ヒューマンインタフェース 67－1
（1996. 7. 11）

Multitrees における対話操作の実現

増田尚則　　今宮淳美

山梨大学　　電子情報工学科

〒400 山梨県甲府市武田 4-3-11

情報の再利用と再構成という考えに基づいて情報検索を効率よく行うためのデータ構造として Multitrees が提案されている．本論文では，Multitrees における基本操作を定義し，それらをユーザが対話的に行えるシステムの設計・実装について述べる．

Implementing the interactive operations for Multitrees

Hisanori Masuda　　Atsumi Imamiya

Department of Electrical Engineering and Computer Science,
Yamanashi University

4-3-11 Takeda, Kofu, Yamanashi 400, Japan

Furnas defined and discussed Multitrees as an effective data structures for information representation and retrieval. Multitrees is based on an idea of reuse and restructure of existing information structure. This paper describes the internal data structure, algorithms and an interactive system for operating the Multitrees.

1 はじめに

私たちは, 効率よく情報検索をするためにしばしば情報構造として tree を用いる. しかし, その tree 構造が大きくなると, 必要とする部分へのアクセスに時間がかかる. これを解決するため, 必要な部分を再利用し, 効率のよい検索を可能とする情報構造として Multitrees が提案されている [1]. たとえば, 対象資源を本として, Multitrees を用いると, 多人数が本の中の必要な章や節を引用するための情報構造が表現可能になる. さらに, 他人が構成した tree 構造からも, 章や節による引用するための情報構造が表現可能になる.

論文 [1] において, Furnas らは Multitrees の定義および概念を提案しているが, この構造を実現するための具体的なデータ構造やアルゴリズムを示していない. 本論文では, Multitrees を実現するためのデータ構造と基本操作アルゴリズムを示すとともに, それら操作を対話的に行うシステムの実装について述べる.

第 2 節で Multitrees の例と Furnas らが提案した定義を示す. 次に, 第 3 節では Multitrees を実現するためのデータ構造と基本操作アルゴリズムについて述べる. 第 4 節では, 第 3 節で述べた基本操作を対話的に行うためのシステムの設計・実装について述べる.

2 Multitrees

この節では, Multitrees の例と Furnas らの提案した Multitrees の定義について述べる.

2.1 二人の教師

まず, Multitrees の構築例を説明する. 今, 教師 A がうけもつ授業で使用する reader のシラバスや資料を作るとする. 教師 A は, 白紙の状態から書き始める代わりに, 既にある書物から一部分を参照するかもしれない. 書物を 1 つの大きな tree 構造で表現した時 (図 1.1), 教師 A は, ここからある章を, また他からある節をという具合にこの大きな tree 構造から授業のための reader の構造をまとめることができる. この作業の結果, 教師 A は元の構造から章や節を選ぶと同時に, それに従属する subtree を選び, 自分の新しい tree 構造を作ったことになる (図 1.2).

さらに, 教師 B が教師 A のシラバスから一部分を借りて使ったり, 全く新しい tree 部品を使って書いてもよい (図 1.3). 唯一の制約は, それぞれのシラバスは tree であり, 必要な部分を集め, いくつかの新しい階層構造を作ることである. その結果としての全体の情報構造を **Multitrees** と呼ぶ.

2.2 Multitrees の定義

論文 [1] において, Furnas らは次の 4 条件を満たす DAG(非循環有向グラフ) として Multitrees を定義している. それぞれの条件は等価である.

1. 既に存在する tree の上に新しい tree を付け加えることによって構築される DAG.
2. どのノードの子孫も tree を形成する DAG.
3. どのノードの先祖も逆 tree を形成する DAG.
4. Diamond free な DAG (図 2)

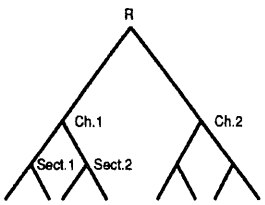


図 1.1 資源の tree 構造

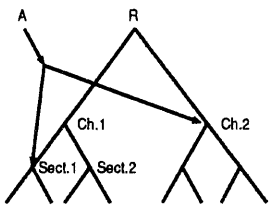


図 1.2 教師 A がシラバスを作成

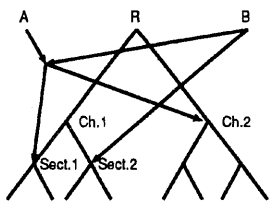


図 1.3 教師 B もシラバスを作成

図 1 Multitrees の構築例

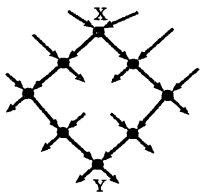


図 2 Diamond の制約: ノード X で分岐した経路が, 再びノード Y で一緒になると, ノード X とノード Y の間に Diamond ができる. これは, Multitrees においては許されてない. すなわち, Multitrees においては, 2 ノード間では高々 1 つの経路のみが存在する.

