Title	AC規則を含む項書換え系の停止性について
Author(s)	中野，賢司
Citation	
Issue Date	1997-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1018
Rights	
Description	Supervisor:外山 芳人，情報科学研究科，修士

修 士 論 文

AC 規則を含む項書換え系の停止性について

指導教官　外山芳人 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム額専攻

中野賢司

1997 年 2 月 14 日

目 次

参考文献 **39**

第 1 章

はじめに

項書換え系 (Term Rewriting System, TRS) とは、項の書き換えによって計算の過程を表す数学的な計算モデルである。TRS は、等式を左辺から右辺への書換え規則と見ることにより、自然な計算モデルを与えることができるので、関数型言語、代数的仕様のような等式に基づくプログラミングや、等式論理による定理自動証明やプログラムの変換・検証などを研究する上で重要な役割を果たしている [BP85,D87,H80]。

TRS の計算モデルとしての重要な性質に合流性と停止性があげられる。TRS が合流性を満たしているならば、どのように書換えを行なってもその結果の一意性は保証される。従って、合流性が保証されている TRS では、計算順序に関わりなく効率の良い計算が可能である。

一方、TRS が停止性を満たしているならば、書き換えによる計算は必ず終了する。TRS をプログラムと見なせば、停止性が保証されている場合には、任意の入力に対し TRS は必ず出力を返すことが保証される。また、停止性が保証されている TRS では、合流性を満たすかどうかの解析が容易になることも知られている。

しかし、TRS の停止性は一般には決定不能である [D82]。そのため、TRS が停止性を持つための十分条件の研究が進められている [D82]。TRS の停止性を示す方法として、項に順序をつけて証明する方法が広く用いられている。項上の順序には意味論的順序と構文論的順序の２つがある。意味論的順序は、項の集合に整礎な領域への対応を与えることにより、停止性を証明する。この順序では、領域の決定に人間の経験と直観を必要とする面が大きい。また、しばしば領域として用いられる多項式集合では、多項式上での不等式を解く必要があり、順序 の判定の自動化が困難である。一方、構文論的順序では、与えられた関数記号上の順序を項の構造に基づいて項上の順序に拡張する。この順序は、機械的な手続きによる二項の比較が容易であるとういう利点をもつ。このため複雑な項にも対応でき、逆に比較の前提となる関数記号上の順序が項の構造から推定できるなど、順序判定の自動化に有利である。構文論的順序では、単純化順序と呼ばれる諸性質を満たす順序が停止性の証明に用いられる [D87]。そのような順序の代表的な例としては、 D.Plaisted による部分項経路順序（Path ofSubterm Ordering）[P78]、N.Dershowitz による再帰経路順序（Recursive Path Ordering）[D79]、および P.Leacanne による再帰分解順序 (Recursive Decomposition Ordering）[L81] などが知られている。

AC 項書換え系（AC-TRS）は、簡略化の方向が定められない等式を含む項書換え系（E-TRS）で、AC は結合律（Associativity）と交換律（Commutativity）を表す等式の集

合を表す。TRS が等式を含むことで、書き換え規則として簡略化の方向が定めることが困難な等式も取り扱うことが可能となるため、AC-TRS は自動証明システム等で広く利用されている。

しかし、項書換え系に書換えの方向が定まらない等式が含まれると、同じ等式を両方向に適用することで無限の書換えが可能となり、通常の TRS のような停止性は満たされない [BP85]。そこで、AC-TRS では TRS のような項と項の関係ではなく、等式による同値類上での停止性を考える必要がある。つまり、$s > t$ という関係が成立していて、$s =_{AC} s'$, $t =_{AC} t'$という等式が成立しているならば、与えられた時、$s' > t'$が成立する必要がある。このような性質を AC に関する適合性という。AC-TRS の停止性の証明では、項の上での順序を AC に関する適合性を満たすものに拡張する。

AC-TRS の停止性に関する研究では、順序判定の自動化が容易であるために構文論的順序がよく用いられている。しかし、TRS の構文論的順序は AC-TRS のような同値類と同値類の関係の比較に直接用いることはできない [BP85]。そこで、項の順序を項の同値類上の順序へ拡張するために、平坦化が提案された [BP85]。これは、項に連続して現れるの AC 規則の性質を持つ関数記号（AC 関数記号）を縮退させ、AC 規則に関する同値類の代表元を得る変形である。このような代表元によって比較することにより、AC に関する適合性を満たすことができる。

しかし、平坦化による変形のみでは単純化順序とならないことがわかっている [BP85]。従来の研究では、他にいくつかの変形を組み合わせた順序を定義することにより、単純化順序の性質を実現している。Bachmair らの分配規則による変形での APO（Associative Path Ordering）では、関数記号の順序関係について AC 関数記号が最小等のの制限が必要である [BP85]。また、Kapur らの AC 順序による疑似項のコピーおよび 繰り上げ操作との組合わせでは、APO の関数記号の制限は必要でないが、変形の過程で非決定的な手続きを含んでいるため、コンピューターへの実装が困難である [KSZ90]。Rubio らによる解釈操作では、項の集合から最大の順序をもつ項を選択する手続きを持つために、基本的に項は全順序であることを仮定している [RN93]。

本研究では、平坦化と異なる視点から AC-TRS の停止性を証明するための順序を考察するために、 平坦化に代わる変形として積み上げ化を提案する。従来の平坦化では、AC 関数記号が連続で表れたときには関数記号を 1 個に縮退させるため、項の AC 関数記号に関する深さに関する情報が失われていた。一方積み上げ化では、項の深さに関する情報を最外関数の次数によって保存するので、AC に関する適合性を満たすと同時に、より効率的な比較が期待できる。しかし平坦化の場合と同様、積み上げ化も単独では単純化順序を満たすことがない。そのため、別の変形と組み合わせて単純化順序の性質を実現する必要がある。本論文では単純化順序を満たすための条件について考察し、いくつかの手法を提案する。

本論文の構成は以下のようになっている。2 章では、項書換え系の基本的な定義とその

基本的な性質である停止性について述べる。3 章では、AC 規則を含んだ項書換え系について述べ、その停止性と証明するための従来の手法について説明する。4 章では、本研究で提案した積み上げ化について考察し、積み上げ化を基礎とした単純化順序の実現のためにいくつかの手法を提案する。しかしながら、本論文で提案している手法のみでは単純化順序を実現することができない。ここでは具体的な例を示し、その原因と改良方法について考察する。

第 2 章

AC 規則を含む項書換え系について

2.1 項書換え系について

2.1.1 項書換え系

本節では、[BP85,D87,NST95,RN93] に準じて本稿に必要となる項書換え系、および AC 規則に関する諸定義を与える。関数記号の集合を $\mathcal{F} = \{f, g, h, \cdots\}$、変数の集合を $\mathcal{V} = \{x, y, z, \cdots\}$ $(\mathcal{F} \cap \mathcal{V} = \emptyset)$ とする。また、自然数の集合を $\mathcal{N} = \{0, 1, 2, \cdots\}$ と表す。

定義 1 (項) $arity : \mathcal{F} \to \mathcal{N}$を関数記号に対しその引数の数を与える写像とする。このとき、項を以下のように定義する。

1. $x \in \mathcal{V}$は項である。
2. $f \in \mathcal{F}, arity(f) = n$ で $t_1, \cdots, t_n$が項のとき、$f(t_1, \cdots, t_n)$ も項である（ただし、$arity(f) = 0$ のとき fを定数とし、fを項とする）。

□

関数記号$\mathcal{F}$と変数$\mathcal{V}$によって構成される項の集合を$\mathcal{T}(\mathcal{F}, \mathcal{V})$ とし、変数が含まれない項を基礎項といい、その集合を$\mathcal{T}(\mathcal{F})$ とする。項s のサイズ$|s|$ を、s が定数、変数の場合は$|s| = 1$、$s = h(s_1, \cdots, s_n)$ の場合は$|s| = 1 + |s_1| + \cdots + |s_n|$ で定める。$top(h(s_1, \cdots, s_n)) = h$ によって、項 s の最外の関数記号を表す。また、項$t_1, \cdots, t_n$を要素とする多重集合を $[t_1, \cdots, t_n]$ と表す。

定義 2 (代入) 代入σは変数から項への写像である。σは $f(t_1, \cdots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ に対して

$$\sigma(f(t_1, \cdots, t_n)) = f(\sigma(t_1), \cdots, \sigma(t_n))$$

として項から項への写像に拡張することができる。また、$[x \leftarrow t]$ によって変数 x に項 t を代入することを表す。

□

文脈 $C[\ ]$ とは $\mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{V})$ の要素で □ を 1 つ含む項である。ただし □ は$\mathcal{F}$に含まれない定数とする。このとき $C[t]$ は $C[\ ]$ の □ を t に置き換えて得られたものを表す。

定義 3 (項書き換え系 (TRS)) 項書換え系 Rは書き換え規則の集合である。また、書換え規則は以下の条件を満たす *2* 項の対 (l, r) である。

1. lは変数ではない。
2. lで出現する変数はrでも出現する。

以下では書き換え規則 (l, r) を $l \to r$で表す。このとき、Rによる s から t への書き換え $s \to_R t$ は以下のようにして定められる。

$s \to_R t \quad \Longleftrightarrow \quad \exists l \to r \in R, \exists C[\ ], \exists \sigma, t = C[\sigma(l)], s = C[\sigma(r)]$。

ここで$\sigma(l)$ をリデックスと呼ぶ。 □

書換え規則の集合 Rによる以下のような有限、または無限の書き換え列を、Rのリダクション列と定める。

$s \to_R s_1 \to_R s_2 \to_R \cdots$

Rによる 0 回以上の書換えで、s から t への書き換え列が得られるなら $s \xrightarrow[R]{*} t$ とする。また、Rによる 0 回以上の書き換えによって得られる項の内、Rにおけるリデックスが存在しない項を正規形と呼ぶ。

