複数の部分木の置換を許した属性木に対する属性評価アルゴリズムの提案

5N-7

馮　安　　菊野亨　　鳥居宏次
大阪大学　基礎工学部　情報工学科

1．まえがき

ソフトウェア開発支援システムなどに代表されるインタラクティブ・システムの開発に属性文法を利用する試みは，近年盛んに研究されている[1],[4]．本報告では，そのための一般的なモデルを紹介すると共に，既存の属性評価方法における問題点を解明し，新しいアルゴリズムを提案する．

2．モデル

インタラクティブ・システムは対象オブジェクトと操作から構成されると仮定する．更に，操作をオブジェクトを変更するもの（変更操作と呼ぶ）とオブジェクトの性質を参照するもの（参照操作と呼ぶ）に分ける．

属性文法 G で記述されたシステムでは，オブジェクト $w \in L(G)$ を属性木 $T_w(G)$で表す．各変更操作に対し内部処理 R を行い，属性木を変更する．各参照操作に対し内部処理 G(a) を行い，$T_w(G)$ の属性 a の値を求める．

変更操作はオブジェクト w の一部分 v を v' に置換して，新しいオブジェクト w' を作成する． 誤った変更操作（$w' \notin L(G)$）を防ぐために，変更操作に次の条件をつける．条件：v ， v' と対応する部分木$T_v(G)$, $T_{v'}(G)$の頂点のラベル（非終端記号）は同じである．

内部処理Rでは$T_v(G)$を$T_{v'}(G)$で置換し$T_w(G)$を導出する．その際に一部の属性の値を変更する必要がある．

3．属性評価方法

処理系列「$R_{11}...R_{1n1}\ G(a_1)...R_{m1}...R_{mnm}\ G(a_m)$」に対して，３通りの属性評価方法が考えられる：

M1：各$R_{ij}(1 \leq i, 1 \leq j \leq n_i)$毎に，$R_{ij}$によって値が変更される全ての属性を再評価する[3]．

M2：各$G(a_i)$毎に，$\{R_{ij}\}(1 \leq j \leq n_i)$によって変更される全ての属性をまとめて再評価する[2]．

M3：$G(a_i)$毎に，$\{R_{kj}\}(1 \leq k \leq i, 1 \leq j \leq n_k)$で変更される属性の内$a_i$に影響するものだけを再評価する．

M1でno$(1 \leq no \leq n_i)$回の評価が必要な属性をM2では高々1回で済ませる．しかし，M2では$\{a_i\}$と関係のないものまで再評価してしまう．M2で複数回の評価を行う属性は，M3では高々１回しか評価しない．

図１に属性文法の記述例を示す．非終端記号PRCはデータフロー図のバブルを表す．属性 PRC.s と PRC.e はバブルの番号を表す．

```
PRC = PRC PRC
  { PRC[2].s := PRC[1].s;
    PRC[3].s := PRC[2].e;
    PRC[1].e := PRC[3].e  }
PRC = BUB
  { PRC.e := PRC.s + 1  }
PRC = ⊥
  { PRC.e := PRC.s + 1  }
```

図１　属性文法記述

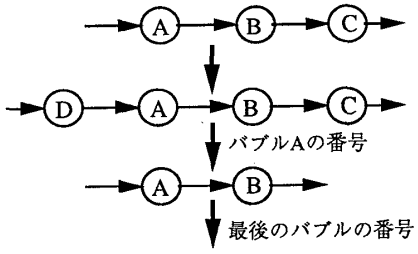
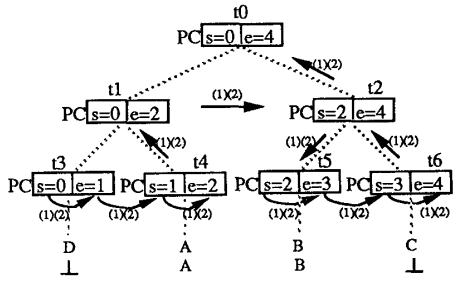
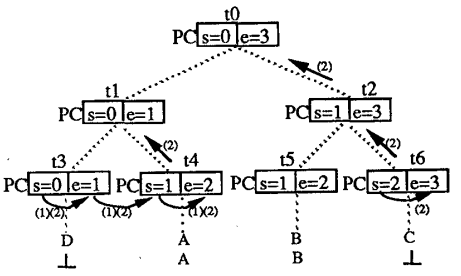


図２　データフロー図の変更と参照



(a)従来の方法



(b)提案する方法

図３　評価法の比較

次に，図２にデータフロー図に関する一連の操作を示

On Updating Algorithm for Attributed Tree with Multiple Subtree Replacements
An Feng, Tohru Kikuno and Koji Torii
Faculty of Engineering Science, Osaka University

す．↓は変更操作を表す．更に，バブルAの番号の参照(図3の属性(t4.e))と最後のバブルの番号の参照(属性t0.e)がある．

評価法M2とM3による属性評価を図3に示す．各矢印の番号(i)はそれがi回目の評価であることを表す．箱の中に示す各属性の値は1回目の評価の結果である．M2とM3での評価数はそれぞれ22個，10個である．

