>
<!ATTLIST  book
  price  CDATA  #IMPLIED
  id     ID     #IMPLIED
  >
<!ELEMENT  author   (name, age)>
<!ELEMENT  title    (#PCDATA)>
<!ELEMENT  name     (#PCDATA)>
<!ELEMENT  age      (#PCDATA)>
```

図 2.1 DTD

このとき，上記 DTD は 3 次組 $(d, \alpha$,book) で表され，ここで $d(\text{book}) = \text{title author}^+$，$d(\text{author}) = \text{name age}$，$d(\text{title}) = d(\text{name}) = d(\text{age}) = \epsilon$ である．木 t の頂点 v に対して，v のラベルを $l(v)$ と表す．このとき，木 t と DTD $D = (d, s)$ に対して，もし t のルートのラベルが s であり，かつ，t の任意の頂点 n に対して $d(l(n))$ が $l(n_1)l(n_2)\cdots l(n_m)$ にマッチするならば，t は D に対して妥当であるという．ここで，$n_1, n_2, \cdots, n_m$ は n の子である．また，$\alpha(\text{book}) = \{\text{price, id}\}$，$\alpha(\text{author}) = \alpha(\text{title}) = \alpha(\text{name}) = \alpha(\text{age}) = \epsilon$ である．

次に上記 DTD で定義される親子関係や兄弟関係，属性について述べる．例えば，$d(\text{book}) = \text{title author}^+$ について考える．ここで，親子関係については，親 book の子は book の内容モデルに出現する

title と author である．また，兄弟関係については，book の内容モデルに出現する各要素を兄としたときの，その兄の（必ずしも隣接しない）弟を考える．まず，title を兄としたとき，その弟は兄の後ろに出現し得る要素すべてとなり，ここでは author しかないので，兄 title の弟は author である．次に，兄 author の弟は author となる．これは，演算子 '+' は要素が 1 回以上出現することを示しているためである．また，$\alpha(\text{book}) = \{\text{price, id}\}$ では，book が属性 price と id を持つことを表す．

2.2　CSS の概要

XML/HTML に適用されることで Web ページのスタイルを指定するスタイルシートとして CSS[1][2] がある．CSS は規則集合で構成され，一つの規則はセレクタとプロパティで構成される．セレクタによりプロパティを適用する要素を指定する．例えば ul li {font-color:red} という規則において，セレクタは ul li，プロパティは color:red であり，この規則によって，ul の子孫要素 li 内の文字列は赤字で表示されることになる．次に CSS 規則について定義を述べる．

2.2.1　CSS 規則

Σ をラベル集合とする．Σ に属するラベルおよび記号 '$*$' を単純セレクタという．単純セレクタ s に対して，もし $s =$'$*$'，または，s がラベルでありかつ $s = l(v)$ ならば，s は v にマッチするという．

要素間関係セレクタ

セレクタとは，単純セレクタを結合子で繋げたものである．ここで，結合子とは空白を表す '␣'（子孫），'$>$'（子），'+'（隣接兄弟），および，'$\sim$'（一般兄弟）である．セレクタ sel の長さを $len(sel)$ と表し，sel に含まれる単純セレクタの数と定義する．例えば，$sel = a\text{␣} * > c$ のとき，$len(sel) = 3$ である．

s, s' を単純セレクタ，(v, v') を木 t の頂点の組とする．

- もし s が v にマッチし，s' が v' にマッチし，かつ，v' が v の子孫であるならば，子孫セレクタ $s\text{␣}s'$ は (v, v') にマッチするという．
- もし s が v にマッチし，s' が v' にマッチし，かつ，v' が v の子であるならば，子セレクタ $s > s'$ は (v, v') にマッチするという．
- もし s が v にマッチし，s' が v' にマッチし，かつ，v' が v の（必ずしも隣接しない）弟であるならば，一般兄弟セレクタ $s \sim s'$ は (v, v') にマッチするという．
- もし s が v にマッチし，s' が v' にマッチし，かつ，v' が v に隣接する弟であるならば，隣接兄弟セレクタ $s + s'$ は (v, v') にマッチするという．

$sel = s_1 c_1 s_2 c_2 s_3 \cdots s_{n-1} c_{n-1} s_n$ をセレクタ，(v, v') を木 t の頂点の組とする．ここで，s_i は単純セレクタ，c_i は結合子である．もし n 個の頂点の系列 $v = v_1, v_2, \cdots, v_n = v'$ で，任意の $2 \leq i \leq n$ に対して $s_{i-1} c_{i-1} s_i$ が (v_{i-1}, v_i) にマッチするものが存在するならば，sel は (v, v') にマッチするという．

CSS 規則を $sel\ p : v$ と表す．ここで，sel はセレクタ，p はプロパティ，v はプロパティ値である（ただし，実際には，1 つの CSS 規則にプロパティとその値の組を複数記述することができる．そのような CSS

規則は，$sel\ p : v$ のように 1 つのプロパティとその値をもつ CSS 規則を複数記述することで表せる．例えば，$sel\ \{p_1 : v_1, p_2 : v_2\}$ は 2 つの CSS 規則 $sel\ p_1 : v_1$ と $sel\ p_2 : v_2$ として表せる．)．CSS 規則 r に対して，r のセレクタを $sel(r)$，r のプロパティを $prop(r)$ と表す．セレクタ sel に対して，sel の終端の単純セレクタを $tail(sel)$ と表す．例えば，$r = a\_b + c\ p : v$ とすると，$sel(r) = a\_b + c$，$prop(r) = p$，$tail(sel(r)) = c$ である．以下では簡単のため，任意の CSS 規則 r に対して，$tail(sel(r))$ はラベルであると仮定する．

属性セレクタ

本研究では一部の属性セレクタを扱う．以下に示す属性セレクタが対象である．

- id セレクタ：記号 '#' で表されるセレクタである．単純セレクタの後に記述することでその単純セレクタが id 属性を持つことになる．
- class セレクタ：記号 '.' で表されるセレクタである．単純セレクタの後に記述することでその単純セレクタが class 属性を持つことになる．
