増進的複製による型付き素性構造差分計算手法

小暮 潔†

本論文は増進的複製による型付き素性構造の差分演算の計算手法を提案する．制約に基づく言語理論で使用される型付き素性構造の集合には，表す情報の包含関係を示す半順序関係が定義され，この半順序関係から2つの束演算—最小上界である汎化と最大下界である単一化—が定義される．差分演算は，表す情報の間に包含関係がある2つの型付き素性構造に対して，それらの情報の差を表す—すなわち，情報の少ない方との単一化が情報の多い方を与える—型付き素性構造で，極小情報の型付き素性構造の中の特定の1つを得る演算である．差分演算は単一化や汎化よりも素性アドレス集合，型記号，共参照関係が入出力で複雑な関係になる．そこで，差分を計算するために必要な有向グラフ操作技法を開発した．差分演算は，言語情報記述の階層化による記述の効率化，情報の共有や分割による言語処理の効率化などを行う際に有用である．

Typed Feature Structure Difference Calculation with Incremental Copying

KIYOSHI KOGURE†

A difference calculation algorithm with incremental copying has been developed for typed feature structures. The set of typed feature structures has a partial ordering denoting the subsumption relationship between the information they capture. For any typed feature structures t_1 and t_2 such that t_1 is more informative than t_2, the difference operation gives one of the minimally informative typed feature structures t such that the unification of t and t_2 yields t_1. Between the input and output typed feature structures of this operation their sets of feature-addresses, their type symbol assignments, and their coreference relations are related in much more complex manners than those of unification and generalization operations are. Graph manipulation techniques required for calculating this operation have thus been devised. The difference operation has such useful applications in natural language processing using typed feature structures as hierarchization of linguistic descriptions and information sharing and splitting, which makes natural language processing more efficient.

1. はじめに

制約に基づく言語理論[2),11),12)]は言語的対象の記述に素性論理を用い，素性論理はモデルとして素性構造や型付き素性構造を用いる☆．論理式のモデル，すなわち，素性構造が有限に表現できるとき，論理式上の定理証明をモデル演算に置換できることがあり，このようなときにモデル演算の効率的アルゴリズムが存在するならば，モデル演算が有利であることが期待できる[12)]．モデルである素性構造の有用な演算として，情報の包含関係の半順序関係に基づく束演算である単一化と汎化が定義されている．

素性構造には上記の半順序関係に基づき，素性構造の持つ情報の差を抽出する差分も定義される[9)]．この演算は，表す情報の間に包含関係がある2つの素性構造に対して，それらの情報の差を表す素性構造を与える．換言すると，素性構造 t_1 とそれ以下の情報を表す素性構造 t_2 に対して，t_2 との単一化が t_1 になるような極小情報の素性構造の特定の1つを与える．差分演算には素性構造を用いた言語処理で様々な応用がある．言語情報記述の階層化による記述の効率化，情報の共有や分割による言語処理の効率化などである☆☆．

本論文は型付き素性構造の差分演算の増進的複製による計算手法を提案する．差分は単一化や汎化

† 日本電信電話株式会社基礎研究所
NTT Basic Research Laboratories

☆ 本章では特に断らないかぎり，「素性構造」で通常の素性構造と型付き素性構造の双方を示す．

☆☆ 例えば，2つの語義を持つ曖昧な語に関して，語義それぞれに関する素性構造 t_1 と t_2 があるとする．この2つの素性構造が共通に持つ情報を表す素性構造 t_3 を汎化演算で計算し，t_1，t_2 と t_3 の表す情報の差を表す素性構造 t_1'，t_2' を差分演算で計算することにより，'t_3 and (t_1' or t_2')' の形式の記述が得られる．このような記述により効率的な選言的素性記述単一化法[5)]を有効利用できる．

より入出力の関係が複雑で，単一化や汎化の計算手法[1),4),6),8),10),13),14)]の単純な修正では差分結果が得られないので，必要なグラフ操作技法を開発した．型付き素性構造は通常の素性構造の拡張であるから，本論文の手法は通常の素性構造にも適用される．

本論文では，第2章で型付き素性構造とその上の演算を簡単に説明し，第3章で計算手法を提案する．

2. 型付き素性構造と汎化，単一化，差分

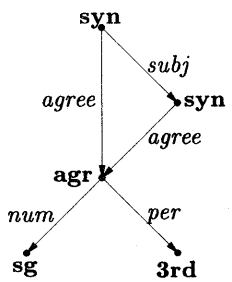
型付き素性構造は簡潔な表現を可能にするための素性構造の拡張である．型付き素性構造は対象の種類を表す型記号と，対象の側面を表す素性—素性記号と値の対—の集合から構成され，この値も型付き素性構造である．型記号は型付き素性構造で表そうとする対象の集合を示す．型記号の集合 $\mathcal{T}$ は，この対象の集合の包含関係に関する半順序関係 $\leq_{\mathcal{T}}$ を持つ．$\mathbf{a}, \mathbf{b} \in \mathcal{T}$ は，$\mathbf{a}$ の示す集合が $\mathbf{b}$ の示すものに包含されるとき，$\mathbf{a} \leq_{\mathcal{T}} \mathbf{b}$ である．$\mathcal{T}$ には，$\leq_{\mathcal{T}}$ に関して最大の要素 $\top$ と最小の要素 $\bot$ が存在する．$\top$ は無情報を意味し，$\bot$ は矛盾を意味する．また，$\mathcal{T}$ は $\leq_{\mathcal{T}}$ に基づく Brouwer 束とする☆．すなわち，任意の $\mathbf{a}, \mathbf{b} \in \mathcal{T}$ に関して，最小上界 $\mathbf{a} \vee_{\mathcal{T}} \mathbf{b}$，最大下界 $\mathbf{a} \wedge_{\mathcal{T}} \mathbf{b}$，$\mathbf{c} \wedge_{\mathcal{T}} \mathbf{b} \leq_{\mathcal{T}} \mathbf{a}$ である最大の元 $\mathbf{c} = \mathbf{b} \rightarrow_{\mathcal{T}} \mathbf{a}$ が存在する．

型付き素性構造は図1(a)のようなマトリクスで示される．図の構造は型記号 **syn** と [ ] 間の素性を持つ．素性がない場合は後の部分を省略する．埋め込まれた部分構造の指定に有限素性記号列である素性アドレスを用いる．長さ0の素性アドレスを ϵ で，素性記号列の結合演算子を "·" で示す．型記号 **agr** の前の X はタグ記号である．同一タグ記号の素性アドレス値は同一で，このような素性アドレスは共参照関係にある，あるいは，共参照すると言う．例えば，図の構造では *agree* と *subj* · *agree* が共参照関係にある．素性アドレスは自分と自明な共参照関係にある．型付き素性構造で素性アドレス p と q が共参照関係にあり，かつ，空列でない r に関して，$p \cdot r$ がこの構造に含まれるとき，$q \cdot r$ もこの構造に含まれ，$p \cdot r$ と $q \cdot r$ が真の接頭語（すなわち，自分以外の接頭語）間の関係により必然的な共参照関係にある．これを共参照関係の右不変性と言う．ゆえに，図の構造は素性アドレス集合 {ϵ,*agree*,*agree* · *num*,*agree* · *per*,*subj*,*subj* · *agree*,*subj* · *agree* · *num*,*subj* · *agree* · *per*} を持つ．

型付き素性構造は図1(b)のような有向グラフでも示される．型付き素性構造は型記号（とタグ記号）の

$$\mathbf{syn}\begin{bmatrix} agree: & \mathsf{X}:\mathbf{agr}\begin{bmatrix} num: & \mathbf{sg} \\ per: & \mathbf{3rd} \end{bmatrix} \\ subj: & \mathbf{syn}\begin{bmatrix} agree: & \mathsf{X} \end{bmatrix} \end{bmatrix}$$

(a)



(b)

図1 型付き素性構造の例
Fig. 1 Example of a typed feature structure.