例 4 (書換えの例) 以下のような TRS R が与えられたとする。

$$R = \begin{cases} x + 0 & \to \quad x \\ x + S(y) & \to \quad S(x + y) \end{cases}$$

$S(0) + S(S(0))$ は Rによって以下のように書き換えられる。

$\underline{S(0) + S(S(0))} \to_R S(\underline{S(0) + S(0)}) \to_R S(S(\underline{S(0) + 0})) \to_R S(S(S(0)))$

（下線部はリデックス、$S(S(S(0)))$ が R の正規形） □

2.1.2 AC 規則を含む項書換え系

TRS の書き換え規則 $s \to t$ は、書換えが一方向にのみ行なわれる。一方、等式 $s = t$ では $s \to t,\ t \to s$ のように両方向への書換えが可能である。このような等式の集合を Eとし、Eによる書き換えを$\sim_E$とする。また、Eによる 0 回以上の書換えを$\overset{*}{\underset{E}{\sim}}$とし、$s \overset{*}{\underset{E}{\sim}} t$ が成立するとき $s =_E t$ と表す。

定義 5 (等式を含む項書換え系 (E-TRS)) 等式の集合 E と、書き換え規則の集合 R' からなる $TRS R = E \cup R'$を、E-TRS と表す。

□

例 6 (E-TRS の例)

$$R' = \begin{cases} x + 0 & \to \quad x \\ x + S(y) & \to \quad S(x + y) \end{cases}$$

$$E \quad : \quad x{+}y = y{+}x$$

とすると $S(0) + S(S(0))$ は Rによって以下のように書き換えられる。

$$S(0) + S(S(0)) \sim_E \underline{S(S(0)) + S(0)} \rightarrow_{R'} S(\underline{S(S(0)) + 0}) \rightarrow_{R'} S(S(S(0)))$$

（下線部は R' によるリデックス）

□

TRS は、項の簡略化の方向が決まっている規則によって構成される。しかし、簡略化の方向が決定できない等式が含まれると、通常の方法では停止性などの性質を取り扱うことができない。そこで、E-TRS では簡略化の方向が決定できない規則の集合 E と、通常の規則の集合 R' に分けて考える。

定義 7 (AC 規則を含む項書換え系（AC-TRS）) 交換律および結合律を満たす関数記号の集合を $\mathcal{F}_{\mathcal{AC}}$ （ただし、$\mathcal{F}_{\mathcal{AC}} \subset \mathcal{F}$ とする）、およびそれぞれ交換律、結合律を表す等式の集合を

$AC = \{f(x,y) = f(y,x),\ f(x,f(y,z)) = f(f(x,y),z) \mid f \in \mathcal{F}_{\mathcal{AC}}\}$

とする。このとき、等式の集合として AC を用いた E-TRS を AC-TRS と定義する。 □

なお、$f \in \mathcal{F}_{\mathcal{AC}}$ のとき、$arity(f)$ は後述の平坦化、積み上げ化などの項の変形の際には 1 以上の適当な値がとれるものとする [BP85]。

2 つの項 $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ が、等式集合 AC に関して $s \overset{*}{\underset{AC}{\sim}} t$ を満たすとき、$s =_{AC} t$ と表記する。このとき、s と t は結合律、交換律から作られる同値関係であることに注意する。

2.2 停止性

停止性は一般には決定不能であるが、特定の TRS に関してはその停止性を判定する方法がいくつか提案されている。本節では構文論的順序による方法をとりあげる。

2.2.1 TRS の停止性

TRS Rが停止性を持つとは、Rによる無限のリダクション列が存在しないことである。従って、任意の項 t から生成されるリダクション列は有限で、いつかは正規形 t' に到達する。

例 8 (停止性を満たす TRS)

$$R = \left\{ \ f(a) \rightarrow a \right.$$

この TRS R は停止性を持つ。 □

例 9 (停止性をみたさない TRS)

$$R = \left\{ \ f(a) \to f(f(a)) \right.$$

この *TRS* R は以下のような無限の書換えが行なわれるので停止性を持たない。

$$f(a) \to_R f(f(a)) \to_R f(f(f(a))) \to_R \cdots$$

□

TRS の停止性を証明するために半順序>を用いる方法がある。半順序とは推移的、反対称的な 2 項関係のことである。停止性を証明するために用いる半順序は以下の性質を満たさなくてはならない。

定義 10 (整礎性) 集合 U 上の半順序>が整礎であるとは、U の要素による無限減少列 $s_1 > s_2 > s_3 > \cdots$が存在しないことである。
□

項の集合 $\mathcal{T}(\mathcal{F} \cup \mathcal{V})$ 上の半順序>が任意の項 t、関数記号 $f \in \mathcal{F}$に対し $f(\cdots, t, \cdots) > t$ を満たすとき、部分項性を満たすという。また、任意の項 s, t および、任意の文脈 $C[\ ]$ に対し、$s > t$ のときに $C[s] > C[t]$ が成立するならば単調性を満たすという。さらに、$s > t$ のときに任意の代入σに対して、$\sigma(s) > \sigma(t)$ が成立するならば、代入に関して閉じているという。

定義 11 (単純化順序) 半順序>が単純化順序であるとは、単調性および部分項性を満たし、かつ代入に閉じている場合である *[D87]*。 □

単純化順序を用いると、TRS の停止性に関して以下のことが言える [D87]。

定理 12 (TRS の停止性) 単純化順序>が与えられた時、*TRS* の任意の規則 $l \to r$で、$l > r$が成立するなら、R は停止性を持つ。 □

2.2.2 再帰的経路順序 (RPO)

単純化順序の代表例として再帰的経路順序 (RPO) があげられる [D78]。経路順序とは関数記号上の順序 (precedense) による比較を項全体に再帰的に拡張した順序である。$f(s_1 \cdots s_m) \equiv g(t_1 \cdots t_n)$ とは、$f = g$ かつ、$m = n$ かつ、$[s_1, \cdots, s_m] \equiv_{ms} [t_1, \cdots, t_n]$ が成立する場合である。ただし、$\equiv_{ms}$は$\equiv$を多重集合に拡張したものである。また、変数と関数記号、あるいは異なる変数間には順序がつかないものとする。

定義 13 (RPO) 関数記号上の順序を$>_{\mathcal{F}}$としたとき、$s = f(s_1 \cdots s_m) >_{RPO} g(t_1 \cdots t_n) = t$ が成立するのは、以下のいずれかが成り立つ時である *[D82]*。

1. $f >_{\mathcal{F}} g$ かつ 任意の jについて、$s >_{RPO} t_j$が成立するか
2. $g >_{\mathcal{F}} f$ かつ ある i が存在して、$s_i >_{RPO} t$ または $s_i \equiv t$ が成立するか
3. $f = g$ 、$[s_1, \cdots, s_m] \gg_{RPO} [t_1, \cdots, t_n]$

□

ただし≫は項に関する順序>を項の多重集合同士の比較に拡張した多重集合順序であり、以下のように定義する。

定義 14 (多重集合順序) 項の集合$\mathcal{T}(\mathcal{F} \cup \mathcal{V})$ 上の半順序>が与えられた時に、項の多重集合 M,N について $M = [s_1, \cdots, s_m] \gg [t_1, \cdots, t_n] = N$ が成立するのは以下のいずれかの場合である *[RN93]*。

1. $M \neq \emptyset$ かつ $N = \emptyset$ 。
2. $[s_1 \cdots s_m]$ の中の s_iと $[t_1 \cdots t_n]$ の中の t_jについて、$s_i = t_j$かつ $M - [s_i] \gg N - [t_j]$ 。
3. $[s_1 \cdots s_m]$ の中の s_iと $[t_1 \cdots t_n]$ の中の $t_{j1}, \cdots, t_{jk}$について $s_i > t_{j1}, \ \cdots, \ s_i > t_{jk}$で、かつ

   (a) $M - [s_i] \gg N - [t_{j1}, \cdots, t_{jk}]$ または
   (b) $M - [s_i] = N - [t_{j1}, \cdots, t_{jk}]$。

□

また RPO の拡張として、関数記号の状態を取り入れた状態付きの再帰的経路順序（RPOS）が知られている [RN93]。状態とは各々の関数記号の種類によって決まる順序関係の比較の方法を表すものである。ここでは$\{lex, multi\}$ の２つ状態が存在するとし、任意の関数記号 fについて$\tau(f) \in \{lex, mult\}$ とする。このとき状態付の再帰的経路順序を以下のように定義する [RN93]。

定義 15 (状態付の再帰的経路順序（RPOS）) $s = f(s_1 \cdots s_m) >_{RPOS} t = g(t_1 \cdots t_n)$ が成立するのは以下のいずれか成り立つ時である。

1. $f >_{\mathcal{F}} g$ かつ 任意の jについて、$s >_{RPOS} t_j$が成立する。
2. $g >_{\mathcal{F}} f$ かつ ある i が存在して、$s_i >_{RPOS} t$ または $s_i \equiv t$ が成立する。
3. $f = g$,$\tau(f) = lex$ かつ、

   (a) $[s_1, \cdots, s_m]$ のある s_iについて、$s_i >_{RPOS} t$ または $s_i \equiv t$ のとき。　もしくは

(b) $[s_1, \cdots, s_m]$ のある i が存在して、$j < i$ を満たす任意の t_jについて $s_j \equiv t_j$で、$s_i >_{RPOS} t_i$を満たし、かつ $[t_1, \cdots, t_m]$ の任意の t_kで $s >_{RPOS} t_k$。