- 一般属性セレクタ：att を属性とするとき，$[att]$ と表されるセレクタである．単純セレクタの後に記述することでその単純セレクタが att を持つことになる．

2.2.2　優先度

CSS スクリプトにおいて，CSS 規則の衝突，すなわち，同じプロパティをもつ複数の CSS 規則が XML/HTML データの同じ要素にマッチする，ということがしばしば発生する．このような衝突を解決するために，CSS では優先度と呼ばれる仕組みが用意されている．最も優先度が高い CSS 規則のみが適用される．

セレクタ sel に出現するラベルの数，id セレクタの数，class セレクタの数，一般属性セレクタの数をそれぞれ $ltot(sel)$，$idtot(sel)$，$cltot(sel)$，$atttot(sel)$ と表す．$pri(sel)$ は sel の優先度を表す．このとき，$pri(sel) = ltot(sel) + idtot(sel) \times 100 + cltot(sel) \times 10 + atttot(sel) \times 10$ である．例えば，$sel = a\#1\_* + c$ のとき，$ltot(sel) = 2$，$idtot(sel) = 1$，$cltot(sel) = 0$，$atttot(sel) = 0$ であり，$pri(sel) = 102$ となる．CSS スクリプトは CSS 規則のリスト R として定義される．CSS 規則 r の R における位置を $index_R(r)$ と表す．例えば，$R = [r, r', r'']$ としたとき，$index_R(r) = 1$ かつ $index_R(r'') = 3$ である．r が R に出現するとき，$r \in R$ と書く．木 t と t の頂点 v に対して，もし CSS 規則 $r \in R$ が次の条件を満たすならば，r は v に適用されるという．

- t のある頂点 v' に対して，$sel(r)$ は (v', v) にマッチし，かつ，
- 任意の CSS 規則 $r' \in R$ に対して，もし $sel(r')$ がある頂点 v'' に対して (v'', v) にマッチし，かつ，$prop(r) = prop(r')$ であるならば，(a) $pri(sel(r)) > pri(sel(r'))$，または，(b) $pri(sel(r)) = pri(sel(r'))$ かつ $index_R(r) > index_R(r')$，が成り立つ．

2.3　充足不能な CSS 規則の定義

本論文では，DTD の下で充足不能な CSS 規則を検出する問題について考える．ここで，CSS 規則 r と DTD D に対して，もし D に妥当などの XML/HTML データのどの要素にも r が適用されない場合，r は D の下で充足不能であるという．本論文では r が充足不能となるパターンを 3 つに限定し，順に検証する．

1. $sel(r)$ において，D の下で単純セレクタの持つ属性が正しく記述されていないならば r は充足不能である．
2. $sel(r) = s_1c_1s_2c_2s_3 \cdots s_{n-1}c_{n-1}s_n$ を r のセレクタ，$D = (d, s)$ を DTD とする．D に妥当などの木 t に対しても，$sel(r)$ がマッチするような頂点の組 (v, v') が t に存在しないならば，r は充足不能である．簡単に言うと，$sel(r)$ において，D の下で要素間関係が正しく記述されていないならば r は充足不能である．
3. CSS 規則の衝突が発生している場合，$pri(sel(r))$ より優先度の高い規則（もし 2 つの CSS 規則が同じ優先度をもつ場合，後に出現する CSS 規則）集合の適用要素の集合に，r の適用要素の集合が包含されているならば r は充足不能である．

3 つのパターンすべてで充足不能とならない場合は，r は充足可能である．

以下に充足不能な CSS 規則の例を示す．

```
<!ELEMENT   book     (title, author+)>
<!ATTLIST   book
  price   CDATA   #IMPLIED
  id      ID      #IMPLIED
  >
<!ELEMENT   author   (name, age)>
<!ELEMENT   title    (#PCDATA)>
<!ELEMENT   name     (#PCDATA)>
<!ELEMENT   age      (#PCDATA)>
```

図 2.2　DTD

```
book.a          {font-family:serif}
author title    {font-family:serif}
book age        {font-family:serif}
author age      {font-family:sans-serif}
name + age      {font-family:fantasy}
```

図 2.3　充足不能な CSS 規則の例

図 2.2 の DTD の下で図 2.3 の CSS 規則 book.a {font-family:serif} は，パターン 1 において DTD で要素 book が class 属性を持たないことから，充足不能である．CSS 規則 author title {font-family:serif} は，パターン 2 において DTD で要素 author の子孫要素に要素 title が出現しないことから，充足不能である．また，book age {font-family:serif} は充足不能である．その理由は次のとおりである．パターン 3 においてまず 4 番目または 5 番目の規則集合とプロパティの衝突が発生している．次にどの要素 age も要素 book を親としてもつまたは要素 name を隣接する兄としてもち，よってどの要素 age に対しても 4 番目または 5 番目の CSS 規則が適用されるため（もし 2 つの CSS 規則が同じ優先度をもつ場合，後に出現する CSS 規則が適用される），book age {font-family:serif} は充足不能である．言い換えると，book age {font-family:serif} の優先度より優先度の高い規則集合の適用要素の集合に，book age {font-family:serif} の適用要素の集合が包含されているため，book age {font-family:serif} は充足不能である．

2.4　オートマトンの概要

提案手法ではオートマトンを用いるため，オートマトンの概要を述べる．オートマトンとは，ある特定の性質を満たす文字列を認識する抽象機械である．オートマトンの認識する文字列の集合を言語という．図 2.4 に二進数を受理するオートマトンを示す．q_0，q_1，q_2 が状態を表し，初期状態は太矢印で示された q_0 となる．受理状態は q_1，q_2 となる．遷移関数は矢印で表し，例えば，q_0 において 1 が入力された場合，q_2 に遷移する．このオートマトンを M としたとき，このオートマトンの認識する言語は $L(M) = \{0, 1, 10, 11, \cdots\}$ となる．

