ソフトウェア基礎論　16－5
プログラミング言語　4－5
（1986. 2. 7）

属性評価における大域化可能属性の発見アルゴリズム

森山　孝男　・　佐々木　尚　・　片山　卓也
（東京工業大学・情報工学科）

1．まえがき

本報告では属性文法の評価器における大域的領域割り当て発見について議論を行う。すなわち属性生起の集合が与えられたときに、それが共通の大域的領域を共用できるかどうかを決定するアルゴリズムに基づいて領域割り当てを見い出す。属性文法から品質の高い属性評価器を生成するときに、導出木についている属性のインスタンスに適切に領域を割り当てる方法を見い出すことが重要な問題であることは広く知られている。非常に原始的な方法としては、各々の属性インスタンスごとに個々の領域を割り当てることが考えられる。しかしこの方法では属性評価の間じゅうに一回だけしか各領域が使用されず、望ましくない。多数の人が属性評価における効率的な領域の使用法について考えてきた。Saarinenは属性生起をsignificant なものとそうでないものに分け、significant な属性生起をスタックに割り当てることを考えた。片山[1] はすべての属性生起をスタックに入れることを提案した。

属性文法の意味規則はapplicative であるので、文法から直接的に評価器を実現すると、記号表のような大きいサイズのデータをコピーすることになり時間的にも領域的にも効率が悪くなる。このような大きいサイズのデータを効率よく取り扱うためには次のふたつのことを行わなければならない。第一は共通の領域を割り当てることができる属性のインスタンスを見い出すことである。第二はデータのアクセスの方法を値ではなく更新による方法に変更することである。すなわち値すべてを代えるかわりにデータの一部分を更新するのである。Farrowはこの問題の有用性を示している。このようにして、属性インスタンスに大域的領域を割り当てることによって、領域的にも時間的にも非常に効率的になることが期待される。

本論文では、与えられた属性生起の集合が共通の領域場所を占めることができるかどうかを決定する問題について考える。Sethi [2] は有限の大きさのdag （非循環有向グラフ）について領域を大域化する問題について考え、ある種のpebble game の概念を導入した。彼の方法は与えられたグラフが有限の大きさを持つときには有効であるが、我々は有限の大きさを持たない属性依存グラフを取り扱うので、属性評価については直接適用できない。彼もまた属性文法に対して提案を行っているが、それは評価戦略を考慮しておらず十分なものではない。

一般に属性の領域割り当て戦略は属性評価器の構造に大きく依存している。我々の考えている評価器は、非終端記号とその合成属性の組みに手続きを割り当てて属性文法を手続きの集合へと変換するという原理によって構成される。したがって評価器は本質的に再帰的であり、属性大域化テストの問題は生成規則レベルでの問題に帰着される。すなわち有限のdag についてのSethi のアルゴリズムを修正することによって問題が解決される[3] 。

本論文では、上の方法に基づき大域的領域割り当ての発見方法について考える。まずそのために、領域割り当て発見の発見的方法および効率化を考える。次にそれに基づいてアルゴリズムを実現する。

2．準備

2．1　属性文法の定義

属性文法は意味規則を持った文脈自由文法のことである。これは形式的には次のように定義される。

(G,A,F)

ここで3つ組みのなかは次のように定義される。

(1)　G=(VN,VT,P,S) は、VNを非終端記号の集合、VTを終端記号の集合、P を生成規則の集合、S を初期記号の集合とする文脈自由文法である。一般性を失うことなく初期記

号は生成規則の右辺にでてこないと仮定できる。 $w_i \in VT$、 $X_i \in VN$としたとき、生成規則p は

p: $X_0 \rightarrow w_0 X_1 w_1 \cdots w_{n-1} X_n w_n$

の形をしている。

(2) A は属性の集合である。各々の $X \in VN$は Aの部分集合 A[X]を持っており、A[X]の要素は Xの属性と呼ばれる。A[X]は、相続属性の集合INH[X]と合成属性の集合SYN[X]との合併集合である。INH[X]∩SYN[X]= ϕである。