ラベルを持つ節点で，素性は素性記号のラベルを持つ弧である．共参照素性アドレス値は同一節点である☆☆．

型付き素性構造の集合は型記号集合の半順序関係を拡張した半順序関係を持つ．型付き素性構造 t_1 は矛盾（型記号$\bot$）を含むか，あるいは，次の条件を満足するとき，かつ，そのときにかぎり，t_2 以下であると言い，$t_1 \leq_t t_2$ と書く．すなわち，(1) t_1 の型記号が t_2 の型記号以下で，(2) t_2 に存在する各素性 f が t_1 に存在し，t_1の値 $t_{1,f}$ が t_2 の値 $t_{2,f}$ 以下であり，かつ，(3) t_2 の各共参照関係が t_1 で成り立つ．関係 $\leq_t$ では，表す情報が多い型付き素性構造ほど小さくなる．関係 $\leq_t$ に基づき，t_1 と t_2 の最小上界 $t_1 \vee_t t_2$ と最大下界 $t_1 \wedge_t t_2$ が定義される．最小上界は引数が共通に持つ情報を抽出する汎化で，最大下界は両方の引数の持つ情報を集積する単一化である．

型付き素性構造の差分演算は型付き素性構造の表す情報の差を得る演算である[9)]．無矛盾で循環構造を含まない型付き素性構造 t_1，t_2 で，$t_1 \leq_t t_2$ であるものに関して，差分 $t = t_2 \rightarrow_t t_1$ は $t \wedge_t t_2 = t_1$ を満足する $\leq_t$ に関して極大な型付き素性構造で，特定の素性アドレスに情報が最も集中するものである．

単一化結果の共参照関係が入力の共参照関係の和を推移的かつ右不変的になるようにする最小の拡張であることなどから，極大な型付き素性構造が複数存在することがあるが，最後の条件で一意に決定される．この条件は以下で定義される半順序関係 $\preceq$ で形式化される．素性記号集合上の優先関係を示す全順序関係を $\leq_{\mathcal{F}}$，これの辞書式順序である素性アドレス集合の優

☆ このような束の構成方法に関しては文献7)などを参照．

☆☆ 節点は素性アドレスの共参照同値類を表すとも見なせる．

先関係を $\leq_{\mathcal{F}^*}$ とする．型付き素性構造 t_1, t_2 の素性アドレス集合の和の要素を $\leq_{\mathcal{F}^*}$ の昇順に並べたものを $p_1, p_2, \ldots, p_n$ とする．$i=1,2$, $1 \leq j \leq n$ に関して，$\mathbf{a}_{i,j}$ を t_i の p_j での型記号，$S_{i,j}$ を t_i の p_j と共参照関係にある素性アドレスの集合とする．ただし，t_i が p_j を含まないときには，$\mathbf{a}_{i,j}=\top$, $S_{i,j}=\emptyset$ とする．型記号と素性アドレス集合の対に関する優先順序 $\preceq'$ を次のように定義する．任意の型記号 $\mathbf{a}_1$, $\mathbf{a}_2$, 素性アドレス集合 S_1, S_2 に関して，$\mathbf{a}_1 \leq_{\mathcal{T}} \mathbf{a}_2$, かつ，$S_2 \subseteq S_1$ であるとき，かつ，そのときにかぎり，$\langle \mathbf{a}_1, S_1\rangle \preceq' \langle \mathbf{a}_2, S_2\rangle$ である．このとき，列 $\langle \mathbf{a}_{1,1}, S_{1,1}\rangle$, $\langle \mathbf{a}_{1,2}, S_{1,2}\rangle$, …, $\langle \mathbf{a}_{1,n}, S_{1,n}\rangle$ が $\preceq'$ の辞書式順序に関して列 $\langle \mathbf{a}_{2,1}, S_{2,1}\rangle$, $\langle \mathbf{a}_{2,2}, S_{2,2}\rangle$, …, $\langle \mathbf{a}_{2,n}, S_{2,n}\rangle$ 以下であるとき，かつ，そのときにかぎり，$t_1 \preceq t_2$ であるとする．型記号集合が Brouwer 束であるとき，極大な型付き素性構造の集合中に関係 $\preceq$ で最小のものが存在し，これを t_1 と t_2 の差分とする．

差分 $t = t_2 \rightarrow_t t_1$ は次の性質を持つ．共参照関係は次で構成される☆．型付き素性構造の共参照関係は共参照する素性アドレスの対の集合である．t_1 と t_2 の共参照関係の差集合中の対を優先順序 $\leq_{\mathcal{F}^*}$ の辞書式順序の昇順に並べた列を $\langle p_{1,1}, p_{1,2}\rangle$, …, $\langle p_{n,1}, p_{n,2}\rangle$ とする．この列を以下の繰り返し手続きで並べ替える．$\langle p_{i,1}, p_{i,2}\rangle$ を順に調べ，$p_{i,1}$ と t_1 で共参照する素性アドレスの真の接頭語を含む対でそれ以降に出現するものがあれば，そのような対を $\leq_{\mathcal{F}^*}$ の辞書式順序の昇順に並べ，$\langle p_{i,1}, p_{i,2}\rangle$ の直前に移動する．この操作をこのような $\langle p_{i,1}, p_{i,2}\rangle$ がなくなるまで繰り返した結果を $Seq = \langle p'_{1,1}, p'_{1,2}\rangle$, …, $\langle p'_{n,1}, p'_{n,2}\rangle$ とする．型付き素性構造は素性アドレス ϵ を含むから暫定的共参照関係を $\{\langle \epsilon, \epsilon\rangle\}$ とし，Seq 中の対の必要性を順に調べ，必要な場合に暫定的関係に追加する．$\langle p'_{i,1}, p'_{i,2}\rangle$ を追加する際，暫定的関係を対称にするために $\langle p'_{i,2}, p'_{i,1}\rangle$ も追加し，$p'_{i,1}$ と $p'_{i,2}$ の各接頭語 q に関して反射的にするために $\langle q, q\rangle$ も追加する．対は，直前の暫定的関係と t_2 の共参照関係の和を t_1 の素性アドレス集合上で推移的かつ右不変的になるようにする最小の拡張に含まれないときに必要である．以上の逐次的構成過程で得た暫定的関係を t_1 の素性アドレス集合上で推移的かつ右不変的になるようにする最小の拡張 κ を，κ を用いて計算する素性アドレス集合上に制限したものが t の共参照関係である．上の逐次的構成過程では，Seq 中の以下の自明でない関係が不要である．

(a) 素性アドレスがいずれも t_1 での共参照同値類中の最優先（$\leq_{\mathcal{F}^*}$ で最小）要素でない関係は不要である☆☆．(b) 少なくとも一方の素性アドレスが t_1 で共参照同値類中の最優先要素でない真の接頭語を持つ関係は不要である．(c) t_1 で $p, q_1, \ldots, q_m$ がそれぞれ共参照同値類中の最優先要素であり（$i \neq j$ で $q_i = q_j$ の場合も含む），$1 \leq i \leq m$ に関して，各 q_i がそれぞれ q'_i, q''_i と共参照関係にあり，空列でない r_i が存在し，$q_i \cdot r_i$ が p と共参照関係にあり，かつ，t_2 で $1 \leq i \leq m-1$ に関して $q'_i \cdot r_i$ と $q''_{i+1} \cdot r_{i+i}$ が共参照関係にあるとき，$q_i \cdot r_i$ 中で最優先のものを p' とすると，p' 以外の $q_i \cdot r_i$ と p の間の関係は不要である．

