DEIM Forum 2018 G2-4

# CSS 規則充足不能性問題の計算複雑さ

鈴木　伸崇† 　岡田　拓也†† 　權　 娟大†††

† 筑波大学図書館情報メディア系　〒 305-8550 茨城県つくば市春日 1-2
†† 筑波大学大学院図書館情報メディア研究科　〒 305-8550 茨城県つくば市春日 1-2
††† 東京大学大学院農学生命科学研究科　〒 113-8657 東京都文京区弥生 1-1-1
E-mail: †nsuzuki@slis.tsukuba.ac.jp, ††s1721657@s.tsukuba.ac.jp, †††ayekwon@mail.ecc.u-tokyo.ac.jp

**あらまし**　CSS は HTML/XML データのためのスタイル記述言語として広く用いられている．$D$ を DTD，$R$ を CSS 規則の集合とする．CSS 規則には詳細度による優先順位があるため，$D$ の下で充足不能な規則が $R$ に含まれる可能性がある．ここで，もし CSS 規則 $r$ が $D$ に妥当などの XML データのどの要素にも適用されないならば，$r$ は充足不能であるという．本稿では，充足不能な CSS 規則を発見する問題の計算複雑さについて考える．対象とする CSS 規則は，結合子として子孫，子，一般兄弟，隣接兄弟の 4 種を用いたセレクタが記述可能なものとする．まずこの問題が，CSS の結合子を 1 種類のみに限定し，DTD を閉包なし，重複なし，かつ再帰なしに限定しても coNP 困難であることを示す．次に，この問題が閉包なしかつ再帰なし DTD の下で $\mathrm{P}^{\mathrm{NP}}$ に属すること，および，一般の DTD の下で PSPACE に属することを示す．最後に，この問題が多項式時間可解となるための条件を示す．

**キーワード**　CSS, DTD, XML, 計算複雑性

## 1. はじめに

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，DocBook [1] や MathML [2] など）．CSS スクリプトは，“`ul li {font-color:red}`” などの CSS 規則を列挙して記述される．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 HTML/XML 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 では詳細度と呼ばれる仕組みが用意されている．簡単に言うと，より詳細に記述されたセレクタほど優先されるという仕組みであり，最も詳細度が高い CSS 規則のみが適用される．

本稿では，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 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．例えば，図 1 の DTD と CSS について考える．2 番目の CSS 規則 “`c {font-family:sans-serif}`” は充足不能である．なぜならば，1 番目の CSS 規則の方が詳細度が大きく，かつ，どの `c` 要素も `a` 要素を祖先にもつからである．また，3 番目の CSS 規則 “`a f {font-family:serif}`” も充足不能である．なぜならば，どの `f` 要素も `b` 要素を親としてもつか，`c` 要素を隣接する兄としてもち，よってどの `f` 要素に対しても 4 番目か 5 番目の CSS 規則が適用されるからである（もし 2 つの CSS 規則が同じ詳細度をもつ場合，後に出現する CSS 規則が適用される）．充足不能な CSS 規則は明らかに冗長であり，CSS スクリプトの可読性やメンテナンス性の悪化，ブラウザによる描画速度の低下などの要因となる．したがって，充足不能な CSS 規則の検出は極めて重要な問題と考えられる．

DTD

```
<!ELEMENT a (bc)>
<!ELEMENT b (fe)>
<!ELEMENT c (#PCDATA)>
<!ELEMENT e (cf)*>
<!ELEMENT f (#PCDATA)>
```

CSS

```
a c {font-family:serif}
  c {font-family:sans-serif}
a f {font-family:serif}
b > f {font-family:sans-serif}
c + f {font-family:fantasy}
```

図 1　充足不能な CSS 規則の例

CSS 規則自体は特に複雑なものではないため，一見するとこの問題は容易に解けるように思われる．しかし本稿では，この問題がほとんどの場合において計算困難であることを示す．本稿の対象とする CSS 規則として，結合子として子孫 ‘␣’（空白記号），子 ‘>’，隣接兄弟 ‘+’，および，一般兄弟 ‘∼’ を用いたセレクタで構成されるものを考える．本稿で得られた結果を表 1 に示す．各セルには複雑さの上限と下限が示されている．まず，この問題は閉包なし，重複なし，かつ，再帰なし DTD という，非常に限定された DTD の下で coNP 困難である．さらに，この限定された DTD の下で，上記 4 種の結合子のうちの 1 つのみを許した場合でも coNP 困難である．これらの結果から，DTD の内容モデルや CSS セレクタの結合子に制限を課すという方法では，この問題を効率よく解くための条件を得るのは難しいと考えられる．本稿では，この問題が効率よく解けるための 2 つの条件（表 1 の R1 と R2）を示す．これらの条件は，CSS 規則の衝突の仕方に着目しており，R1 は，CSS 規則 $r$ に衝突する規則が存在しない場合，$r$ の充足不能性は多項式時間で決定できるという自明な条件である．一方，R2 は，$r$ に衝突する CSS 規則が存在した場合でも，それら CSS 規則のセレクタの長さが 2 以下であれば，$r$ の充足不能性が多項式時間

で決定できることを示している．

**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 [3]．しかし，DTD の下での CSS 規則の充足不能性やその計算複雑さについては考察されていない．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 [4]，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 [5], [6]．これらの手法は DTD を考慮しておらず，また計算複雑さも考察されていない．著者の知る限り，DTD の下での CSS 規則充足不能性問題の計算複雑さについて考察した研究は存在しない．