(3) F は意味関数の集合である。 $a \in SYN[X_0]$ または $a \in INH[X_k]$ $(k=1...n)$となるような各々の属性生起 $v = a.X_k$ に対してひとつの意味関数fp,vが存在する。これは、規則p の他の属性生起から vの値をいかに計算するかを指定する。fp,vの引数の集合をDp,vと表しfp,vの依存集合と呼ぶ。もしDp,v= $\{ v_1, \ldots, v_m \}$ であればfp,vはdomain $(a_1) \times \ldots \times$domain$(a_m)$ からdomain(a) への写像である。ここでdomain(a) は属性a の定義域を表し、 $v_i = a_i.X k_i$ である。以上のことを

$$v = fp,v(v_1, \ldots, v_m)$$

と表す。

２．２ 属性依存関係

属性の依存関係を表す幾つかのグラフを定義していこう。

(1) p を生成規則とする。DGp =(DVp,DEp) は次のように定義される。

DVp= $\{ a.X_k \mid 0 \leq k \leq n$ かつ $a \in A[X_k] \}$

DEp= $\{(v,u) \mid v \in Dp,u\}$

(2) 導出木 Tが与えられたとき、DG[T] は次のように Tの構造に従ってDGp を幾つも貼り合わせて得られる。 Tの根に適応される規則をp とし、k 番目の部分木をTkとする。DG[T] はDG[T_1]...DG[Tn]とDGp から次のように作る。

DG[T]=(DV[T],DE[T])

DV[T]=DV'p∪ ∪DV[Tk]

DE[T]=DE'p∪ ∪DE[Tk]

ここで kは１から nまでの範囲を動く。さらに、DG'p=(DV'p,DE'p)はDGp より得られるグラフである。すなわち、Tk の根をr[k]とするときに（r[0]は Tの根とする） $a.X_k$ が $a.X_k.r[k]$ に対応するとして、

DV'p= $\{ a.X_k.r[k] \mid a.X_k \in DVp \}$

DE'p= $\{(a.X_k.r[k], b.X_j.r[j]) \mid (a.X_k, b.X_j) \in DEp \}$

上で導出木 Tの Xでラベルされている節 tの属性 aの生起をa.X.t と表した。これを Tにおける属性インスタンスと呼ぶ。

(3) sを導出木 Tの根 nの合成属性とする。 iを nの相続属性とする。このとき i.X.nから s.X.nへの道の上にないすべての辺と節をDG[T] から取り除いたグラフをDG[s.X.n, T] とする。

(4) 導出木 Tの根を Xとする。IO[X,T]=(A[X],Eio[T]) を次のように定義する。(a,s) ∈Eio[T]となるのは Xの属性 aとs に対応した vとu なる節について、v からu への道がDG[T] に存在するときそのときに限る。さて、IO(X) を {IO[X,T] | Tは Xを根とする導出木} とし、その要素をIOk=(A[X],Ek) とする。このとき、IO[X]=(A[X],Eio)をEio=∪Ekにより定義する。

(5) DGp* = (DGp* ,DEp*) を次のように定義する。

DGp* =DVp,

DEp* =DEp∪ $\{(a.X_k, b.X_k) \mid (a,b)$ は IO[X_k] の辺であり $1 \leq k \leq n\}$

(6) $i.X \in I = \{ i.X \mid (i,s) \in IO[X] \}$ からs.X へ至る道の上にない辺をDGp * から取り除いてできたグラフを

DGp* [s.X] = (DVp* [s.X], DEp* [s.X])

とする。

(7) すべての生成規則p について、 DGp* がサイクルを持たないとき、属性文法は絶対非循環であるという。

２．３ 属性評価器

絶対非循環である属性文法G に対しては、次のようにしてその属性評価器を構成できる。 $X \in VN$, $s \in SYN[X]$とする。組み s.Xについて手続きRs.X ($u_1, \ldots, u_m$,T ;v) を構成する。ここで、 $u_1, \ldots, u_m$ は $I = \{i \mid (i,s)$ はIO[X] の辺} に対応するパラメータであり、 Tは導出

木に対応するパラメータであり、 vは sに対応するパラメータである。この手続きはIから sを求めるためのものである。左辺記号 X_0 の相続属性 i_1 , ..., i_m の値が与えられたとき、 s_0 の値を次の手続き呼び出し文の実行により求める。

call Rs.X (u_1 ,...,u_m ,T;v)

ただしこのときの u_1 ,...,u_m は i_1 , ..., i_m の値のための変数であり、v は s_0 の値のための変数である。

手続きRs.X は次のようにして構成する。

```
procedure Rs.X (u1 ,...,um ,T;v)
  case production(T) of
     p 1 : H p1 ,s
     p 2 : H p2 ,s
        ...
        ...
  end
end
```

ここでHp,s は s∈SYN[X_0] の値を求めるための文の列である。production(T) は導出木T の根での生成規則を示す。Hp,s はさらに次のように属性の値を求めていく。x を生成規則p の中の属性生起とする。もしも xが右側非終端記号の相続属性もしくは左側非終端記号の合成属性であれば、意味関数を用いてその値を決定する。またもしも x が右側非終端記号の合成属性であれば、それについての手続きを呼び出すことにより、その値を決定する。