素性アドレス p の型記号は次の値を取る．(a) p が t_1 での p の共参照同値類中の最優先要素 q と差分結果で共参照する場合☆☆☆: t_1 の p での型記号を $\mathbf{a}$, t_1 での p の共参照同値類の要素で，t_2 に含まれる各 p_i（$1 \leq i \leq n$）での t_2 の型記号を $\mathbf{b}_i$ とすると，$(\mathbf{b}_1 \wedge_{\mathcal{T}} \cdots \wedge_{\mathcal{T}} \mathbf{b}_n) \rightarrow_{\mathcal{T}} \mathbf{a}$ である．(b) 他の場合：$\top$ である．

素性アドレス集合は以下の素性アドレスから構成される．(a) t_1 だけに含まれ，t_1 でこれを含む共参照同値類中の最優先要素であり，かつ，t_2 に含まれる素性アドレスと t_1 で共参照関係にないもの．(b) 型記号が $\top$ 以外のもの．(c) 差分結果で自分以外と，真の接頭語間の関係から必然的に成立しない共参照関係にあるもの．(d) (a–c) の接頭語．(e) (a–d) と差分結果で共参照関係にあるもの．以上から，素性アドレスは t_1 の共参照同値類中の最優先要素と差分で共参照関係にないかぎり差分に含まれないことが証明できる．

型付き素性構造の汎化，単一化，差分の例を**図 2** に示す．ここで，$agree \leq_{\mathcal{F}} subj$ とする．単一化と差分の例から明らかなように，差分結果と差分の第 1 引数の単一化は差分の第 2 引数になる．差分の第 1 引数との単一化が第 2 引数となる極大型付き素性構造は他にも存在するが，上記の優先関係により差分結果は例に示したものに一意に決定される．

☆ 文献 9) の構成方法では正しい結果が得られないことがあり，以下で述べる方法により正しい共参照関係が得られる．

☆☆ t_1 でのある共参照同値類中の最優先要素を p, 他の任意の異なる要素を p', p'' とする．Seq に $\langle p', p''\rangle$ が出現するとき，$\langle p, p'\rangle$ と $\langle p, p''\rangle$ は出現すれば，$\langle p', p''\rangle$ より前に出現し，出現しなければ，t_2 の共参照関係に含まれ，いずれの場合にも $\langle p', p''\rangle$ を取り扱う直前の暫定的関係と t_2 の共参照関係の和を推移的かつ右不変的になるようにする最小の拡張が $\langle p', p''\rangle$ を含むから，$\langle p', p''\rangle$ は不要である．以下の (b) と (c) も同様の論法で示せる．

☆☆☆ 差分の形式的構成法では，q と p の対を κ が含む場合．

$$\mathbf{syn}\left[agree:\ \mathbf{agr}\left[per:\ \mathbf{2nd}\right]\right]$$

$$\vee_t\ \mathbf{syn}\begin{bmatrix} agree: & \mathsf{X}:\mathbf{agr}\begin{bmatrix} num: & \mathbf{sg} \\ per: & \mathbf{3rd}\end{bmatrix} \\ subj: & \mathbf{syn}\left[agree:\ \mathsf{X}\right]\end{bmatrix}$$

$$=\mathbf{syn}\left[agree:\ \mathbf{agr}\left[per:\ \mathbf{2nd\_or\_3rd}\right]\right]$$

$$\top\left[agree:\ \top\begin{bmatrix} num: & \mathbf{sg} \\ per: & \mathbf{3rd}\end{bmatrix}\right]$$

$$\wedge_t\ \mathbf{syn}\begin{bmatrix} agree: & \mathsf{X}:\mathbf{agr} \\ subj: & \mathbf{syn}\left[agree:\ \mathsf{X}\right]\end{bmatrix}$$

$$=\mathbf{syn}\begin{bmatrix} agree: & \mathsf{X}:\mathbf{agr}\begin{bmatrix} num: & \mathbf{sg} \\ per: & \mathbf{3rd}\end{bmatrix} \\ subj: & \mathbf{syn}\left[agree:\ \mathsf{X}\right]\end{bmatrix}$$

$$\mathbf{syn}\begin{bmatrix} agree: & \mathsf{X}:\mathbf{agr} \\ subj: & \mathbf{syn}\left[agree:\ \mathsf{X}\right]\end{bmatrix}$$

$$\rightarrow_t\ \mathbf{syn}\begin{bmatrix} agree: & \mathsf{X}:\mathbf{agr}\begin{bmatrix} num: & \mathbf{sg} \\ per: & \mathbf{3rd}\end{bmatrix} \\ subj: & \mathbf{syn}\left[agree:\ \mathsf{X}\right]\end{bmatrix}$$

$$=\top\left[agree:\ \top\begin{bmatrix} num: & \mathbf{sg} \\ per: & \mathbf{3rd}\end{bmatrix}\right]$$

図2 型付き素性構造の汎化，単一化，差分の例
Fig. 2 Examples of generalization, unification, and difference of typed feature structures.

3. 差分計算手法

差分の入力型付き素性構造を表す2つの有向グラフを取り，差分結果の有向グラフを返す手法を示す．本手法は入力を保存するために，入力を複製しながら出力を構成する増進的複製アプローチ[14]を取る．差分は単一化や汎化より入出力構造が複雑に対応するので，種々のグラフ操作技法を考案した．以下では最初に差分計算用のデータ構造を示し，次に計算の各側面を検討し，最後に差分アルゴリズムを示す．

3.1 増進的複製とデータ構造

差分計算のためのデータ構造

型付き素性構造の有向グラフを表すのに**図3**のデータ構造を用いる．型付き素性構造を表す *node* 構造はフィールドとして型記号のための *tsymbol*，素性を表す *arc* 構造を管理する *arcs* と，計算に必要な *generation*, *copy*, *forward*, *preference* を持つ．

世代管理用 *generation* フィールド． データ構造の破壊的変更可能性を判断するために，個々の差分グラフを計算する過程に識別子（整数）を割り当て，*node* 構造作成時にこの識別子を *generation* 値として付与する．*node* 構造はこの値が計算過程の識別子と等しいとき，その過程で作成されたことを意味し，カレントであると言う．カレントな構造を破壊的に変更しても，入力 *node* 構造が表す素性構造は変更されない．

node 構造	
tsymbol	⟨型記号⟩
arcs	⟨*arc* 構造の集合⟩
generation	⟨整数⟩
copy	⟨*node* 構造⟩ | *NIL*
forward	⟨*node* 構造⟩ | *NIL*
preference	⟨整数⟩

arc 構造	
label	⟨素性記号⟩
value	⟨*node* 構造⟩

図3 差分計算のためのデータ構造
Fig. 3 Data structures for typed feature structure difference.

