社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－GN－68（12）
2008 / 5 / 23

ユーザ毎に変更可能な仮想ディレクトリ構造に基づくWeb文書の構造化

吉田　翔†　奥居　哲††

† 中部大学工学研究科情報工学専攻
†† 中部大学工学部情報工学科
E-mail: †yoshida@isc.chubu.ac.jp, ††okui@cs.chubu.ac.jp

あらまし 仮想ディレクトリに基づきWebサイトの文書を構造化する枠組みを提案する．仮想ディレクトリ構造はWeb文書の意味内容を反映したメタ情報とその配置情報とに応じて生成される．後者は各利用者に固有の情報であり，これにより利用者毎に適切なディレクトリ構造が得られる．さらに，仮想ディレクトリ構造は閲覧中の状況に応じて利用者が動的に再構築可能である．発信者の想定を外れたディレクトリ生成を抑制する仕組みとして，メタ情報の配置に制約を設ける方法についても議論する．最後に，本提案手法の適用事例として，Wikiサイトの自動リンクに関する問題が改善可能であることを示す．

キーワード 仮想ディレクトリ，Web文書，メタ情報，タグ

Structuring Web Documents via User-Configurable Virtual Directories

Sho YOSHIDA† and Satoshi OKUI††

† Graduate School of Computer Science, Chubu University
†† Department of Computer Science, Chubu University
E-mail: †yoshida@isc.chubu.ac.jp, ††okui@cs.chubu.ac.jp

1. はじめに

近年，Webサイトの大規模化・複雑化にともない，Webサイト上の情報を適切に整理し提示する必要性が益々高まっている．ディレクトリ構造は，そのためにもっとも広く用いられている情報の構造化手法である（例えば，[5]）．適切なディレクトリ構造を用いてWeb文書を分類することで，必要な情報へのアクセスが容易になる．ディレクトリ構造は，アクセス制御やコンポーネントの再利用等の基盤としても重要である[6]．

しかしながら，ディレクトリ構造を用いた情報の構造化には様々な問題点も指摘されている[1]．そのひとつが，異なる観点に基づく異なるディレクトリ構造を両立できないという問題である．「撮影日」，「撮影場所」，「撮影者」のような互いに直交した（つまり，明確な依存関係のない）属性に基づきディレクトリ構造を設計する場合，異なるディレクトリ構造が許されるにもかかわらず，単一のディレクトリ構造を排他的に採用せざるを得ない．このため，採用したディレクトリ構造が，ある利用者にとっては適切であっても別の利用者にとってはそうではないという状況が起こり得る（さらに言えば，同じ利用者であっても状況により適切なディレクトリ構造は異なる）．

そこで，各Web文書のメタ情報と各利用者に固有の情報とから生成される仮想的なディレクトリ構造を活用することで，問題の解決をはかろうというのが本研究である．メタ情報から生成される仮想ディレクトリ構造に関しては，多くの試みがある（例えば，[1]～[3]）．本研究がこれらの研究と大きく異なるのは，属性の取捨選択とその順序付けに基づくディレクトリ構造の生成に主眼を置いている点である．属性の取捨選択と順序付けを各利用者に固有の情報とすることで，各利用者毎に異なるディレクトリ構造を許容する．これにより，利用者は自身にとって最も望ましいディレクトリ構造を自ら選択できるようになる．さらに，文書閲覧中においても属性の取捨選択と順序付けを随時変更可能にすることで，利用者の状況に応じた動的なディレクトリ構造の変更が可能になる．

一方，利用者によるディレクトリ構造変更の無制限な許容は，Webサイト作成者（発信者）の想定の枠を超えたディレクトリ構造の生成をもたらす．そこで，属性の取捨選択と順序付けに適切な制約を設ける方法についても検討する．

Wikipedia[4]に代表される大規模なWikiサイトの運用に関

しては，サイトの文書が構造化されていないことに起因するいくつかの問題点が指摘されている．引用元の文脈にそぐわない不適切な自動リンクが生成される問題もその一例である．この問題が，本稿で提案する利用者固有の変更可能な仮想ディレクトリを Wiki に適用することでどのように改善されるかを示し，本提案の枠組みの有効性について議論する．