2．4 EVAL[G]-評価可能性

値の依存関係を表しているdag についての計算のモデルとしてpebblingが知られている。pebblingは値が失われないように計算を逐次化することを表している。D=(V, E) を有限のdag とする。D のpebblingとは条件

もし (v_i ,v_j) ∈E ならばi<j となる

を満足するような Vの節 vの列

v_1 ,v_2 ,...,v_n

である。

ここでは、我々の属性評価器の構造を考慮して、属性文法 Gに関してのpebblingを定式化する。T を Gの導出木とする。DG[T] のpebblingにおいて、属性インスタンスはDG[T] の依存関係により決められる順序に従って評価されていた。ここでは評価器が再帰的であることによりその順序がさらに制限される。

X∈VN、s ∈SYN[X]とする。DG[T] のなかでの s のインスタンスを評価することを考えよう。評価器が手続きRs.X を呼び出すと、その手続きRs.X はその左側非終端記号が Xであるような生成規則 pを選んでから、さらに一連の文を実行する。それら一連の文は、右側非終端記号 Yの相続／合成属性を評価しようとするかまたは左側非終端記号 Xの合成属性を評価しようとする。もし右側非終端記号 Zの合成属性 tを評価しようとするなら、それらに対して手続きRt.Z が呼び出される。

評価器はこのように構成されていることから、属性評価の順序に次のような制限を加える。属性 tのインスタンスを評価し始めたときには、終了するまでその評価にその評価器はかかりっきりになっている。すなわち、その評価のあいだは、他のいかなる属性インスタンスも tを評価するのに必要なものを除いては、評価することができないということである。このことを次のように定式化する。

[定義] 再帰的pebbling

その根 nが X∈VNでラベルされている導出木を T とし、 p: X→ w_0 X_1 w_1 X_2 ... X_m w_m , (X_i ∈VN, w_i ∈VT*) がその根で適用されているとする。 s ∈SYN[X]についてs.X.n のためのDG[T] の部分グラフである依存グラフDG[s.X.n,T] についてのpebbling C[s.X.n,T] について考える。 DGp* においてs.X が依存している合成属性生起を r.X_i ,t.X_j ... とし、 T_i , T_j ... はその根がそれぞれ n_i , n_j ... である Tの部分木と仮定する。またC[r.X_i .n_i ,T_i], C[t.X_j .n_j ,T_j] ... は DG[r.X_i .n_i ,T_i], DG[t.X_j ,T_j], ...それぞれのpebblingであるとする。また、 source(DG[s.X.n,T])= {i.X.n |(i,s) ∈IO[X]}とする。このとき C[s.X.n,T] が再帰

的pebblingであるというのは、次の条件が満足されることである。

（1）もし Tが空木であるならば、C[s.X.n,T]は空列である。

（2） C[s.X.n,T] は、DG[s.X.n,T] についてのpebblingである。

（3）source(DG[s.X.n,T])= { w_1 ,w_2 ,...,w_h } とする。このとき、ある { i_1 ,..., i_h } = { 1,...,h }について、wi_1 ,..., wi_h は C[s.X.n,T] の先頭の列である。

（4）(4a) C[r.X_i .n_i ,T_i], C[t.X_j .n_j ,T_j] ... はみな再帰的pebblingである。(4b) C[s.X.n,T] は C[r.X_i .n_i ,T_i]',C[t.X_j .n_j ,T_j]'... を部分列として含んでいる。ここでC[s.X.n,T]' は C[s.X.n,T] において v∈ source(DG[s.X.n,T])∪ {s.X.n}なるすべての節 v を取り除くことによって得られた列とする。

文法Gが 2.3節の評価器EVAL[G] により評価可能であることを定式化する。

[定義] GはEVAL[G]-評価可能

その根 nが X∈VNでラベルされている導出木を T とする。任意の X∈VN,s∈SYN[X],Tについて、次の条件が成立するとき、属性文法GはEVAL[G]-評価可能であるという。

（1）DG[s.X.n,T] についてのpebblingC[s.X.n,T] は再帰的pebblingである。

（2）T1とT2を、それらの根で同じ生成規則が適用されている任意の導出木とする。このとき、

C#[s.X,T1]=C#[s.X,T2]

が成立する。ここでC#[s.X,T] は C[s.X.n,T] において (a)部分列 C[r.X_i .n_i ,T_i]',C[t.X_j .n_j ,T_j]'... すべてを対応する r.X_i ,t.X_j ,...に入れ替え、(b)その他の a.X_k .vなる属性インスタンスは a.X_k に入れ替えることにより得られた列とする。