複製管理用 *copy* フィールド． 共参照関係の決定などに必要な入出力節点の対応を取るために *copy* フィールドを使用する．差分では各入力節点を単一出力節点と対応させるので，*copy* 値は *node* 構造か *NIL* を取る．入出力節点の対応が意味を持つのは1つの差分グラフの計算中だけで，*copy* 値で有意味なのはカレントな節点である．カレントでない，例えば，別の差分グラフ計算中に作成された *node* 構造は無視される．

同値類管理用 *forward* フィールド． 節点は素性アドレスの共参照同値類を表すが，計算過程で同値類の和を取る必要が生じる．そこで，Union-Find アルゴリズム[3]を用い，その中でこのフィールドを用いる．使用法はグラフ単一化[6),14]と同様である．複数の *node* 構造が表す同値類の和を取るために，代表構造を決め，他の構造の *forward* 値をこれに設定する．*forward* 値をたどり，代表構造を得る操作を参照解読と言う．参照解読結果の構造が共参照同値類の情報を持つ．

優先順序管理用 *preference* フィールド． 共参照関係決定の際，2つの複製節点が表す共参照同値類の最優先素性アドレスのどちらが優先するか判定する必要が生じる．そこで，最優先素性アドレスの優先性を示す整数を複製に *preference* 値として付与する．この値が小さい節点の最優先素性アドレスが優先する．

単一化や汎化のためのデータ構造

単一化，汎化，差分を任意に適用するには，データ構造変換を回避するためにデータ構造の共用が望まれる．単一化では，非破壊的単一化法[14]と LING 法[6]が図3の *node* 構造から *preference* を除いた構造を，LIC 法[1]がさらに *forward* を除いた構造を用い，これらの手法と差分はデータ構造を共用できる．汎化では，増進的複製による手法[8]が図3の *node* 構造から *preference* と *forward* を除いた構造で，*copy* 値が *node* 構造の集合のものを用い，*copy* 値の扱いの簡単

な変更で，この手法ともデータ構造を共用できる．

3.2 基本アイデア

2つの有向グラフが与えられたとき，両入力の同一素性アドレスの節点☆の対を遷移しながら複製を作成し，出力を構成する．差分の各素性アドレスの構造は入力の共参照同値類中での最優先性に依存して決定される．最優先性を判断するために，素性アドレスの優先順序 $\leq_{\mathcal{F}^*}$ で遷移し，遷移した節点の *copy* 値を複製節点にする．これにより，節点が複製を持たないこと☆☆が遷移中の素性アドレスが節点の表す同値類中で最優先であることを意味する．複製作成順序が複製の表す最優先素性アドレスの優先順序に対応するから，複製の *preference* 値に差分計算中の何番目に作成された複製であるのかを示す整数を付与する．この遷移は，$\leq_{\mathcal{F}^*}$ が素性記号の優先順序 $\leq_{\mathcal{F}}$ に基づく辞書式順序であるから，各節点でそれを出発する弧をその素性記号の優先順序で選択していく縦型の遷移になる．

以下では，共参照関係，型記号，素性アドレス集合に関する条件を順に吟味する．入力は $t_1 \leq_t t_2$ である型付き素性構造 t_1, t_2 を表す2つの有向グラフ g_1, g_2 の根 $n_{1,\epsilon}$, $n_{2,\epsilon}$ であるとし，$i, j = 1, 2, \ldots$ に関して，$i \leq j$ ならば，$f_i \leq_{\mathcal{F}} f_j$ であるとする．

3.2.1 共参照関係

第2章で述べた差分の共参照関係の性質がグラフ上で意味することを考える．グラフでは節点が同一素性記号の弧を複数持たないかぎり共参照関係は右不変的な同値関係となるから，逐次的構成過程だけを考えれば十分である．この過程で不要な関係の条件 (a) は，g_1 の各節点に最初に到達した素性アドレス p が g_1 で最優先☆☆☆であるから，このような p との関係に注目すればよいことを示す．(b) は，g_1 の節点に2番目以降に到達する素性アドレスを真の接頭語に持つ素性アドレスとの共参照関係が考慮不要であることを示す．例えば，**図4** の g_1 と g_2 の差分では，g_1 で f_1, f_2, f_3 が共参照関係にあり，f_2, f_3 を真の接頭語とする素性アドレスとの関係（例えば，$f_1 \cdot f_4$ と $f_2 \cdot f_4$, $f_1 \cdot f_4$ と $f_2 \cdot f_5$ の関係）は考慮不要である．(c) は，グラフ遷移中に共参照関係の不要性が後で発見される可能性を示す．図4の場合，g_1 で f_1 と f_2 が，g_2 で $f_1 \cdot f_6$ と $f_2 \cdot f_4$ が共参照関係にあり，$f_1 \cdot f_4$ と $f_1 \cdot f_6$ の関係は不要である．この不要性は g_2 の $f_2 \cdot f_4$ を調べないと発見できない．

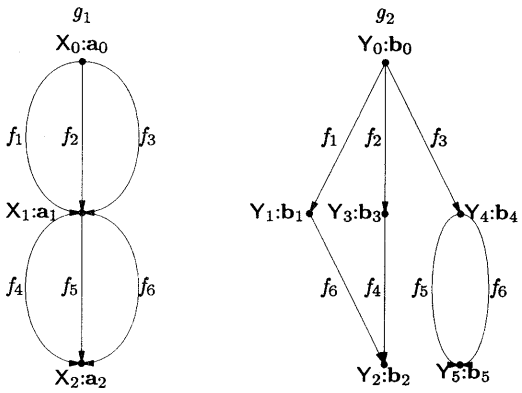


図4 共参照関係の取り扱い
Fig. 4 Treatment of coreference relations.

共参照関係の検出

入力グラフを遷移する際，入力節点に最初に到達したときに *copy* 値に複製節点を設定する．素性アドレス p の入力節点が既に複製を持つ場合，この複製を設定した素性アドレスと p が共参照関係にあることを意味する．入力条件から，g_2 で成立する関係は g_1 でも成立するから，g_2 側の *copy* 値の検査で両入力共通の共参照関係を検出できる．

共参照関係の構成

素性アドレス p の入力節点が出力の q の節点 n を *copy* 値として既に持つ場合，出力での p と q の共参照関係は，(1) n を p の出力節点とするか，(2) p の出力節点 n' を別に作成し，n' の *forward* 値を n にすることで構成できる．方法 (1) は複製量が少ないが，成立させた関係の除去に弧の削除などが必要で，削除する弧の決定は簡単ではない．そこで，方法 (2) を用いる．不要な関係の条件 (a–c) から節点滞在中に不要性が示せない，成立させる可能性のある共参照関係にある素性アドレス対を表す節点対を記録し，後で条件 (c) が示す不要性が判明したら，対応する節点対を記録から削除する．節点滞在中に不要性を示せないのは，g_1 の共参照同値類の最優先要素と，$p = p_1 \cdot f$（f は素性記号）で，p_1 が g_1 で最優先である p の関係で，後の不要性の判断はこの種の関係を調べる．グラフ遷移終了後の共参照関係確定時に方法 (2) で共参照関係を構成する．

3.2.2 型記号

差分の素性アドレス p での型記号は p が g_1 の共参照同値類中の最優先素性アドレスと差分で共参照関係に

☆ 正確にはグラフの根から同一素性アドレスをたどって到達する節点．適宜この簡略表現を用いる．また，「節点の表す（最優先）素性アドレス」で節点に到達する（最優先）素性アドレスを示す．

☆☆ 正確には，*copy* 値としてカレントな *node* 構造を持たないこと．適宜この簡略表現を用いる．

☆☆☆ 正確には，g_1 で p を含む共参照同値類中の最優先素性アドレス．適宜この簡略表現を用いる．

あるときだけ⊤以外の値を取ることがある．他の素性アドレスは不要である．最優先性は g_1 側での複製の有無で判定でき，最優先素性アドレスと共参照関係にあるものは同一節点で表される．また，$(\mathbf{b}_1 \wedge_\tau \cdots \wedge_\tau \mathbf{b}_n) \rightarrow_\tau \mathbf{a} = (\mathbf{b}_n \rightarrow_\tau \cdots \rightarrow_\tau (\mathbf{b}_1 \rightarrow_\tau \mathbf{a}) \cdots)$ である．ゆえに，g_1 のすべての節点 $n_{1,p}$（*tsymbol* 値 $\mathbf{a}_{1,p}$）と（存在すれば）対応する g_2 の節点 $n_{2,p}$（*tsymbol* 値 $\mathbf{a}_{2,p}$）を遷移し，以下を行うことで型記号を割り当てられる．(a) $n_{1,p}$ が複製を持たない場合：$n_{2,p}$ が存在すれば，*tsymbol* 値 $\mathbf{a}_{2,p} \rightarrow_\tau \mathbf{a}_{1,p}$ の，存在しなければ，$\mathbf{a}_{1,p}$ の新規複製を作成する．(b) $n_{1,p}$ が複製を持つ場合：$n_{2,p}$ が存在し，複製を持たなければ，$n_{1,p}$ の複製の *tsymbol* 値 $\mathbf{a}_{3,p}$ を $\mathbf{a}_{2,p} \rightarrow_\tau \mathbf{a}_{3,p}$ に変更する．