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 [7] や Chrome Developer Tools [8] 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 [9]．Mesbah らは，HTML データ上で冗長な CSS 規則を検出する手法を提案している [10]．これらの HTML データ上での検証と，本稿での DTD に基づいた検証には以下のような違いがある．Web サイトや XML で構成された文書では，多くの場合 1 つの CSS スクリプトが複数の文書によって共有される．さらに，HTML/XML 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 HTML/XML データのどの要素にも適用されない CSS 規則として $r$ が検出されたとしても，$r$ が本当に不要なのか否かは自明でない．そのような場合，もし $r$ が充足不能と判定されれば，$r$ は他の HTML/XML 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，不要であると判断できる．

他の関連研究として，XPath 充足可能性問題がある．この問題は，XPath 式 $p$ と DTD $D$ に対して，$D$ に妥当かつ $p$ の解が空でない XML データが存在するか否かを決定する問題である．Benedikt らはこの問題の計算複雑さについて調査し，多くの場合でこの問題が計算困難であることを示している [11]．この計算困難性に対処するため，DTD や XPath 式に対する制限がいくつか提案されている．Montazerian らは，child 軸と述語を許す XPath 式の充足可能性問題が重複なし DTD の下で多項式時間可解であることを示している [12] ．また，著者らによって，重複なし DTD の下で，多項式時間可解となる XPath のクラスが示されている [13]．Ishihara らは，MRW-DTD という重複なし DTD より広いクラスの DTD を提案しており [14]，充足可能性問題が多項式時間可解となるいくつかの XPath のクラスを示している．XPath 充足可能性問題とは異なり，本研究の CSS 規則充足不能性問題は，CSS や DTD にかなりの制限を加えても計算困難である．

## 2. 諸 定 義

### 2.1 CSS

$\Sigma$ をラベル集合とする．$\Sigma$ に属するラベルおよび記号 ‘$*$’ を単純セレクタという．木 $t$ の頂点 $v$ に対して，$v$ のラベルを $l(v)$ と表す．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 $v$ にマッチするいう．

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，‘∼’（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，$sel = a␣* > c$ 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 $s␣s'$ 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 $2 \leqq i \leqq n$ 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 (注1)．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，$sel$ の先頭の単純セレクタを $head(sel)$，$sel$ 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，$sel(r) = a␣b + c$，$prop(r) = p$，$head(sel(r)) = a$，$tail(sel(r)) = c$ 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．

セレクタ $sel$ に出現するラベルの数を $spec(sel)$ と表す．例えば，$sel = a␣* + c$ のとき，$spec(sel) = 2$ である．本稿では属性セレクタは扱わないので，$spec(sel)$ は $sel$ の詳細度を表す．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 $R$ に置ける位置を $index_R(r)$ 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) $spec(sel(r)) > spec(sel(r'))$，または，(b)

(注1)：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，本稿のように 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．

表 1　本研究で得られた結果

| CSS | | | | | DTD | |
|---|---|---|---|---|---|---|
| ␣ | > | ∼ | + | 制限 | 閉包なし，重複なし，かつ，再帰無し DTD | 任意 DTD |
| + | | | | - | $\mathrm{P}^{\mathrm{NP}}$ (定理 5)<br>coNP-hard (定理 1) | PSPACE (定理 7)<br>coNP-hard (定理 1) |
| | + | | | - | $\mathrm{P}^{\mathrm{NP}}$ (定理 5)<br>coNP-hard (定理 2) | $\mathrm{P}^{\mathrm{NP}}$ (定理 6)<br>coNP-hard (定理 2) |
| | | + | | - | $\mathrm{P}^{\mathrm{NP}}$ (定理 5)<br>coNP-hard (定理 3) | PSPACE (定理 7)<br>coNP-hard (定理 3) |
| | | | + | - | $\mathrm{P}^{\mathrm{NP}}$ (定理 5)<br>coNP-hard (定理 4) | $\mathrm{P}^{\mathrm{NP}}$ (定理 6)<br>coNP-hard (定理 4) |
| | + | | + | - | $\mathrm{P}^{\mathrm{NP}}$ (定理 5)<br>coNP-hard (定理 2,4) | $\mathrm{P}^{\mathrm{NP}}$ (定理 6)<br>coNP-hard (定理 2,4) |
| + | + | + | + | - | $\mathrm{P}^{\mathrm{NP}}$ (定理 5)<br>coNP-hard (定理 1〜4) | PSPACE (定理 7)<br>coNP-hard (定理 1〜4) |
| + | + | + | + | R1 | PTIME (定理 8) | |
| + | + | + | + | R2 | PTIME (定理 9) | |

$spec(sel(r)) = spec(sel(r'))$ かつ $index_R(r) > index_R(r')$, が成り立つ．

$CSS$ を CSS 規則の全体集合とする．$CSS$ のサブセットを，そのサブセットで使用が許される結合子を示すことで表す．例えば，$CSS^{\{\llcorner,>\}}$ は結合子として '␣' と '>' のみが使用可能な CSS 規則の全体集合を表す．

### 2.2　DTD と CSS 規則の充足不能性

DTD を 2 次組 $D = (d, s)$ と表す．ここで，$d$ は $\Sigma$ から $\Sigma$ 上の正規表現集合への写像，$s \in \Sigma$ 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，book を文書要素とする次の DTD ををを考える．

```
<!ELEMENT book   (title, author+)>
<!ELEMENT author (name, age)>
<!ELEMENT title  (#PCDATA)>
<!ELEMENT name   (#PCDATA)>
<!ELEMENT age    (#PCDATA)>
```

