コンテンツ一貫性制約を用いたWebサイト管理手法の提案

澤　菜津美† 　森嶋　厚行†† 　飯田　敏成† 　杉本　重雄†† 　北川　博之†††

† 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
†† 筑波大学大学院 図書館情報メディア研究科/知的コミュニティ基盤研究センター
〒 305–8550 茨城県つくば市春日 1–2
††† 筑波大学大学院 システム情報工学研究科 〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †{sawa,mori,toshi,sugimoto}@slis.tsukuba.ac.jp, ††kitagawa@cs.tsukuba.ac.jp

あらまし　現在，Web サイトの構築手法として，大きく分けて，各ページのコンテンツの直接作成と， ページとは別の情報源 (DB 等) からオンデマンドで動的にページを作成するシステムを構築，の二つがある．前者で構築された Web サイトは，サイトの規模が大きくなるにつれ，コンテンツの一貫性維持が非常に困難になってくる．一方，後者で構築された Web サイトは，バックエンドの DB 等を使用して容易にコンテンツの一貫性維持が行えるが，構築するための初期コストが大きい．本稿では，バックエンドに DB 等を利用せずに Web サイトのコンテンツ一貫性維持を容易に行うための仕組みとして，Web ページ間のコンテンツ一貫性制約を用いた Web サイト管理手法を提案する．
キーワード　Web サイト管理，コンテンツ一貫性

Proposal of A Web-Site Management Method Using Content Integrity Constraints

Natsumi SAWA†, Atsuyuki MORISHIMA††, Toshinari IIDA†, Shigeo SUGIMOTO††, and Hiroyuki KITAGAWA†††

† Grad. Sch. of Library, Information and Media Studies, Univ. of Tsukuba,　1–2 Kasuga, Tsukuba, Ibaraki, 305–8550 Japan
†† Grad. Sch. of Library, Information and Media Studies/ Research Center for Knowledge Communities, Univ. of Tsukuba.　1–2 Kasuga, Tsukuba, Ibaraki, 305–8550 Japan
††† Grad. Sch. of Sys. and Info. Eng., Univ. of Tsukuba. 1–1–1 Tennohdai, Tsukuba, Japan
E-mail: †{sawa,mori,toshi,sugimoto}@slis.tsukuba.ac.jp, ††kitagawa@cs.tsukuba.ac.jp

Abstract　Methods for the Web-site construction are classified into the following two categories: (1) to create directly each Web page, and (2) to develop a system that derives Web-pages from other information sources such as databases. The Web sites constructed by the first-category methods have difficulties in maintaining their content integrities, especially when the sizes of the Web sites are large. As for the Web sites built by the second-category methods, the maintenance of its content integrity is easy because the updates of background information sources are automatically reflected by the derived Web pages. But the initial costs of Web site construction by the second category methods tend to be large. This paper proposes a novel Web-site management method that introduces content integrity constraints on Web pages, which allows us to maintain the content integrities of Web contents in an easy way, without requiring backend information sources.
Key words　Web-site Management, Content Integrity

1. はじめに

現在，Web サイトを通じた情報発信が広く普及している．これら Web サイトを構築する方法には，大きく分けて次の二つの手法がある．

(手法 A) 各ページのコンテンツの直接作成: 例えば，テキストエディタを用いて HTML ドキュメントを直接作成する方法，HTML 作成支援ツール等を用いる方法，Wiki などを通じて作成する方法，などがある．コンテンツの更新は各ページを更新することにより行われる．

(手法 B) ページとは別の情報源からページを作成するシステムを構築: 例えば，バックエンドに DB システムを配置し，DB に格納されているデータから Web ページを作成する方法．コンテンツの更新は DB の更新により行われる．

一般に，Web サイトに含まれるコンテンツは互いに関連していることが多い．例えば，大学の研究室の Web サイトでは，各人のページに研究室の名前，住所，電話番号が含まれており，これらは一致するはずである．また，研究室メンバの発表論文の一覧は，研究室の発表論文一覧のサブセットであることが一般的である．このような関連を表す制約を，本論文では**コンテンツ一貫性制約**と呼ぶ．Web サイトのコンテンツの変更があった場合には，これらのコンテンツ一貫性制約が保持されるように更新されることが望ましい．しかし，Web サイトの規模が大きくなるにつれ，手法 A ではコンテンツ一貫性の維持が困難になる．したがって，ある程度大規模な Web サイトは手法 B で構築される．

問題は，(1) ページ数の少ない Web サイトでは手法 A の方が構築コストが小さいこと，および，(2) サイトの規模が大きくなってしまうと，手法 A で構築されたサイトから手法 B への移行コストが大きいこと，である．したがって，あらかじめ大規模サイトであることが分かっており，かつ，構築のためのリソースがあり，かつ，コンテンツ一貫性が維持されないことが致命的な問題になる (ビジネス関係など) 場合のみ，最初から手法 B で構築される事が多い．それ以外の場合では，手法 B の採用を決断するケースは少なくなってしまう．

我々は現在手法 A で構築された Web サイトが実際どれくらい存在し，またその規模についても調べるため，tsukuba.ac.jp 内の Web サイトの調査を行った (図 1).