3.2.3 素性アドレス集合

第2章で述べた素性アドレス集合の要素に関する各条件のグラフ遷移時の判定可能性を考える．条件 (a) に関しては，一方の入力だけに含まれるという条件は同一素性アドレスの節点対を遷移する際に節点滞在中に簡単に判定できる．g_1 の節点だけが含む素性記号の弧の到達点を根とする部分グラフ中の節点に対応するからである．また，g_1 での素性アドレスの最優先性は複製の有無で判定できる．しかし，g_2 の素性アドレスと g_1 で共参照関係にあるかどうかはグラフ全体の遷移後にしか判定できない．(b) は，後で型記号が⊤になることがあり，グラフ全体の遷移後でないと判定できない．(c–e) は，共参照関係が確定するグラフ全体の遷移後でないと判定できない．

そこで，最初に入力グラフを遷移し，必要となる可能性がある素性アドレスの節点を作成し，後で不要な部分を削除する．既に述べたように，差分は g_1 での最優先素性アドレスと差分で共参照する素性アドレスしか含まないので，この可能性がある素性アドレスの節点を作成するだけで十分である．

各条件は複製作成後の再遷移により判定できる．条件 (a) は遷移時に g_1 の節点だけが含む素性記号の弧の到達点を根とする部分グラフを複製する際，出力で g_1 での共参照同値類の最優先素性アドレスを表す，各節点の最初に作成される複製を記録し，後で g_2 に含まれる素性アドレスとの共参照関係を発見した時点で対応する節点を記録から削除することで判定できる．(b) は再遷移だけで判定できる．(c) は，確定した共参照関係を *forward* 値で成立させる際に *forward* 値を記録することで判定できる．(d) は (a–c) で必要な節点への途中の節点であることで判定できる．(e) は共参照関係にある素性アドレスが同一節点で表されるから問題がない．

```
FUNCTION Difference(node1, node2)
  GLOBAL generation, preference;
  GLOBAL apairs, dpairs, cnodes, nnodes;
  generation := generation + 1;
  preference := 0;
  apairs := ∅;
  dpairs := ∅;
  cnodes := ∅;
  nnodes := ∅;
  newnode := Make_DGraph(node1, node2);
  FOR ⟨gnode, snode⟩ IN apairs DO
    snode.forward := gnode;
    cnodes := cnodes ∪ {gnode}
  ENDFOR;
  Remove_Unnecessary(newnode);
  return(newnode)
ENDFUNCTION
```

図5 差分関数 (1)
Fig. 5 Difference function (1).

3.3 差分アルゴリズム

差分グラフを関数 *Difference*（**図5**）で計算する．この関数は引数として2つの *node* 構造（*node1* と *node2*）を取り，差分グラフの根（*newnode*）を返す☆．この関数は第3.2節の検討に基づき，(1) 骨格の構成，(2) 共参照関係の確立，(3) 不要な構造の除去の3段階から構成される．

関数 *Difference* は以下の大域変数を使用する．

generation: 計算過程の識別子を格納する．*Difference* は呼び出されると，この値を1だけ増加させ，既存の節点をカレントでないようにする．関数 *Create_Node* が *node* 構造を作成するとき，この変数値をその *generation* 値として付与する．

preference: 計算過程で作成された *node* 構造の数を格納する．*Create_Node* が *node* 構造を作成するとき，この変数値をその *preference* 値として付与した後，この変数値を1だけ増加させる．

apairs: 共参照関係を構成する素性アドレスの出力 *node* 構造の対の集合を格納する．

dpairs: 出力 *node* 構造の対の集合を格納する．各対は，第1要素の表す素性アドレスに関する共参照関係から第2要素の表すものに関する関係が不要になる可能性があることを示す．

cnodes: 共参照関係を成立させた出力 *node* 構造（*forward* 値）の集合を格納する．

nnodes: 素性アドレスの条件 (a) を満足する素性アドレスの出力 *node* 構造の集合を格納する．

☆ *node1* と *node2* が表す型付き素性構造をそれぞれ t_1 と t_2 とすると，$t_1 \leq_t t_2$ であるとする．

```
FUNCTION Make_DGraph(node1, node2)
  node1 := Dereference(node1);
  node2 := Dereference(node2);
  IF ¬Current_Node_p(node1.copy) THEN
    newcopy := Create_Node();
    node1.copy := newcopy;
    node2.copy := newcopy;
    newcopy.tsymbol
      := node2.tsymbol →_T node1.tsymbol;
    arcpairs := Sorted_Arc_Pairs(node1, node2);
    FOR ⟨arc1, arc2⟩ IN arcpairs DO
      IF Arc_p(arc2) THEN
        value := Make_DGraph(arc1.value, arc2.value)
      ELSE
        value := Copy_DGraph(arc1.value)
      ENDIF;
      newarc := Create_Arc(arc1.label, value);
      newcopy.arcs := newcopy.arcs ∪ {newarc}
    ENDFOR;
    return(newcopy)
  ELSE IF ¬Current_Node_p(node2.copy) THEN
    newcopy := Create_Node();
    newcopy.tsymbol := ⊤;
    apairs := apairs ∪ {⟨node1.copy, newcopy⟩};
    Remove_Redundancy(node1, node2, newcopy);
    return(newcopy)
  ELSE
    newcopy := Create_Node();
    newcopy.tsymbol := ⊤;
    pnode2 := Get_Preferred_Node(node2.copy);
    dpairs := dpairs ∪ {⟨newcopy, pnode2⟩};
    return(newcopy)
  ENDIF
ENDFUNCTION
```

図6 差分関数 (2)
Fig. 6 Difference function (2).

ステップ (1)

関数 *Make_DGraph*（図6）で出力の骨格を作成する．この関数は再帰適用で両入力の同一素性アドレスの節点の対を遷移し，共通複製を作成する．入力 *node1* 側だけが持つ素性アドレスは，関数 *Copy_DGraph* で複製を作成する．値として節点を返す．この関数は，入力節点を関数 *Dereference* で参照解読した後，カレントな *node* 構造を *copy* 値として (a) *node1*（と入力条件から *node2*）が持たない場合，(b) *node1* だけが持つ場合，(c) 両方とも持つ場合に場合分けする．

場合 (a) では，現在の素性アドレス p が *node1* 側で最優先である．型記号 $node2.tsymbol \rightarrow_T node1.tsymbol$ の p の出力節点 *newcopy* を作成し，*node1* と *node2* の *copy* 値に設定した後，弧を扱う．*node1* と *node2* の同一素性記号の *arc* 構造の対のリストで，素性記号の優先順序でソートされたものを関数 *Sorted_Arc_Pairs* で作成し☆，*node2* の *arc* 構造が存在する対には *value* 値に *Make_DGraph* を適用し，ほかの対には *node1* の *arc* 構造の *value* 値に *Copy_DGraph* を適用し，複製を作成し，複製への弧を *newcopy* に追加する．値 *newcopy* を返す．