4. $f = g, \tau(f) = mult$ かつ、$[s_1, \cdots, s_m] \gg_{RPOS} [t_1, \cdots, t_n]$ (ただし、$\gg_{RPOS}$ は $>_{RPOS}$ の多重集合への拡張である)。

□

RPO、RPOS に対して、以下の定理が成立する [D79],[RN93]。

定理 16 (RPO、RPOS と単純化順序) *RPO,RPOS* は単純化順序である。

□

よって、定理 16 と定理 18 により RPO、RPOS と停止性について、以下のような定理が成立する。

定理 17 (RPO、RPOS と停止性) TRS Rの任意の規則 $l \to r$で、$l >_{RPO} r$が成立するなら、R は停止性を持つ。同様に、TRS Rの任意の規則 $l \to r$で、$l >_{RPOS} r$が成立するなら、R は停止性を持つ *[D79],[RN93]*。

第 3 章

AC 規則を含む項書換え系の停止性について

3.1 AC 規則を含んだ場合への適用

本章では、AC-TRS の停止性に関する諸性質と従来までの研究について述べる。

等式を含んでいる AC-TRS は、通常の TRS のような停止性を持たない。それは等式のみを適用することにより無限の書換えが可能なためである。そこで AC-TRS では項から項への書換えではなく、AC 規則から作られる同値類から同値類への書換えと見なして停止性について考察する。

一般の TRS Rでは、任意の項 $s \in \mathcal{T}(\mathcal{F} \cup \mathcal{V})$ が、以下のような無限リダクション列を生成しない場合に停止性を満たすと定義した。

$$s \to_R s_1 \to_R \cdots \to_R s_n \to_R \cdots$$

AC-TRS $R = R' \cup AC$では、以下のように等式による書換えをリダクション列に含め、AC 規則による同値類から同値類への書換えと見なす。このとき、任意の項 s が以下のような無限リダクションを生成しない場合に停止性を満たすと定義する。

$$s \to_{R'} s_1 \underset{AC}{\overset{*}{\sim}} s'_1 \to_{R'} s_2 \underset{AC}{\overset{*}{\sim}} s'_2 \cdots \to_{R'} s_n \underset{AC}{\overset{*}{\sim}} s'_n \to_{R'} \cdots$$

AC-TRS の停止性を証明する場合にも、単純化順序の代表的な手法である RPO（または RPOS）が利用可能である。しかし、RPO を AC-TRS の停止性の証明に適用する時には AC に関する同値類について考慮する必要がある。

例 18 (AC-TRS によるリダクション列への RPO の適用) $a, b \in \mathcal{F},\ f \in \mathcal{F}_{\mathcal{AC}},\ x, y, z, w \in \mathcal{V},\ f >_{\mathcal{F}} a >_{\mathcal{F}} b$ とする。このとき、*AC-TRS* $R = R' \cup AC$について

$$R' = \begin{cases} g(f(x, f(y, z)), w) & \to f(x, f(y, z)) \\ f(b, x) & \to b \end{cases}$$

とする。この規則はRPOにより、それぞれ$g(f(x, f(y, z)), w) >_{RPO} f(x, f(y, z)),\ \ f(b, x) >_{RPO} b$ が成立しているので、R'は停止性を満たしている。

一方、以下のような s, t が与えられると、

$$\begin{cases} s = f(b, f(a, b)) \\ t = f(a, f(b, b)) \end{cases}$$

$$s = f(b, f(a, b)) \sim_{AC} f(f(b, a), b) \sim_{AC} f(a, f(b, b)) = t$$

より、s と t は AC 同値類である。また、この両者を RPO で比較すると

$$f(b, f(a, b)) >_{RPO} f(a, f(b, b))$$

という順序がつく。このとき、$u = g(f(a, f(b, b)))$ という項が与えられたとする。この項に Rを適用すると

$$g(f(a, f(b, b)) \rightarrow_{R'} f(a, f(b, b)) \underset{AC}{\overset{*}{\sim}} f(b, f(a, b)) \rightarrow_{R'} b$$

というリダクション列が生成される。しかし、これを RPO で比較してみると

$$g(f(a, f(b, b)) >_{RPO} f(a, f(b, b)) <_{RPO} f(b, f(a, b)) >_{RPO} b$$

となる。　□

整礎順序による停止性の証明では、TRS R による書き換えも項の順序が単調減少であることを示すことによって、書換えが有限で停止することを証明する。しかし、書き換えに等式が含まれると、両方向での書き換えが可能であるから、リダクション列を単調減少させることは難しい。つまり、RPO を用いて R'のどの書き換え規則を順序づけることができても、AC-TRS についての停止性を導くことはできない。

そこで、以下のような性質を考える。

定義 19 (AC に関する適合性) $s,\ s',\ t,\ t' \in \mathcal{T}(\mathcal{F} \cup \mathcal{V})$ について $s =_{AC} s',\ t =_{AC} t'$が成立している時に

$$s >_{RPO} t \Rightarrow s' >_{RPO} t'$$

が成立するならば、RPO は AC に関する適合性をもつという。　□

AC-TRS $R = R' \cup AC$において、R'の任意の規則が AC 適合性を満たす順序によって停止性を保証されているならば、等式に関係なく Rの書き換えにおいても停止性を満たすことが保証される。

よって、AC-TRS の停止性を証明するための順序は定理 17 の拡張として以下の性質を満たす必要がある。

定理 20 (AC-TRS の停止性条件) *AC-TRS* $R = R' \cup AC$について、AC に関する適合性を持つ単純化順序>が与えられたとする。このとき、*AC-TRS* の全ての書き換え規則 $l \rightarrow r \in R'$に対して $l > r$ならば、*AC-TRS* は停止性を持つ *[DM79]*。　□

3.2 従来までの手法

前節で述べた AC に関する適合性を満たすための手法として、平坦化と呼ばれる変形があげられる。

3.2.1 平坦化による変形

AC に関する適合性を満たす方法として、AC 同値類の代表元へ変形して順序を定義する手法がある。

ある AC 同値類の任意の項 s_1, s_2について$\alpha(s_1) = \alpha(s_2)$ となる変形を仮定する。このとき,

$$s > t \;\Leftrightarrow\; \alpha(s) >_{RPO} \alpha(t)$$

によって、項上の順序>を定義する。すると、$s =_{AC} s'$, $t =_{AC} t'$ならば$\alpha(s) = \alpha(s'), \alpha(t) = \alpha(t')$ より $s' > t'$が成立することになり、AC に関する適合性が満たされる。このような変形としてαとして、以下のような平坦化が提案されている [BP85]。

定義 21 (平坦化) 関数記号 $f \in \mathcal{F}_{\mathcal{AC}}$、項の多重集合 $X, Y, Z \subset \mathcal{T}(\mathcal{F})$ に対し、平坦化の変形を以下のように定義する。

$$f(X, f(Y)) \rightarrow_{Fl} f(Z)$$

ここで、$Z = X \cup Y$である。ただし、$X = [X_1, \cdots, X_n]$ としたとき、$f(X)$ は $f(X_1, \cdots, X_n)$ を表すものとする。また、$arity(f)$ は *1* 以上の適当な値をとる。 □

平坦化により、結合律に関して等しいものは同一の構造の項に変形される。また、Zを多重集合として扱うと、交換律による変形は考慮する必要がなくなる。よって平坦化は、AC に関する同値類の代表元を作り出す操作となる。このとき、平坦化の正規形は以下のような性質を持っている。

補題 22 (平坦化の一意性) 任意の項$s \in \mathcal{T}(\mathcal{F} \cup \mathcal{V})$ に対し、$Fl(s)$ は一意に定まる *[BP85]*。
□

補題より、平坦化は正規形を持つ。任意の項t の平坦化による正規形を $Fl(t)$ とする。

例 23 (平坦化の例) $f \in \mathcal{F}_{\mathcal{AC}}, a, b \in \mathcal{F}$

$$f(a, f(b, b)) \rightarrow_{Fl} f(a, b, b) \leftarrow_{Fl} f(f(a, b), b)$$

□

項の平坦化の正規形を RPO で比較する手法は AC に関する適合性を満たすことが知られている [BP85]。しかし、平坦化の正規形を RPO で比較する方法は、以下のように単調性に関する反例が存在する [BP85]。

例 24 (単調性に関する反例) $f \in \mathcal{F}_{\mathcal{AC}}, h \in \mathcal{F}, f >_{\mathcal{F}} h$、$C[\ ] = f(a, \Box)$ とし、

$$\begin{cases} s = f(a,a) \\ t = h(a,a) \end{cases}$$

とする。$Fl(s) = f(a,a) >_{RPO} h(a,a) = Fl(t)$ だが

$$\begin{cases} C[t] = f(a, h(a,a)) \\ C[s] = f(a, f(a,a)) \end{cases}$$

について $Fl(C[t]) = f(a, h(a,a)) >_{RPO} f(a,a,a) = Fl(C[s])$ となり単調性を満たさない。□

これは s, t を RPO で比較する過程で現れる $\underline{f}(a,a) >_{RPO} h(a,a)$ において順序のポイントとなる下線部の f が

$$f(a, \underline{f}(a,a)) \rightarrow_{Fl} f(a,a,a)$$

のように平坦化の過程で縮退してしまい、順序の決定に関与できなくなってしまうことが原因である。

3.2.2 解釈操作 (Interpretation) による手法

関数記号の集合 $\mathcal{F}$ に、最低の順序を持つ関数記号 $f_\perp$ （$f > a > f_\perp$）を導入すると、平坦化による関数記号の縮退は以下のように表現することができる。