ここでは，クローラを用いて収集した tsukuba.ac.jp 内のサイトから無作為に選んだ 300 個の Web サイトを対象とした．その内，リンク切れである Web サイトが 4 個あったが，それらについては結果から除外した．まず，動的な Web ページ生成を行っている可能性のあるサイトをカウントした．具体的には，ファイルの拡張子が php 等のサイトの数をカウントした．これらは，手法 B によって構築されている Web サイトである可能性がある．その結果，18%のサイトが何らかの動的なページ生成を行っていることが分かった．これら以外のサイトは，手法 A で作成されている可能性が高いが，他の情報源から作られている可能性もある．そこで，手法 B で作成されていそうなサイトをチェックし，それらは手法 B に分類した．残りの Web サイトについて，ページ数を数えたところ，34%が 100 ページ以上の Web ページを持っていた．以上の結果から，ある程度多くの Web ページを持つ Web サイトであっても，手法 A で構築されている Web サイトが多いことが強く推測される．このような状況である原因はいくつか考えられる．例えば，(1) 手法 B の Web サイトを構築するためのリソースが存在しない，(2) サイトの内容が非定型であり，手法 B に適さない，(3) 最初は少ないページであったので手法 A で構築していたが，いつの間にか規模が大きくなった，等である．手法 A で作成された Web サイトのコンテンツの一貫性を保持するためには，各ページの

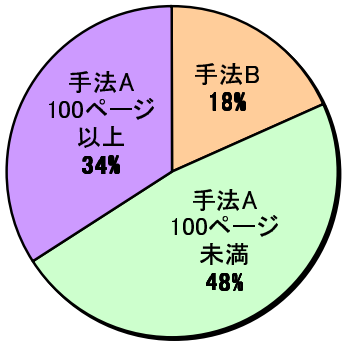

図 1　tsukuba.ac.jp 内の Web サイト調査結果

入念なチェックとページ毎の更新が必要であるが，多数の Web ページについて行おうとすると，非常に労力がかかることは明白である．

本論文では，手法 A で作成された Web サイトのコンテンツ一貫性維持を低コストで行うための仕組みについて提案する．鍵は，コンテンツ一貫性制約の明示的な記述と保持である．コンテンツ一貫性制約を明示的に保持しておけば，コンテンツ一貫性制約が満たされているか否かを監視し，制約が破られたときには報告もしくは自動修正などを行う仕組みが容易になる．特に本論文では，Web ページコンテンツ一貫性制約として木包含従属性を定義し，それを自動発見するための仕組みについて提案する．これらの技術により，低コストでの Web サイト構築と Web サイト管理の両立を可能とする新しいアプローチを実現できるのではないかと期待している．

本稿の構成は次の通りである．2 章では，コンテンツ一貫性制約を用いた Web サイト管理手法の提案を行う．3 章では，我々が考えるコンテンツ一貫性制約の一種として木包含従属性を定義し，説明する．4 章では，木包含従属性を発見するアルゴリズムについて説明する．5 章では，簡単な予備実験について述べる．6 章では，関連研究について述べる．7 章はまとめと今後の課題である．

2. コンテンツ一貫性制約を用いた Web サイト管理手法

本章では，コンテンツ一貫性制約を用いた Web サイト管理手法について述べる．図 2 は，明示的なコンテンツ一貫性制約を用いた Web サイトのコンテンツ一貫性管理 (図 2(a)) と，手法 B である Web-DB 連携システムを利用した場合 (図 2(b)) を比較したものである．コンテンツ一貫性制約は，Web のコンテンツで成立する制約を**直接的に**表現する．例えば，あるページのコンテンツの一部と他ページの一部が一致する，という条件を明示的に表現する．一方，手法 B では，制約は DB 問合せを通じて**間接的に**表現されていることに注意してほしい．したがって，データを管理するための情報源 (この場合は DBMS) を用意する必要がある．

コンテンツ一貫性制約としては，次のようなものが考えられる．(1) コンテンツの一致や包含関係 (2) あるコンテンツの値が他のコンテンツから計算できる．(3) 過去 1 週間のコンテンツ