3 データ構造と基本操作アルゴリズム

この節では, Multitrees 実現のためのデータ構造とアルゴリズムについて述べる.

3.1 データ構造

ノードは次の 3 領域からなる (図 3).

1. キーデータ領域
2. 親ノードへのポインタ領域
3. 子ノードへのポインタ領域

親ノードへのリンク, 子ノードへのリンクは補助ノードを介して行う. Multitrees は任意のノードが複数の親を持つという特徴がある. 後述する基本操作のうち Add 操作と Delete 操作においては対象とするノードが複数の親をもつ場合, 通常の tree のためのデータ構造では不都合が生じる. 補助ノードは次の 2 領域からなる (図 4).

1. ノードへのポインタ領域
2. 次の補助ノードへのポインタ領域

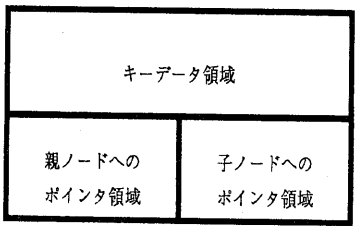


図 3 ノードの構造

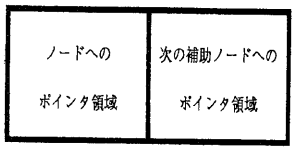
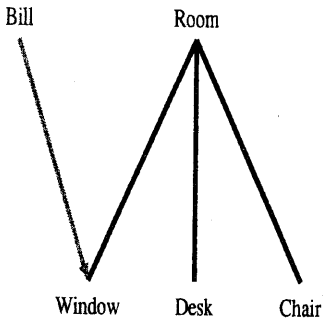
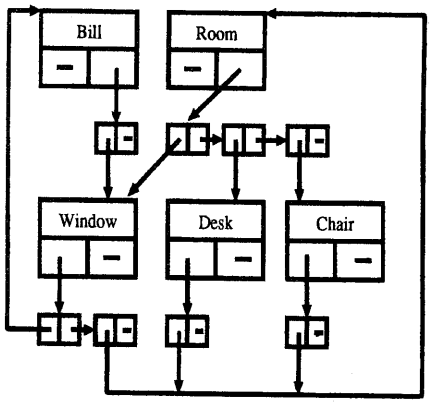


図 4 補助ノードの構造



(a)



(b)

(a) の Multitrees の内部表現. Room ノードから子ノードへのリンクをたどると, 3 つの補助ノードがあり, それらの補助ノードは Window と Desk と Chair ノードへのリンクを持つ. もう一方の root ノードである Bill から子ノードへのリンクをたどると 1 つの補助ノードを介して Window ノードに致る. 次に, Window ノードから親ノードへのリンクをたどると, 2 つの補助ノードがあり, 一方は Bill へ, もう一方は Room へリンクしている.

図 5 Multitrees の例

3.2 基本操作について

Multitrees に対する 3 つの基本操作アルゴリズム Search, Add, Delete を示す.

1. **Search(rootname, searchame)**

Search 関数は rootname を根とする tree に searchname が存在するかどうかを探索する関数である. はじめに, rootname が存在するかをチェックし, そのノードの名前が探索している名前 (serchname) ならば探索を終了する. そうでない場合, 再帰的に子ノードおよび兄弟ノードを探索する. rootname の指す子ノードへのポインタを child_p として, Search 関数のコードを図 6 に示す.

```
int Search(rootname, searchname){
  if (rootname が存在) then
   if (rootname == searchname ) then
     True;
   else{
    flag = 0;
    if(Search2(child_p, searchname) == 1)
    then True;
    if(flag == 0) then False
   }
  else False;
   int Search2(補助ノードポインタ, name){
     if( 補助ノードポインタ != NULL) then
     {
       to_node := 補助ノードがもつ
                  ノードへのポインタ;
       current_name := to_node の指す
                       ノードの名前;
       if (current_name == name) then{
          flag = 1; return 1;
       }
       else {
          Search2(子ノード, name)
          if(flag == 1) return 1;
          Search2(兄弟ノード,name)
       }
     } /* if( != NULL) */
   } /* int Search2 */
} /* int Search */
```