場合 (b) では，現在の素性アドレス p と *node1* の *copy* 値を設定した *node1* 側での最優先素性アドレス q が *node1* 側だけで共参照関係にある．不要性が示されないかぎり，この関係を成立させるために型記号 $\top$ の p の出力節点 *newcopy* を作成し，q の出力節点 *node1.copy* とこれの対を *apairs* に追加する．p は，この関係が成立すると，*node1.copy* で表され，成立しないと，*node1* 側での最優先素性アドレスと差分で共参照しないから不要で，いずれの場合も *newcopy* からの弧は不要である．また，逐次的構成過程で不要な関係の条件 (b) から p を真の接頭語に持つ素性アドレスに関する共参照関係の構成を考慮する必要はない．出力に *node2* の情報を反映するために関数 *Remove_Redundancy* を *node1*, *node2*, *newcopy* に適用し，値 *newcopy* を返す☆☆．

場合 (c) では，現在の素性アドレス p と，*node1* 側で r の節点滞在中に作成した r の出力節点を *node2* の *copy* 値に設定した r' が *node2* 側で共参照関係にあり，*node1* 側で r と r' が真の接頭語間の関係から必然的な共参照関係にあり，$r \leq_{\mathcal{F}^*} r' <_{\mathcal{F}^*} p$ である．共参照関係の逐次的構成過程で不要な関係の条件 (c) から *node1* 側での p の共参照同値類中の最優先要素 q と p の関係は不要である☆☆☆．また，この条件から，*node1* 側で p と真の接頭語間の関係により必然的な共参照関係にある素性アドレスと他との *node2* 側での共参照関係を後で *Remove_Redundancy* が発見すると，p の代わりに r を考慮する必要がある．この p と r の関係は，型記号 $\top$ の p の出力節点 *newcopy* を作成し，これと *node2.copy* の対を *dpairs* に追加することで記録できる．ここで，既に r が他とこのような関係にある可能性があるから，r の代わりに扱う素性アドレスの出力節点を関数 *Get_Preferred_Node*

☆ この関数は第1引数の節点が持つ各弧に対して同一素性記号の第2引数の節点の弧との対を作成し，それらの素性記号の優先順序でソートされたリストを返す．第2引数の節点に同一素性記号の弧がないとき，代わりに *NIL* を持つ対を作成する．

☆☆ q と p の共参照関係が成立しないと，*newcopy* はステップ (3) で削除される．

☆☆☆ 節点を作成する *Make_DGraph* と *Copy_DGraph* が任意の真の接頭語が *node1* 側で最優先である素性アドレスだけで呼ばれ，p も r もそのような素性アドレスであり，条件 (c) が適用される．

```
FUNCTION Copy_DGraph(node)
  node := Dereference(node);
  IF Current_Node_p(node.copy) THEN
    newcopy := Create_Node();
    newcopy.tsymbol := ⊤;
    apairs := apairs ∪ {⟨node.copy, newcopy⟩};
    return(newcopy)
  ELSE
    newcopy := Create_Node();
    newcopy.tsymbol := node.tsymbol;
    node.copy := newcopy;
    nnodes := nnodes ∪ {newcopy};
    arcs := Sorted_Arcs(node);
    FOR arc IN arcs DO
      newvalue := Copy_DGraph(arc.value);
      newarc := Create_Arc(arc.label, newvalue);
      newcopy.arcs := newcopy.arcs ∪ {newarc}
    ENDFOR;
    return(newcopy)
  ENDIF
ENDFUNCTION
```

図7 差分関数 (3)
Fig. 7 Difference function (3).

で決定し☆, *newcopy* とこの節点の対を *dpairs* に追加する．場合 (b) と同様に *newcopy* からの弧が不要で，値 *newcopy* を返す．

関数 *Copy_DGraph*（図7）は引数として節点 *node* を取り，これを根とする部分グラフの複製を返す．引数の *copy* 値 *node.copy* がカレントな *node* 構造の場合，*Make_DGraph* の場合 (b) と同様に，型記号 ⊤ の複製 *newcopy* を作成し，*node.copy* と *newcopy* の対を *apairs* に追加する．他の場合，現在の素性アドレス p が *node1* 側で最優先であるので，*node1* と同一型記号の複製 *newcopy* を作成する．なおかつ p は *node1* 側だけに含まれ，素性アドレスの条件 (a) を満たす可能性があるので，*newcopy* を *nnodes* に追加する．そして，素性記号の優先順序で弧を複製する．

関数 *Remove_Redundancy*（図8）は引数として入力構造 *node1*, *node2* と出力構造 *node3* を取り，出力から冗長な情報を除去する．現在の素性アドレスを $p = p_1 \cdot f$ (f は素性記号)，p_1' を *node1* 側で p_1 の共参照同値類中の最優先要素とすると，*node3* が $p' = p_1' \cdot f$ の出力節点で，*node1.copy* が *node1* 側での p の共参照同値類中の最優先要素 q の出力節点であるように呼び出される．*node2* が *copy* 値としてカレントな節点を (a) 持たない場合と，(b) 持つ場合に分ける．前者の場合，*node2* の *copy* 値を *node3* にする．q は *node2* 側に含まれる p と *node1* 側で共参照関係にあり，素

☆ この関数は入力として節点を取り，これを第1要素とする対を *dpairs* が含む場合は第2要素に自分自身を再帰適用し，他の場合は入力を返す．

```
FUNCTION Remove_Redundancy(node1, node2,
                           node3)
  node1 := Dereference(node1);
  node2 := Dereference(node2);
  IF ¬Current_Node_p(node2.copy) THEN
    node2.copy := node3;
    nnodes := nnodes − {node1.copy};
    node1.copy.tsymbol
      := node2.copy.tsymbol →_T node1.copy.tsymbol;
    arctriples
      := Sorted_Arc_Triples(node2, node1, node1.copy);
    FOR ⟨arc2, arc1, arc3⟩ IN arctriples DO
      Remove_Redundancy(arc1.value, arc2.value,
                        arc3.value)
    ENDFOR
  ELSE
    pnode2 := Get_Preferred_Node(node2.copy);
    pnode3 := Get_Preferred_Node(node3);
    IF pnode3.preference < pnode2.preference THEN
      apairs := apairs − {⟨node1.copy, pnode2⟩};
      dpairs := dpairs ∪ {⟨pnode2, pnode3⟩}
    ELSE IF pnode2.preference < pnode3.preference
    THEN
      apairs := apairs − {⟨node1.copy, pnode3⟩};
      dpairs := dpairs ∪ {⟨pnode3, pnode2⟩}
    ENDIF
  ENDIF;
  return(node3)
ENDFUNCTION
```

図8 差分関数 (4)
Fig. 8 Difference function (4).