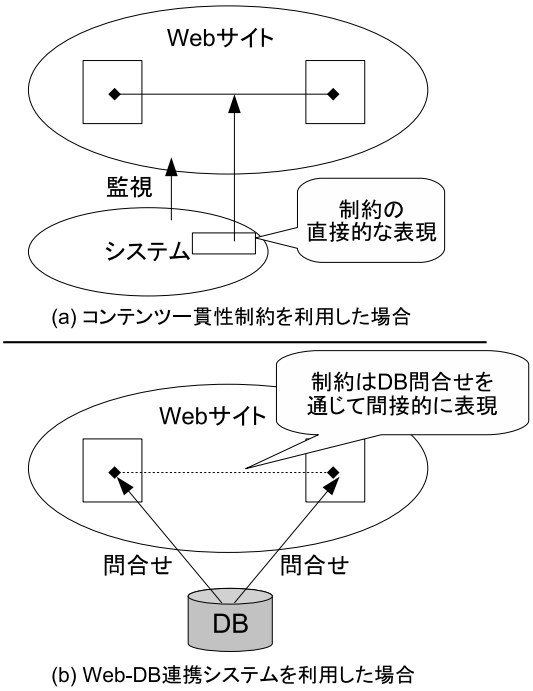

図 2　コンテンツ一貫性管理方法の比較

だけが表示される．(4) ある場所に含まれていないコンテンツのみが表示される．

図 3 はコンテンツ一貫性制約を用いた Web サイト管理の仕組みを表したものである．以下にその手順を述べる．まず，利用者がコンテンツ一貫性制約を登録する (図 3 (1))．登録の際には利用者が直接コンテンツ一貫性制約を作成しても良いが，既存のコンテンツからコンテンツ一貫性制約の候補を自動発見させて，適切と考えられるものを一部採用しても良い．4 章では，我々が考えるコンテンツ一貫性制約の一種である木包含従属性を自動発見させるアルゴリズムを提案する．制約が登録されると，システムは定期的もしくは Web サイトの更新が行われた際などに Web サイトのチェックを行い，先に発見しておいた制約と照らし合わせて，制約が破られていないかどうか調べる (図 3 (2))．その際，もし制約違反を発見したら，Web サイト管理者に報告もしくは自動修正を行う (図 3 (3))．

手法 B で構築する Web-DB 連携システム等の場合，DB のデータから Web ページを動的に生成するための変換プログラムなどを Web サイト毎に個別に実装する必要があるためそのコストが大きくなるが，本システムの利用では Web サイト毎に個別にシステムを実装する必要はないため，大きなコストがかからないと予想される．

以上のような Web サイト管理手法を用いる利点は，手法 A による Web サイト構築手法と一貫性管理を両立できることである．したがって，下記のような利用方法が考えられる．(1) 既存の手法 A で作成された Web サイトに本システムを適用し，サイトを再構築することなくコンテンツの一貫性管理を実現する．(2) 手法 B での構築に適さない非定型サイトなどの一貫性管理を行う手法として，手法 A と本提案手法の組合せを新たな Web サイトの構築の際に採用する．(3) コンテンツが分散管理

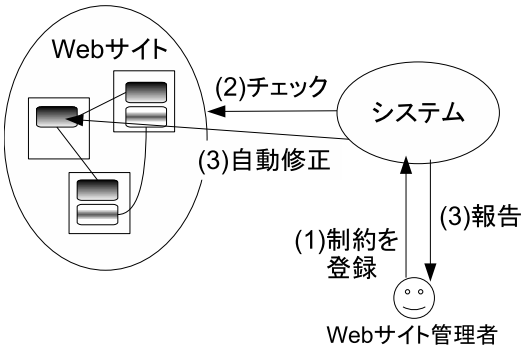

図 3　コンテンツ一貫性制約を用いた Web サイト管理手法

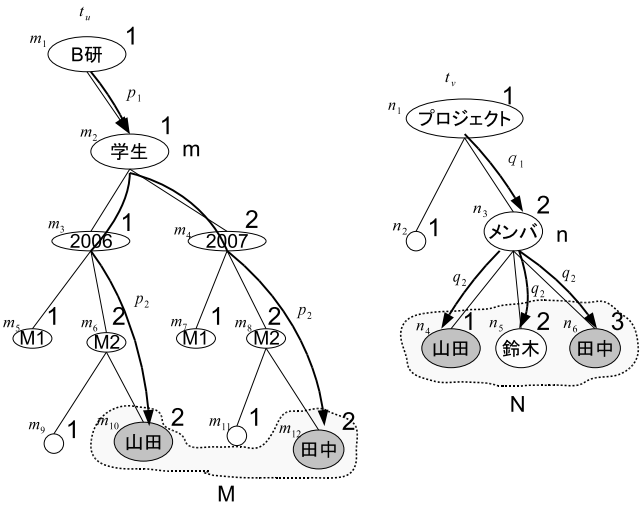

図 4　木包含従属性が成立する例 (左: t_u, 右: t_v)

されており，物理的に手法 B での Web サイトの再構築が困難である場合にコンテンツの一貫性管理を実現する．

以下では，先に述べたようなコンテンツ一貫性制約の一種である木包含従属性を定義し，既存の Web サイトのコンテンツで成立する木包含従属性を発見するためのアルゴリズムについて提案する．利用者は発見されたものから適切なものだけを選択し，利用すればよい．