図 6: Search 関数のコード

2. **Add(rootnameA, nodenameA, rootnameB, nodenameB)**

Add 手続きは rootnameA を根とする tree の nodenameA に rootnameB を根とする tree の nodenameB を付け足す手続きである. まず Add する 2 つのノードが実際に存在するか調べる. 次に, 両ノードが存在したら nodenameA の子ノードへのリンク先を新たな補助ノードを介して nodemameB にリンクする. 一方, nodenameB は親ノードへのリンク先を新たな補助ノードとすることによって nodenameA とリンクする. Add 手続きのコードを図 7 に示す.

```
void Add(rootnameA, nodenameA,
         rootnameB, nodenameB){
 if( Search(rootnameA, nodenameA) AND
 Search(rootnameB, nodenameB)) then {
   nodenameA の子ノードへのリンク先に
                補助ノードを挿入;
   追加した補助ノードのリンク先を
                nodenameB とする;
   nodenameB の親ノードへのリンク先に
                補助ノードを挿入;
   追加した補助ノードのリンク先を
                nodenameA とする;
 }
 else エラー処理
} /* void Add */
```

図 7: Add 関数の疑似コード

3. **Delete(rootname, nodename)**

Delete 手続きは rootname を根とする tree から ノード nodename を削除する手続きである. 通常の tree 構造の Delete 操作と違うのは, root ノードと削除したいノードの間に複数の親をもつノードが存在している場合である. たとえば, 図 9(a) において A を根とする tree から x を削除する場合, A と x の間に 親 A と 親 B をもつ M が存在する. この時, 単に M から x へのリンクを切ると, A を根とする tree から x が削除されるだけでなく, B を根とする tree からも削除されてしまう. A を根とする tree だけから x を削除する場合, この要求を満たすために, A と x の間にあるノードのうち複数の親を持つすべてのノード (図 9(a) では M) をコピー

ることによって Delete を実現する. 尚, 本論文で定義されている Delete 操作は対象ノード以下をすべて削除する. 図 8 に示すコードにおいて, root ノードを A, 削除ノードを x, A と x の間にあるノードを M とする.

```
void Delete(rootname nodename){
  if( Search(rootname, nodename)) then {
    rootname(ノード A とする) に移る;
    while(1){
      if( A の子ノード M が x) then {
        A から x へのリンクを切る;
        Delete 操作終了;
      }
      else{
        if( M の親が複数存在 ) then {
          M を M' にコピー;
          A から M へのリンクを切る;
          A から M' へリンクを張る;
          if(M' の子ノードが x) then {
            M' から x へのリンクを切る;
            Delete 操作終了;
          }
          else M' を A に置き換える;
        }
        else M を A に置き換える;
      }
    } /* while(1) */
  } /* if Search */
  else エラー処理
} /* void Delete */
```