性アドレスの条件 (a) を満足しないから，q の出力節点 *node1.copy* を *nnodes* から削除する．*node2* の型記号の情報を出力に反映させるために *node1.copy* の *tsymbol* 値を *node2.tsymbol* $\rightarrow_T$ *node1.copy.tsymbol* に変更する．そして，*node2* の弧の情報を出力に反映させるために，*node2* が持つ各 *arc* 構造とこれと同一素性記号の *node1*, *node1.copy* の *arc* 構造との3つ組を要素とするリストで，素性記号の優先順序でソートされたものを関数 *Sorted_Arc_Triples* で作成し☆☆，各3つ組の *node1*, *node2*, *node1.copy* の *arc* 構造の *value* 値に *Remove_Redundancy* を適用する．

場合 (b)，すなわち，*node1* 側の r の節点滞在時に作成した節点を *node2* が r' で *copy* 値に設定されているとき，両入力で r' と p が共参照関係にあり，*node1* 側で r と r'，および，前述のように p' と p が真の接頭語間の関係により必然的な共参照関係にある．逐次的構成過程で不要な関係の条件 (c) で不

☆☆ この関数は第1引数の節点が持つ各弧に対して同一素性記号の第2，第3引数の節点の弧との3つ組を作成し，それらの素性記号の優先順序でソートされたリストを返す．第2，3引数の節点に同一素性記号の弧がないとき，代わりに *NIL* を持つ3つ組を作成する．

要な関係を発見する可能性がある．まず，*node2.copy* と *node3* に *Get_Preferred_Node* を適用し，r と p' の代わりに考慮する素性アドレス r'' と p'' の出力節点 *pnode2* と *pnode3* を決定する．(1) $p'' <_{\mathcal{F}^*} r''$ (*pnode3.preference* < *pnode2.preference*) のとき，q と r'' の共参照関係が不要で，これを表す *node1.copy* と *pnode2* の対を *apairs* から除去し，(2) $r'' <_{\mathcal{F}^*} p''$ のとき，q と p'' の共参照関係が不要で，これを表す *node1.copy* と *pnode3* の対を *apairs* から除去する．*Make_DGraph* の場合 (c) と同じ理由から場合 (1) では *pnode2* と *pnode3* の対を，場合 (2) では *pnode3* と *pnode2* の対を *dpairs* に追加する．既に *node2* の型記号と弧の情報は出力に反映されているので，*node3* を返して終る．

ステップ (2)

確定した共参照関係を成立させるために，*apairs* 中の各対の第 2 要素の *forward* 値を第 1 要素にする．そして，第 1 要素を *cnodes* に追加し，その必要性を示す．

ステップ (3)

関数 *Remove_Unnecessary*（**図 9**）で不要な構造を除去する．この関数は引数として節点を取り，素性アドレスの各条件に基づき節点の必要性を判断し，必要な場合は節点を，不要な場合は *NIL* を返す．必要な節点は，条件 (a) に対応する *nnodes* 中のもの，(b) に対応する *tsymbol* 値が $\top$ 以外のもの，(c) に対応する *cnodes* 中のもの，以上のいずれかに到達可能なものである．この関数は引数の節点の参照解読後，各弧の到達点に再帰適用し，値が *NIL* の場合に弧を削除する．そして，節点が条件 (a–c) のどれかを満足するか，弧を持つ場合に節点を，他の場合に *NIL* を返す．以上で不要な部分を除去した結果が差分グラフである．

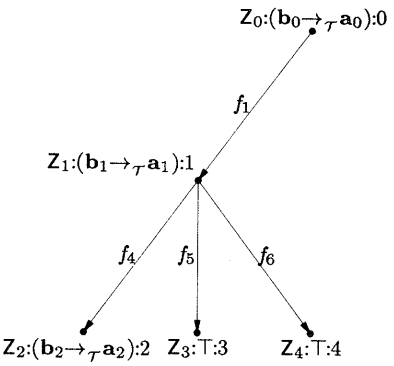
トレース例

計算過程を例で説明する．**図 10** から**図 12** は図 4 の X_0 と Y_0 の *node* 構造を入力とする計算のスナップショットである．図で節点はラベルとしてタグ記号☆，*tsymbol* 値，*preference* 値を持つ．*Difference* は X_0 と Y_0 の構造に *Make_DGraph* を適用する．この関数は両入力が複製を持たないので，型記号 $\mathbf{b}_0 \rightarrow_{\mathcal{T}} \mathbf{a}_0$ の節点（Z_0）を作成し，弧を順に扱う．f_1 の弧を扱うために $\mathsf{X}_1, \mathsf{Y}_1$ の構造に自分を適用する．両入力とも複製を持たないので，$\mathbf{b}_1 \rightarrow_{\mathcal{T}} \mathbf{a}_1$ の節点（Z_1）を作成し，弧を順に扱う．f_4 の弧は X_1 の構造だけが持つので，到達点（X_2）に *Copy_DGraph* を適用する．この節点は複製を持たないので，$\mathbf{a}_2$ の複製（Z_2）を作成し，*nnodes* に追加し，これに到達する弧を Z_1 の構造に追加する．次の f_5 の弧も X_1 の構造だけが持つので，到達点（X_2）に *Copy_DGraph* を適用する．X_2 の構造が複製（Z_2）を持つので，型記号 $\top$ の複製（Z_3）を作成し，Z_2 と Z_3 の構造の対を *apairs* に追加する．次の f_6 の弧は両入力が持つので，到達点（X_2 と Y_2）に *Make_DGraph* を適用する．X_2 の構造が *copy* 値として Z_2 の構造を持つので，$\top$ の複製（Z_4）を作成し，Z_2 と Z_4 の構造の対を *apairs* に追加し，$\mathsf{X}_2, \mathsf{Y}_2, \mathsf{Z}_4$ の構造に *Remove_Redundancy* を適用する．Y_2 の構造が *copy* 値としてカレントな節点を持たないので，*copy* 値を Z_4 の構造にし，X_2 の構造の *copy* 値（Z_2）を *nnodes* から削除し，Z_2 の

☆ 節点を指定するために便宜上使用する．

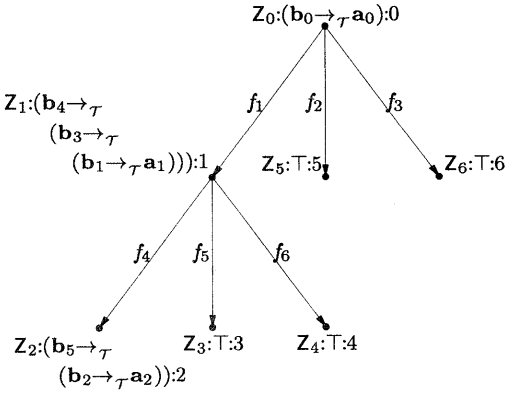
```
FUNCTION Remove_Unnecessary(node)
  node := Dereference(node);
  FOR arc IN node.arcs DO
    value := Remove_Unnecessary(arc.value);
    IF ¬Node_p(value) THEN
      node.arcs := node.arcs − {arc}
    ENDIF
  ENDFOR;
  IF node ∈ nnodes OR node.tsymbol ≠ ⊤ OR
     node ∈ cnodes OR node.arcs ≠ ∅ THEN
    return(node)
  ELSE
    return(NIL)
  ENDIF
ENDFUNCTION
```

図 9 差分関数 (5)
Fig. 9 Difference function (5).



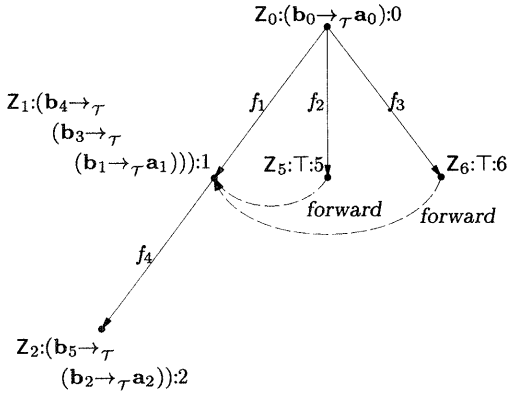
$apairs = \{\langle \mathsf{Z}_2, \mathsf{Z}_3 \rangle, \langle \mathsf{Z}_2, \mathsf{Z}_4 \rangle\}$,
$dpairs = \emptyset$,
$cnodes = \emptyset$,
$nnodes = \emptyset$.

図 10 スナップショット (1)
Fig. 10 Snapshot (1).