3. 木包含従属性

本章では，我々が考えるコンテンツ一貫性制約の一種である**木包含従属性** (tree inclusion dependencies) について説明する．木包含従属性を用いれば，1 章の発表論文の例であげたように，ある Web ページに含まれているコンテンツの一部が，他の Web ページに含まれているコンテンツの一部に対して，サブセットになっているような関係を表す事ができる．下記では，HTML ページを木構造としてモデル化する．

図 4 は，URL u と v で表される Web ページ (の木構造) 間に，木包含従属性 $u[/1/1,/*/2/2] \subseteq v[/1/2,/*]$ が成立する例である．この図では，ノードの左肩にノード id($m_1, m_2, ...$), ノードの右肩に兄弟番号 (1,2,3) が振ってある．また，ノード内には，そのノードに含まれるテキスト (学生，山田等) を記述してある．$u[/1/1,/*/2/2] \subseteq v[/1/2,/*]$ は，URL u の Web ページコンテンツを表す木構造 t_u において，兄弟番号で表されるパス

/1/1 と /∗/2/2 を連結したパス /1/1/∗/2/2 で求められるノード集合 M と, 同様に URL v の Web ページコンテンツを表す木構造 t_v において，/1/2 と /∗ を連結したパス /1/2/∗ で求められるノード集合 N の間に $M \overset{\text{text}}{\subseteqq} N$ の関係が成立することを表す．パス中のアスタリスク (∗) は任意のノードを表す．ここで，$M \overset{\text{text}}{\subseteqq} N$ とは，

$$\{text(m_i)|m_i \in M\} \subseteqq \{text(n_i)|n_i \in N\}$$

の事である．$text(m_i)$ は，ノード m_i が持つテキストを表す．図 4 では，例えば，$text(m_{10})$ は”山田”，$text(m_{12})$ は”田中”となる．したがって，この場合，$M \overset{\text{text}}{\subseteqq} N$ は {”山田”,”田中”}$\subseteqq$ {”山田”,”鈴木”,”田中”} と同値である．

一般には，木包含従属性 $u[p_1, p_2] \subseteqq v[q_1, q_2]$ は下記のように定義される．M_u を t_u のノード集合，N_v を t_v のノード集合，p_i, q_j をそれぞれパス式とする．このとき，木包含従属性 $u[p_1, p_2] \subseteqq v[q_1, q_2]$ が成立するとは，下記が成立することと同値である．

$$\begin{aligned}&\exists m, n, M, N\\&(\{m\} = eval(u, p_1) \wedge \{n\} = eval(v, q_1) \wedge\\&M = \{m_i | m_i \in eval(u, p_1.p_2)\} \wedge N = \{n_j | n_j \in eval(v, q_1.q_2)\}\\&\qquad \wedge m = lca(M) \wedge n = lca(N) \wedge M \overset{\text{text}}{\subseteqq} N)\end{aligned}$$

ここで，$p_1.p_2$ は二つのパス式 p_1, p_2 の連結を表す．$eval(u, p)$ は，URL u のページに対してパス式 p を適用した結果求められるノード集合である．$lca(M)$ は，M に含まれるノードの least common ancestor を求める式である．図 4 では，m は m_2, p_1 は /1/1, p_2 は /∗/2/2, M は $\{m_{10}, m_{12}\}$ になる．また，n は n_3，q_1 は /1/2, q_2 は /∗, N は $\{n_4, n_5, n_6\}$ になる．定義に当てはめると，

$$\begin{aligned}&\{m_2\} = eval(u, /1/1) \wedge \{n_3\} = eval(v, /1/2) \wedge\\&M = \{m_i | m_i \in eval(u, /1/1/*/2/2)\} = \{m_{10}, m_{12}\}\\&\quad \wedge N = \{n_j | n_j \in eval(v, /1/2/*)\} = \{n_4, n_5, n_6\}\\&\quad \wedge m_2 = lca(\{m_{10}, m_{12}\}) \wedge n_3 = lca(\{n_4, n_5, n_6\}) \wedge M \overset{\text{text}}{\subseteqq} N\end{aligned}$$

となる．

木包含従属性以外のコンテンツ一貫性制約の例として以下のようなものが考えられる．**(1) コンテンツの表示期間の指定．**例えば，最新のニュース一覧を表示する場合に，各ニュースの表示期間を一週間と指定する．したがって，古いニュースから自動的に消えていくことになる．**(2) コンテンツの表示件数の指定．**例えば，研究室メンバの論文リストを表示する場合に，表示する件数を指定する．