以下，本稿は次のように構成される．まず第2節で，既に言及したディレクトリ構造の問題点について更に検討した後，その解決策として第3節で仮想ディレクトリ構造を導入する．続く第4節では，利用者による（閲覧中の）仮想ディレクトリ構造の動的な再構築について述べる．第5節では仮想ディレクトリ構造の変更に関する制約の導入について述べる．第6節では，本提案の枠組みを Wiki に適用した事例について論じる．最後に第7節で，結論を述べる．

2. ディレクトリ構造に基づく Web サイトの問題点

本節では，ディレクトリ構造に基づく Web サイトの構造化の問題について具体例を示し，詳細に検討する．

ディレクトリ構造を階層構造と見なしたとき，各階層は分類のための何らかの観点を表している．図1の (1) は，**公開年度**，**講義**，**講義回**という観点に基づいて授業の資料を配布する Web サイトのディレクトリ構造の例である．

観点が直交している場合，すなわち互いの間に依存がない場合には，観点と階層の対応する順序を変更することにより異なるディレクトリ構造が得られる．この例では，(2) のようなディレクトリ構造も可能である．

この2つのディレクトリ構造を比較すると，利用者によって，その利便性に大きな違いがある．例えば，2008 年度にオペレーティングシステム（OS）とデータベース（DB）の授業を初めて履修する受講者にとっては (2) より (1) のほうが利便性が高いであろう．この利用者にとって，昨年度の講義資料は参考程度にしか必要ない．それにもかかわらず，他の講義回の資料や他の講義の資料を閲覧する際に，(2) では公開年度のディレクトリを横断しなければならないからである．一方，例えば，2007 年度に OS の授業を履修し 2008 年度にも再履修する受講者が，各講義回の資料を昨年度の資料と見比べながら閲覧するような状況では (2) のほうが利便性が高い．

このように，複数の異なるディレクトリ構造が可能であり利用者や状況によって望ましいディレクトリ構造が異なる場合であっても，従来のディレクトリ構造による Web サイトの構築では，いずれかひとつのディレクトリ構造を排他的に選択し，それに基づいて情報を分類せざるを得ない．このため，すべての利用者にとって最も望ましいディレクトリ構造を設計するのは一般に困難である．

3. 仮想ディレクトリ構造の導入

この問題を解決するには，利用者毎に異なるディレクトリ構造を許容する枠組みが必要である．そこで本節では，**仮想ディレクトリ木**と呼ぶ枠組みを導入する．

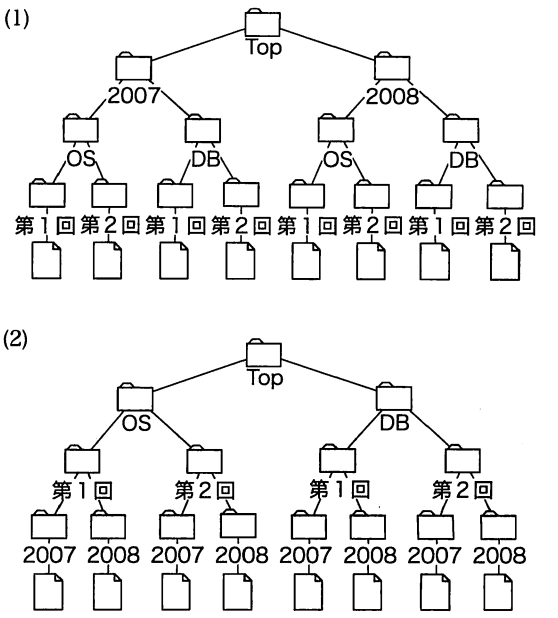

図1　直交した属性の順序によるディレクトリ構造の違い

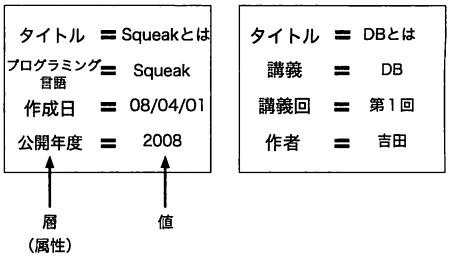

図2　メタ情報の例

3.1 層

一般に仮想ディレクトリ構造とは，ファイルの意味内容を反映するメタ情報から機械的に生成されるディレクトリ構造を意味する [1]〜[3]．本研究における仮想ディレクトリ構造も，Web 文書に付与されたメタ情報から生成される．各 Web 文書のメタ情報は属性とその値の対の集まりからなる．図2に一例を示す．本研究の手法では，属性は仮想ディレクトリ構造の階層に対応し属性値がその階層のディレクトリ名に対応する．これは [1]〜[3] における方式とは異なる．(注1) この点を強調して，以下では属性のことを**層**（layer）と呼ぶ．