4．評価アルゴリズム

内部処理Rで置換してきたノードの集合をChange(T)とし，更新する必要がある属性の集合をAFFECTとする．AFFECT の属性の中で，属性依存グラフD(T)の上で，属性aに至る道のあるものの集合を Needed(a) とする．

有向グラフ$A=(V_A,E_A)$, $B=(V_B,E_B)$と$V'\subseteq V_A$に対し，

$A\cup B=(V_A\cup V_B, E_A\cup E_B)$

$A-B=(V_A, E_A-E_B)$

$A/V'=(V',E')$，但し，$E'=\{(x,y)\mid x,y\in V'$，かつ，

A上でV'の要素を含まないxからyへの道が存在する}

と定める．属性木上の各ノードrに対し，下方特徴グラフ[4]Sub(r)，上方特徴グラフ[4]Sup(r)，親子グラフPar(r)を持たせる．(以下ではノードxを頂点とする部分木をT_xとし，xの親をp(x)とする．)

$Sub(r)=D(T_r)/A(r)$

$Sup(r)=(D(T)-D(T_r))/A(r)$

$Par(r)=(D(T_{p(r)})-D(T_r))/(A(r)\cup A(p(r)))$

属性木上の祖先s_0から子孫s_kに至る道を$(s_0,s_1,\ldots,s_{k-1},s_k)$とすると，$s_0$と$s_k$の属性間の依存関係[3]DP($s_0$, s_k)は次式

$(Par(s_1)\cup\ldots\cup Par(s_{k-1}))/(A(s_0)\cup A(s_k))$

として求まる．属性木T上のノードの集合$X=\{x_1,\ldots,x_n\}$に対し，圧縮木Compress(X)=(CV,CE)を構成する．

CV=X∪{x_i, x_jのT上の最小の共通の祖先}

CE={(x,y) | xはCV上のy($x\neq y$)の最小の祖先}

とする．CT=Compress(X)(=CTと表す)の頂点をrt，葉の集合をLFで表す．CTの属性依存関係[3]DEP(CT)を

$$Sup(rt)\cup\Big(\bigcup_{z\in LF} Sub(z)\Big)\cup\Big(\bigcup_{(x,y)\in CE} DP(x,y)\Big).$$

と定める．以上の準備の下に，図4に属性評価アルゴリズム Propagate(a) の概要を示す．Treenode(a)は属性木T上で属性aが含まれているノードを表す．アルゴリズムではM上のノードbの入力次数が0の時，bをSに加える．b∈S∩Arrivedなら，bを評価し，Needed(a)に属すか否かを判定する．bがNeeded(a)に属する場合は関数ExpandDG(b)によって圧縮木を拡張し，属性をMに加える．このとき導入される属性の数はO(1)なので，M の最大のサイズは O(|Needed(a)|) となる．Propagate(a) の時間計算量は O((|Needed(a)|+|Change(T)|+1)*log n)となる．ここでn はT のノード数とする．

```
PROPAGATE(a)
  CT:=Compress(Change(T) ∪ {TreeNode(a)});
  M := DEP(CT);
  Arrived:=M上でaに到達可能なノードの集合;
  S := 入力次数が0であるMのノードの集合;
  while c∈Arrived∩Sが存在する do
    M, S, Arrived から c を削除する;
    c を評価する;
    if cの値が変更した then ExpandDG(c)
    while M上でcから出る枝(それを(c,d)とする)が
          存在する do
      辺 (c,d) をMから削除する
      if dの入力次数が0である then dをSに加える
    od
  od
  Change(T) := M上のノードの集合
end PROPAGATE
ExpandDG(b)
  while D(T)上に次の条件を満たすノードcが存在する
    [条件:辺(b,c)がD(T)上に存在し，かつ，c∉M] do
    TreeNode(c) を CT に加える;
    M := DEP(CT);
    Arrivedに属するあるノードへの道が存在するTreeN
    ode(c)の全ての属性をArrivedに加える;
    入力次数が0であるTreeNode(c)の属性をSに加える;
  od
end ExpandDG
```

図4　属性評価アルゴリズム

5．まとめ

今回提案したアルゴリズムは一般の属性文法に適用できる．更に，評価に要する計算時間を従来のものに比べて改善したものとなっている．なお，アルゴリズムの正当性などに関する詳細については別の機会に報告する予定である．

参考文献

[1]馮，他："属性文法による構造化分析法の形式的記述"，信学技報,COMP88-3, pp.21-30(1988).

[2]T.Reps, et al.:"Remote attribute updating for language-based editors", Proc. 13th ACM Symp. POPL, pp.1-13(1986).

[3]T.Reps, et al.:"Incremental context-dependent analysis for language-based editors",ACM Trans. Program Lang. Syst., 5, 3, pp.449-477(1983).

[4]片山卓也："属性文法に基づくソフトウェア自動生成システム構成法の研究"，昭63年文部省科研(一般研究(A))報告書(1989).