4. 木包含従属性を発見するアルゴリズム

本章では，2 つの木が与えられた時，3 章で定義した木包含従属性を発見するアルゴリズムについて述べる．実際には，Web ページコンテンツを表す木構造を作成する問題があるが，ここでは，既に木は与えられたものとする．提案アルゴリズムの概要は次の通りである．まず，2 つの木構造を比較して，共通して現れるテキストを含むようなそれぞれの部分領域を求める．次に，これを手がかりとして，木包含従属性を発見する．具体的には次の 2 つのフェーズから構成される．

[フェーズ 1] 2 つの木構造 t_u，t_v において共通して現れるテキスト集合を含むようなそれぞれの部分領域を求める．具体的な手順は次のようになる．(1)t_u，t_v をそれぞれ，サブツリーに分割し，集合 T_u, T_v を求める．ここで，T_u とは，　$T_u = \{t'_u | t'_u$は t_uの葉以外をルートとするようなサブツリー $\}$ の事である．T_v も同様とする．(2) これら全ての組み合わせ $(t'_u, t'_v) \in T_u \times T_v$ について，次を行う．すなわち，t'_u, t'_v を高さのレベル (深さ) ごとに分割し，それぞれの集合を比較した際に，どちらの集合にも共通して現れるテキストの集合を求める．4.1 節で (2) の詳細は説明する．

[フェーズ 2] フェーズ 1 で求めた共通テキスト集合を手がかりにして，木構造 t_u，t_v の木包含従属性を発見する．詳細は 4.2 節で説明する．

4.1 フェーズ 1: 木構造の共通テキスト集合を求めるアルゴリズム

木構造の共通テキスト集合を求めるアルゴリズムは図 5 である．入力は T_u，T_v である．2 つの木構造 $(t'_u, t'_v) \in T_u \times T_v$ に対して，t'_u のノード集合を M_u とし，M_u^i は t'_u の深さ i に存在するノードの集合 (例えば，M_u^0 はルートノード 1 つからなる集合) であるとする．この時，M_u は下記のように M_u^i の直和集合として表される (N_v も同様)．

$$M_u = M_u^0 + M_u^1 + \ldots M_u^i$$

フェーズ 1 アルゴリズムの出力は集合 $Candidates$ である．これは三つ組み $(common, M_u^i, N_v^j)$ を要素に持つ．ここでの $common$ は M_u^i と N_v^j で共通して現れるテキストの集合である．6 行目から 14 行目で，M_u^i と N_v^j の全ての組み合わせについて同じテキストを持つノードがないかどうか調べている．9 行目では，$texts(M_u^i)$ と $texts(N_v^j)$ の積集合を求め，$common$ に格納している．ここでの $texts(M_u^i)$ は集合 M_u^i に含まれる全てのノードが持つテキストの集合を返す関数である ($texts(N_v^j)$ も同様)．例えば，図 4 の $N_v^2 = \{n_4, n_5, n_6\}$ であれば，{”山田”, ”鈴木”, ”田中”} が返される．$common$ が空でなかった場合には，$Candidates$ に，$(common, M_u^i, N_v^j)$ を追加する (12 行目)．

4.2 フェーズ 2: 木包含従属性の集合を求めるアルゴリズム

フェーズ 2 では，フェーズ 1 で出力された集合 $Candidates$ を用いて，木包含従属性の集合を求める．図 6 がそのアルゴリズムである．

すべての $Candidates$ の要素 $(common, M_u^i, N_v^j) \in Candidates$ に対して，以下を実行する．まず，M_u^i 内の $common$ に含まれるテキストを持つノード集合 $\pi_{common}(M_u^i)$ の least common ancestors である m を求める (2 行目)．t_u は木なので，m はただ一つに定まる．次に，m から $\pi_{common}(M_u^i)$ までの共通パスを生成するため，関数 $GeneralizedPath$ を呼び出す (3 行目)．

ここで，関数 $GeneralizedPath(m, N)$ について説明する (17 行目から 28 行目)．引数 m はノード，N はノードの集合である．返り値は m から各 $n \in N$ へのパス全ての共通パス p である．ここでいう共通パスとは，例えば，パス 1 が/2/A/apple/ であり，パス 2 が/3/A/banana/ であった場合，2 つのパスの異な

Input: T_u, T_v
Output:
Set of Triples $Candidates$ =
$\{(common_0, M_u^i, N_v^j), (common_1, M_u^i, N_v^j), ...\}$

```
1  Candidates ← φ;
2  for each (t'_u, t'_v) ∈ T_u × T_v {
3      M_u = t'_u.nodes(); // t'_u に含まれるノード集合.
                   ただし，M_u = M_u^0 + M_u^1 + ... M_u^i
4      N_v = t'_v.nodes(); // t'_v に含まれるノード集合.
                   ただし，N_v = N_v^0 + N_v^1 + ... N_v^i
5      // M_u のノード集合 M_u^i，N_v のノード集合 N_v^j の組み合わせ
6      for each M_u^i in M_u {
7          for each N_v^j in N_v {
8              // texts(M_u^i) と texts(N_v^j) の積集合を求める
9              common ← texts(M_u^i) ∩ texts(N_v^j);
10             if (common ≠ φ)
11                 // Candidates との和集合を Candidates に格納
12                 Candidates ← Candidates
                                   ∪{(common, M_u^i, N_v^j)};
13         }
14     }
15 }
```