$f(a, \underline{f}(a,a)) \rightarrow_{Fl} f(a, \underline{f_\perp}(a,a))$

本来は f によって項の順序が決定されていたが、$f_\perp$ に変化したために単調性において順序の逆転が起きていると考えられる。

$f \in \mathcal{F}_{\mathcal{AC}},\ h,\ a \in \mathcal{F},\ f >_{\mathcal{F}} h >_{\mathcal{F}} a,\ \ s = f(a, f(a,a)),\ t = f(a, h(a,a))$ とすると、上記の表現を用いれば、

$$\begin{cases} s = f(a, f(a,a)) \rightarrow_{Fl} f(a, f_\perp(a,a)) \\ t = f(a, h(a,a)) \end{cases}$$

となる。RPO で 2 項を比較すると、変形前は $f(a,a) >_{RPO} h(a,a)$ で $s >_{RPO} t$ が成立していたが、変形後は $h(a,a) >_{RPO} f_\perp(a,a)$ により $t >_{RPO} s$ となってしまう。そこで、そ

のような状態を回避するため、上記の例の h のように fが $f_\perp$に変化することにより、相対的に順序に影響を受ける fより小さい順序の関数記号に注目する。そして、fの変化と同様な変化を h にも起こして、順序の逆転を回避する方法が考えられている。それが解釈操作による手法である [RN93]。

この方法は平坦化に加えて、最外の関数記号と内部の部分項の関数記号の順序に注目して変形を行なう方法である。ただし、関数記号の順序は全順序とする [RN93]。

定義 25 (解釈操作の規則) $s \in \mathcal{T}(\mathcal{F})$ について、その解釈操作 $I(s)$ を以下のように定義する。

- 平坦化の規則を Flとする。
- 以下の規則を R_Iとする。

  $f(x_1, \cdots, x_m, g(t_1, \cdots, t_r), y_1, \cdots, y_n) \to f(x_1, \cdots, x_m, t_j, y_1, \cdots, y_n)$

  ただし、$f \in \mathcal{F}_{\mathcal{AC}}, > \in \mathcal{F}, f \succ_F g,\ m + n \geq 1,\ t_1, \cdots t_r \in \mathcal{T}(\mathcal{F})$, かつ $t_j = max_I\{t_1, \cdots t_r\}$ とする。$max_I\{t_1, \cdots, t_n\}$ は、後述する順序$>_I$によって $t_1, \cdots, t_n$ の中から最大の順序の項を選択する。また $arity(g) = 0$、すなわち gが定数 c の時には、

  $f(x_1, \cdots, x_m, c, y_1, \cdots, y_n) \to f(x_1, \cdots, x_m, \perp, y_1, \cdots, y_n)$

  とし、$\perp$は最小の順序の関数記号とする。

□

このような解釈操作の規則 $Fl \cup R_I$によって項の解釈操作を以下のように定める [RN93]。

定義 26 (解釈操作) 任意の基礎項 $s \in \mathcal{T}(\mathcal{F})$ について、解釈操作の標準形を $I(s)$ とすると、$I(s)$ は解釈操作の規則 $R_F \cup R_I$を次の戦略に従って適用することにより得られる。

1. 規則の適用は最左最内戦略をとる。
2. Flと R_Iでは Flを優先して適用する。

□

このようにして求まった $I(s)$ は s に対して一意に決定する。さらに解釈操作の変形が同一になった時の場合も考えて順序を以下のように定義する。

定義 27 (解釈操作を利用した順序) $s, t \in \mathcal{T}(\mathcal{F})$ とし、s, t に Flのみを適用した標準形をそれぞれ $s' = f(s_1, \cdots, s_m),\ t' = g(t_1, \cdots, t_n)$ とする。また、任意の $f \in \mathcal{F}_{\mathcal{AC}}$は、$\tau(f) = mult$ 任意の $h \notin \mathcal{F}_{\mathcal{AC}}$は$\tau(h) = lex$ とする。このとき $s >_I t$ が成立するのは

- $I(s) >_{RPOS} I(t)$。　または、
- $I(s) = I(t)$ かつ

  1. $top(s) \in \mathcal{F}_{\mathcal{AC}}$でかつ、$[s_1, \cdots, s_m] \gg_I [t_1, \cdots, t_n]$。または、
  2. $top(s) \notin \mathcal{F}_{\mathcal{AC}}$でかつ、$[1, \cdots, m]$ のある i と、$j < i$ を満たす任意の jについて $s_j =_{AC} t_j$で、$s_i >_I t_i$を満たす。

□

解釈操作は以下の定理が成立する。

定理 28 (解釈操作を利用した順序の性質 [RN93) *]*
解釈操作を利用した順序は AC に関する適合性を満たし、かつ単純化順序の性質を満たす。

□

よって定理 20 より、AC-TRS $R = R' \cup AC$で R'の任意の規則 $l \to r$について、解釈操作を利用した順序$>_I$で $l >_I r$が成立しているなら、R の停止性が成立する。

例 29 (解釈操作による比較の例) $h, a, b \in \mathcal{F},\ f \in \mathcal{F}_{\mathcal{AC}},$ 関数記号の順序を $a >_{\mathcal{F}} f >_{\mathcal{F}} h >_{\mathcal{F}} b$ としたとき、

$$\begin{cases} s = f(a, f(a, a)) \\ t = f(a, h(a, a)) \end{cases}$$

について項の変形を行なうと

$$\begin{aligned} s &: f(b, f(a, a)) \to_{Fl} f(b, a, a) \to_{R_I} f(\bot, a, a) \\ t &: f(b, h(a, a)) \to_{R_I} f(\bot, a) \end{aligned}$$

となる。従って

$$\begin{cases} I(s) = f(\bot, a, a) \\ I(t) = f(\bot, a) \end{cases}$$

となり、$I(s) >_{RPOS}$ *I(t)* が得られる。

□

第 4 章

積み上げ化を基本にした順序について

4.1 積み上げ化

前章の平坦化は AC 適合性を満たすという利点がある反面、AC 関数記号の縮退により単調性に反例が生じる欠点があった。本研究では、AC 適合性を満たす新しい変形である積み上げ化を提案する。ただし、本章で扱う項は、断りがない限り全て基礎項とする。

4.1.1 積み上げ化について

従来までの手法では、AC に関する適合性を成立させるために、AC 同値類の代表元を生成する方法として平坦化を用いてきた。しかし、前章で示したとおり関数記号の縮退によって、順序に矛盾が生じることが判明している。そこで本論文では、AC 同値類の代表元を生成する新しい方法として、AC 関数記号の縮退に対処した積み上げ化を提案する [NST95]。

定義 30 (積み上げ化) 関数記号 $f \in \mathcal{F}_{\mathcal{AC}}$、任意の項の多重集合 $X,\ Y\ \subset \mathcal{T}(\mathcal{F})$ によって積み上げ化の変形を以下のように定義する。

$f(X, f(Y)) \rightarrow_{St} f(f(Z))$

ここで、$Z = X \cup Y$ である。また、項 t の積み上げ化による正規形を $St(t)$ と表記する。$\underbrace{f(\cdots(f}_{n}(T)\cdots)$ は、$f^n(T)$ ($n \geq 1$)と表記する。ここで、f^n の引数は $n+1$ である。 □

平坦化と積み上げ化の大きな相違点は、平坦化が内部の AC 関数記号 f が変形の結果縮退するのに対し、積み上げ化では内部に連続して現れる f を項の最外に移動し、関数記号の出現数の情報を f の次数という形で保持している点にある。

例 31 (積み上げ化の例) $f \in \mathcal{F}_{\mathcal{AC}},\ a, b, c \in \mathcal{F}$ とすると $t = f(a, f(f(a, b), a))$ に対し、積み上げ化の正規形は以下のようにして求まる。

$$f(a, f(f(a, b), c)) \rightarrow_{St} f^2(a, f(a, b, c)) \rightarrow_{St} f^3(a, a, b, c)$$

よって $St(t) = f^3(a, a, b, c)$ である。

□

積み上げ化は以下のような性質を満たす。

補題 32 (積み上げ化の正規形) 平坦化による変形は以下の２つの性質を満たしている。

1. 停止性を満たす（積み上げ化の書き換えは必ず終了する）。
2. 合流性を満たす（適用の仕方が何通りあっても、書換えが進むと同じ形に書き換えられる）。

□

よって、積み上げ化による正規形が一意に決まる。

補題 33 (積み上げ化の AC に関する適合性) $s,\ s',\ t,\ t' \ \in \mathcal{T}(\mathcal{F})$ が与えられ、$s =_{\mathcal{AC}} s',\ t =_{\mathcal{AC}} t'$が成立しているときに、$s > t$ を $St(s) >_{RPO} St(t)$ と定義すると

$$s > t \Rightarrow s' > t'$$

が成立する。 □

<**証明**> 積み上げ化の定義より、積み上げ化の結果 $St(s) \equiv St(s'), St(t) \equiv St(t')$ が成立するので、$St(s) > St(t)$ ならば $St(s') > St(t')$ が成立する。 □