このとき，上記 DTD は 2 次組 $(d, \mathrm{book})$ で表され，ここで $d(\mathrm{book}) = \mathrm{title}\,\mathrm{author}^+$，$d(\mathrm{author}) = \mathrm{name}\,\mathrm{age}$，$d(\mathrm{title}) = d(\mathrm{name}) = d(\mathrm{age}) = \epsilon$ である．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．

$R$ を CSS スクリプト，かつ $r \in R$ とする．. もし $D$ に妥当などの木 $t$ のどの頂点に対しても $r$ が適用されないならば，$r$ は $D$ の下で $R$ に関して充足不能であるという．次章以降，次の CSS 規則充足不能性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 $R$ に関して充足不能であるか否かを決定せよ

## 3.　CSS 規則充足不能性問題の計算困難さ

本章では，CSS 規則充足不能性問題の計算困難さについて考える．まず，計算複雑さの下限について考察し，次に上限について考察する．

以下の議論では，一般の（制限のない）DTD に加えて，以下の制限された DTD について考える．

- 正規表現 $e$ に対して，もし $\Sigma$ に属するどのラベルも高々 1 回しか $e$ 出現しないとき，$e$ を重複なしという．DTD $D$ のどの内容モデルも重複無し正規表現ならば，$D$ は重複なしという．
- 正規表現 $e$ に対して，$e$ が Kleene 閉包を含まないならば，$e$ を閉包なしという．もし DTD $D$ のどの内容モデルも閉包ならば，$D$ は閉包なしという．
- DTD $D = (d, s)$ に対して，もし (1)$b$ が $d(a)$ に出現するか，または，(2) あるラベル $c$ に対して，$c$ が $a$ から到達可能かつ $b$ が $d(c)$ に出現するならば，$b$ は $a$ から到達可能であるという．任意のラベル $a$ に対して，$a$ が $a$ から到達可能でない場合，$D$ は再帰なしという．

### 3.1　計算複雑さの下限

まず，CSS 規則充足不能性問題の計算複雑さの下限について考える．以下に示すように，この問題はかなり制限された DTD と CSS に対しても計算困難である．まず，子孫・子セレクタについて考え，次に一般兄弟・隣接兄弟セレクタについて考える．

[定理 1]　$CSS^{\{\llcorner\}}$ に対する CSS 規則充足不能性問題は，閉包なし，重複なし，かつ，再帰なし DTD の下で coNP 困難である．

証明:　3DNF-tautology 問題は coNP 完全である [15]．この問題を CSS 規則充足不能性問題に帰着する．3DNF-tautology 問題は次のように定義される．

入力：　3DNF 式 $\phi$
問題：　$\phi$ がトートロジー，すなわち，$\phi$ に対する任意の真偽値割り当てに対して $\phi$ が真であるか否かを決定せよ

$\phi = (l_{11} \wedge l_{12} \wedge l_{13}) \vee (l_{21} \wedge l_{22} \wedge l_{23}) \vee \cdots \vee (l_{m1} \wedge l_{m2} \wedge l_{m3})$ を 3DNF 式とする．ここで，$l_{ij}$ はリテラルである．また，$\{x_1, x_2, \cdots, x_n\}$ を $\phi$ に出現する変数の集合とする．一般性を失わず，各節のリテラルは変数の添字順に並んでいると仮定する．例えば，$(x_1 \wedge \neg x_3 \wedge x_4)$ はこの仮定を満たし，一方 $(x_1 \wedge x_4 \wedge \neg x_3)$ はこの仮定を満たさない．

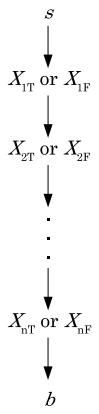


図 2　定理 1 と定理 2 で用いられる木の構造

$\phi$ に基づいて，DTD $D$，結合子として '␣' のみを用いた CSS 規則からなる CSS スクリプト $R$，および，CSS 規則 $r \in R$ を定義する．まず，DTD $D = (d, s)$ を次のように定義する．

$$\begin{aligned} d(s) &= X_{1T}|X_{1F} \\ d(X_{1T}) = d(X_{1F}) &= X_{2T}|X_{2F} \\ d(X_{2T}) = d(X_{2F}) &= X_{3T}|X_{3F} \\ &\vdots \\ d(X_{n-1T}) = d(X_{n-1F}) &= X_{nT}|X_{nF} \\ d(X_{nT}) = d(X_{nF}) &= b \\ d(b) &= \epsilon \end{aligned}$$

図 2 は $D$ に妥当な木の構造を示している．ここで，$X_{iT}$ は「$x_i$ が真である」ことを表すラベル，$X_{iF}$ は「$x_i$ が偽である」ことを表すラベルである．よって，$D$ に妥当な木は $\phi$ に対する真偽値割り当てを表す．

次に，CSS スクリプト $R$ を次のように定義する．

$$R = [r_1, r_2, \cdots, r_m, r_B]$$

ここで，

$$\begin{aligned} r_i &= L_{i1}␣L_{i2}␣L_{i3}␣b\ p{:}v_i \quad (1 \leqq i \leqq m) \\ r_B &= b\ p{:}v \end{aligned}$$

かつ $L_{ij}$ は次式で表されるラベルである．

$$L_{ij} = \begin{cases} X_{kT} & \text{if } l_{ij} = x_k \\ X_{kF} & \text{if } l_{ij} = \neg x_k \end{cases} \quad (1 \leqq i \leqq m, 1 \leqq j \leqq 3) \quad (1)$$