上の条件(2) は、任意の導出木T について、T の根での生成規則p の属性生起の評価順序は同じであることを述べている。

3．属性文法における大域的領域割り当て

3．1 大域的領域割り当てとpebbling

Sethi は大域的な領域を用いた計算のモデルとしてdag についてのpebblingを導入した。そのpebblingは大域的領域に入っている値が失われないように計算を逐次化することが本質である。D=(V,E) を有限のdag とし、その節は集合Lの中の領域場所によりラベル付けされるとする。この領域場所を

g : V → L

により表す。すなわちg(v)は節v に割り当てられる領域場所を示している。D のSethi のpebblingとは以下の条件を満足するようなV の節の列

v_1 ,v_2 ,...,v_n

である。

（1）もし（ v_i ,v_j ）∈E ならばi<j となる。

（2）すべてのｉ＝１，．．．，ｎについて，もしi<j かつ（ v_i ,v_j ）∈E であるようなｊが存在すれば、i<k<j かつg(v_k) = g(v_i) となるようなｋは存在しない。

Sethi は与えられた(D,g,L) についてSethi のpebblingが存在するかどうかを決めるアルゴリズムを与えた。

3．2 属性文法におけるSethi のpebbling

今から属性文法についてのSethi のpebblingについて考える。属性文法G の導出木をT とする。T の中での属性インスタンスについての依存関係を表すグラフはDG[T] であった。そこでDG[T] の中の幾つかの属性インスタンスに大域的領域を割り当てることを考える。すなわち(DG[T],g,L) についてのpebblingを考える。DG[T] の節をDV[T] とすると、ラベル付け関数g は

g : DV[T] → L

である。node(T) を導出木Tの節の集合とすると、DV[T] は A×VN×node(T) の部分集合である。この方式では、規

則p の属性生起a.X に割り当てられる領域が Tの個々の節により異なってもよく、 Tの節に依存しているといえる。我々は前節で述べたように属性評価を行うので、評価器は導出される個々の木について独立である構造を持つ。したがって上のことにさらに条件を加える。すなわち、我々の領域割り当て関数は、導出木T に現れる属性生起に対して決まったひとつの領域を割り当てることにする。それで我々の gは

$$g : A \times VN \rightarrow L$$

となり、各T に対しては

$$g^* : A \times VN \times node(T) \rightarrow L$$

である g* にもとづいてpebblingを行う。ここで

すべてのv∈node(T) に対して

$$g^*(a.X.v) = g(a.X)$$

である。つまり X∈VNと a∈A[X]を持つDG[T] の節の各々は、もし共有されるならば、共有領域場所g(a.X)が与えられる。(DG[T],g* ,L) についてのpebblingは、大域的領域を用いてT の属性インスタンスを評価する順序を決めていることになる。このpebblingは、我々が考える特別なクラスの評価器のためには、まだ十分とは言えない。なぜなら、その評価器は属性インスタンス評価の順序にさらに 2.4節の制限を加えたものだからである。

4．属性文法Gの (EVAL[G],g,L)- 評価可能性

この節では、さきの2節と3節に基づいて属性文法のあるクラスを定義する。すなわち、(1) 属性評価器の構成法と(2) 領域割り当ての構成方法によりそのクラスの特徴付けを行う。

[定義] 属性文法G の (EVAL[G],g,L)- 評価可能性

Gを属性文法とする。 gを領域割り当て関数、 Lを領域の集合、EVAL[G] を2節で構成される評価器とする。GはEVAL[G]-評価可能であるとする。このとき、次の条件が、任意の X∈VN、任意の s∈SYN[X]と Xでラベルされている nを根に持つ任意の導出木 Tについて成立するときに限って、 Gは (EVAL[G],g, L)-評価可能であるという。

(DG[s.X.n,T], g * ,L) についての再帰的pebbling C[s.X.n,T] はSethi の意味でのpebblingである。

5．(EVAL[G], g, L) －評価可能性の判定法

この節では属性文法が (EVAL[G],g,L)- 評価可能であるかを調べる方法について述べる。判定は生成規則レベルに帰着されることが知られている。すなわち、以下に定義する DGp° [s.X] なるグラフについての問題に還元される。

生成規則 p : X → w_0 X_1 w_1 X_2 ... X_n w_n と s∈SYN[X]について、拡張された依存グラフ DGp° [s.X] を定義する。

[定義]

DGp° [s.X] = (DVp° [s.X], DEp° [s.X])

DVp° [s.X] = DVp* [s.X] ∪ {[t.X_k] | k=1,..,n, s∈SYN[X_k] }

DEp° [s.X] = DEp* [s.X] ∪ {(i.X_k ,[t.X_k]) | k=1,...,n, (i,t)∈IO[X_k] } ∪ {([t.X_k], t.X_k) | k=1,...,n, t∈SYN[X_k] } − {(i.X_k ,t.X_k) | k=1, ...,n, (i,t)∈IO[X_k] }