積み上げ化を用いると、平坦化における単調性の反例は以下のように解消される。

例 34 (平坦化での反例への積み上げ化の適用) $f \in \mathcal{F}_{\mathcal{AC}},\ h, a \in \mathcal{F},\ f >_{\mathcal{F}} h >_{\mathcal{F}} a,\ C[\,] = f(a, \Box)$ とし、

$$\begin{cases} s = f(a, a) \\ t = h(a, a) \end{cases}$$

とする。ここで $St(s) = f(a, a) >_{RPO} h(a, a) = St(t)$ となる。このとき、

$$\begin{cases} C[s] = f(a, f(a, a)) \\ C[t] = f(a, h(a, a)) \end{cases}$$

について $St(C[s]) = f^2(a, a, a) >_{RPO} f(a, h(a, a)) = St(C[t])$ となり単調性を満たす。□

この例では平坦化においては縮退した fが、積み上げ化では次数の情報として残り、順序の決定に関与できるため単調性が成立する。

積み上げ化の性質の一つとして、積み上げ化を行なった項は行なう前の項より順序が大きくなることがあげられる。

補題 35 (積み上げ化の性質) 任意の項 t について $St(t) \neq t$ であるなら

$$St(t) >_{RPO} t$$

が成立する。 □

例 36 (積み上げ化による順序の変化) $f \in \mathcal{F}_{\mathcal{AC}}, a \in \mathcal{F}$、$f >_{\mathcal{F}} a$ としたとき、

$$\begin{cases} t & = f(f(a,a),a) \\ St(t) & = f^2(a,a,a) \end{cases}$$

より

$$f^2(a,a,a) >_{RPO} f(f(a,a),a)$$

が成立する。

□

4.1.2 積み上げ化の欠点

積み上げ化は前節の平坦化の欠点を補っているが、以下のような別の欠点を持つ。$s > t$ が成立していて $C[s], C[t]$ という項で単調性を考える場合、$C[s], C[t]$ の順序関係は文脈部分 $C[\ ]$ は順序に関係せず（例えば RPO の定義のように相殺して）、s, t 部分で独立に順序が決定できることが望ましい。しかし、積み上げ化の結果文脈部分と s, t の関係が変化してしまい、$s > t$ の関係を独立に適用することは不可能となることがある。

例えば、例 34 の $C[\ ] = f(\underline{a}, \Box)$ に注目すると、$\underline{a}$の上には fだけが出現している。しかし、$f(\underline{a}, f(a,a))$ においては積み上げ化の結果

$$f(a, f(a,a)) \rightarrow_{St} \underline{f^2}(a,a,a)$$

となり、$\underline{f^2}$のように文脈の形自体が変化する。このような fを一種の重み付けと解釈すると、積み上げ化の結果$\underline{a}$の重み付けは、$C[\ ]$ においては fであったものが、$St(f(\underline{a}, f(a,a))) = f^2(\underline{a}, a, a)$ においては f^2へと変化することになる。すると、単調性に関して以下のような反例が生ずる。

例 37 (積み上げ化の単調性に関する反例 (1)) $f \in \mathcal{F}_{\mathcal{AC}},\ g, h, a \in \mathcal{F}$、$g >_{\mathcal{F}} f >_{\mathcal{F}} h >_{\mathcal{F}} a$ のとき, $s = h(a, f(a,a)),\ t = f(a,a), C[\ ] = f(g(a,a), \Box)$ とする。すると

$St(s) = h(a, f(a,a)) >_{RPO} f(a,a) = St(t)$ だが

$$\begin{cases} St(C[s]) = f(g(a,a), h(a, f(a,a))) \\ St(C[t]) = St(f(g(a,a), f(a,a))) = f^2(g(a,a), a, a) \end{cases}$$

という積み上げ化が行なわれるため、
$St(C[t]) = \underline{f^2(g(a,a)}, a, a) >_{RPO} \underline{f(g(a,a)}, h(a, f(a,a))) = St(C[s])$
が成立して単調性を満たさない。

□

この例では文脈 $C[\ ]$ 中の $g(a,a)$ の重み付けに注目すると、s 側では積み上げ化が起こらず重み付けが f のまま変化しないのに対し、t 側では積み上げ化によって f^2 に変化している。すると、本来は多重集合順序によって相殺されるはずの $g(a,a)$ が、文脈の変形による f の重み付けの差によって順序全体に影響を及ぼし、順序の逆転が生じていることがわかる。

例 38 (積み上げ化の単調性に関する反例 (2)) $f \in \mathcal{F}_{\mathcal{AC}},\ g, h, a \in \mathcal{F}$、$g >_{\mathcal{F}} f >_{\mathcal{F}} h >_{\mathcal{F}} a$ のとき, $s = h(a, f(a,a)),\ t = f(a,a), C[\ ] = f(a, \square)$ とする。すると
$St(s) = h(a, \underline{f(a,a)}) >_{RPO} f(a,a) = St(t)$ だが

$$\begin{cases} St(C[s]) = f(a, h(a, \underline{f(a,a)})) \\ St(C[t]) = St(f(a, f(a,a))) = f^2(a,a,a) \end{cases}$$

と積み上げ化が行なわれるので $St(C[t]) = f^2(a,a,a) >_{RPO} f(a, h(a, f(a,a))) = St(C[s])$ となり単調性を満たさない。

□

これ例では、s 内部の$\underline{f(a,a)}$が t と部分項関係となっているため $s > t$ が成立していたが、積み上げ化により t 側が $C[\ ]$ の a を取り込み、$f(a,a,a)$ に変化したために部分項関係が崩れてしまい順序の逆転が起こっている。例 37 とは異なり、文脈中の要素 a が項全体に影響をあたえるわけではないが、a が s, t の内部に取り込まれることにより、s, t の順序を微妙に変化させてしまう。

4.1.3 反例について

補題 35 より、積み上げ化は元の項よりも順序を大きくする作用がある。前節での反例では、$s >_{RPO} t$ が成立している場合に単調性を考えると、$C[s]$ で積み上げ化が起こらず、$C[t]$ で積み上げ化が起こっている。そのため、$C[t]$ の順序が大きくなって、順序が逆転してしまう。しかし、常に順序の逆転が起こるわけではなく、逆転が生ずるためにはいくつかの条件が必要である。以下では、これらの条件について考察する。

まず、$s,\ t,\ C[\ ] = f(X, \square)$ として、以下の条件を仮定する。

1. $f \in \mathcal{F}_{\mathcal{AC}}$

2. $top(s) \neq f$
3. $top(t) = f$

すなわち、s は文脈に関して積み上げ化が起きないが、t では変形が起きる場合である。

次に、文脈 $C[\,]$ について反例の条件を考察する。まず、任意の項 $s \in \mathcal{T}(\mathcal{F})$ について、$max(s)$ を、s に現れる関数記号の中で最大の順序を持つものとすると、以下のような補題が成立する。

補題 39 (RPO と最大の関数記号) 任意の項 $s, t \in \mathcal{T}(\mathcal{F})$ について、

$$s >_{RPO} t \rightarrow max(s) \geq_{\mathcal{F}} max(t)$$

□

補題 39 より、$s >_{RPO} t$ が成立する時、$max(s) \geq_{\mathcal{F}} max(t)$ が成立する。このとき、反例の条件を $max(s)$ について以下のように場合分けする。

1. $max(s) >_{\mathcal{F}} f$ の場合。

   s, t よりも順序の大きい部分項 $X_i \in X$ が文脈に含まれ、積み上げ化で f の重み付けに差がでたために順序関係が崩れる（例 (1)）。

2. $max(s) = f$ の場合。

   s 内部に f を含み、t との比較では f が順序に関与しているとする。しかし、積み上げ化の結果 s 側では積み上げ化が起こらず、t 側では、積み上げ化によって X を f 以下の部分項の中に取り込んでいるために、順序が逆転する（例 (2)）。

上記の 1 の重み付けと 2 の取り込みは本質的には同じ現象である。

$max(s) >_{\mathcal{F}} f$ の場合、X の要素を X_i とすると $C[s], C[t]$ の順序は、最終的に s, t の順序に帰着すべきであるが、文脈に $X_i >_{RPO} s, t$ を満たす X_i が含まれると、項の順序は X_i への f による重み付けで順序が決定され、s, t の情報が反映されなくなってしまう。

例 40 (反例の条件 1 について) $g, h \in \mathcal{F},\ f \in \mathcal{F}_{\mathcal{AC}},\ g >_{\mathcal{F}} f >_{\mathcal{F}} h$、$s = h(S),\ t = f(T),\ C[\,] = f(g(X), \Box)$ とする。また、$g(X) >_{RPO} h(S),\ g(X) >_{RPO} f(T)$ と仮定する。すると、

$St(s) > St(t)$ から、$h(S) >_{RPO} f^n(T')$ であるが

$$\begin{cases} St(C[s]) = \underline{f(g(X)}, h(S)) \\ St(C[t]) = St(f(g(X), f(T))) = \underline{f^{n+1}(g(X)}, T') \end{cases}$$

より

$$St(C[t]) = f^n(g(X), T') \gg_{RPO} f(g(X), h(S)) = St(C[S])。$$

よって、単調性を満たさない。

□

$max(s) = f$の場合は、s 内部に出現する fが t に出現する fと比較する上で対応して、両項の f以下に出現する部分項で順序がついていた。しかし、$C[t]$ 側で積み上げ化が起こり、文脈中の要素 X_iを f以下に取り込むために、順序の逆転が生ずる。

例 41 (反例の条件 2 について) $s = h(f^n(S)),\ t = f^m(T),\ C[\ ] = f(g(X), \Box),\ f >_{\mathcal{F}} h,\ f \in \mathcal{F}_{\mathcal{AC}}$とすると、$s > t$ より、$f^n(S) >_{RPO} f^m(T)$ が成立している。単調性について、

$$St(C[t]) = f^{n+1}(X, T) >_{RPO} f(X, h(f^m(S))) = St(C[t])$$

が成立するのは、RPO の定義から部分項に注目すると、$f^n(X, T) >_{RPO} h(f^m(S))$ の場合である。これは

$$f^n(\underline{X}, T) >_{RPO} f^m(S)$$

となっていて, 取り込まれた X について、例えば $n = m, X \equiv_{ms} S$とすると、順序の逆転が成立する。

□

例 41 が例 40 と異なるのは、例 40 では文脈の要素 X_iについて $X_i >_{RPO} s, t$ が成立しているが、例 41 では必ずしも成立している必要がない点である。例えば、例 40 で $n = m$ と仮定すると、$S \gg_{RPO} T \gg_{RPO} X_i$で、かつ $[X_i] \cup T \gg_{RPO} S$という場合が考えられる。