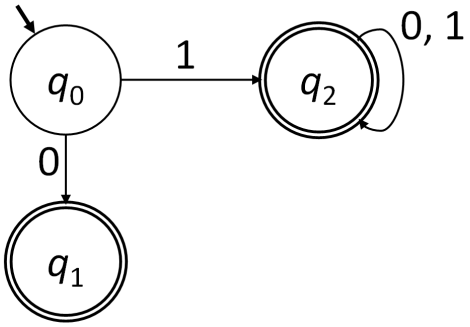

図 2.4　二進数を受理するオートマトン

形式的には，オートマトンは 5 次組 $M = (Q, \Sigma, \delta, q_0, F)$ と定義される．ここで，Q は状態集合，Σ は入力記号の集合，δ は遷移関数，q_0 は初期状態，F は受理状態の集合を表す．遷移関数 δ は，状態 $p \in Q$ と入力 $a \in \Sigma$ に対して，次の状態集合 $Q_{nex} \subseteq Q$ を定める関数であり，$\delta(p, a) = Q_{nex}$ のように表される．以下に二進数を受理するオートマトンの形式的定義を示す．

- 状態集合：$Q = \{q_0, q_1, q_2\}$
- 入力記号の集合：$\Sigma = \{0, 1\}$
- 遷移関数：$\delta(q_0, 0) = \{q_1\}$, $\delta(q_0, 1) = \{q_2\}$, $\delta(q_2, 0) = \{q_2\}$, $\delta(q_2, 1) = \{q_2\}$

- 初期状態：初期状態は q_0
- 受理状態の集合：$F = \{q_1, q_2\}$

2.4.1　積オートマトン

オートマトン $A_L = (Q_L, \Sigma_L, \delta_L, p, F_L)$ とオートマトン $A_M = (Q_M, \Sigma_M, \delta_M, r, F_M)$ があるとき，A_L の認識する言語と A_M が認識する言語の共通集合を受理する積オートマトン $A_{L\times M}$ は，$A_{L\times M} = (Q_L \times Q_M, \Sigma_L \times \Sigma_M, \delta, (p, r), F_L \times F_M)$ と表せる．ただし，p を A_L の状態，r を A_M の状態，(p, r) を $A_{L\times M}$ の状態としたとき，$\delta((p,r),1) = (\delta_L(p,1) \times \delta_M(r,1))$ である．図 2.5 にオートマトン A_L とオートマトン A_M の積オートマトン $A_{L\times M}$ を示す．

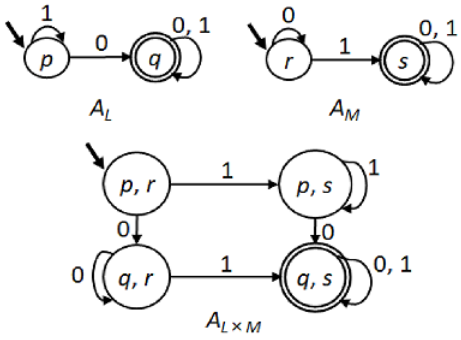

図 2.5　積オートマトン $A_{L\times M}$

2.4.2　和オートマトン

オートマトン $A_L = (Q_L, \Sigma_L, \delta_L, p, F_L)$ とオートマトン $A_M = (Q_M, \Sigma_M, \delta_M, r, F_M)$ があるとき，A_L の認識する言語と A_M が認識する言語の和集合を受理するオートマトン（以降簡単のため和オートマトンとする）A_{L+M} は，$A_{L+M} = (Q_L \cup Q_M \cup \{sum_0\} \cup \{f\}, \Sigma_L \cup \Sigma_M \cup \{\epsilon\}, \delta, sum_0, f)$ と表せる．ここで，‘ϵ’ は空記号である．また，$Q_{sum} = Q_L \cup Q_M \cup \{sum_0\} \cup \{f\}$ を和オートマトンの状態集合，$\Sigma_{sum} = \Sigma_L \cup \Sigma_M \cup \{\epsilon\}$ を和オートマトンの入力記号の集合とすると，遷移関数 δ は状態 $q \in Q_{sum}$ と入力 $a \in \Sigma_{sum}$ に対して，以下のように定義される．

$$\delta(q,a) = \begin{cases} \delta_L(q,a) & q \in Q_L \text{の場合} \\ \delta_M(q,a) & q \in Q_M \text{の場合} \\ \{p,r\} & q = sum_0 \text{かつ } a = \epsilon \text{の場合} \\ \{f\} & q \in F_L \cup F_M \text{かつ } a = \epsilon \text{の場合} \end{cases}$$

2.4.3　補集合オートマトン

ある決定性オートマトン $A = (Q, \Sigma, \delta, q_0, F)$ に対し，決定性オートマトン $B = (Q, \Sigma, \delta, q_0, Q - F)$ を A の補集合を受理するオートマトン（以降簡単のため補集合オートマトンとする）という．すなわち，A の非受理状態を受理状態に，受理状態を非受理状態にすると B になる．アルファベット Σ 上において，

L を A の認識する言語，$\overline{L}$ を B の認識する言語とするとき，$\overline{L} = \Sigma^*$ - L が成り立つ．また，決定性オートマトンとはどの状態と入力記号の組に対しても，遷移先が高々 1 つになるようなオートマトンである．そのため，ここでの決定性オートマトン A, B の遷移関数 δ は，状態 $p \in Q$ と入力 $a \in \Sigma$ に対して，次の状態 $q \in Q$ を定める関数となり，$\delta(p, a) = q$ のように表される．

第 3 章

充足不能な規則の検出手法

本章では，次の DTD の下で充足不能な CSS 規則を検出する問題について考える．
入力： DTD D，CSS スクリプト R，CSS 規則 $r \in R$
問題： r が D の下で充足不能であるか否かを決定せよ
本論文では r が充足不能となるパターンを 3 つに限定し，順に検証する．

1. $sel(r)$ において，D の下で単純セレクタの持つ属性が正しく記述されていないならば r は充足不能である．
2. $sel(r)$ において，D の下で要素間関係が正しく記述されていないならば r は充足不能である．
3. CSS 規則の衝突が発生している場合，$pri(sel(r))$ より優先度の高い規則（もし 2 つの CSS 規則が同じ優先度をもつ場合，後に出現する CSS 規則）集合の適用要素の集合に，r の適用要素の集合が包含されているならば r は充足不能である．

3 つのパターンすべてで充足不能とならない場合は，r は充足可能である．

3.1 提案手法の概要

次に $r \in R$ が DTD D の下で充足不能か否かを決定する本手法の流れを示す．

1. $sel(r)$ に出現する属性を持つ単純セレクタそれぞれについて，DTD の α を参照し，単純セレクタに属性セレクタが正しく指定されていないものがあれば，上記パターン 1 にて r は充足不能と判定する（以降簡単のためこの判定を属性判定と呼ぶ）．
2. D の DTD オートマトン M_D を構成する．
3. $sel(r)$ を正規表現 $re(sel(r))$ に変換する（r の CSS 規則オートマトン M_r を構成する）．
4. 以下が成り立つか否かを判定する
   $L(M_D) \cap L(M_r) = \emptyset$
   上記の式が成立するならば，パターン 2 にて r は充足不能と判定する（以降簡単のためこの判定を優先度無し要素間関係判定と呼ぶ）．