とする。上の[t.X_k] なる記号は、根が X_k でラベルされている部分木 T_k における tの計算を表現するために導入した仮想的な節である。

(EVAL[G],g,L)-評価可能であることの必要十分条件は、上のグラフの DGp° [s.X] すべてについて、あるグラフが存在しており (1)同じ領域を持つ節はすべて全順序を付けられており、 (2)あるグラフ変換規則について閉じており、さらに (3)サイクルを持たないことが成立することである。以下、この条件を詳しく述べる。

5．1 EXDGp° [s.X]

まず、 DGp° [s.X] についての領域割り当て関数 g° p[s.X]を考える。節[t.X_k] について、 X_k から導出される木 T_k における属性評価において使用されるかもしれない領域を割り当てる。一般的に T_k においては複数個

の領域が使用されるので、[t.X_k] には Lの部分集合 USE [t.X_k] を割り当てなければならない。 g° p[s.X]: DVp° [s.X] →2L （2L は Lのべき集合）を v∈ DVp° [s.X] について次のように定義する。

[定義]

g° p[s.X](v) =

{ g(v) }　　v≠[t.X_k] のとき

USE[t.X_k]　v=[t.X_k] のとき

なお g° p[s.X]は gが与えられると唯一つに定まる。

g° p[s.X]により同じ領域を割り当てられた節は、かならず順序付けられなければならないこと、すなわち一つの領域には一時期に一つの値しか記憶しておくことができないを表現するために、次のグラフを定義する。

[定義]

EXDGp° [s.X] は DGp° [s.X] のサイクルを持たない２項関係としての拡大で、次のように定義する。

EXDGp° [s.X] = (EXDVp° [s.X], EXDEp° [s.X])

EXDVp° [s.X] =　DVp° [s.X],

EVDEp° [s.X] =　DEp° [s.X] ∪ {(x,y) または(y,x) | g° p[s.X](x) ∩ g° p[s.X](y) ≠φ} ∪ {(x,y) |x,a ∈ {i.X | (i,s)∈IO[X], i.X∈ DVp° [s.X]}, (a,y) ∈ DEp° [s.X] }

ここで、後者の辺の集合は、評価器を構成する手続きの引数すべてがそろってから評価が実行されることを表している。

また、このような EXDGp° [s.X] は DGp° [s.X] について数多く存在しているかもしれない。ここではそのうちのひとつを EXDGp° [s.X] で表現している。

５．２　変換規則→

EXDGp° [s.X] の節の２項関係としての拡大には次のグラフの変換規則が適用される。

[定義]　Rを EXDGp° [s.X] の節の上の二項関係の拡大とする。→はRの変換規則であり、次のように定義する。

(1)　x,y,zは異なる節であり、 (x,z)∈ DEp° [s.X] であり、さらに g° p[s.X](x) ∩ g° p[s.X](y) ≠φであるとする。このとき、

$\rightarrow_e$ ：R→$_e$ R ∪ {(y,x) | (y,z) ∈R}

→L ：R→L R ∪ {(z,y) | (x,y) ∈R}

(2) また、 t.X_k を右非終端記号 X_k の合成属性 tの生起とし、 vを任意の節とする。このとき、

$\rightarrow_a$ ：R→$_a$ R ∪ {(t.X_k ,v) | ([t.X_k],v)∈R}

$\rightarrow_b$ ：R→$_b$ R ∪ {(v,[t.X_k]) | (v,t.X_k) ∈R}

(3) またx,y,z を異なる節とすると、

$\rightarrow_t$ ：R→$_t$ R ∪ {(x,z) | (x,y) ∈R,(y,z)∈R} と定義する。このときQ→RはQ→$_e$ RまたはQ→L RまたはQ→$_a$ RまたはQ→$_b$ RまたはQ→$_t$ Rであるとき、またそのときに限る。

R≠QでありQ→Rが偽であるそのときに限り、Qをexclusion closedであるという。変換→はChurch-Rosser propertyを持つ。Q→* RかつRがexclusion closedであるときRをQ* と書いて、Qのexclusion closure と呼ぶ。

５．３　判定定理

最終的に次の定理のかたちで判定方法が述べられる。

[定理]