4.2 再編化の定義と問題点

以下の節では前節の例 40,41 を基にして、その解決方法について検討を行なう。現在のところ、上の 2 つの反例に完全に対処できる手法は発見されていない。そこで現在まで考案した手法による順序とその反例、および問題点について示す。

4.2.1 再編化

前節の例 40 に注目すると、本来 s, t とは関係のない $C[s], C[t]$ に含まれる共通の要素 X_iに、重み付けに差がでることが問題である。そこで、原因となる要素 X_iが $C[s], C[t]$ の中に共通して存在している点を利用して, $C[s], C[t]$ の 2 項を比較するときに両項の部分項の集合からその共通部分を探せば、X_iに該当する要素が発見できる。このような X_iが

発見できれば、fの重み付けを操作することにより、両項のX_iの候補となる項への重み付けを同一にすることが可能である。

また、例 41 では、Xの取り込みが問題となっている。これはs, t との比較の際には存在しない$C[\ ]$ の Xの要素が取り込まれているためである。そこで、積み上げ化によって最外に集められた AC 関数記号 fを部分項に再分配し、その際に Xへの分配を避けることを考える。この方法を用いることにより、例 41 のような Xの取り込みを回避することが期待できる。そこで、以下のような再編化という操作を考える。再編化は積み上げ化によって最外に集められた AC 関数記号を、部分項へ分配して項を再構成する変形である [NST96b]。

定義 42 (再編化) 2つの項$s = f^n(s_1 \cdots s_{n+1})$ と$t = f^m(T)$ を比較する場合に、$n > m > 0$ とすると $f^n(s_1, \cdots, s_{n+1}) \rightarrow_{Re} f^m(f^{n-m}(S_1), S_2)$。ただし、$S_1 \cup S_2 = [s_1, \cdots, s_n]$ とし、$\forall s_1 \in S_1, \forall t_2 \in S_2$について、$s_1 \geq_{RPO} t_2$が成立する。ただし、この s_1, t_2の比較で最外の関数記号が AC 関数記号で、その次数が異なる場合には s_1, t_2に再編化を行なって順序を再帰的に決定する。 □

再編化は、同じ最外関数記号で次数の異なる項を比較するときに高い次数を低い次数に合わせ、残りの次数を部分項に分配する方法である。

例 43 (再分配の例) $s = h(f^n(S)),\ t = f^m(T),\ C[\ ] = f(g(X), \Box),\ f >_{\mathcal{F}} h,\ f \in \mathcal{F}_{\mathcal{AC}}$とすると、$s > t$ より、$f^n(S) >_{RPO} f^m(T)$ が成立している。単調性を考えると、

$$\begin{cases} St(C[s]) = f(g(X), h(f^m(S))) \\ St(C[t]) = f^{n+1}(g(X), T) \end{cases}$$

$C[t]$ に再編化を行なうと、$f^{n+1}(g(X), T) \rightarrow_{Re} f(g(X), f^n(T))$ となる。このとき RPO で $f(g(X), h(f^m(S)))$ と $f(g(X), f^n(T))$ の比較を行なえば、仮定 $s >_{RPO} t$ より、
$f(g(X), h(f^m(S))) >_{RPO} f(g(X), f^n(T))$ となり単調性を満たす。
□

しかしながら、この操作ではXの要素がS_1に分類される可能性が高く、また、例 40 での X_iに対する重み付けの差が避けられない。また例 41 に関しても、Xの要素が s, t よりも大きい順序であると、元の t の要素よりも Xの要素に対して fの分配を行なうため、やはり取り込みが起こる可能性がある。

例 44 (再編化の欠点) $s = h(f^n(S)),\ t = f^m(T),\ C[\ ] = f(g(X), \Box),\ g >_{\mathcal{F}} f >_{\mathcal{F}} h,\ f \in \mathcal{F}_{\mathcal{AC}}$とする。このとき $s > t$ より、$f^n(S) >_{RPO} f^m(T)$ で、かつ文脈の $g(X)$ について、$[g(X)] \gg_{RPO} S \gg_{RPO} T$が成立していると仮定する。すると、

$$\begin{cases} St(C[t]) = f(g(X), h(f^m(S))) \\ St(C[t]) = f^{n+1}(g(X), T) \end{cases}$$

となるので、$C[t]$ に再編化を行なうと定義より $f^{n+1}(g(X), T) \rightarrow_{Re} f(t_i, f^n(T - [t_i] \cup [g(X)]))$ が得られる。ただし、$t_i \in T$で $g(X) >_{RPO} t_i$。

このとき、RPOで $f(\underline{g(X)}, h(f^m(S)))$ と $f(t_i, \underline{f^n(T - [t_i] \cup [g(X)])})$ の比較を行なえば、$f(t_i, \underline{f^n(T - [t_i] \cup [g(X)])}) >_{RPO} f(\underline{g(X)}, h(f^m(S)))$ となり、単調性を満たさない。

□

この例は、両項に共通した部分項に同一の重み付けを行なうという再編化の目的に反している。そこで、再編化に次のような戦略を導入する。

定義 45 (差分戦略) $n > m > 0$, $f \in \mathcal{F}_{\mathcal{AC}}$とし、$S, T$は任意の多重集合、$ST = S \cap T$ とする。

$f^n(S)$ と $f^m(T)$ を比較するとき、以下の方法で比較する戦略を差分戦略という。

1. STが存在するとき

   $f^n(S)$ の再編化において S_2に STの要素を優先的に集める。ただし、STの要素数が S_2の要素数を上回った時は STの要素を RPO で順序づけして、順序の小さい要素から順に ST の要素数と S_2の要素数の差分だけ S_1に集める。

2. STが存在しないとき

   通常の再編化を行なう。

□

例 40、41 の反例において原因となる Xの要素 X_iが S_2に属すことにより例 40 での X_i への重み付けは、両項で低い方の項の次数で一致する。また例 41 の取り込みの問題も、X_iが S_2に優先的に送られるので、順序の逆転を起こすような X_iが fの分配に関連することはない。

差分戦略を組み込んだ再編化を改めて以下のように定義する。

定義 46 (差分戦略を含む再編化 (Reconstruction)) $>$を基礎項上の全順序とする。このとき $Re(s,\ t,\ >)$ を $s' > t'$と定める。ただし、$s',\ t'$は以下の変形で定める。

$s = f^m(S),\ t = f^n(T),\ \ f \in \mathcal{F}_{\mathcal{AC}}$のとき

- $m \geq n$: $s' = f^m(f^{n-m}(S_1), S_2),\ t' = t$。ただし、多重集合 S_1, S_2（それぞれ要素数 $n - m + 1,\ m$）は Sの中から以下のように定める。

1. s, t のそれぞれの部分項の集合 S, Tが共通部分 STをもつならば、Sを $S' \cup ST$ とし S', STのそれぞれの多重集合について順序の大きい順に並べ直す。その上で STの順序の大きい要素を優先的にに S_2側へ選択していく。
   (a) STの要素数が S_2の要素数を上回ったなら、順序の大きい要素から順に S_2 へ選択していき、残りの要素を S_1とする。
   (b) STの要素数が S_2の要素数を下回ったなら、S'の要素を順序の大きい要素から順に S_1に選択し、残りの要素を S_2とする。
2. 共通部分がない場合は、Sを順序の大きいものから順番に $n-m+1$ 個を S_1に選択し、残りを S_2とする。

- $m < n$：$s' = s,\ t' = f^n(f^{n-m}(T_1), T_2)$ は $s,\ t$ を入れ換えて上と同様に変形する。

これ以外のときには $s' = s,\ t' = t$。

□

このような再編化を用いた再編化順序を以下のように定義する。

定義 47 (再編化順序 (RCO)) 任意の２項$s, t \in \mathcal{T}(\mathcal{F})$ とし、$St(s) = h^m(s_1, \cdots, s_{m+1}),\ St(t) = g^n(t_1, \cdots, t_{n+1})$ とする。このとき、$s >_{RCO} t$ を以下で定義する。

1. $top(St(s)) = top(St(t)) \in \mathcal{F}_{\mathcal{AC}}$、かつ $m \neq n$ の場合。$Re(St(s),\ St(t),\ >_{RCO})$。
2. それ以外の場合。
   (a) $h >_{\mathcal{F}} g$の場合、$[s] \gg_{RCO} [t_1, \cdots, t_{n+1}]$。
   (b) $h = g$の場合、 $[s_1, \cdots, s_{m+1}] \gg_{RCO} [t_1, \cdots, t_{n+1}]$。
   (c) $h <_{\mathcal{F}} g$の場合、$[s_1, \cdots, s_{m+1}] \gg_{RCO} [t]$。

□

上記の定義が曖昧な点を持たないことは、関数記号やその次数の場合分けにより簡単に示すことができる。RCO では、再編化が起きない場合にはRPO と同様に比較を行なう。また、内部の部分項の比較で RCO の定義の 1 の条件を満たした場合には、再帰的に再編化が行なわれる。

補題 48 (再編化が起きない RCO) 再編化順序において再編化が起きない項の比較は、RPO で比較することと同値である。

例 49 (差分戦略による比較) $f \in \mathcal{F}_{\mathcal{AC}},\ g, h, a \in \mathcal{F},\ g >_{\mathcal{F}} f >_{\mathcal{F}} h >_{\mathcal{F}} a$ のとき $s = h(f(a,a)),\ t = f(a,a),\ C[\ ] = f(g(a,a), \Box)$ とすると、$St(s) = s, St(t) = t$ より $h(f(a,a)) >_{RCO} f(a,a)$。一方、$C[s], C[t]$ では、

$$\begin{cases} St(C[s]) = f(g(a,a), h(f(a,a))) \\ St(C[t]) = St(f(g(a,a), f(a,a))) = f^2(g(a,a), a, a) \end{cases}$$