5. $r_1, r_2, \cdots, r_k \in R$ を，r と衝突し得る CSS 規則とする．すなわち，任意の $1 \leq i \leq k$ に対して，

$tail(sel(r_i)) = tail(sel(r))$，$prop(r_i) = prop(r)$，かつ，(a) $pri(sel(r_i)) > pri(sel(r))$ または (b) $pri(sel(r_i)) = pri(sel(r))$ かつ $index_R(r_i) > index_R(r)$ が成り立つ．$1 \leq i \leq k$ に対して，$sel(r_i)$ を正規表現 $re(sel(r_i))$ に変換する（r_i の CSS 規則オートマトン M_{r_i} を構成する）．

6. 以下が成り立つか否かを判定する
   $L(M_D) \cap L(M_r) \subseteq L(M_D) \cap \bigcup_{1 \leq i \leq k} L(M_{r_i})$
   上記の式が成立するならば，パターン 3 にて r は充足不能と判定する（以降簡単のためこの判定を優先度あり要素間関係判定と呼ぶ）．
7. 1. 4. 6. すべて（パターン 1, 2, 3 すべて）で充足不能とならない場合は，r は充足可能と判定する．

以降各ステップについて具体的に述べる．

3.2　属性判定

$sel(r)$ に属性を持つ単純セレクタが存在する場合，r の属性判定を行う．ここで，$sel(r)$ 中の属性を持つ単純セレクタの集合を A_s とし，$a_s \in A_s$ は属性を持つ単純セレクタである．a_s は要素 $ele(a_s)$ とその要素の持つ属性 $att(a_s)$ で構成される．また，DTD D の要素 Σ から属性集合への写像を α とする．ここで，任意の $a_s \in A_s$ に対して $att(a_s) \notin \alpha(ele(a_s))$ が成り立つ a_s が存在するならば，r は充足不能と判定する．

例えば，sel = a#1␣b.1 + c である r の属性判定について考える．ここで，D について，α(a) = {id, class}，α(b) = {id}，α(c) = {id, class} であるとする．このとき，A_s = {a#1, b.1} となる．また，a_s = a#1 に対して，$ele(a_s)$ = a，$att(a_s)$ = id であり，a_s = b.1 に対して，$ele(a_s)$ = b，$att(a_s)$ = class である．このとき，b.1 $\in A_s$ に対して $att(b.1) \notin \alpha(ele(b.1))$ が成り立つため，r の属性判定の結果は充足不能となる．

3.3　DTD オートマトンの構成

DTD の要素間関係を表す DTD オートマトンを構成する．例えば，図 3.1 は DTD $D = (d, s)$（ここでは，α は考慮しない）の DTD オートマトンを示しており，ここで $d(s) = ab^*$，$d(a) = \epsilon$，$d(b) = sc$，$d(c) = \epsilon$ である（演算子 '$*$' は要素が 0 回以上出現することを表す）．この図において，横方向の状態遷移は D の内容モデルにおける状態遷移つまり兄弟関係を表し，縦方向の状態遷移は親子関係を表す．

DTD オートマトンを形式的に定義する．$D = (d, s)$ を Σ 上の DTD とする．$\Sigma_v = \{a_v \mid a \in \Sigma\}$ かつ $\Sigma_h = \{a_h \mid a \in \Sigma\}$ と定義する．ここで，Σ_v と Σ_h は，それぞれ縦ラベルと横ラベルと呼ばれる．正規表現 $d(a)$ に対して，$d(a)$ に出現する各ラベル b を対応する横ラベル b_h に置き換えたものを $d_h(a)$ と表す．$M_h(a) = (Q_a, \Sigma_h, \delta_a, q_0^a, F_a)$ を，$d_h(a)$ のオートマトン，r を CSS 規則とする．このとき，r に関する D の DTD オートマトンを次に示す M として定義する．

$M = (Q, \Sigma_h \cup \Sigma_v, \delta, q_0, F)$

ここで，　Q，δ，F は次のように定義される．

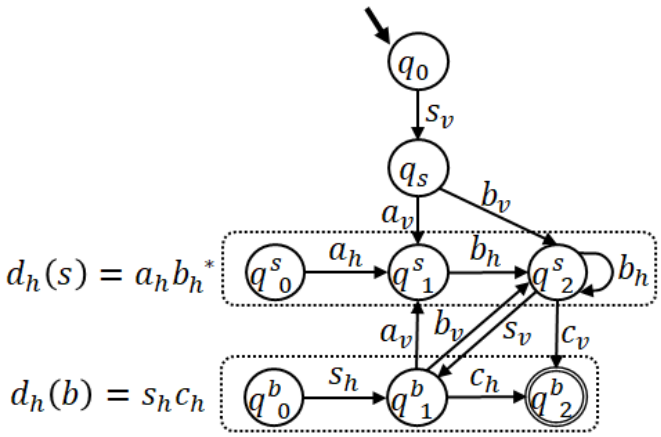

図 3.1　DTD オートマトンの例

- まず，Q は次のように定義される．
  $Q = \bigcup_{a \in \Sigma} Q_a \cup \{q_0, q_s\}$
  ここで，Q_a は $M_h(a)$ の状態集合，q_0 は初期状態，q_s は開始ラベル s を表す状態である．
- δ は，各ラベル a の δ_a（$M_h(a)$ の遷移関数）と δ_v をマージすることにより得られる．δ_a は横方向の状態遷移つまり兄弟関係を表し，δ_v は縦方向の状態遷移つまり親子関係を表す．ここで，δ_v は次のように定義される．$a \in \Sigma$ をラベル，b を $d(a)$ に出現するラベル，c を $d(b)$ に出現するラベルとする．Q_a に属する状態のうち，b_h で到達するものの集合を $Q_a(b_h)$ と表す．すなわち，
  $Q_a(b_h) = \{q \in Q_a \mid q \in \delta_a(q', b_h), q' \in Q_a\}$
  直観的には，$q \in Q_a(b_h)$ は $d_h(a)$ における「b の状態」を表す．次に，$\delta_v(q, c_v)$ は，c が $d(b)$ に出現する場合に，任意の $q \in Q_a(b_h)$ が c_v により $Q_b(c_h)$ に属する状態に遷移するように定義される（図 3.2）．すなわち，

$$\delta_v(q, c_v) = \begin{cases} Q_s(c_h) & q = q_s \text{かつ } c \text{ が } d(s) \text{ に出現するとき} \\ Q_b(c_h) & \text{ある } a \in \Sigma,\ d(a) \text{ に出現する } b, \\ & \text{および } d(b) \text{ に出現する } c \text{ に対して}, \\ & q \in Q_a(b_h) \text{ のとき} \\ \emptyset & \text{それ以外のとき} \end{cases}$$

  δ は，δ_a と δ_v を用いて次のように定義される．

$$\delta(q, c) = \begin{cases} \{q_s\} & q = q_0 \text{かつ } c = s_v \text{のとき} \\ \delta_a(q, c) & \text{ある } a \in \Sigma \text{とある } c \in \Sigma_h \text{に対して } q \in Q_a \text{のとき} \\ \delta_v(q, c) & \text{ある } a \in \Sigma \text{とある } c \in \Sigma_v \text{に対して } q \in Q_a \text{のとき} \\ \emptyset & \text{それ以外のとき} \end{cases}$$