図 8 Delete 関数の疑似コード

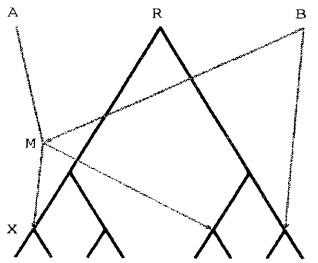


図 9(a) Multitrees の初期状態

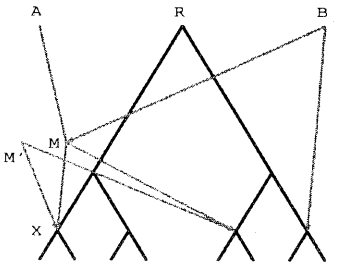


図 9(b) ノード M を M' にコピー

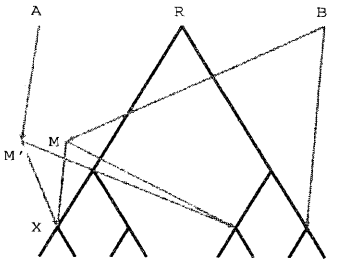


図 9(c) A-M のリンクを切り
A-M' のリンクを張る

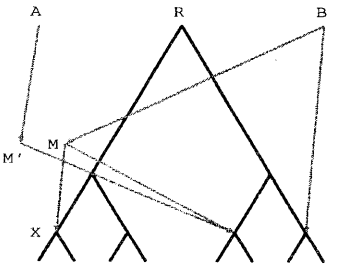


図 9(d) M'-x のリンクを切る

図 9 Delete(A, x) の様子. root ノード A と 削除するノード x の間に複数の親を持つノード M が存在する場合

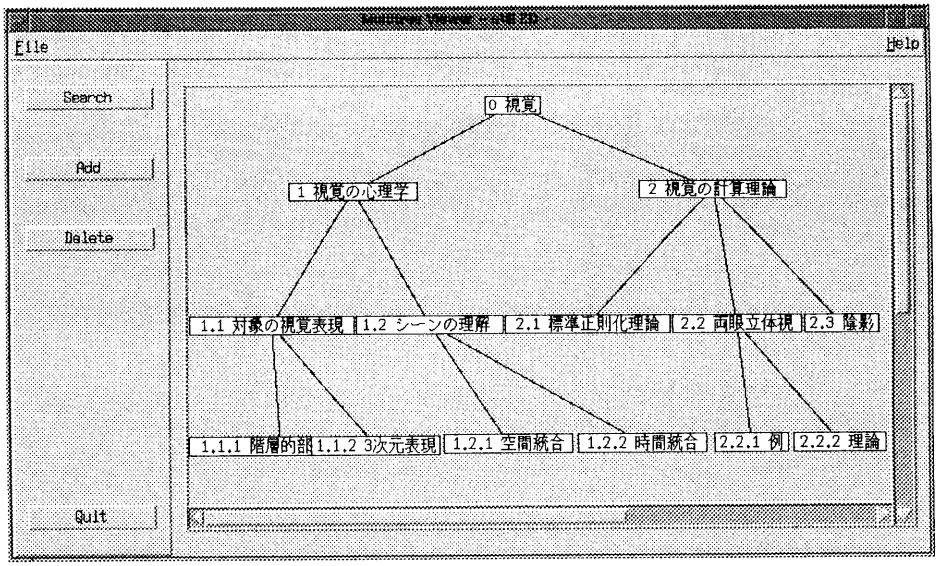


図 10　対話システム画面

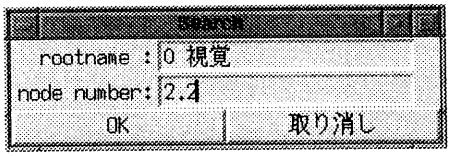


図 11　パラメータ入力用ウィンドウ

4　対話システム

4.1　設計

前述の 3 つの基本操作を画面上で対話的に行えるシステムの実現について述べる.

まず, システムがウィンドウ中に tree を表示する. ユーザは tree のノードをマウスで指定することにより, Add 操作と Delete 操作 を行うことができる. ウィンドウをスクロールして Search 操作を行うことができるが, この方法はユーザへの負荷が大きい. そこで, ユーザがパラメータ (rootname と nodenumber) を直接指定できる Search 機能も用意する. これは, ユーザが Search ボタンを押した時, システムがパラメータ入力用ウィンドウを開き, ユーザがパラメータを入力する方法で実現する (図 11).

同様の機能を Add 操作と Delete 操作でも利用できるように, Add ボタンと Delete ボタンをつける.

また, 基本操作を対話的に行うために, マウスのクリックによる新規ノードの作成の機能も用意する. ユーザがマウスをクリックしたら, システムはその場所に新しいノードを表示し, Multitrees の内部構造においても新たなノードを生成する.