Gを属性文法とする。さらに、 Lを領域の集合とし、 gを領域割り当て関数、EVAL[G] を２節で構成される評価器とする。このとき、

Gは (EVAL[G],g,L)- 評価可能

←→

すべての pとすべてのs.X について、ある EXDGp° [s.X] が存在して EXDGp° [s.X] * がサイクルを持たない

ここで、生成規則p の左側非終端記号 Xの合成属性を sとする。

６．大域的領域割り当ての自動発見

前章までで示された判定定理をもとにして、与えられた属性文法から大域的領域の割り当てを自動的に捜し

出すアルゴリズムを考える。基本的には２段階の検索が考えられる。ひとつは、大域的領域の割り当て関数 gについての検索、もうひとつは、特定の gについて (EVAL[G],g,L)- 評価可能かどうかを判定するための探索である。すなわち、 EXDGp°[s.X]* がサイクルを持たないような EXDGp°[s.X] を実際に見い出す探索である。

６．１　アルゴリズム概略

（１）属性文法を入力し属性依存グラフ DGpを作る。

（２）拡張依存グラフ DGp* を作る。

（３） DGp*[s.X] を作る。

（４）最適なg を求めるため大域的領域割り当て関数g について以下のサーチをする。

（５）g からUSE[s.X]を求める。

（６）USE[s.X]とDGp *[s.X] から、大域的領域割り当て g° p[s.X]付きのグラフ DGp°[s.X] を作る。

（７）すべての DGp°[s.X] について以下を行い、それらがすべて成功すればそのg は成功である。

（８） DGp°[s.X] に対して以下を満たすようなEXをサーチする。

（９）EXと DGp°[s.X] から EXDGp°[s.X] を作る。

（１０）変換規則→を適用してexclusion closure EXDGp°[s.X] 作る。

（１１）そのexclusion closure がサイクルを持たなければ良い。

７．領域割り当て発見のための種々のヒューリステックス

我々は、前節の定理に基づいて、領域割り当て関数 gの中で効率の良いものを発見したい。このとき、 gの発見について２段階の探索が存在する。ひとつは、最適の gを発見するために gそのものについて探索することであり、もう一つはEXを探索することである。以降これらの探索について効率化を考える。また、最適な gの発見についてその他の効率化についても考える。

７．１　領域割り当て関数 gの発見法

すべての領域割り当て関数 gを生成して、それが実際可能かどうかを調べる方法は非常に素朴なものである。しかしこれでは、対象としている属性非終端記号の組みが増えるにしたがって、調べる回数が爆発的に増加してしまう。ここでは、探索回数を減らすことを考える。

a.X （ a∈SYN[X]、 X∈VN）についていかなる大域的領域 mも割り当てが不可能であると仮定する。このとき、他のいかなる b.Y（ b∈SYN[Y]、 Y∈VN）に大域的領域 mの割り当てが可能であるとしても、 a.Xとb.Y にともに同じ大域的領域 mを割り当てることは不可能である。すなわち、はじめから大域的領域割り当てが不可能なa.X に対しては、いかに大域的領域割り当て関数g を工夫してみても、a.X を大域化して割り当てを行うことができないのである。したがって、このように最初からそれ自身で大域化できないa.X については、大域的領域を割り当てないように除いて考えることが効率的である。そこで、各g についてひとつひとつ大域的領域割り当て可能かどうかを直接に判定するのではなく、まず前処理として、それ自身で大域化可能である属性非終端記号の組みを見い出すことにする。次にそれら可能な組みについて同じ大域的領域を割り当てる組み合わせを考えて大域的領域割り当てが可能であるかを調べる。

次に探索のための評価関数を考える。大域化できた属性非終端記号。組みの数 | {s.X | g(s.X =m ∈L} | が多ければ多いほど評価器の効率は良くなる　また大域的領域の個数 | Im(g) | がすくなければすくないほど、評価器は効率がよい。しかし実際に大域的領域割り当て関数 g を効率の良い評価器を構成するように決定しようとすると、上の両者にはtrade off が存在する。したがってどちらの指標が効率により影響を与えるかについて考えなければならない。

結論としては大域化できた属性非終端記号の組みの多さのほうがより多く効率に影響を与える。理由は次の

とうりである。大域的領域２個で割り当て可能であるものを大域的領域１個の領域を用いて割り当て可能であるものに効率化しても、領域としては１個の節約にしかならない。これに対して、s.X が大域的になれば、導出木中で存在するすべてのs.X に対して１個の領域ですむ。したがって、後者のほうがより効率的である。このことを考慮して評価関数を作る。すべての属性非終端記号の組みの数をN とする。大域的領域割り当て関数が与えられたとき、大域化される属性非終端記号組みの数をn1、使用する大域的領域の数をn2とする。このとき、

$$f(g) = N * n1 - n2$$

を評価関数とする。この評価関数の値が大きいほどよい。なお、 N>n1であるから、これはn1がn2に優先して調べられることを意味している。

７．２　EXについての探索法

ある特定のg を用いて割り当てが可能かどうかを調べるときに、同じ領域を持つ節についてはかならず全順序が与えられている EXDGp°[s.X] を考える必要があった。このような全順序を与える関数をEXとする。すなわち、 DGp°[s.X] から EXDGp°[s.X] を求める関数をEXとする。したがって、g が可能な割り当てであることを判定するときには、 EXDGp°[s.X] * にサイクルを生じさせないようなEXをひとつ見い出す必要がある。最も簡単な方法としては、すべての場合についてEXを調べることがある。しかしこれでは効率が良くないので徐々にEXを構成していくことを考える。