- F は，r に出現する末尾の結合子と末尾の単純セレクタにより定義される．$r = s_1 c_1 s_2 \cdots c_{n-1} s_n$ $p : v$ とする．F は $sel(r)$ の末尾の結合子 c_{n-1} と末尾の単純セレクタ s_n で到達可能な状態の集合である．すなわち，

$$F = \begin{cases} \bigcup_{q' \in Q} \delta(q', (s_n)_v) & c_{n-1} \in \{\sqcup, >\} \text{ のとき} \\ \bigcup_{q' \in Q} \delta(q', (s_n)_h) & c_{n-1} \in \{+, \sim\} \text{ のとき} \end{cases}$$

図 3.1 において，もし $r = a \_ c\ p : v$ ならば，$F = \{q_2^b\}$ である．

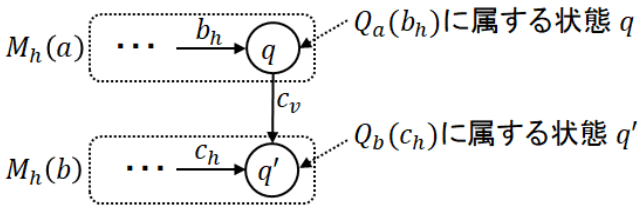

図 3.2　$Q_a(b_h)$ に属する状態から $Q_b(c_h)$ に属する状態への遷移

3.4　CSS 規則オートマトンの構成

$sel = s_1 c_1 s_2 \cdots c_{n-1} s_n$ をセレクタとする．sel を表す正規表現 $re(sel)$ を，$re(sel) = c'_0 s'_1 c'_1 s'_2 \cdots c'_{n-1} s'_n$ と定義する．ここで，$0 \leq i \leq n$ に対して

$$c'_i = \begin{cases} (\Sigma_v)^* & \text{(a) } i = 0\text{，または，} \\ & \text{(b) } i \geq 1 \text{ かつ } c_{i-1} = \text{‘␣’ のとき} \\ (\Sigma_h)^* & c_{i-1} = \text{‘}\sim\text{’ のとき} \\ \epsilon & c_{i-1} \in \{\ >, +\ \} \text{ のとき} \end{cases}$$

かつ，$1 \leq i \leq n$ に対して

$$s'_i = \begin{cases} (s_i)_v & s_i \in \Sigma\text{，かつ，} \\ & i = 1 \text{ または } c_{i-1} \in \{\text{␣}, >\ \} \text{ のとき} \\ |_{a \in \Sigma}\ a_v & s_i = \text{‘}*\text{’ かつ } c_{i-1} \in \{\text{␣}, >\ \} \text{ のとき} \\ (s_i)_h & s_i \in \Sigma \text{かつ } c_{i-1} \in \{\ +, \sim\} \text{ のとき} \\ |_{a \in \Sigma}\ a_h & s_i = \text{‘}*\text{’ かつ } c_{i-1} \in \{\ +, \sim\} \text{ のとき} \end{cases}$$

例えば，$sel = a \sim b > c$ のとき，$re(sel) = (\Sigma_v)^* a_v (\Sigma_h)^* b_h c_v$ である．

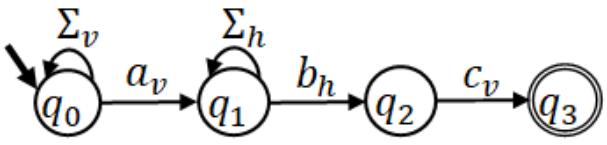

図 3.3　$re(sel) = (\Sigma_v)^* a_v (\Sigma_h)^* b_h c_v$ のオートマトン M_r（CSS 規則オートマトン）

また，正規表現 re(sel) はオートマトン M_r に変換可能である（本論文では変換アルゴリズムについては省略する）．図 3.3 に $re(sel) = (\Sigma_v)^* a_v (\Sigma_h)^* b_h c_v$ のオートマトン M_r（CSS 規則オートマトン）を示す．

3.5　優先度無し要素間関係判定

D の DTD オートマトン M_D と CSS 規則オートマトン M_r との積オートマトンを求め，その空性判定を行うことで，$sel(r)$ で記述される要素間関係が DTD の要素間関係に適合しているかを調べること（優先度無し要素間関係判定）ができる．$L(M_D) \cap L(M_r) = \emptyset$ が成り立つならば，r は D の下で充足不能となる．

また，次の 3.6 節の優先度あり要素間関係判定は計算困難であるのに対し，一方優先度無し要素間関係判定は多項式時間で計算可能である [14]．本論文では，優先度あり要素間関係判定の前に優先度無し要素間関係判定を行い充足不能な規則を検出する．ここで規則が充足不能と判定されれば，以降の優先度あり要素間関係判定を行う必要がなく，これにより優先度あり要素間関係判定の回数を減らすことができる．

3.6　優先度あり要素間関係判定

$r_1, r_2, \cdots, r_k \in R$ を，r と衝突し得る CSS 規則とする．すなわち，任意の $1 \leq i \leq k$ に対して，$tail(sel(r_i)) = tail(sel(r))$，$prop(r_i) = prop(r)$，かつ，(a) $pri(sel(r_i)) > pri(sel(r))$ または (b) $pri(sel(r_i)) = pri(sel(r))$ かつ $index_R(r_i) > index_R(r)$ が成り立つ．$1 \leq i \leq k$ に対して，$sel(r_i)$ を正規表現 $re(sel(r_i))$ に変換する（r_i の CSS 規則オートマトン M_{r_i} を構成する）．以下が成り立つか否かを判定する

$$L(M_D) \cap L(M_r) \subseteq L(M_D) \cap \bigcup_{1 \leq i \leq k} L(M_{r_i})$$

上記の式が成立するならば，r は充足不能と判定する．また，$L(M_D) \cap L(M_r) \subseteq L(M_D) \cap \bigcup_{1 \leq i \leq k} L(M_{r_i})$ が成り立つか否かは，$M_R = \bigcup_{1 \leq i \leq k} M_{r_i}$ を任意の $1 \leq i \leq k$ に対しての M_{r_i} の和オートマトンとし，$M_{\overline{R}}$ を M_R の補集合オートマトンとするとき $L(M_D) \cap L(M_r) \cap L(M_{\overline{R}}) = \emptyset$ が成り立つか否かと同じである．この式が成立するならば，r は充足不能である．この判定を直観的に表した集合関係図を示す．図 3.4 は $L(M_D) \cap L(M_r) \cap L(M_{\overline{R}}) = \emptyset$ が成り立ち r が充足不能と判定される場合の集合関係図であり，図 3.5 は $L(M_D) \cap L(M_r) \cap L(M_{\overline{R}}) = \emptyset$ が成り立たず r が充足不能と判定されない場合の集合関係図である．また，ここまで述べた属性判定，優先度無し要素間関係判定，優先度あり要素間関係判定すべてで充足不能とならない場合は，r は充足可能と判定する．