以下，$\phi$ がトートロジーであることと，$r_B$ が $D$ の下で $R$ に関して充足不能であることが同値であることを示す．

($\Rightarrow$) $\phi$ がトートロジーであると仮定する．このとき，$\phi$ に対する任意の真偽値割り当てに対して，真となる節 $(l_{i1} \wedge l_{i2} \wedge l_{i3})$ が存在する．したがって，$D$ に妥当な任意の木に対して，$t$ の葉頂点 $b$ に適用される CSS 規則 $r_i = L_{i1}␣L_{i2}␣L_{i3}␣b\ p{:}v_i$ が存在する．さらに，$spec(r_i) > spec(r_B)$ である．よって，$r_B$ は葉頂点 $b$ に適用されることはなく，$r_B$ は充足不能である．

($\Leftarrow$) $D$ に妥当などの木 $t$ に対しても，$r_B$ が $t$ の葉頂点 $b$ に適用されることがないと仮定する．このとき，$D$ に妥当な任意の木 $t$ に対して，$r_1, r_2, \cdots, r_m$ のうち少なくとも 1 つが葉頂点 $b$ に適用される．よって，$\phi$ に対する任意の真偽値割り当てに対して，$\phi$ は真となる．　□

次に示すように，CSS の結合子として '>' のみを用いる場合でも，この問題は coNP 困難である．

[定理 2]　$\mathrm{CSS}^{\{>\}}$ に対する CSS 規則充足不能性問題は，閉包なし，重複なし，かつ，再帰なし DTD の下で coNP 困難である．

証明:　定理 1 と同様，この定理も 3DNF-tautology 問題からの帰着により示す．$\phi = (l_{11} \wedge l_{12} \wedge l_{13}) \vee (l_{21} \wedge l_{22} \wedge l_{23}) \vee \cdots \vee (l_{m1} \wedge l_{m2} \wedge l_{m3})$ を 3DNF 式とする．まず，DTD $D = (d, s)$ は定理 1 と同様に定義される．次に，CSS スクリプト $R$ を次のように定義する．

$$R = [r_1, r_2, \cdots, r_m, r_B]$$

ここで，$r_B = b\ p{:}v$，かつ，$r_i$ $(1 \leqq i \leqq m)$ は '␣' を '$*$' と '>' を繋げたものに置き換えて構成する．すなわち，

$$\begin{aligned} r_i = L_{i1} > \underbrace{* > \cdots > *}_{\substack{\text{'>' で結合された} \\ dist(L_{i1}, L_{i2}) \text{個の '*'}}} > L_{i2} > \underbrace{* > \cdots > *}_{\substack{\text{'>' で結合された} \\ dist(L_{i2}, L_{i3}) \text{個の '*'}}} > \\ L_{i3} > \underbrace{* > \cdots > *}_{\substack{\text{'>' で結合された} \\ dist(L_{i3}, b) \text{個の '*'}}} > b\ p{:}v_i \end{aligned}$$

ここで，$L_{ij}$ は式 (1) と同様である．また，$L_{ij} \in \{X_{kT}, X_{kF}\}$ と $L_{ij+1} \in \{X_{k'T}, X_{k'F}, b\}$ に対して，$dist(L_{ij}, L_{ij+1})$ を次のように定義する．

$$dist(L_{ij}, L_{ij+1}) = \begin{cases} k' - k - 1 & \text{if } L_{ij+1} \in \{X_{k'T}, X_{k'F}\} \\ n - k & \text{if } L_{ij+1} = \text{'}b\text{'} \end{cases} \quad (2)$$

ここで，$n$ は $\phi$ に出現する変数の数である．例えば，$\phi$ の $i$ 番目の節が $(x_2 \wedge \neg x_5 \wedge x_7)$ であり，かつ，$n = 9$ のとき，$r_i = X_{2T} > * > * > X_{5F} > * > X_{7T} > * > * > b\ p{:}v_i$ である．

以上から，定理 1 と同様にして，$\phi$ がトートロジーであることと，$r_B$ が $D$ の下で $R$ に関して充足不能であることが同値であることが示せる．　□

次に，兄弟関係の結合子のみを用いた CSS を考える．まず，結合子として '∼' のみを用いた場合に，CSS 規則充足不能性問題が coNP 困難であることを示す．

[定理 3]　$\mathrm{CSS}^{\{\sim\}}$ に対する CSS 規則充足不能性問題は，閉包なし，重複なし，かつ再帰なし DTD の下で coNP 困難である．

証明:　定理 1 と同様，この定理も 3DNF-tautology 問題からの帰着により証明する．

$\phi = (l_{11} \wedge l_{12} \wedge l_{13}) \vee (l_{21} \wedge l_{22} \wedge l_{23}) \vee \cdots \vee (l_{m1} \wedge l_{m2} \wedge l_{m3})$ を 3DNF 式とする．$\phi$ に基づいて，DTD $D$，結合子として

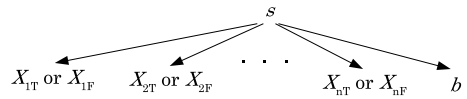


図 3　定理 3 と定理 4 で用いられる DTD に妥当な木の構造

‘∼’ のみを用いた CSS からなる CSS スクリプト $R$，CSS 規則 $r \in R$ を構成する．まず，DTD $D = (d, s)$ を次のように定義する．

$$\begin{aligned} d(s) &= (X_{1T}|X_{1F})(X_{2T}|X_{2F})\cdots(X_{nT}|X_{nF})b \\ d(X_{iT}) = d(X_{iF}) &= \epsilon \quad (1 \leqq i \leqq n) \\ d(b) &= \epsilon \end{aligned}$$