図 5　フェーズ 1:共通テキスト集合を求めるアルゴリズム

る部分を任意の文字列としてアスタリスク (*) で表したパス式 /*/A/*/の事をいう．

p が求まると，次に 4 行目では，木 t_u から $getNodes$ によってノード集合 M を得る．$getNodes$ のパラメータ $path(m).p$ はルートから m までのパスを表す $path(m)$ と p を連結したパスである．以上の操作を N_v^j についても同様に行う (6 行目から 8 行目)．M と N が得られたら，$M \overset{\text{text}}{\subseteqq} N$ が成立するかどうかを調べ，成立する場合には制約を出力する (10 行目から 12 行目)．$M \overset{\text{text}}{\supseteqq} N$ の場合も同様である (13 行目から 15 行目)．

次に $GeneralizedPath(m, N)$ のアルゴリズムについて説明する．19 行目から始める for 文中では，N に含まれるノード n について，まず，ルートから n までのパスからルートから m までのパスを除去したパスを求める (20 行目)．例えば，ルートから n までのパスが /food/fruit/1/A/orange/ であり，ルートから m までのパスが /food/fruit/であった場合は/1/A/orange/となる．次に，n が複数個ある場合に，上で説明した共通パスを生成する (21 から 25 行目)．24 行目の $generalize$ は，2 つのパスの共通パスを返す関数である．$GeneralizedPath$ は以上を繰り返して，N 中のすべてのパスの共通パス p を返す (27 行目)．

5. 予 備 実 験

4 章で説明した木包含従属性を発見するアルゴリズムを実際の Web ページに適用した場合，どのような結果が出力される

Input:
Set of Triples $Candidates$ =
$\{(common_0, M_u^i, N_v^j), (common_1, M_u^i, N_v^j), ...\}$
Output: 木包含従属性の集合

```
1  for each (common, M_u^i, N_v^j) ∈ Candidates {
2      m ← lca(π_common(M_u^i)); // least common ancestors を求める
3      p ← GeneralizedPath(m, π_common(M_u^i)); // 共通パスを生成
4      M ← t_u.getNodes(path(m).p); // ノード集合 M を得る
5
6      n ← lca(π_common(N_v^j));
7      q ← GeneralizedPath(n, π_common(N_v^j));
8      N ← t_v.getNodes(path(n).q);
9
10     if (M ⊆(text) N) {
11         Output (u[path(m), p]⊆v[path(n), q]) // 従属性の出力
12     }
13     if (M ⊇(text) N) {
14         Output (u[path(m), p]⊇v[path(n), q]) // 従属性の出力
15     }
16 }

17 path GeneralizedPath(node m, Set of nodes N) {
18     path p ← null;
19     for each n ∈ N {
20         p' ← pathFromRoot(n) から
                先頭の pathFromRoot(m) を除去したもの;
21         if (p = null) {
22             p ← p';
23         } else {
24             p ← p.generalize(p'); // 共通パスを生成
25         }
26     }
27     return p;
28 }
```

図 6　フェーズ 2: 木包含従属性の集合を求めるアルゴリズム

のか確認するため，実際の Web ページを用いて簡単な予備実験を行った．

今回の予備実験で使用した Web ページは，ある大学の研究科におけるある分野の紹介ページと同研究科の教員一覧のページである．まず，これらの各ページの URL u, v を入力として木構造を抽出する．今回この木構造は，HTML ドキュメント中の単純な繰り返し構造 (リストを表すタグ ($< ol >, < ul >, < li >$) およびカンマ (,)) に着目して自動生成したものである．抽出した木をそれぞれ t_u, t_v とし，木包含従属性に関連する部分だけを取り出したものを図 7 に示す．図 7 では，ノード内にテキストを表し，ノードの左肩にノード id, ノードの右肩に兄弟番号が振ってある．なお，テキスト中の人物名は変更してある．