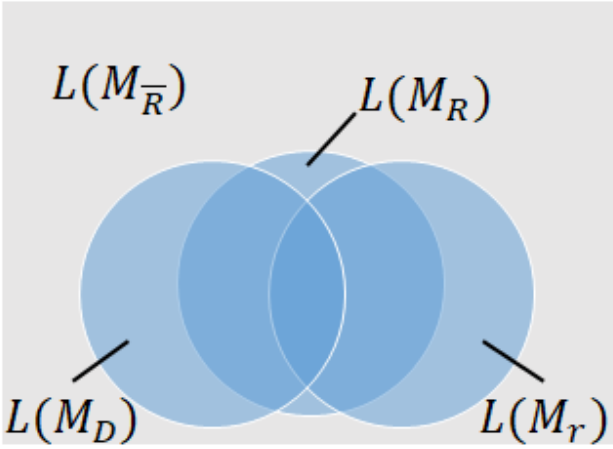

図 3.4　r が充足不能と判定される場合の集合関係

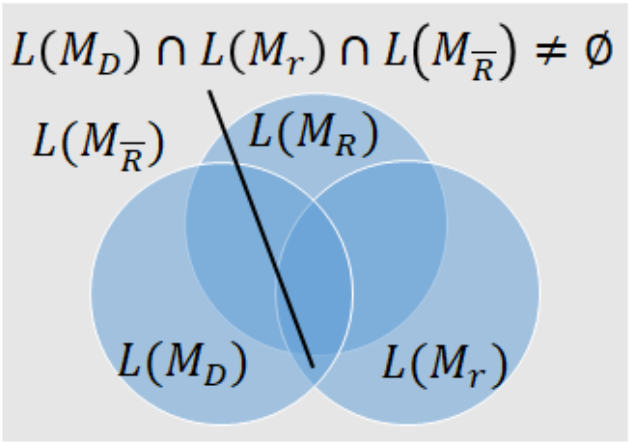

図 3.5　r が充足不能と判定されない場合の集合関係

第 4 章

評価実験

本章では，提案アルゴリズムに関する評価実験の結果について述べる．評価実験の環境は以下の通りである．

- CPU: Intel Xeon E5-2623 v3 3.0Ghz CPU
- メモリ: 16GB RAM
- OS: Linux CentOs 7 64bit
- 使用言語: Ruby 2.4.1

また，評価実験では W3C が公開している DTD（XHTML-1.0-Transitional）[*1]を用いる．CSS にはこの DTD を用いている Web サイトで使用されている CSS を用いる．本論文では，社会工学類 | 筑波大学 理工学群の Web サイト[*2]の CSS と食べログの Web サイト[*3]の CSS を用いる．

まず，社会工学類 | 筑波大学 理工学群の Web サイトの CSS での提案アルゴリズムの評価について述べる．以降，D を DTD（XHTML-1.0-Transitional），C_1 を社会工学類 | 筑波大学 理工学群の Web サイトの CSS（実験では CSS の一部を利用）とする．C_1 に属性判定で充足不能と判定される CSS 規則や結合子 '+'，および，'∼' を含むセレクタを持つ CSS 規則が存在しなかったため，筆者がそれらの CSS 規則を C_1 に追加した．また，図 4.1 は C_1 についてプロパティごとに CSS 規則を収集した表記である．これを C_{1p} とする．C_{1p} では '$*$' の連続でプロパティごとに CSS 規則が区分されている．また，先頭にプロパティ，後ろにそのプロパティを持つ CSS 規則集合が記述されている．表 4.1 は C_{1p} で各充足不能性判定パターン（属性判定，優先度無し要素間関係判定，優先度あり要素間関係判定）で検出された充足不能な CSS 規則の数と充足可能判定された CSS 規則の数の結果である（簡単のため以降，属性判定をパターン 1，優先度無し要素間関係判定をパターン 2，優先度あり要素間判定をパターン 3 とする）．

また，実験では DTD と CSS に詳しい筑波大学大学院博士前期課程の院生 2 人の実験協力者に充足不能性判定を行ってもらった．実験は以下のように行われた．

1. 事前に DTD，CSS，パターン 1，パターン 2，パターン 3 の定義や関連する例を実験協力者に説明

[*1] https://www.w3.org/TR/xhtml1/
[*2] https://www.sk.tsukuba.ac.jp/College
[*3] http://m.tabelog.com/

表 4.1　実験 1 の充足不能性判定の結果

パターン 1	パターン 2	パターン 3	充足可能	計
2	1	9	32	44

した．

2. DTD とプロパティごとに CSS 規則を収集した表記の CSS スクリプトを実験協力者に提示し，手動で各充足不能性判定パターンを順に検証してもらった．

表 4.2 はその結果である．提案アルゴリズムの実行時間は 20.38 分であった．表の各セルは実験協力者によって正しく判定された CSS 規則の割合である．充足不能性判定に各実験協力者平均約 46.79 分かかった．

表 4.2　実験 1 の結果

実験協力者	パターン 1	パターン 2	パターン 3	充足可能	計
1	2/2	1/1	7/9	32/32	42/44（95%）
2	1/2	1/1	5/9	31/32	38/44（86%）

パターン 1 にて，提案アルゴリズムで検出した 2 つの充足不能な CSS 規則について，実験協力者 1 は 2 つの充足不能な CSS 規則すべてを検出した．一方，実験協力者 2 は 2 つの充足不能な CSS 規則のうち 1 つを検出した．提案アルゴリズムによりパターン 1 で検出された充足不能な CSS 規則には，例えばプロパティ font-size を持つ p[type=“submit”] input がある．DTD では p 要素は type 属性を持たないため，この CSS 規則は提案アルゴリズムにより充足不能と判定された．またパターン 2 にて，提案アルゴリズムで検出した 1 つの充足不能な CSS 規則について，両実験協力者はこの充足不能な CSS 規則を検出した．提案アルゴリズムによりパターン 2 で検出された充足不能な CSS 規則には，例えばプロパティ font-size を持つ * html p#search_box input がある．DTD では html 要素は親要素を持たないため，この CSS 規則は提案アルゴリズムにより充足不能と判定された．またパターン 3 にて，提案アルゴリズムで検出した 9 つの充足不能な CSS 規則について，実験協力者 1 は 9 つの充足不能な CSS 規則のうち 7 つを検出した．一方，実験協力者 2 は 9 つの充足不能な CSS 規則のうち 5 つを検出した．提案アルゴリズムによりパターン 3 で検出された充足不能な CSS 規則には，例えばプロパティ text-decoration を持つ dl.footer_navi dd ul li a がある．まず dl#news dd a または ul.footer_link li a の規則集合とプロパティ text-decoration の衝突が発生している．次にどの要素 a も要素 dl を親としてもち，よってどの要素 a に対しても dl#news dd a または ul.footer_link li a の CSS 規則が適用されるため，dl.footer_navi dd ul li a は提案アルゴリズムにより充足不能と判定された．言い換えると，dl.footer_navi dd ul li a の優先度より優先度の高い規則集合の適用要素の集合に，dl.footer_navi dd ul li a の適用要素の集合が包含されているため，dl.footer_navi dd ul li a は提案アルゴリズムにより充足不能と判定された．なお実験協力者による充足不能性判定には少々誤りがあったが，これらの誤りは実験協力者のケアレスミスによるものである．

次に，食べログの Web サイトの CSS での同様の提案アルゴリズムの評価について述べる．以降，C_2 を食べログの Web サイトの CSS（実験では CSS の一部を利用）とする．C_2 に属性判定で充足不能と判定される CSS 規則や結合子 ‘+’，および，‘∼’ を含むセレクタを持つ CSS 規則が存在しなかったため，筆者がそれらの CSS 規則を C_2 に追加した．また，図 4.2 は C_2 についてプロパティごとに CSS 規則を収集した表記である．これを C_{2p} とする．C_{2p} では ‘∗’ の連続でプロパティごとに CSS 規則が区分されている．先頭にプロパティ，後ろにそのプロパティを持つ CSS 規則集合が記述されている．また “#headline a:hover” や “#breadcrumb-wrap #location a:hover” などの CSS 規則では，擬似クラス “hover” が出現するが，本論文では擬似クラスは考慮しない（ただし実験では優先度の計算に疑似クラスも含めた．なお擬似クラス 1 つにつき優先度は 10 である．）．表 4.3 は C_{2p} で各充足不能性判定パターン（属性判定，優先度無し要素間関係判定，優先度あり要素間関係判定）で検出された充足不能な CSS 規則の数と充足可能判定された CSS 規則の数の結果である．