3.2 層 配 置

仮想ディレクトリ木の生成には，Web 文書のメタ情報の他に (1) ディレクトリ生成に使用する層の選択と (2) 選択した層の順序付けの情報を用いる．これを層の列 $A_1, \ldots, A_n$ で表現し，**層配置**（layer configuration）と呼ぶ．層配置は本研究における中心的概念であり，以下に述べる仮想ディレクトリ木の階層構造を決定するガイドラインとしての役割を担う．

(注1)：本研究の方法とデータマイニングにおける決定木の生成との間には類似点がある．

3.3 仮想ディレクトリ木

層配置から導出されるディレクトリ木を**仮想ディレクトリ木**と呼ぶ．正確には，以下のように帰納的に定義される．まず，文書の集合 D に属する文書のうち層 A を有するものからなる集合を $D|_A$ で，層と値の対 $A=a$ を有するものからなる集合を $D|_{A=a}$ でそれぞれ表し，D に属する文書のいずれかに出現する層の値からなる集合を $V(D)$ で表すとする．このとき，層配置 $A_1,...,A_k$ と文書の集合 D から生成される根が r の仮想ディレクトリ木を以下のように定義する．

[$k=0$ の場合]D の要素（文書）を葉として内包する名前が r のディレクトリ

[$k>0$ の場合]$V(D|_{A_1})=\{a_1,\ldots,a_n\}$ とする時，層配置 $A_2,...,A_k$ と文書の集合 $D|_{A_1=a_i}$ から生成される根が a_i の仮想ディレクトリ木（$1 \leqq i \leqq n$）を部分木として内包する名前が r のディレクトリ

以下，単に層配置 $A_1,\ldots,A_n$ から生成される仮想ディレクトリ木と言った場合，層配置 $A_1,\ldots,A_n$ と全文書から生成される根が Top の仮想ディレクトリを意味するものとする．ここで Top はディレクトリ木全体の根を表す特別なディレクトリ名である．

この定義は直観的には，仮想ディレクトリ木の生成が，層と値の対をフィルタとみなし文書を絞り込みながらトップダウンに行われるを述べている．つまり (i) 層配置を左から順に走査し，(ii) その層が取り得る値に対応するサブディレクトリを生成し，(iii) その層と値の対を有する文書のみに絞り込んだ上で，(i) から (iii) の操作を反復することで仮想ディレクトリ木が生成される．

従来の（固定された）ディレクトリ構造と比較して，本研究の仮想ディレクトリ木は以下のような特徴を持つ．

第 1 に，仮想ディレクトリ木は階層化されており，各ディレクトリは何れかの階層に所属している．これに対して，従来の通常のディレクトリ構造は必ずしも階層化されておらず部分木毎に異なる高さを持つことが許される．この特徴と関連して，本研究の仮想ディレクトリ木においては，Web 文書は最下層のディレクトリにのみ出現する．それ以外のディレクトリはサブディレクトリを内包するが，Web 文書を直接に内包することはない．また，サイトの全文書が仮想ディレクトリ木に出現するわけではない．定義より，一般に層配置 $A_1,\ldots,A_n$ から生成される仮想ディレクトリ木に出現する文書は，メタ情報として層 A_1 から A_n をすべて有する文書に限られる．出現しない Web 文書を閲覧したい場合，利用者は次節で述べる仮想ディレクトリの（閲覧中における）変更操作を行う．

第 2 に，ディレクトリ木は文書を絞り込みながらトップダウンに生成されるので，余分なディレクトリを含まない．従来のディレクトリ構造の場合には，サブディレクトリの階層を辿った末に空ディレクトリに到達することがある．本研究の仮想ディレクトリ木では，これは起こり得ない．