EXは次のようにして構成される。(1) 大域的領域すべてについて一度に全順序を与えず、ひとつの大域的領域について全順序を与える。(2) 同じ領域を割り当てられる節すべてについても始めから一度に全順序を与えず、ひとつひとつの領域を割り当てられる節について徐々に全順序を与える。

次にEX発見の効率化について考える。EXを与える前では完全な判定はできないが、以下に述べるように判定を行いEX発見の効率化を行うことができる。まず、EXを与える前に変換規則を適用して、その段階でのexclusion closure を求めることができる。その段階でのexclusion closure がもしもサイクルを持っていれば、いかなるEXを考えても最終的にはサイクルが生じてしまうので、現在の大域的領域割り当て関数については大域化が不可能であることがわかる。またサイクルが生じない場合には、この段階で加わった辺はいかなるEXを考えても付け加わるので、その現段階でのexclusion closure を出発点としてEXを徐々に構成することができる。

７．３　g° p[s.X]を求める方法

領域割り当て関数g が与えられると、 X∈VN,s∈SYN[X]の組みs.X についてUSE[s.X]を求め g° p[s.X]を求める必要がある。最も簡単な方法では、g が変更されるごとにこのUSE[s.X]を求めることが考えられる。しかし gは 4.2節のような性質を持っているものだけを考えていたので、g の変更に対していちいちUSE[s.X]を求めなくてもよい。そのかわりに与えられた属性文法について一度だけデータフロー解析を行うだけでよい。

次のようにしてUSE[s.X]を求めることができる。s.X について、まず Y∈VN,a∈SYN[Y]の組みa.Y がs.X を求めるのに必要（すなわちs.X のインスタンスs.X.n を求めるための依存グラフDG[s.X.n,T] の中にa.Y のインスタンスa.Y.m が存在するような導出木T が存在する可能性がある）であるかを調べる。そのようなa.Y の集合をuseattr(s.X)によりあらわす。次に各領域割り当て関数に応じてuseattr(s.X)をもとにしてUSE[s.X]を求めることにする。すなわち、USE[s.X]= {q | g(a.Y)=q ∧ a.Y∈useattr(s.X)} であるので、USE[s.X]を求めるためにはuseattr(s.X)に属しているすべての属性と非終端記号の組みについて gよりその領域を求めそれらの和集合をとればよい。

８実行例

例として、図のような簡単な属性文法を考える。これは、2 進小数の値とその桁数を求めるものである。これについて、大域的領域割り当て関数の自動生成をおこな

った。

```
Vn= {F, L, B}
Vt= {'.', '0', '1'}
INH[F]= φ     SYN[F]= {val,len}
INH[L]= {pos} SYN[L]= {val,len}
INH[B]= {pos} SYN[B]= {val}
p0: F -> '.' L
    pos.L = 1
    val.F = val.L
    len.F  len.L
p1: L -> B
    pos.B = pos.L
    val.L = val.B
    len.L = pos.L
p2: L0 -> B L1
    pos.B = pos.L0
    pos.L1 = pos.L0 + 1
    val.L0 = val.B + val.L1
    len.L0 = len.L1
p3: B -> '0'
    val.B = 0
p4: B -> 1
    val.B = 2^(-pos.B)
```

この場合、属性非終端記号の組としてはval.F, len.F, pos.L, val.L, len.L, pos.B, val.B の７個がある。前章までのアルゴリズムを実行した結果、val.F, len.F, pos.L, val.L, len.L, pos.Bにすべて同一の大域的領域を割り当てることが可能であることがわかった。実行時間はCPU としてMC68010(10MHz)を使用して４秒ほどである。