であるので、再編化を行ない $C[t]$ 側で f の再分配を行なう。

ここで $[g(a,a), h(f(a,a))]$ と $[g(a,a), a, a]$ の共通部は $g(a,a)$ なので、再編化によって $f^2(g(a,a), a, a) \rightarrow_{Re} f(f(a,a), g(a,a))$ となる。よって $f(g(a,a), h(f(a,a))) >_{RCO} f(f(a,a), g(a,a))$ となり、$C[s] >_{RCO} C[t]$ が成立する。

□

上記の例のように、$X = [g(a,a)]$ には両項とも f しか重み付けが行なわれていないので、順序の整合性を崩さない。

4.2.2 再編化順序の性質

RCO で定義される項上の順序が AC に関する適合性をもつ単純化順序となるためには、以下の性質を満たす必要がある。

1. AC に関する適合性
2. 半順序
   (a) 非反射律
   (b) 推移律
3. 部分項性
4. 単調性

以下では１～４についてそれぞれ考察を行なう。

補題 50 (AC に関する適合性) $s =_{AC} s',\ t =_{AC} t',\ s >_{RCO} t \Rightarrow s' >_{RCO} t'$。 □

<**証明**> 積み上げ化の性質より、$St(s) \equiv St(s'),\ St(t) \equiv St(t')$ が得られるので明らか。
□

補題 51 (非反射律) 任意の項 t について、$t >_{RCO} t$ は成立しない。

□

<**証明**>　t 自身と比較すると、最外の関数記号が AC 関数記号であるかどうかとは無関係に再編化は起きない。よって最外の関数記号の定義の 2 に従い、部分項同士の比較となる。また、部分項は一致するので、多重集合順序の定義により順序はつかない。

□

補題 52 (部分項性) $t \in \mathcal{T}(\mathcal{F}),\ C[\ ] = f(x, \Box),\ X$は任意の多重集合とする。この時 $C[t] >_{RCO} t$。

□

<**証明**>　$top(C[\ ])$ および $top(t)$ が $\mathcal{F}_{\mathcal{AC}}$に属するかどうかによって場合分けを行ない、$|t| + |C[t]|$ のサイズに関する帰納法で証明する。サイズが３の時、t は定数、$C[t]$ は関数記号と１つの引数をもつ項であるから積み上げ化、再編化が起きないので RPO での比較と同様である。よって、RPO の部分項性により成立する。

サイズが３より大きい場合、$St(t)$ と $St(C[t])$ について再編化が起きるかどうかについて場合分けを行ない証明する。

1. $top(C[\ ]) = top(t) = f \in \mathcal{F}_{\mathcal{AC}}$の場合。

   $St(t) = f^n(T),\ st(C[t]) = f^{n+m+1}(X', T),\ t$ の要素数 $n+1,\ X'$の要素数 $m+1$ とすると、再編化により $St(C[t]) = f^{n+m+1}(X', T) \rightarrow f^n(f^{m+1}(S_1), S_2)$ に変形される。すると、$f^n(T),\ f^{n+m+1}(X', T)$ に共通する要素はTであるから、定義にしたがって部分項集合を分類する。ただし、S_1側は $m+1$ 個、S_2側は n 個の要素からなるので、$S_1 = X' \cup [t_i],\ S_2 = T - [t_i]$ となる。

   定義より $f^n(f^{m+1}(X \cup [t_i]), T - [t_i])$ と $f^n(T)$ を比較すると $[f^{m+1}(X' \cup [t_i]), T - [t_i]]$ と t の比較となる。多重集合順序の定義より、$[f^{m+1}(X' \cup [t_i])]$ と、$[t_i]$ の比較となるが、帰納法の仮定により $[f^{m+1}(X' \cup [t_i])] \gg_{RCO} [t_i]$ が成立する。
2. それ以外の場合。

   $St(C[t])$ と $St(t)$ の間には再編化が起きないため、定義に従い部分項の比較になり、帰納法の仮定により成立する。

□

補題 53 (単調性) 任意の項 $s,\ t,\ C[\ ] = f(X, \Box)$, および任意の多重集合 Xについて、$s >_{RCO} t \Rightarrow C[s] >_{RCO} C[t]$。

□

<**証明**>　$top(s),\ top(t),\ top(C[\ ])$ によって場合分けを行なう。ここでは特に $top(s) = top(t) = top(C[\ ]) = f \in \mathcal{F}_{\mathcal{AC}}$の場合で、$St(s) = f^m(S),\ St(t) = f^n(T),\ m > n$ のときの証明を示す。

$St(C[s]) = f^{l+m+1}(X', S), St(C[t]) = f^{l+n+1}(X', T)$ とすると

$s >_{RCO} t$ の関係より $[f^{m-n}(S_1)] \cup S_2 \gg_{RCO} T$が成立する。このとき、再編化により $f^{l+m+1}(X', S)) \rightarrow f^{l+n+1}(f^{m-n}(XS_1), XS_2)$。すると S_1と XS_1の要素数が等しいので、再編化の定義より $S_1 = XS_1,\ [S_2, X'] = XS_2$ が成立する。

よって RCO の定義によって $[f^{m-n}(S_1)] \cup X' \cup S_2$と $X' \cup S_2$の比較となり多重集合順序により、$C[s] >_{RCO} C[t]$ が成立する。他の場合についても同様に示すことができる。　□

4.2.3　推移律の証明について

前節までに、AC に関する適合性、非反射律、部分項性、単調性について証明を行なった。残る推移律について証明が完成すれば RCO は AC 適合性を満たす単純化順序であることが示される。

しかし、現在のところ RCO は推移律が成立するかどうかが未確認である。これは、再編化自体の推移律の証明が複雑なことに加え、差分戦略によって比較する項に共通部がどのように存在するかなど、場合分けが非常に多岐に分かれるためである。

項 s,t,u に対し $s > t$、$t > u$ とすると

1. 再編化が起こるかどうか（２項の最外の AC 関数記号が一致するか）。

   (a) s,t,u いずれも一致しない。
   (b) s,t のみ、または t,u のみ、または s,u のみで一致。
   (c) s,t,u で一致（さらに、再編化がどの項に起こるかの組合せを考える必要がある）。

2. 差分戦略が適用されるか（部分項に共通部が存在するか）。

   (a) s-t 間、t-u 間、s-u 間で全く存在しない。
   (b) s-t 間のみ、または t-u 間のみ、または s-u 間のみで存在。
   (c) s-t 間と t-u 間、または s-t 間と s-u 間、または t-u 間と s-u 間で存在。
   (d) s-t-u 間で存在。

この 1 と 2 はそれぞれ独立の条件であり、さらにそれぞれ細かい場合分けが存在する。例えば、1-(c) について各 s, t, u の次数を n, m, lとすれば、その大小関係によってどの項に再編化が適用されるか決まるので、n, m, lの大小関係についての場合分けが存在する。このような場合分けの多様さが推移律の証明を困難なものにしている。

4.3 SDL 順序の定義と問題点

前節での方法は、比較する項ごとにそれぞれの標準形が異なるために、証明が非常に繁雑になってしまった。そこで、本節では RPO を基本として一意な標準形が得られる変形を検討する。

4.3.1 分離操作

再編化においては、例 40、41 の反例の原因となる X を 2 項の共通部分から特定し、その重み付けを同一にすることによって、単調性の反例を解消した。しかし、この方法では比較する 2 項の組合せによって変形が異なるので、証明が複雑になってしまう。そこで反例の X となる項の、もう少し具体的な条件について考える。

例 40 では X が反例の原因となるのは、s, t よりも X のある要素 X_i が RPO で比較して大きい場合である。そこで反例が起きる前提条件として s, t について

- $s > t$
- $top(s) \neq f$
- $top(t) = f$
- $f \in \mathcal{F}_{\mathcal{AC}}$

という条件が考えられる。また、X_i が s, t より RPO で比較して大きくなるには、補題 39 より少なくとも X 内部に f、または f より大きい関数記号 g が含まれていなければならないということがわかる。g を含む X では、RPO に従えば f の影響は g で打ち消されると考えられる。

例 54 (g と f の関係) $g, h, a \in \mathcal{F}, f \in \mathcal{F}_{\mathcal{AC}}$ とし、関数記号の順序を $g >_{\mathcal{F}} f >_{\mathcal{F}} h >_{\mathcal{F}} a$ とする。このとき $g(a, a) >_{RPO} h(a, a)$ である。また同様に $g(a, a) >_{RPO} f(a, f(a, \cdots f(a, h(a, a)) \cdots))$ が成立し、f がどれだけ現れてもこの関係は変わらない。

もし X_i に含まれる最大の関数記号が f の場合には、例 54 とは異なり X 側でも f が打ち消されるため、f の出現数によって順序が変化する。また例 41 の反例の原因とも関連するので、ここでは f よりも真に大きい関数記号が含まれた場合に限定して対応を考える。

g を含む X には、上記のように f による重み付けを無くしても順序に影響を与えないことから、元の項から分離させて f の重み付けを回避する方法が考えられる。

定義 55 (分離操作) $t_1, \cdots, t_n \in \mathcal{T}(\mathcal{F}), f \in \mathcal{F}_{\mathcal{AC}}$ とする。このとき以下の操作を分離操作と呼ぶ。

$$f(t_1, \cdots, t_n) \rightarrow_{De} [f(t_{i1}, \cdots, t_{ip}, \perp), t_{j1}, \cdots, t_{jq}]$$