第 3 に，仮想ディレクトリ木には，複数の異なるディレクトリに同一の文書を配置することが可能である．これは Web 文書に重複した層の出現を許すことで実現される．例えば，ある講義資料を 2007 年度と 2008 年度で共用する場合には，その文書のメタ情報に**公開年度=2007** という対と**公開年度=2008** という対の両方を含めればよい．

3.4 利用者固有の仮想ディレクトリの並立

層配置の情報を利用者固有の情報として利用者毎に用意することで，利用者毎に異なる仮想ディレクトリ木が生成可能になる．これにより，第 2 節で検討した問題が解決される．例えば，新規履修受講生に適した図 1 の (1) のディレクトリ構造を提示するには，新規履修受講生に固有の層配置情報として，順に**公開年度**，**講義**，**講義回**を用意すればよい．同様に，再履修受講生に適した (2) のディレクトリ構造を提示するには，**講義**，**講義回**，**公開年度**という層配置情報を用意すればよい．

4. 仮想ディレクトリ構造の動的な変更

前節では，利用者毎に固有の仮想ディレクトリ構造を導入することで，各々の利用者にとっての利便性を改善できることをみた．さらに，閲覧中の利用者が状況に応じて随時，ディレクトリ構造を変更できれば，さらに利便性を改善できる．本節では，このための枠組みを導入する．

4.1 閲覧時点の情報の保存

仮想ディレクトリ木において利用者が現在閲覧中のディレクトリを**カレントディレクトリ**と呼ぶ．また，カレントディレクトリの属する階層を**カレント層**と呼ぶ．

閲覧中における仮想ディレクトリ構造の変更で最も問題となるのは，変更前の履歴情報が喪失することである．一般に，ルートディレクトリからカレントディレクトリに至るパスの情報は利用者にとって重要な履歴情報である．層配置における層の削除や挿入，順序の交換に伴い仮想ディレクトリ木の構造が変化し，この履歴情報が変更後の新しいディレクトリ構造と対応しなくなる．特に，変更直前のカレントディレクトリは変更後，喪失するかもしれないので，新たなカレントディレクトリを設定し，利用者を誘導する必要がある．このとき，変更前の情報を可能な限り保持しつつ，変更後のディレクトリ構造に対応したパス情報を利用者に与えることが望ましい．これに対して，カレントディレクトリが無効になった際に，例えば，無条件にルートディレクトリを新しいカレントディレクトリにする方法では，利用者の閲覧履歴が完全に失われることになり，利便性を大きく損ねるであろう．

以下，この問題について議論するにあたり，まず，閲覧時点の情報を表す記法を導入する．層配置 $A_1,\ldots,A_n$ と層 A_1 から A_k（$1 \leqq k \leqq n$）に対する値 a_1 から a_k が与えられたとき，

$$[A_1=a_1]\ldots[A_k=a_k][A_{k+1}]\ldots[A_n]$$

と書いて，これを**時点表示**あるいは**様相**と呼ぶ．A_1 から A_k を，この時点表示における**確定層**と呼び，A_{k+1} から A_n を**未確定層**と呼ぶ．

時点表示は，仮想ディレクトリ木における利用者の現在の位置を表す．すなわち，A_k はカレント層，a_k はカレントディレクトリ，$a_1,\ldots,a_k$ はルートディレクトリからカレントディレクトリに至るパスに対応する（図 3）．そこで，混同を許し，A_k