その場合の属性評価器は以下のようになる。

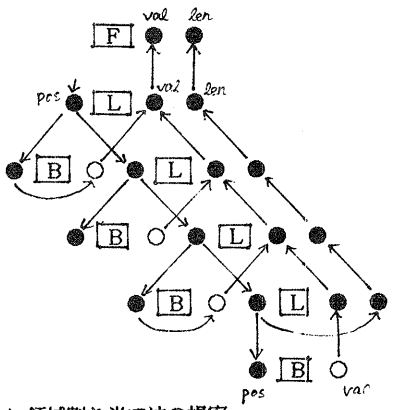
```
procedure evaluator for G1
    var glob
    procedure Rval.L(T;)
        var val.B
        case production(T) of
        p1: call Rval.B(T1; val.B)
            glob <- val.B
        p2: call Rval.B(T1; val.B)
            glob <- glob + 1
            call Rval.L(T2;)
            glob <- val.B + glob
        end
    end;

    procedure Rval.B(T; v)
        case production(T) of
        p3: v <- 0
        p4: v <- 2^(-glob)
        end
    end

    procedure Rlen.L(T;)
        case production(T) of
        p2: call Rlen.L(T2;)
        end
    end

    T <- initial value of T
    glob <- 0
    call Rval.L(T;)
    write(glob)
    glob <- 0
    call Rlen.L(T;)
    write(glob)
end
```

入力文字列".1010" に対する計算木を示すと図のようになる。このうち黒丸●で表された属性のインスタンスはすべて同一の大域的領域を使用し、白丸○はすべてスタック上の個々の領域が使用される。従って、すべての属性インスタンス２２個を５個の領域で計算できることがわかる。



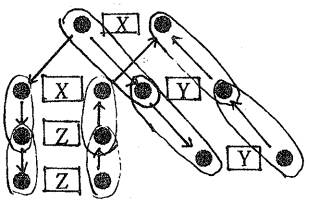
９．新しい領域割り当て法の提案

以上の論議で扱われてきた領域割り当ての方法では、もしもa.X が大域化が可能であればa.X のインスタンスはすべて一つの大域的領域を使用するので大幅に空間効率を上げることができた。しかし、この大域化の方法はかなり粗い大域化の方法であるので、あるa.X を大域化したために他のもっとコストの高い属性が大域化されるのを阻んでしまう、といったことも十分に考えられる。

そこで、ここでは属性を完全に大域化するのではなしに、もうすこし狭い範囲での領域の共有化を行うためにその記憶管理方法をを多少変更する。導出木が与えられる前にデータフロー解析を行ってルール内での属性評価順序を決定してしまうのは２．４章の属性評価器と同じであるので、大域化の判定法はこれまでの方法とほとんど同じで良い。

これまでの属性評価器では、手続き中から他の手

続きを呼び出す場合、子供の非終端記号の属性の領域は、手続きの中でスタック上に確保された領域又は大域的領域（この場合は引数として明示的に渡す必要は無い）が使用されていた。属性が大域化されていない場合には親の非終端記号の属性の領域は、その手続きを呼び出す側で用意することになる。ここで新たに提案する方法は、子供の非終端記号の属性の領域として、スタック内に確保された領域又は親の非終端記号の属性の領域（この領域も、そのまた親の手続きから渡されたものである）を使用するものである。この領域の大域化（共有化）はルール内だけのローカルなものであるが、そのような属性の共有化が積み重なってゆく事によって広範囲な大域化が達成できる。また、この大域化はこれまでのように導出木全体に及ぶものではなく、適当なところで大域化の範囲を制限できるので、全体的にみて大域化の可能性を高めることができる。その様子を図に示す。



この方法ではルールごとに領域割り当てを決めることができるため、属性の依存関係を参考にして効率良く領域割り当てを生成できるという特徴がある。さらに、トランスファー属性を考慮してUSE[s.X]を求めることによって大域化の度合いを高めることが出来る。

１０．むすび

属性評価において、大域的領域割り当ての自動生成を行った。対象となった文法はまだ簡単なものであり、実用的なものについては現在のところ実際的な時間で生成できない。このために次のことが考えられる。(1) さらに効率化を行うための発見的手法を考える。(2) 多項式時間内に解ける領域割り当てのクラスを発見する。(3) 最適ではないが、近似的に解を求めることにより現実的な時間内に領域割り当てを発見する。

また、大域化可能な属性の範囲を広げるために、ローカルな属性の共有化を積みかさねることにより全体的な属性大域化を達成する形の属性評価器の提案を行った。

今後は同時評価可能な合成属性の集合に対して一つの手続きを構成し、属性評価の効率をさらに高めることを考えている。

［参考文献］

[1] Katayama,T.: Transration of Attribute Grammarsin to Procedures, ACM Transaction on Programming Languages and Systems, Vol.6, No.2, pp345-369 (1984).

[2] Sethi,R., Pebble games for studying storage sharing. Theoretical Computer Science Vol.19, No.1, pp.69-84, July 1982.

[3] Katayama,T.,Sakaki,H.: Global Storage Allocation in Attribute Evaluation, Conference Record of the 13th Annual ACM Symposium on Principles of Programming Languages, pp.26-37 (1986).