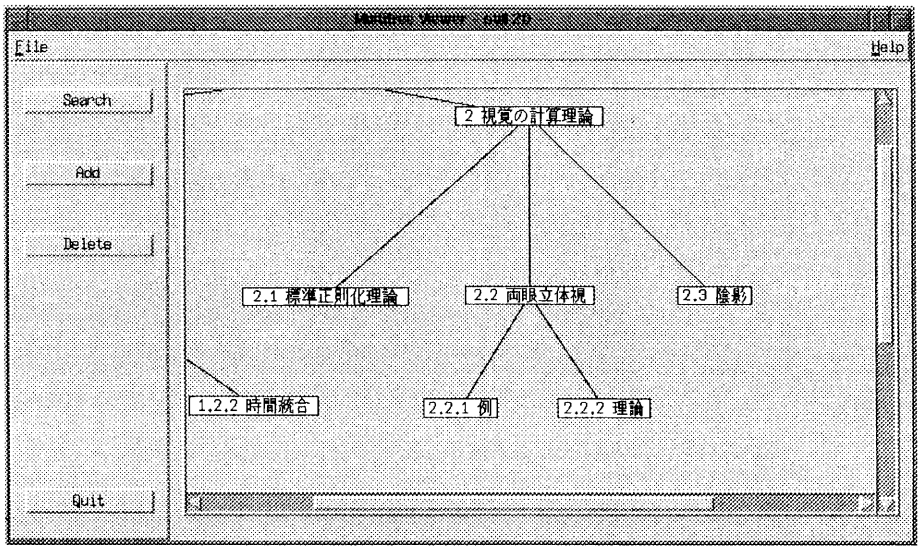


図 12　Search 操作後の画面

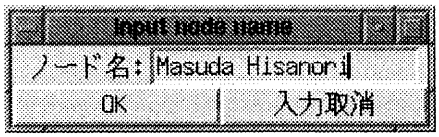


図 13　ノード名入力用ウィンドウ

4.2　実装

対話型システムは C 言語と Tcl/Tk [3] を用いて作成した．基本操作は C 言語で記述し，インタフェース部分は Tcl/Tk で記述した．

ここでは，Multitrees の対象資源を本とする実装例を示す．　対話システム画面を図 10 に示す．　画面は大きく 3 領域からなる．　すなわち，メニューバー，基本操作指定ボタン領域，Multitrees 表示領域である．ここで，ユーザが Search ボタンを押すとシステムはパラメータ入力用ウィンドウ (図 11) を表示する．ユーザがパラメータを入力した後 OK ボタンを押すと，システムは Search を実行する．Search 後の様子を図 12 に示す．

また，ユーザが Multitrees 表示領域でマウスの中ボタンをクリックすると，システムは図 13 に示すノード名入力用ウィンドウを開く．ユーザがノード名を入力した後に OK ボタンを押すと，システムはノード名を持つ新たなノードを作成する (図 14)．　Add 操作をマウスボタンのクリックのみで行う場合は，2 つのノード (例「Masuda Hisanori」と「両眼立体視」) をクリックすると，システムはそれらのノード間にエッジを描画する．この時に Multitrees 内部の構造に対しても Add 操作を行う．図 15 に Add 操作後の Multitrees を示す．

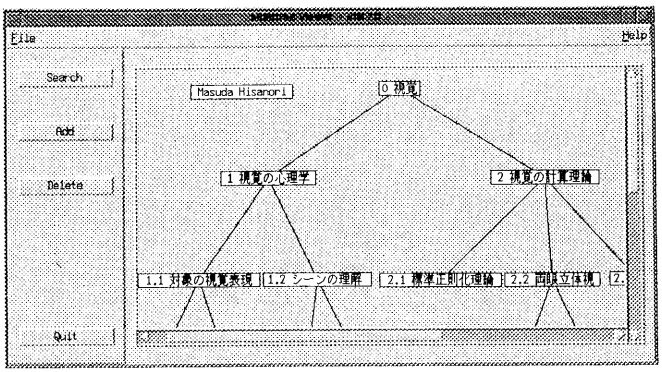


図 14　新規ノード作成後の画面

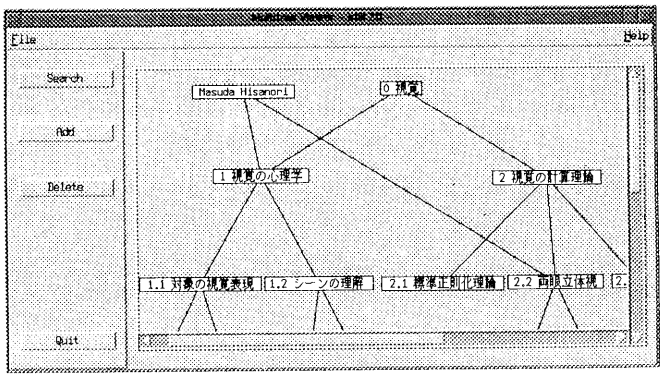


図 15　Add 操作後の画面

また対話操作を容易にするために次の機能も用意した.

1. マウスのドラッグによるノードの移動
2. ノードの移動に伴う, エッジの伸縮

5　おわりに

本論文では, Multitrees における基本操作を対話的に行うシステムの実現について述べた. 今後の課題として, Multitrees とその対話操作システムをさまざまな情報検索問題に応用すること, およびシステムのユーザビリティ評価がある.

本論文について検討およびいくつかの有効な視点を与えてくれた山梨大学茅暁陽講師に感謝します.

参考文献

[1] Furnas, G.W., Zacks, J., Multitrees:Enriching and Reuse Hierarchical Structure, Proceedings of CHI'94 Conference on Human Factors in Computing Systems ACM, 1994, pp.330-336

[2] Nagao, M., Digital Library in 21st Century,Proceeding of The Seventh International Forum on the Frontier of Telecommunications Technology, S3-1,1995,pp.1-30

[3] Ousterhout,J. Tcl and the Tk Toolkit, Addison-Wesley, 1993

[4] 川人他, 認知科学 3 視覚と聴覚, 岩波書店, 1994

[5] 増田尚則, Multitrees における対話的基本操作の実現, 山梨大学工学部卒業論文, 1996