図 3 に $D$ に妥当な木の構造を示す．

次に，CSS スクリプト $R$ を以下のように定義する．

$$R = [r_1, r_2, \cdots, r_m, r_B]$$

ここで，$r_i$ と $r_B$ は次のように定義される．

$$\begin{aligned} r_i &= L_{i1} \sim L_{i2} \sim L_{i3} \sim b\ p{:}v_i \quad (1 \leqq i \leqq m) \\ r_B &= b\ p{:}v \end{aligned}$$

また，$L_{ij}$ は式 (1) と同様である．

このとき，定理 1 と同様に，$\phi$ がトートロジーであることと，$r_B$ が $D$ の下で $R$ に関して充足不能であることが同値であることが示せる．　□

この問題は，結合子として ‘+’ のみを用いた CSS 規則を許す場合でも，やはり coNP 困難である．

［定理 4］ $\mathrm{CSS}^{\{+\}}$ に対する CSS 規則充足不能性問題は，閉包なし，重複なし，かつ，再帰なし DTD の下で coNP 困難である．

証明:　定理 3 と同様にして示せる．$\phi = (l_{11} \wedge l_{12} \wedge l_{13}) \vee (l_{21} \wedge l_{22} \wedge l_{23}) \vee \cdots \vee (l_{m1} \wedge l_{m2} \wedge l_{m3})$ を 3DNF 式とする．DTD $D$ は定理 3 と同様に定義する．CSS スクリプト $R$ を次のように定義する．

$$R = [r_1, r_2, \cdots, r_m, r_B]$$

ここで，$r_B = b\ p{:}v$，かつ，$r_i$ は次の形をした CSS 規則である．

$$\begin{aligned} r_i = L_{i1} + \underbrace{* + \cdots + *}_{\substack{\text{‘+’ で結合された} \\ dist(L_{i1}, L_{i2}) \text{個の ‘*’}}} + L_{i2} + \underbrace{* + \cdots + *}_{\substack{\text{‘+’ で結合された} \\ dist(L_{i2}, L_{i3}) \text{個の ‘*’}}} + \\ L_{i3} + \underbrace{* + \cdots + *}_{\substack{\text{‘+’ で結合された} \\ dist(L_{i3}, b) \text{個の ‘*’}}} + b\ p{:}v_i \end{aligned}$$

ここで，$L_{ij}$ は式 (1) と同様であり，$dist()$ は式 (2) と同様である．

以上から，$\phi$ がトートロジーであることと，$r_B$ が $D$ の下で $R$ に関して充足不能であることが同値であることが示せる．　□

### 3.2　計算複雑さの上限

次に，CSS 規則充足不能性問題の計算複雑さの上限について考える．まず，制限された DTD について考察し，次に一般の DTD について考察する．

#### 3.2.1　制限された DTD の場合

［定理 5］ $\mathrm{CSS}^{\{\sqcup,>,+,\sim\}}$ に対する CSS 規則充足不能性問題は，閉包なしかつ再帰なし DTD の下で $\mathrm{P}^{\mathrm{NP}}$ に属する．

証明:　まず，次の問題が NP に属することを示す．

- 入力：閉包なしかつ再帰なし DTD $D = (d, s)$，CSS スクリプト $R$，CSS 規則 $r \in R$
- 問題：$k \leqq n$ 個のピボット $v_1 = v_{i_1}, v_{i_2}, \cdots, v_{i_k} = v_n$ をもつ，次の条件 (i) と (ii) を満たす経路 $path = v_1, v_2, \cdots, v_n$ (図 4) が存在するか否かを決定せよ．

(i)　$path$ は $D$ に対して「妥当」である．すなわち，$D$ に妥当な木 $t$ で $path$ を含むものが存在する．より具体的には，$path$ は $D$ に妥当なある木に対して次の条件を満たす．

＊　$v_1$ は $t$ のルートである

＊　$spath_j = v_{i_j}, v_{i_j+1}, \cdots, v_{i_{j+1}}$ を，$j$ 番目のピボット $v_{i_j}$ から $j+1$ 番目のピボット $v_{i_{j+1}}$ までの $path$ の部分経路とする．任意の $1 \leqq j \leqq k-1$ に対して，$spath_j$ は次の条件を満たす．

・ $j$ が奇数のとき，任意の $i_j \leqq h \leqq i_{j+1}-1$ に対して $v_{h+1}$ は $v_h$ の子である

・ $j$ が偶数のとき，任意の $i_j \leqq h \leqq i_{j+1}-1$ に対して $v_{h+1}$ は $v_h$ に隣接する弟である

(ii)　$sel(r)$ は $path$ にマッチするが，次の条件を満たすどの $r' \in R$ に対しても $sel(r')$ は $path$ にマッチしない．

＊　$prop(r') = prop(r)$，かつ，(a)$spec(sel(r')) > spec(sel(r))$　または　(b)$spec(sel(r')) = spec(sel(r))$ かつ $index_R(r') > index_R(r)$

ここで，上記条件 (i) と (ii) を満たす正解の経路 $path$ が推測されたとする．$D$ は閉包なしかつ再帰なし DTD なので，$path$ の長さは $D$ のサイズ以下に抑えられる．よって，条件 (i) と (ii) が成り立つことは $D$ と $R$ を用いて多項式時間で確かめることができる．したがって，上記問題は NP に属する．