表 4.3　実験 2 の充足不能性判定の結果

パターン 1	パターン 2	パターン 3	充足可能	計
2	1	16	20	39

また，実験では DTD と CSS に詳しい筑波大学大学院博士前期課程の院生 2 人の実験協力者に充足不能性判定を行ってもらった．実験は前の実験と同様に行われた．

表 4.4 はその結果である．前の実験と同様の実験環境下で提案アルゴリズムの実行時間は 35.89 分であった．この実験では，前の実験より実行時間が長かった．その理由として，この実験では前の実験よりパターン 3 の実行回数が多かったことが考えられる．表の各セルは実験協力者によって正しく判定された CSS 規則の割合である．充足不能性判定に各実験協力者平均約 60.76 分かかった．

表 4.4　実験 2 の結果

実験協力者	パターン 1	パターン 2	パターン 3	充足可能	計
1	1/2	1/1	16/16	19/20	37/39（95%）
2	1/2	1/1	15/16	16/20	33/39（85%）

パターン 1 にて，提案アルゴリズムで検出した 2 つの充足不能な CSS 規則について，両実験協力者は 2 つの充足不能な CSS 規則のうち 1 つを検出した．提案アルゴリズムによりパターン 1 で検出された充足不能な CSS 規則には，例えばプロパティ padding を持つ script.search-box がある．DTD では script 要素は class 属性を持たないため，この CSS 規則は提案アルゴリズムにより充足不能と判定された．またパターン 2 にて，提案アルゴリズムで検出した 1 つの充足不能な CSS 規則について，両実験協力者はこの充足不能な CSS 規則を検出した．提案アルゴリズムによりパターン 2 で検出された充足不能な CSS 規則には，例えばプロパティ margin を持つ li + p がある．DTD では p 要素は li 要素を隣接する兄として持たないため，この CSS 規則は提案アルゴリズムにより充足不能と判定された．またパターン 3 にて，提案アルゴリズムで検出した 16 つの充足不能な CSS 規則について，実験協力者 1 はこれらすべての充

足不能な CSS 規則を検出した．一方，実験協力者 2 は 16 つの充足不能な CSS 規則のうち 15 つを検出した（提案アルゴリズムによりパターン 3 で検出された充足不能な CSS 規則の例は省略する）．なお実験協力者による充足不能性判定には少々誤りがあったが，これらの誤りは実験協力者のケアレスミスによるものである．

これらの実験において，実験協力者は特に優先度無し要素間関係判定・優先度あり要素間関係判定に時間がかかり大変だったと感じていた．このことから，手動での充足不能性判定は CSS を整備する上で大きな負担となることが分かる．また，これらの実験において手動での充足不能性判定より提案アルゴリズムを用いた充足不能性判定の方が実行時間を抑えられた．よって，これらの結果から提案アルゴリズムは CSS 規則の充足不能性判定を行うのに役立つと考えられる．

```
プロパティ:font-size
p#search_box input
* html p#search_box input
p[type="submit"] input
div#center
h4#ttl_about
h4#ttl_examinee
h4#ttl_shako_life
h4#h4_news01
dl#news
div#bg_footer
div#footer
div#bg_footer_txt
****************************************
プロパティ:text-decoration
dl#news dd a
ul.footer_link li a
dl.footer_navi dd ul li a
****************************************
プロパティ:border
p#search_box input
style.footer_navi
div#bg_about
div~p
****************************************
プロパティ:background
div#bg_header
ul#language
ul#size
body li.sizeS a
body.fontM li.sizeS a
head+body.fontM li.sizeS a
body.fontL li.sizeS a
body.fontS li.sizeS a
body.fontM li.sizeM a
body.fontL li.sizeL a
ul#global_navi
p#search_box
p#search_box input
div#box
div#bg_about
div#bg_examinee
div#bg_shako_life
h4#ttl_about
h4#ttl_examinee
h4#ttl_shako_life
#bn_pps a
h4#h4_news01
#bn_symposium a
dl.footer_navi dd ul li
p#copyright
****************************************
```

図 4.1　プロパティごとに CSS 規則を収集した表記 C_{1p}

プロパティ：margin
.found-box table.recaptcha-table
.found-box .submit-btn input[type="submit"]
.found-box .search-title span
.area-wrap ul.area-list li.area p
#footer ul#footer-navi li span
#footer .footer-navi-wrap ul#footer-tabelog-navi
li+p
****
プロパティ：padding
#breadcrumb-wrap #location strong
.found-box table.recaptcha-table
head p~h3
.found-box table .recaptcha_input_area input
.found-box .submit-btn input[type="submit"]
.found-box .search-box input
script.search-box
.found-box .search-box .search-btn input
.area-wrap ul.area-list
.area-wrap ul.area-list li.area p