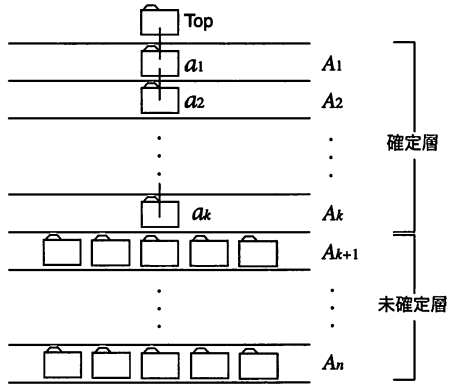

図 3　時点表示の意味

をカレント層，a_k をカレントディレクトリと呼ぶ．以下，この時点表示を用いて，仮想ディレクトリ木の閲覧中の変更について議論する．

仮想ディレクトリ木の構造を変更する操作には，層配置に対する (1) 層の削除，(2) 層の挿入，(3) 層の順序交換の 3 つが考えられる．このうち，層の順序交換は層の削除に引続く層の挿入によって表現可能(注2)であるので以下では直接に取り扱わない．これらの操作により，変更直前の時点表示における確定層の情報が可能な限り保たれることが望ましい．情報を可能な限り保つとは，変更前の確定層をなるべく残す（カレントディレクトリ a_k ができる限り左に移動しないようにする）ということである．

まず，層の削除について考える．**層の削除**とは，層配置 $A_1,\ldots,A_n$ から何れかの層 A_i $(1 \leqq i \leqq n)$ を取り除く操作である．取り除かれる層 A_i がカレント層 A_k 以外の確定層の場合（すなわち，$1 \leqq i < k$ の場合）には，A_i に対応する階層のディレクトリは喪失するが，(仮想ディレクトリの定義により）その配下のディレクトリが喪失することはない（カレントディレクトリも不変）．よって，削除直後の時点表示を

$$[A_1 = a_1]\ldots[A_{i-1} = a_{i-1}]\ \ [A_{i+1} = a_{i+1}]\cdots \\ [A_k = a_k][A_{k+1}]\ldots[A_n]$$

と定める．一方，取り除かれる層 A_i がカレント層 A_k の場合には，カレントディレクトリが喪失する．この場合，階層をひとつ遡った a_{k-1}（$k = 1$ の場合には Top）を新たなカレントディレクトリとする．つまり，削除直後の時点表示を

$$[A_1 = a_1]\ldots[A_{k-1} = a_{k-1}][A_{k+1}]\ldots[A_n]$$

と定める．図 4①にこの例を示す．

次に，層配置への**層の挿入**について考える．層の順序交換を層の削除に続く層の挿入で代替したことにより，挿入される層（以下，B）には確定層の場合と未確定層の場合とがあることに注意を要する．まず，挿入される位置が，ある未確定層 A_i $(k < i \leqq n)$ より下層の場合には，確定層（よって，カレントディレクトリ）は不変である．ただし，B が確定層の場合には，その値は破棄される．ディレクトリ選択はトップダウンに行われるので，未確定層の下層に確定層が現れるのは無意味だからである．次に，挿入される位置がカレント層より上層の場合には，B が確定層か未確定層かで異なる．B が未確定層の場合，その下層は（すでに言及した理由により）すべて未確定層と見なされる．よって挿入した位置の直上の層の確定値が新たなカレントディレクトリになる．すなわち，挿入位置を A_i の左端 $(1 \leqq i < k)$ とすると，挿入直後の時点表示は

$$[A_1 = a_1]\ldots[A_{i-1} = a_{i-1}][B][A_i]\ldots[A_n]$$

となる（新たなカレントディレクトリは a_{i-1}．ただし，$i = 1$ のときは Top）．図 4②にこの例を示す．一方，B が確定層の場合，B は直前の削除操作により確定層から削除されたものに限られる．つまり，先行する削除操作の直前の時点表示を

$$[A'_1 = a'_1]\ldots[A'_k = a'_k][A'_{k+1}]\ldots[A'_{n+1}]$$

とすると，B は $A'_1,\ldots,A'_k$ の何れかと一致する．よって，この挿入操作によって，先行する削除操作直後のカレントディレクトリ a_k は不変である．最後に，挿入位置がカレントディレクトリの直下（A_k と A_{k+1} との間）の場合を考える．これは先ほどの場合と同様に考えることができる．B が未確定層の場合には，挿入直後の時点表示は