次に，CSS 規則充足不能性問題が $\mathrm{P}^{\mathrm{NP}}$ に属することを示す．$A$ を上記問題を解くチューリング機械とする．$A$ をオラクルとして用いて，CSS 規則充足不能性問題を解くためのアルゴリズムを次のように構成する．

（1）　オラクル $A$ を用いて，上記条件 (i) と (ii) を満たす経路が存在するか否かを決定する

（2）　(1) の結果が “yes” ならば，「$r$ は充足可能」と出力する．そうでなければ「$r$ は充足不能」と出力する．

上記アルゴリズムは明らかに多項式時間で動作する．よって定理が成り立つ．　□

#### 3.2.2　一般の DTD の場合

以下，制限のない一般の DTD を仮定する．ここで，定理 5 の証明において，CSS 規則 $r$ が結合子として ‘>’ と ‘+’ のみを用いる場合を考える．この場合，定理 5 の証明における条

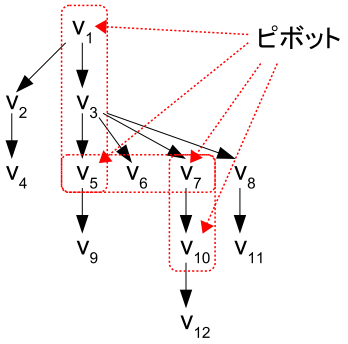


図 4　4 つのピボット $v_1, v_5, v_7, v_{10}$ をもつ $v_1$ から $v_{10}$ への経路

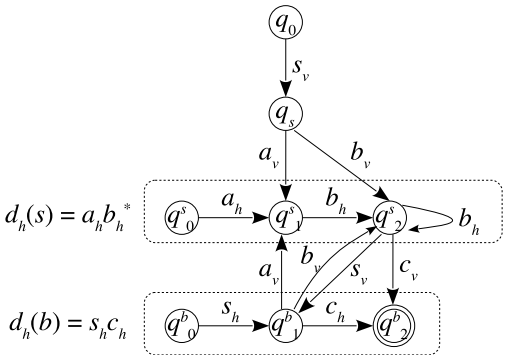


図 5　DTD オートマトンの例

件 (i) と (ii) を満たす任意の経路 $path$ に対して，$path$ の長さは $sel(r)$ に含まれる単純セレクタの数と一致する．このことから，与えられた「正解」の経路に対して，その経路が条件 (i) と (ii) を満たしていることは多項式時間で確かめることができる．よって以下の定理が成り立つ．

［定理 6］ $\mathrm{CSS}^{\{>,+\}}$ に対する CSS 規則充足不能性問題は，一般の DTD の下で $\mathrm{P}^{\mathrm{NP}}$ に属する．

次に，一般の CSS について考える．計算複雑さの上限を示すために，DTD の構造を表す DTD オートマトンを構成する．例えば，図 5 は DTD $D = (d, s)$ 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．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 $\Sigma$ 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，$\Sigma_v$ と $\Sigma_h$ 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 $\epsilon$ なし NFA，$r$ 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 NFA$M$ として定義する．

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

ここで，$Q$，$\delta$，$F$ は次のように定義される．

- まず，$Q$ は次のように定義される．

$$Q = \bigcup_{a \in \Sigma} Q_a \cup \{q_0, q_s\}$$

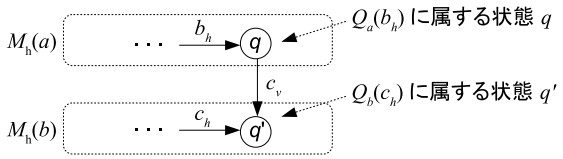


図 6　$Q_a(b_h)$ に属する状態から $Q_b(c_h)$ に属する状態への遷移

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．

- $\delta$ は，各ラベル $a$ の $\delta_a$（$M_h(a)$ の遷移関数）と $\delta_v$ をマージすることにより得られる．$\delta_a$ は横方向の状態遷移を表し，$\delta_v$ は縦方向の状態遷移を表す．ここで，$\delta_v$ 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，

$$Q_a(b_h) = \{q \in Q_a \mid q \in \delta_a(q', b_h), q' \in Q_a\}$$

直感的には，$q \in Q_a(b_h)$ 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 6)）．すなわち，

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

$\delta$ は，$\delta_a$ と $\delta_v$ 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．$r = s_1 c_1 s_1 \cdots c_{n-1} s_n\ p : v$ 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，

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

例えば，図 5 において，もし $r = a \sqcup c\ p : v$ ならば，$F = \{q_2^b\}$ である．

以下の定理が成り立つ．

［定理 7］ $CSS^{\{\sqcup, >, +, \sim\}}$ に対する CSS 規則充足不能性問題は，一般の DTD の下で PSPACE に属する．

証明:　$sel = s_1 c_1 s_2 \cdots c_{n-1} s_n$ 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，$0 \leqq i \leqq n$ に対して