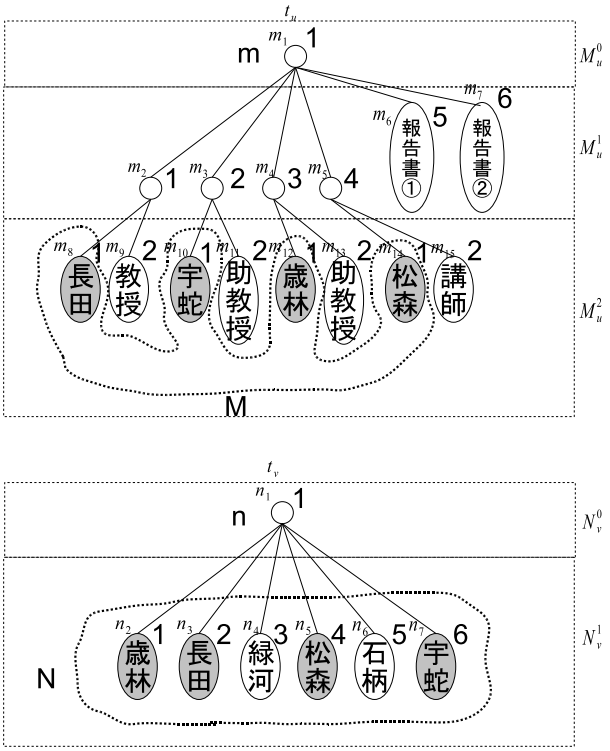

図 7　予備実験で使用した木 (上: t_u, 下: t_v)

この木 t_u, t_v について，フェーズ 1 の木構造の共通部分を求めるアルゴリズムを実行する. 実際は全ての $(t'_u, t'_v) \in T_u \times T_v$ について処理が行われるが，ここでは簡単化のため $t'_u = t_u, t'_v = t_v$ の場合についてのみ説明する．t_u, t_v の共通部分 $common$ は，{”長田”,”宇蛇”,”歳林”,”松森”}(図中では色つきのノード) となるので，候補 $Candidates$ は，{({”長田”,”宇蛇”,”歳林”,”松森”}, M_u^2, N_v^1)} である．ここで，M_u^2 は t_u の深さ 2 のノード集合，N_v^1 は t_v の深さ 1 のノード集合を表す.

続いて，フェーズ 2 の木包含従属性を満たす制約の集合を求めるアルゴリズムを実行する．t_u, t_v の least common ancestors は，図 7 の木では，それぞれノード m_1, n_1 になる．このとき，ルートから m_1 までのパス p_1, およびルートから n_1 までのパス q_1 は, この場合は両方とも /1 となる．また，m_1 から $common$ までの共通パス p_2 は $/*/1$, n_1 から $common$ までの共通パス q_2 は $/*$ となる．ルートから least common ancestors までのパスおよび共通パスを連結して得られるノード集合は，それぞれ，$M = \{m_8, m_{10}, m_{12}, m_{14}\}$, $N = \{n_2, n_3, n_4, n_5, n_6, n_7\}$ である．この M, N について，図 7 を見ると，$M \overset{\text{text}}{\subseteq} N$ が成り立つ事がわかる．すなわち，木包含従属性 $u[/1, /*/1] \subseteq v[/1, /*]$ が成り立つので，この制約が出力される事になる.

同様にして，他のページについても実験を行った．u: ある図書館のこれまでのお知らせの一覧 と v: 全館のこれまでのお知らせの一覧について実行した．発見された制約には，u が v のサブセットであるという関連を表す制約が含まれていた.

今回の予備実験では，ごく簡単な例ではあるが，実際の Web ページから制約を発見できる事を確認した．注意したい点は，提案アルゴリズムを効果的に適用させるには，木構造の抽出を適切に行う必要があるという事である．今回の実験では，単にリストタグおよびカンマを元に木構造の抽出を行うだけですんだが，実際の HTML ドキュメントの構造は多種多様であるため，どのような HTML ドキュメントについても適切に木構造を抽出するためには，ある程度工夫が必要と思われる．また，表記のゆれなどにも対応する必要がある.

6. 関 連 研 究

第 1 章で述べたように，Web サイトを構築する方法は,「手法 A: 各ページのコンテンツの直接作成」と「手法 B: ページとは別の情報源からページを作成するシステムを構築」に大別される.

手法 A としては，テキストエディタを用いて HTML ドキュメントを直接作成する方法以外に，例えば，Dreamweaver [1] 等の Web サイト製作ツールや，Wiki などのツールがある．これらのツールは，凝った Web デザインを施した Web ページを作成したり，HTML に関する知識が無くとも気軽に Web ページを作成できるものとして普及している．これに類似したツールも数多く存在する．これらのツールのおかげで，誰もが気軽に Web ページを作成できるようになった．しかし，Web サイトのコンテンツの管理の面から見ると，これらのツールは必ずしも充分な機能を持つとは言い難い．Web ページのコンテンツの管理は，今もなお，人手によるチェックが主流である．Web ページの数が増えるほど，こうした Web ページのコンテンツ管理の労力が増えることは容易に想像できる.