$$[A_1 = a_1]\ldots[A_k = a_k][B][A_{k+1}]\ldots[A_n]$$

となり（カレンドディレクトリ不変），B が確定層の場合には，

$$[A_1 = a_1]\ldots[A_k = a_k][B = b][A_{k+1}]\ldots[A_n]$$

となる（B の確定値 b が新たなカレントディレクトリになる）．図 4③にこの例を示す．

層配置の変更後の時点表示を以上のように定めることで，閲覧中の利用者の履歴情報が可能な限り保持される．

4.2　ナビゲータ

閲覧中の利用者による仮想ディレクトリの変更（層配置の変更）操作を可能にするために，**ナビゲータ**と呼ぶユーザインターフェースを導入する（図 5）．ナビゲータは現在の時点表示のグラフィカルな表示であり，閲覧中の Web ページの上部に現れる．ドラッグ・アンド・ドロップ操作による層の削除，挿入，順序交換の操作を可能にする．本研究の現在の試験実装では Ajax 技術を用いて実装されている．また，任意の時点のナビゲータをドラッグ・アンド・ドロップで保存し再利用するインターフェースを備えることにより，利用者は異なる仮想ディレクトリ木を切り替えつつ閲覧することが可能になる．

5.　仮想ディレクトリ構造への制約の導入

前節の仮想ディレクトリ構造の動的な再構築は，利用者が状況に応じて随時，ディレクトリ構造を変更を可能にすることによる利便性を述べた．その一方で，利用者によるディレクトリ

(注2)：もっとも，仮想ディレクトリの再構築に伴う計算コストを考える上では，層の順序交換をアトミックな操作とみなしたほうがよい．本稿では計算コストの問題は議論しない．

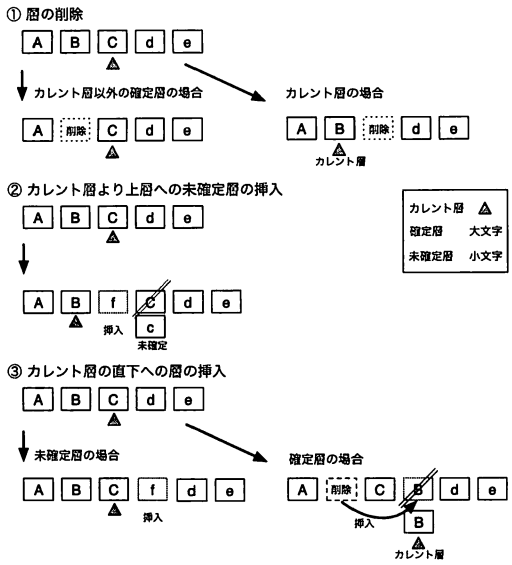

図 4　層配置の変更

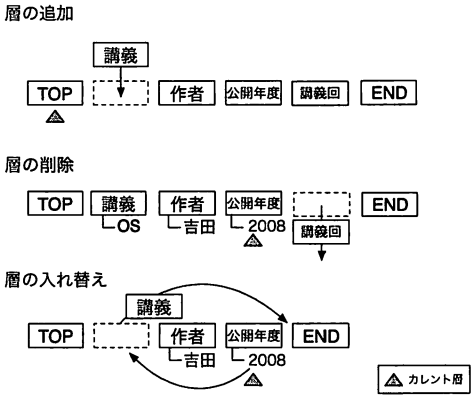

図 5　ナビゲータ

構造の無制限の許容は発信者の想定の枠を超えたディレクトリ構造の生成をもたらす．そこで本節では，層の取捨選択と順序付けに制約を導入する枠組みについて述べる．

5.1　制約の必要性

前節までで取り上げた例（**講義**，**公開年度**，**講義回**や**撮影日**，**撮影場所**，**撮影者**）では，層の間に明確な依存関係がなかった（直交性）．このため，層の順序の変換により異なる仮想ディレクトリ木が得られ，利用者はこれらの中で自身にとって最も望ましいものを選択することが可能になった．

その一方で，層の順序の変換に適切な制約を設けたほうが利用者の利便性を向上させる場合もある．層の間に明確な関数依存性が見られる場合がその一例である．例えば，**講義**から**講師**が一意に決定される（つまりオムニバス形式の授業がない）場合，この順に層を配置すると，多数の講義名を表す各ディレクトリの直下には講師を表す単一のディレクトリのみが内包されることになり，ディレクトリによる整理の意味が無くなる．そこで，**講師**が必ず**講義**より上の階層に出現するという制約を情報発信者が導入できれば，利用者が不適切な階層構造を回避するガイドラインとして役立つであろう．