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

かつ，$1 \leqq i \leqq n$ 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．

DTD オートマトンとセレクタの正規表現を用いると，$r \in R$ が DTD $D$ の下で $R$ に関して充足不能か否かは以下のようにして決定できる．

（1） $D$ の DTD オートマトン $M$ を構成する

（2） $sel(r)$ を正規表現 $re(sel(r))$ に変換する

（3） $r_1, r_2, \cdots, r_k \in R$ を，$r$ と衝突し得る CSS 規則とする．すなわち，任意の $1 \leqq i \leqq k$ に対して，$tail(sel(r_i)) = tail(sel(r))$，$prop(r_i) = prop(r)$，かつ，(a) $spec(sel(r_i)) > spec(sel(r))$ または (b) $spec(sel(r_i)) = spec(sel(r))$ かつ $index_R(r_i) > index_R(r)$ が成り立つ．$1 \leqq i \leqq k$ に対して，$sel(r_i)$ を正規表現 $re(sel(r_i))$ に変換する

（4） 以下が成り立つか否かを判定する

$$L(re(sel(r))) \cap L(M) \subseteqq \bigcup_{1 \leqq i \leqq k} L(re(sel(r_i))) \cap L(M).$$

明らかに，上記の式が成立するときかつそのときのみ，$r$ は充足不能である．

上記 (4) において，正規表現の包含判定問題は PSPACE 完全である [16]．よって，上記手続きは PSPACE で実行できる． □

## 4. 多項式時間可解となるための条件

本章では，CSS 規則充足不能性問題が多項式時間可解となる条件について考える．前章の結果から，CSS の結合子が 1 種のみ使用可能，かつ，DTD が閉包なし，重複なし，かつ，再帰なしであるとしてもこの問題は計算困難である．よって，結合子や DTD の内容モデルに制限を加えるのではなく，別の観点から条件を考える必要がある．

### 4.1 自明な条件

まず，CSS 規則充足不能性問題が多項式時間可解となる単純な条件を示す．$R$ を CSS スクリプト，$r \in R$ を CSS 規則とする．$R$ における $r$ と衝突し得る規則の集合を $C_R(r)$ と表す．すなわち，

$$C_R(r) = \{r' \in R \mid tail(sel(r)) = tail(sel(r')),\ prop(r) = prop(r'),\ spec(sel(r')) > spec(sel(r)) \text{ または } (spec(sel(r')) = spec(sel(r)) \text{ かつ } index_R(r') > index_R(r))\}.$$

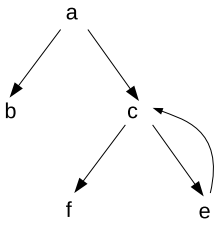


図 7 DTD グラフの例

次の定理が成り立つ．

［定理 8］ $D$ を DTD，$R$ を CSS スクリプト，$r \in R$ を CSS 規則とする．もし次の条件 R1 が成り立つならば，$r$ が $D$ の下で $R$ に関して充足不能であるか否かは多項式時間で決定できる．

R1: $C_R(r) = \emptyset$

証明: 定理 7 のように，$r$ のセレクタは正規表現 $re(sel(r))$ で表せる．よって，条件 R1 が成り立つ場合，$r$ が充足不能か否かは，$re(sel(r))$ と $D$ の DTD オートマトンとの積オートマトンを求め，その空性判定を行うことで決定できる． □

### 4.2 セレクタの長さを制限

次に，CSS 規則充足不能性問題が多項式時間可解となる別の条件を考える．この条件は，たとえ $C_R(r) \neq \emptyset$ であるとしても，互いに衝突するセレクタの長さが 2 以下であれば，この問題が多項式時間可解になるというものである．

提案アルゴルズムは，DTD グラフを用いて CSS 規則の単純セレクタ間の到達可能性を判定する．$D = (d, s)$ を $\Sigma$ 上の DTD とする．$D$ の DTD グラフ $G(D)$ を有向グラフ $(V, E)$ と定義する．ここで，$V = \Sigma$ かつ $E = \{l \to l' \mid l'$ は $d(l)$ に出現するラベル $\}$ である．例えば，$D = (d, a)$ を $d(a) = bc^+$，$d(c) = fe$，$d(e) = c$，$d(b) = d(f) = \epsilon$ なる DTD とする．このとき，$D$ の DTD グラフは図 7 のようになる．

CSS 規則 $r$ に対して，提案アルゴリズムは $G(D)$ をルート $s$ から探索し，ルートから $head(sel(r))$ を経由して $tail(sel(r))$ に到達する経路で，$r$ と競合し得るどの規則にもマッチしないものを求める．前章の結果から，一般には，そのような経路を効率よく求めることは困難である．しかし，後に示すように，次の条件 R2 の下では，そのような経路を効率よく求めることができる．

R2: $len(sel(r)) \leqq 2$，かつ，任意の $r' \in C_R(r)$ に対して $len(sel(r')) = 2$

条件 R2 を仮定する．$C_R(r)$ に属する CSS 規則のうち，結合子が $c$ であるものからなる集合を $C^c_R(r)$ と表す．すなわち，

$$C^c_R(r) = \{r' \in C_R(r) \mid cmb(sel(r')) = c\}$$

ここで，$c \in \{\text{␣}, >, \sim, +\}$，かつ，$cmb(sel(r'))$ は $sel(r')$ の結合子を表す（$len(sel(r')) = 2$ なので，$sel(r')$ は 1 つの結合子

をもつ）．また，次の略記を用いる．

$$head(C^{c}_{R}(r)) = \{head(sel(r')) \mid r' \in C^{c}_{R}(r)\}$$

例えば，$R = [r_1, r_2, r_3, r_4]$，ここで $r_1 = a\_b\ p : v_1$, $r_2 = b\ p : v_2$, $r_3 = c\_b\ p : v_3$，かつ，$r_4 = d\_b\ p : v_4$ とする．このとき，$C_R(r_1) = \{r_3, r_4\}$ かつ $head(C^{\sqsubset}_{R}(r_1)) = \{c, d\}$ である．