手法 B を用いれば，このようなコンテンツ管理の労力を軽減できる．例えば，AT&T 研究所で開発されたシステム Strudel [2] は，Web サイト構築の際に必要な，バックエンドの DB の管理，Web サイトの構造の管理，HTML ページの視覚的表現の管理を分離することができるので，DB の更新を行えば，全ての Web ページが適切に更新される．しかしながら，第 1 章でも述べたように，手法 B の場合，Web サイトを構築するためのコストが非常に大きい事が問題である．それゆえ，手法 B の採用を決断する分岐点は非常に高くなってしまっている．それに対し，提案手法では，低コストでの Web サイト構築と Web サイト管理の両立を目指すものである.

また，最近ではコンテンツ管理システム (CMS) の利用も広がっている．例えば，Xoops [3] や Zope [4] 等があるが，これらのツールを使用すれば手法 B での Web サイトの構築コストはこれらのツールを使わない場合に比べて低くなると思われる．それに対する本提案手法の利点の一つは，既に Web コンテンツが存在する場合に，それらに対して適用することができるので，新たにコンテンツ作成をする必要は無いことである．また，別の利点としては，異なる技術で作成され，分散管理されている複数のサイト間でのコンテンツ一貫性管理も可能にすることである．例えば，手法 A で作成された Web サイトと CMS で作成された Web サイト間のコンテンツ一貫性管理といった事が可能になる.

この他にも Web サイト管理に関する研究は多く存在する．例えば，Web コンテンツを情報部品の集合として捉え，情報部品

間のロジックを定義し，Web サイトの自動生成や管理を行う試みがある [5]．また，Web リンクの管理に着目した研究が挙げられる．例えば，リンク切れではないが，内容的には間違っているリンクを検出する試みが存在する [6]．我々のグループでも，Web ページの移動によって生じたリンク切れを対象として，Web ページ移動によるリンク切れの修正を支援するシステム (WISH システムと呼ぶ) を開発し [7] [8] [9] [10]，実験や公開(注1)を行っている．

7. まとめと今後の課題

本稿では，Web サイト構築と管理の新たなアプローチとして，Web ページ間のコンテンツ一貫性制約を用いた Web サイト管理手法を提案した．また，予備実験により実際の Web ページへの適用が可能である事を示した．

今後の課題としては，まず，提案アルゴリズムに関する大規模実験がある．また，木包含従属性を含むコンテンツ一貫性制約を用いた Web サイト管理手法を実際の Web サイトに適用し，有効性を評価することが必要である．

8. 謝　　辞

ゼミなどでご議論いただきました筑波大学大学院図書館情報メディア研究科の阪口哲男助教授，永森光晴講師に感謝致します．本研究の一部は，文部科学省科学研究費補助金（#18049005）による．

文　　献

[1] Dreamweaver 8. http://www.adobe.com/products/dreamweaver/.

[2] MARY F. FERNANDEZ., DANIELA FLORESCU.,JAEWOO KANG.,ALON Y. LEVY. AND DAN SUCIU. 1997. STRUDEL: a Web site management system. In ACM SIGMOD International Conference on Management of Data, pp.549-552.

[3] Xoops. http://www.xoops.jp/.

[4] Zope. http://zope.jp/.

[5] JOãO CAVALCANTI. AND WAMBERTO VASCONCELOS. 2002. A logic-based approach for automatic synthesis and maintenance of web sites. ACM International Conference Proceeding Series, Vol. 27, 619-626.

[6] 河合 英紀, 河野 泉, 石黒 義英, 福島 俊一, サイト品質管理のためのリンク不整合検出. 電子情報通信学会第 15 回データ工学ワークショップ (DEWS2004), 8 pages, 2004 年 3 月.

[7] 中溝昌佳, 森嶋厚行, 有山智洋, 杉本重雄, 北川博之, WWW コンテンツ一貫性維持のためのリンク更新機構の提案. 日本データベース学会 Letters, Vol.2, No.2, pp.65-68, 2003 年 10 月.

[8] 中溝昌佳, 森嶋厚行, 杉本重雄, 北川博之, WWW における信頼度の高いリンクの発見. 情報処理学会研究報告 Vol.2004, No.72(2004-DBS-134(II)), pp.397-402. 電子情報通信学会技術研究報告 Vol.104, No.177(DE2004-63), pp.87-92, 2004 年 7 月.

[9] 中溝昌佳, 森嶋厚行, 杉本重雄, 北川博之, WWW リンク一貫性維持支援システムにおけるリンク切れ自動修復. 日本データベース学会 Letters, Vol.3, No.3, pp.5-8, 2004 年 12 月.

[10] 中溝昌佳, 飯田敏成, 森嶋厚行, 杉本重雄, 北川博之, Web リンク切れの自動修正における信頼度の高いリンク情報の利用. 電子情報通信学会第 16 回データ工学ワークショップ (DEWS2005), 7 pages, 2005 年 3 月.

（注1）：http://wish.slis.tsukuba.ac.jp/LIM-RO.html.