同様のことが，層の選択に関しても言える．利用者は，仮想ディレクトリ木の生成に用いる層を選択することで，Web 文書の可視性を自ら選択する．このため，例えば，情報発信者がすべての利用者に伝達したい文書（例：休講連絡）が利用者に伝達されない場合があり得る．そこで，層の選択をある程度強制する枠組みが必要になる．

このような制約の導入はアクセス制限を行う場合にも必要である．ディレクトリ構造はサブディレクトリ木単位でのアクセス制御（例：再履修受講生のみ閲覧可や非公開ドラフト等）を可能にする．しかし，仮想ディレクトリにおいてサブディレクトリ単位のアクセス制御を行おうとするならば，そのサブディレクトリの根を横断する層の順序変換を禁止しなければ制限が意味を失う．

5.2　制約の導入

そこで，仮想ディレクトリ木の変更に関する制約を表現する枠組みとして，層の間に 2 種類の制約を導入する．ひとつは，層の層配置への導入に関する制約であり，もうひとつは，導入された層の間の順序に関する制約である．

まず，サイトに存在するすべての層の集合上に，**導入依存**と呼ぶ 2 項関係を仮定する．層 A と B の間に導入依存の関係があるとき，これを $A \preceq B$ と書き，A は B に**依存**している（A depends on B）と言う．この意図は，A を導入した層配置には B も導入しなければならないということである．よって，層配置に関して以下の制約 (C1) を設ける．

(C1) 任意の層配置 Γ について，$A \preceq B$ かつ $A \in \Gamma$ ならば $B \in \Gamma$.

導入依存に対する最小限の仮定は，それが前順序（すなわち反射律と推移律をみたす 2 項関係）であるということである．つまり，導入依存は循環していても構わないとする．例えば，$A \preceq B$ であると同時に $B \preceq A$ であってもよい．このように循環を許すことにより，「A か B の何れか一方を導入する際には他方も導入しなければならない」という制約を表現することが可能になる．

次に，サイトに存在するすべての層の集合上に，**階層依存**と呼ぶ 2 項関係を仮定する．層 A と B の間に階層依存の関係があるとき，これを $A \sqsubseteq B$ と書き，A は B に**先行**している（A precedes B）と言う．この意図は，A と B が同じ層配置の中に出現する際には A が B よりも左に出現しなくてはならない（仮想ディレクトリ木において上の階層に位置しなくてはならない）ということである．よって，層配置に関して以下の制約 (C2) をさらに設ける．

(C2) 任意の層配置 Γ について，$A \sqsubseteq B$ かつ $A, B \in \Gamma$ ならば A は Γ において B より左に出現する．

階層構造は循環を許さないので，導入依存とは異なり，階層

依存は厳格な半順序である．

利用者による層配置の変更は，制約 (C1) と (C2) の範囲内でのみ許される．利用者が（未確定）層を新たに導入する操作を行う毎に，(C1) を満足するように必要な層が自動的に挿入される．また，利用者が (C2) に反する順序交換操作を行おうとした場合には拒否される．

実際に導入依存や階層依存の関係を与えるには，有向グラフの反射推移閉包を求めるアルゴリズムを用いればよい．つまり，情報発信者は一部の層の対に対してのみ具体的に依存関係を明示すれば十分である．

6. 仮想ディレクトリ構造の Wiki への導入

本研究で提案する枠組の適用事例として，Wiki への仮想ディレクトリ構造の導入について検討する．

6.1 Wiki における自動リンクの問題点

Wiki の特徴のひとつは，既に存在するページへのリンクを自動的に生成すること（以下，自動リンク）である．これにより関連するページへのリンクを作成する労力が大幅に軽減される一方で，意図しないリンクが生成される場合も多い．実際に，著者らの一人が管理する授業サポートの Wiki サイトにおいて，以前の公開年度に出題したレポート課題のページに張られた自動リンクを受講者が誤って辿り，間違った課題に取り組んでしまったという事例が生じた．このように，不必要な自動リンクは情報発信者の意図しない不適切な情報伝達を招く恐れがある．