提案アルゴルズムを図 8 に示す．紙面の都合上，$cmb(sel(r)) =$ '␣' の場合のみ示した．簡単のため，条件 R2 において $len(sel(r)) = 2$ かつ $sel(r)$ の単純セレクタはラベルであると仮定する．アルゴリズムのステップ (2) において，$Skip$ を $head(C^{\sqsubset}_{R}(r))$ に設定する．これは，$G(D)$ を探索する際にスキップすべきラベルの集合を表す．ステップ (3) から (5) において，$G(D)$ を探索してルートから $head(sel(r))$ を経由して $tail(sel(r))$ に到達可能な経路で，$r$ と衝突する CSS 規則にマッチしない経路 $p$ を求めている．

入力：DTD $D = (d, s)$, 条件 R2 を満たす CSS スクリプト $R$ および CSS 規則 $r \in R$
出力：「充足可能」または「充足不能」
（1） $D$ の DTD グラフ $G(D)$ を構成する
（2） $Skip \leftarrow head(C^{\sqsubset}_{R}(r))$
（3） $G(D)$ をルート $s$ から $head(sel(r))$ まで，$Skip$ に属するどの頂点も用いずに探索する．もし $head(sel(r))$ が $s$ から到達可能でないならば，「充足不能」と出力する
（4） $G(D)$ を $head(sel(r))$ から $tail(sel(r))$ まで，$Skip$ に属するどの頂点ドも用いずに探索する．もし $tail(sel(r))$ が $head(sel(r))$ から到達可能でないならば，「充足不能」と出力する
（5） もし $G(D)$ において，次の 3 条件を満たす $tail(sel(r))$ の親 $e$ が存在しないならば，「充足不能」と出力する
（a） $e$ は上記 (4) で探索された頂点である
（b） $e \notin head(C^{>}_{R}(r))$ である
（c） $L(d(e))$ が次の条件を満たす文字列 $str$ を含む
- $str$ は $tail(sel(r))$ を含む
- $head(C^{+}_{R}(r))$ に属するどのラベルも，$str$ において $tail(sel(r))$ の隣接する兄として出現することはない
- $head(C^{\sim}_{R}(r))$ に属するどのラベルも，$str$ において $tail(sel(r))$ の（必ずしも隣接しない）兄として出現することはない

（6） 「充足可能」と出力する

図 8　アルゴリズム CheckUnsatisfiability

次の定理が成り立つ．紙面の都合で証明は省略する．

［定理 9］ $D$ を DTD，$R$ を CSS スクリプト，$r \in R$ を CSS 規則とし，条件 R2 が成り立つと仮定する．このとき，$r$ が $D$ の下で $R$ に関して充足不能であるときかつそのときのみ，CheckUnsatisfiability は「充足不能」と出力する．

また，CheckUnsatisfiability の時間計算量は $O(|D|+|R|)$ である．ここで，$|D| = \sum_{a \in \Sigma} |d(a)|$ であり，$|d(a)|$ は正規表現 $d(a)$ のサイズを表す．

## 5. むすび

本稿では，DTD の下で充足不能な CSS 規則を発見する問題について考察し，計算困難となる条件，および，多項式時間可解となるための条件を示した．

今後の課題として，まず，本稿で扱っていない 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，この問題が多項式時間可解となる条件として R1 と R2 を示したが，より実用的な条件を求めていきたいと考えている．さらに，現実の CSS スクリプトを調査し，多項式時間可解となる条件がどの程度適用可能であるかを調べていきたいと考えている．

## 文　献


[1] N. Walsh, “DocBookCssStylesheets.” https://github.com/docbook/wiki/wiki/DocBookCssStylesheets/.
[2] 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/.
[3] 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.
[4] M. Bosch, P. Genevès, and N. Layaïida, “Automated refactoring for size reduction of CSS style sheets,” Proceedings of the 2014 ACM Symposium on Document Engineering, pp.13–16, 2014.
[5] 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.
[6] D. Mazinanian and 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.
[7] Firebug Working Group, “FireBug.” https://www.getfirebug.com/.
[8] Google Inc., “Chrome developer tools.” https://developers.google.com/web/tools/chrome-devtools/.
[9] 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.
[10] 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.
[11] M. Benedikt, W. Fan, and F. Geerts, “XPath satisfiability in the presence of DTDs,” J. ACM, vol.55, no.2, pp.8:1–8:79, May 2008.
[12] M. Montazerian, P.T. Wood, and S.R. Mousavi, “XPath query satisfiability is in ptime for real-world DTDs,” Proceedings of the 5th International Conference on Database and XML Technologies, pp.17–30, 2007.
[13] N. Suzuki, Y. Fukushima, and K. Ikeda, “Satisfiability of simple XPath fragments under duplicate-free DTDs,” IEICE Transactions, vol.96-D, no.5, pp.1029–1042, 2013.
[14] Y. Ishihara, N. Suzuki, K. Hashimoto, S. Shimizu, and T. Fujiwara, “XPath satisfiability with parent axes or qualifiers is tractable under many of real-world DTDs,” Proceedings of the 14th International Symposium on Database Programming Languages, 2013.
[15] M. Garey and D. Johnson, Computers and Intractability - A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[16] L.J. Stockmeyer and A.R. Meyer, “Word problems requiring exponential time(preliminary report),” Proceedings of the Fifth Annual ACM Symposium on Theory of Computing, pp.1–9, 1973.