.area-wrap ul.area-list li.area ul
.area-wrap ul.area-list li.area ul li
#footer ul#footer-navi
#footer #footer-tabelog-navi a strong
****
プロパティ：color
#headline a
#headline a:hover
#headline h1
#headline .user-guide a.info-auth-mobile
h1[charset="a"]
#common-header a
#common-header a:hover
#breadcrumb-wrap #location a:hover
.found-box .submit-btn input[type="submit"]
#econtents .elist p a
#econtents .elist p a:hover
.found-box .search-box input
#footer a
#footer a:hover
#footer ul#footer-navi li span
#footer #footer-tabelog-navi a strong
#footer #footer-tabelog-navi a:hover strong
.footer-copyright__address>a
****

図 4.2　プロパティごとに CSS 規則を収集した表記 C_{2p}

第 5 章

むすび

本論文では，まず，オートマトン理論に基づいて，充足不能な CSS 規則を検出するアルゴリズムを考案した．次に，提案アルゴリズムに関する評価実験を行った．評価実験では，提案アルゴリズムと人手による充足不能な CSS 規則の検出結果を比較し，CSS の整備に関して提案アルゴリズムの有用性を示唆する結果を得た．今後の課題として，まず，本論文で扱っていない CSS 規則，例えば，“first-child” や “last-child” などの擬似クラスを用いたセレクタについて考察することを考えている．次に，より多くの DTD や CSS を使用した評価実験を行いたいと考えている．さらに，プロパティの継承への対応や DTD 以外のスキーマ言語への対応なども今後の課題となる．

謝辞

まず本研究を進めるにあたり多くの温かい励ましとご指導をくださった指導教員の鈴木伸崇准教授に心から感謝いたします．鈴木伸崇准教授の熱心なご指導・ご助言がなければ，本論文の執筆は可能ではありませんでした．また，ご指導・ご助言をくださった副指導教員の阪口哲男准教授に心から感謝いたします．最後に，日頃のゼミ等で多くのご助言をくださった鈴木伸崇研究室のメンバーの皆様本当にありがとうございました．

参考文献

[1] World Wide Web Consortium (W3C), “Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification,” World Wide Web Consortium (W3C), http://www.w3.org/TR/CSS2/.

[2] World Wide Web Consortium (W3C), “CSS Syntax Module Level 3,” World Wide Web Consortium (W3C), http://www.w3.org/TR/css-syntax-3/.

[3] N. Walsh, “DocBookCssStylesheets.” https://github.com/docbook/wiki/wiki/DocBookCssStylesheets/.

[4] B. Bos, D. Carlisle, P.D.F. Ion, B.R. Miller, and eds., “A MathML for CSS profile,.” https://www.w3.org/TR/mathml-for-css/.

[5] World Wide Web Consortium (W3C), “Extensible Markup Language (XML) 1.0 (Fifth Edition),” World Wide Web Consortium (W3C), http://www.w3.org/TR/xml/.

[6] P. Geneves, N. Layaida, and V. Quint, “On the analysis of cascading style sheets,” Proceedings of the 21st International Conference on World Wide Web, pp.809-818, 2012.

[7] M. Bosch, P. Geneves, and N. Layaida, “Automated refactoring for size reduction of CSS style sheets,” Proceedings of the 2014 ACN Symposium on Document Engineering, pp.13-16, 2014.

[8] D. Mazinanian, N. Tsantalis, and A. Mesbah, “Discovering refactoring opportunities in cascading style sheets,” Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp.496-506, 2014.

[9] D. Mazinanian, N. Tsantalis, “Migrating cascading style sheets to preprocessors by introducing mixins,” Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, pp.672-683, 2016.

[10] Firebug Working Group, “FireBug.” https://www.getfirebug.com/.

[11] Google Inc., “Chrome developer tools,” https://developers.google.com/web/tools/chrome-devtools/.

[12] M. Hague, A.W. Lin, and C.H.L. Ong, “Detecting redundant CSS rules in HTML5 applications: A tree rewriting approach,” SIGPLAN Not., vol.50, no.10, pp.1-19, Oct. 2015.

[13] A.Mesbah, and S.Mirshokraie, “Automated analysis of CSS rules to support style maintenance,” Proceedings of the 34th International Conference on Software Engineering, pp.408-418, 2012.

[14] 鈴木 伸崇，岡田 拓也，權 娟大．“CSS 規則充足不能性問題の計算複雑さ”．データ工学と情報マネジメントにおけるフォーラム，2018.