ディレクトリ構造を導入することで，木構造を利用した自動リンクの制御が可能になる．現在閲覧中の Web 文書を含むサブディレクトリ木の階層を考えると，現在閲覧中の Web 文書と関連の深い文書ほど下層のサブディレクトリ木の葉に出現すると考えられる．よって，サブディレクトリ単位で自動リンクを許可する仕組みを導入することで，引用元の文脈にそぐわない不適切な自動リンクの生成を抑制できる．しかしながら，従来のディレクトリ構造では，固定された階層構造が障害となり，サブディレクトリ単位の制御を有効に活用することが難しい．

本研究で提案する仮想ディレクトリ構造に基づく枠組みを用いることで，サブディレクトリ単位のリンク制御を有効に機能させることが可能になる．例えば，講義サポートサイトにおいて，昨年度までの授業資料やレポート課題へのリンクを新規履修受講生に対してはデフォルトで表示しないようにするには，以下のようにすればよい．まず，**公開年度**が**講義**や**講義回**等の層よりも上の階層に位置するように層を配置する（図 6）．次にカレントディレクトリの初期値を **2008 年度**という仮想ディレクトリ（あるいはその配下）に設定する．その上で，自動リンクの生成を許可する範囲を **2008 年度**というサブディレクトリに設定する．この範囲設定はナビゲータに追加された範囲指定のインターフェースを通して行う（図 7(1)）．図 7 において斜線で示した部分が範囲指定のインターフェースであり，これに包含される部分が自動リンクを許可される．この範囲はマウス操作により適宜変更可能である．利用者は，もし必要ならば，この範囲を拡大して過去の資料へのリンクを生成することも可能である（図 7(2)）．

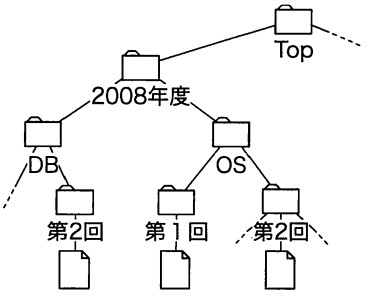

図 6 新規履修生に適した仮想ディレクトリ構造

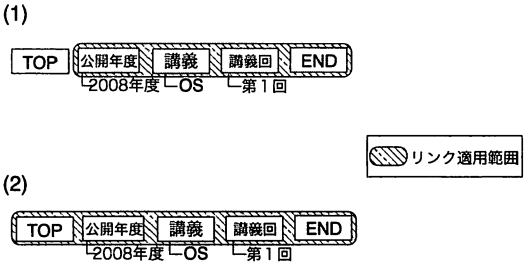

図 7 ナビゲータを用いた自動リンクの許容範囲の設定

7. おわりに

本稿では，仮想ディレクトリ構造に基づく Web 文書の構造化の枠組みを提案した．本提案の仮想ディレクトリ構造は，メタ情報の配置情報（層配置）を用いるという点で，既に提案されている仮想ディレクトリ構造と異なる．層配置を各利用者固有の情報とすることで，利用者が自ら望ましいディレクトリ構造を得ることができ，利用者の利便性が向上することを示した．また，利用者が閲覧中の仮想ディレクトリ構造を再構成する枠組みについて論じ，ナビゲータと呼ぶユーザ・インターフェースを導入した．さらに，情報発信者の想定を外れたディレクトリ構造を抑制するために層の間に 2 種類の依存関係による制約を導入する枠組みを検討した．最後に本提案の枠組みを Wiki に導入することで，自動リンクに関する問題が改善可能であることを示した．

今後の課題としては，本稿で議論しなかった仮想ディレクトリ構造の再構築にかかる計算コストの問題が挙げられる．

文　献

[1] S. Bloehdorn and M. Völkel, TagFS - Tag Semantics for Hierarchical File Systems, In *Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06)* , 2006.

[2] D. K. Gifford, P. Jouvelot, M. A. Sheldon and J. W. O'Tooles Jr, Semantic File Systems, In *Proceeedings of 13th ACM Symposium on Operationg Systems Principles*, pages 16-25, ACM SIGOPS, 1991.

[3] C. Marshall, Birch: A Metadata Search File System, 2006.

[4] Wikipedia, http://ja.wikipedia.org/.

[5] Yahoo! Directory, http://dir.yahoo.com/.

[6] Zope, http://www.zope.org/.