$apairs = \{\langle Z_1, Z_5\rangle, \langle Z_1, Z_6\rangle\}$,
$dpairs = \{\langle Z_3, Z_2\rangle, \langle Z_4, Z_2\rangle\}$,
$cnodes = \emptyset$,
$nnodes = \emptyset$.

図 11 スナップショット (2)
Fig. 11 Snapshot (2).



$apairs = \{\langle Z_1, Z_5\rangle, \langle Z_1, Z_6\rangle\}$,
$dpairs = \{\langle Z_3, Z_2\rangle, \langle Z_4, Z_2\rangle\}$,
$cnodes = \{Z_1\}$,
$nnodes = \emptyset$.

図 12 スナップショット (3)
Fig. 12 Snapshot (3).

構造の *tsymbol* 値を $\mathbf{a}_2$ から $\mathbf{b}_2 \rightarrow_\tau \mathbf{a}_2$ に変更する．図 10 はこの状態を示す．

根に戻り，f_2 の弧を扱うために X_1 と Y_3 の構造に *Make_DGraph* を適用する．前者だけが複製（Z_1）を持つから，複製（Z_5）を作成し，Z_1 と Z_5 の構造の対を *apairs* に追加し，*Remove_Redundancy* を X_1, Y_3, Z_5 の構造に適用する．この関数は X_1 の構造の *copy* 値（Z_1）の *tsymbol* 値を $\mathbf{b}_3 \rightarrow_\tau (\mathbf{b}_1 \rightarrow_\tau \mathbf{a}_1)$ に変更し，f_4 の弧を扱うために X_2, Y_2, Z_2 の構造に自分を適用する．Y_2 の構造は *copy* 値として Z_4 の構造を持つ．Z_4 と Z_2 の構造それぞれに *Get_Preferred_Node* を適用した結果は，*dpairs* 値が空集合であるから，Z_4 と Z_2 の構造である．前者より後者が *preference* 値が小さいので，X_2 の構造の *copy* 値（Z_2）と前者（Z_4）の対を *apairs* から削除し，Z_4 と Z_2 の構造の対を *dpairs* に追加する．再び根に戻り，f_3 の弧を扱うために X_1 と Y_4 の構造に *Make_DGraph* を適用する．f_2 の弧の場合と同様である．$f_3 \cdot f_6$ の構造を扱うために X_2, Y_5, Z_4 の構造に *Remove_Redundancy* を適用する．Y_5 の構造は *copy* 値として Z_3 の構造を持ち，この Z_3 と Z_4 の構造それぞれに *Get_Preferred_Node* を適用した結果は，*dpairs* が Z_4 と Z_2 の構造の対を含むから，Z_3 と Z_2 の構造である．後者の方が *preference* 値が小さいので，Z_2 と Z_3 の構造の対を *apairs* から削除し，*dpairs* 値を更新する．図 11 はこの状態を示す．

得られた *apairs* 値に基づき，f_1 と f_2 の間と f_1 と f_3 の間の共参照関係を成立させる．$\mathbf{b}_5 \rightarrow_\tau (\mathbf{b}_2 \rightarrow_\tau \mathbf{a}_2) \neq \top$ であるとき，不要なのは Z_3 と Z_4 の構造だけで，これらの削除により図 12 の構造を得る．

以上で説明した計算手法により型付き素性構造の差分を表すグラフが得られる．

4. おわりに

本論文は型付き素性構造の差分演算の計算手法を提案した．この演算は，表す情報の間に包含関係のある 2 つの型付き素性構造に対して，それらの情報の差を表す型付き素性構造を与える演算である．差分は単一化や汎化よりも素性アドレス集合，型記号，共参照関係が入出力で複雑な関係になる．そこで，差分計算に必要な種々の有向グラフ操作技法を開発した．

通常の素性構造は型付き素性構造の特殊な場合であるから，本手法は通常の素性構造にも適用可能である．

本差分手法は増進的複製の技法を用い，この技法を使用する単一化手法や汎化手法と同じデータ構造を使用できる．したがって，この 3 種の計算を行う際にデータ構造を変換する必要はない．

本差分手法は，言語情報記述の階層化による記述の効率化，情報の共有や分割による言語処理の効率化などを行う際に有用である．階層化や情報共有の行われていない記述を作成するのは他の記述を考慮することなく，独立に行えるので比較的簡単である．しかし，これは，処理の側面では，同様の記述の重複出現による空間の浪費，類似計算の繰り返しによる時間の浪費を招く．したがって，非階層的記述から階層的記述に変換することが望まれる．本差分手法は，汎化手法と

の組合せにより，そのような記述の変換を可能にする．また，素性構造の持つ一部の情報を先に取り扱い，残差情報を後で取り扱うことにより，処理を効率化できることがあるが，このような場合の残差情報を表す素性構造の計算に差分計算は使用できる．

謝辞 本論文を書くにあたり貴重なコメントをいただいたNTT基礎研究所対話理解研究グループの島津明氏，川森雅仁氏，堂坂浩二氏，中野幹生氏に感謝の意を表する．貴重なコメントをいただいた査読者の方に感謝の意を表する．

参考文献

1) Emele, M.: Unification with Lazy Non-Redundant Copying, *Proc. 29th ACL*, pp.325–330 (1991).
2) Gunji, T.: *Japanese Phrase Structure Grammar*, Reidel (1987).
3) Hopcroft, J. E. and Karp, R. M.: An Algorithm for Testing the Equivalence of Finite Automata, Technical Report TR-71-114, Dept. of Computer Science, Cornell University (1971).
4) Karttunen, L.: D-PATR—A Development Environment for Unification-Based Grammars, Technical Report CSLI-86-61, CSLI (1986).
5) Kasper, R. T.: A Unification Method for Disjunctive Feature Descriptions, *Proc. 25th ACL*, pp.235–242 (1987).
6) Kogure, K.: Strategic Lazy Incremental Copy Graph Unification, *Proc. 13th COLING*, Vol.2, pp.223–228 (1990).
7) Kogure, K.: A Treatment of Negative Descriptions of Typed Feature Structures, *Proc. 14th COLING*, pp.380–386 (1992).
8) 小暮 潔：増進的複製による型付き素性構造汎化手法，情報処理学会論文誌，Vol.34, No.9, pp.1919–1930 (1993).
9) 小暮 潔：型付き素性構造の差分演算，情報処理学会論文誌，Vol.36, No.1, pp.1–11 (1995).
10) Pereira, F. C. N.: Structure Sharing Representation for Unification-Based Formalisms, *Proc. 23rd ACL*, pp.137–144 (1985).
11) Pollard, C. and Sag, I.: *An Information-Based Syntax and Semantics—Volume 1: Fundamentals*, CSLI (1987).
12) Shieber, S. M.: *Constraint-Based Grammar Formalisms—Parsing and Type Inference for Natural and Computer Languages*, PhD Thesis, Stanford Unviersity (1989).
13) Tomabechi, H.: Quasi-Destructive Graph Unification with Structure-Sharing, *Proc. 14th COLING*, pp.440–446 (1992).
14) Wroblewski, D. A.: Nondestructive Graph Unification, *Proc. 6th AAAI*, pp.582–587 (1987).

(平成7年3月13日受付)
(平成7年9月 6 日採録)

小暮 潔（正会員）

1957年生．1979年慶應義塾大学工学部電気工学科卒業．1981年同大学院修士課程修了．同年日本電信電話公社武蔵野電気通信研究所入所．1986年～90年ATR自動翻訳電話研究所出向．現在NTT基礎研究所情報科学研究部対話理解研究グループに所属．主幹研究員．自然言語処理の研究に従事．人工知能学会，電子情報通信学会，日本音響学会，日本認知科学会，言語処理学会，ACL各会員．