ただし
$[t_1, \cdots, t_n] =_{ms} [t_{i1}, \cdots, t_{ip}] \bigcup [t_{j1}, \cdots, t_{jq}]$
$[t_{i1}, \cdots, t_{ip}]$：内部に f より大きい関数記号が現れない項
$[t_{j1}, \cdots, t_{jq}]$：内部に f より大きい関数記号が現れる項
また $f(t_{i1}, \cdots, t_{ip}, \perp)$ に現れる $\perp$ は任意の t_{ik} を置き換えたものとする。

この操作により例 40 の反例の原因となる X は、f の重み付けから分離される。

例 56 (分離操作の例) $f \in \mathcal{F}_{\mathcal{AC}}$, $g, h, a, b \in \mathcal{F}$, $g >_{\mathcal{F}} a >_{\mathcal{F}} f >_{\mathcal{F}} >_{\mathcal{F}} h >_{\mathcal{F}} b$ とする。
$s = f(a, h(b, b)), t = f(g(b, b), b), u = f(b, b)$ それぞれに分離操作を行なうと、

$$\begin{cases} s \rightarrow_{De} [f(\perp, h\ (b, b)), a] \\ t \rightarrow_{De} [f(\perp, b), g(b, b)] \\ u \rightarrow_{De} [f(b, b)] \end{cases}$$

4.3.2 繰り上げ化

例 41 の反例について、s の内部に f が出現したとする。しかし、s 側で積み上げ化が行なわれないために順序の逆転が起こっている。これは s 側で出現する f と f の間に h が存在するため、s 側では積み上げ化が行なわれないためである。

例として $f \in \mathcal{F}_{\mathcal{AC}}$, $g, h \in \mathcal{F}$, $g >_{\mathcal{F}} f >_{\mathcal{F}} h_A >_{\mathcal{F}} a$ のとき、$f_1(h_A(f_2(X)), g(f(Y)))$ を考える。ただし、f_1, f_2 は区別をつけるため便宜上番号を振ったが、本来は両方とも f である。

項の経路を考えると、もし最外の f から内部の f までの間に $g > f$ を満たすような g が出現していれば、分離操作によってそのような部分項は以下のように分離される。

$$f_1(h_A(f_2(X)), \underline{g(f(Y))}) \rightarrow_{De} [f_1(h_A(f_2(X)), \perp), \underline{g(f(Y))}]$$

よって $g(f(Y))$ に現れる f はここでは考慮しない。

次に分離操作の対象外となる項に含まれる f について考える。$\underline{f_1(h_A(f_2}(X)), \perp)$ に注目すると $f_1 - h_A - f_2$ という経路が現れることになる。

f_1 と f_2 の間には h_A が存在するので、構文的には積み上げ化は適用できない。しかし、意味論的にこの部分に注目すると、２つの関数記号 f_1, f_2 は経路上の順序の観点から $f_1 - f_2$ という構造になっていると解釈することができる。なぜなら、もし、f 以上の関数記号 g と経路上で比較すれば、$g > f > h_A$ であるから h_A は順序に貢献せずに f_2 まで比較が進む。また、f より小さい関数記号 h と比較するなら、f_1 の時点で h による比較が終るので、やはり h_A は順序に貢献しない。よって意味論的には f_1 と f_2 は経路上は連続であり、積み上げ化の対象と見なすことができる。このような部分を積み上げ化に関する潜在的リデックスと呼ぶ。

しかし、$f >_{\mathcal{F}} h_B$としたとき、$f_1 - h_A - f_2$ を $f - h_B - f$ という経路と比較する場合には、h_Aの情報が重要となってくる。h_Aは基本的には冗長な情報であるが、他の部分の構造が等しく項の順序が h_Aによって決定するときには必要となる。そこで $[f_1 - h_A - \perp,\ f_1 - f_2]$ という変形を考え、潜在的リデックスと h_Aに関する情報を保存する変形を考える。

定義 57 (繰り上げ化) 以下の２つの関数 $lift_b, lift_s$を用いて繰り上げ化を表す関数 $lift$ を定義する。

- $lift_b(f,t)$: fは関数記号、t は項

  1. $lift_b(f,a) = a$ (a :定数)
  2. $lift_b(f,h(t_1,\cdots,t_n) = h(t_{i1},\cdots,t_{ik},\perp)$
     ただし、$t_{i1},\cdots,t_{ik}$は、$t_1,\cdots,t_n$のうち、最外の関数記号が fでないものとする。
     また、$[t_1,\cdots,t_n]$ のうち、最外の関数記号が fのものは$\perp$に置き換える。

- $lift_s(f,t)$: fは関数記号、t は項

  1. $lift_s(f,a) = [a]$: (a :定数)
  2. $lift_s(f,h(t_1,\cdots,t_n)) = [lift_b(f,h(t_1.\cdots,t_n))] \cup [t_i \mid top(t_i) = f, i = 1,\cdots,n]$

- $lift(f,t)$: fは関数記号、t は項

  1. $lift(f,a) = [a]$: (a :定数)
  2.

$$lift(f,h(t_1,\cdots,t_n)) = \begin{cases} \bigcup lift_s(f,h(p_1,\cdots,p_n)) & (if\ h <_{\mathcal{F}} f) \\ \quad p_i \in lift(f,t_i)\ (i = 1,\cdots,n) & \\ & \\ [St(h(p_1,\cdots,p_n)) \mid p_i \in lift(f,t_i),\ i = 1,\cdots,n] & (if\ h = f) \end{cases}$$

例 58 (繰り上げ化の例) $h, a \in \mathcal{F}, f \in \mathcal{F}_{\mathcal{AC}}$とし、関数記号の順序を $f >_{\mathcal{F}} h >_{\mathcal{F}} a$ とする。この時 $t \equiv \underline{f(h(f}(a,a),b),f(b,b))$ とすると

$lift(f,\underline{f(h(f}(a,a),b),f(b,b))) = lift(f,f(h(\perp,b),f(b,b))) \cup lift(f,\underline{f(f}(a,a),f(b,b)))$
$= [St(f(h(\perp,b),f(b,b))), St(\underline{f(f}(a,a),f(b,b)))]$
$= [f^2(h(\perp,b),b,b), f^3(a,a,b,b)]$

例 58 の下線部の潜在的リデックスは、繰り上げ化の操作によって通常の積み上げ化が可能になる。また、$f^2(h(\perp,b),b,b)$ には潜在的リデックスに存在する h が現れ、h の情報を保持している。

4.3.3 SDL 順序の定義

4.3.2 で定義した分離操作と繰り上げ化を利用して以下のような変形を考える。

定義 59 (分離操作と繰り上げ化による標準形) 任意の t に対し、分離操作と繰り上げ化による標準形 $cf(t)$ を以下のように定義する。

まず、分離操作によって得られる項を以下のように定義する。

$$De(St(t)) = [hat] \cup BOOT$$

ただし、

$$\begin{array}{l} hat \ \ :\text{内部に } top(t) \text{ より大きい関数記号を含まない項} \\ BOOT : \text{内部に } top(t) \text{ より大きい項を含む項の多重集合} \end{array}$$

このとき

$$cf(t) = \begin{cases} lift(top(t), hat) \cup BOOT & if \ \ top(t) \in \mathcal{F}_{\mathcal{AC}} \\ [t] & otherwise \end{cases}$$

このような $cf(t)$ を用いて SDL 順序を以下のように定義する。

定義 60 (SDL 順序) $h, g \in \mathcal{F}$, $s, t, t_1, \cdots, t_n, s_1, \cdots, s_m \in \mathcal{T}(\mathcal{F})$ とし、$s = h(s_1, \cdots, s_n), t = g(t_1, \cdots, t_m)$。

このとき、$s >_{SDL} t$ を以下で定義する。

1. $cf(s) = [s]$ かつ、$f(t) = [t]$ の場合。
   (a) $h > g$の場合 $[s] \gg_{SDL} [t_1, \cdots, t_m]$。
   (b) $h = g$の場合 $[s_1, \cdots, s_n] \gg_{SDL} [t_1, \cdots, t_m]$。
   (c) $h < g$の場合 $[s_1, \cdots, s_n] \gg_{SDL} [t]$。
2. $cf(s) \neq [s]$ または $cf(t) \neq [t]$ の場合。
   $cf(s) \gg_{SDL} cf(t)$。

SDL 順序は AC 関数記号が現れない項や、分離操作、繰り上げ化のリデックスを持たない項の比較は、RPO と同じ手順である定義 1 によって比較を行なっている。

例 61 (SDL 順序による比較の例) $g, h, a \in \mathcal{F}$, $f \in \mathcal{F}_{\mathcal{AC}}$、$g >_{\mathcal{F}} f >_{\mathcal{F}} h >_{\mathcal{F}} a$ のとき $s = h(f(a, a)), t = f(a, a), C[\ ] \equiv f(g(a, a), \Box)$ とすると

$$\begin{cases} cf(St(s)) = [h(\bot), f(a, a)] \\ cf(St(t)) = [f(a, a)] \end{cases}$$

より $cf(St(s)) >_{SDL} cf(St(t))$。
一方、$C[s], C[t]$ では

$$\begin{cases} cf(St(C[s])) & = cf(f(g(a,a), h(f(a,a)))) = [lift(f, f(\bot, h(f(a,a)))), g(a,a)] \\ & = [f(\bot, h(\bot)), f^2(\bot, a, a), g(a,a)] \\ cf(St(C[t])) & = cf(f^2(g(a,a), a, a)) = [f^2(\bot, a, a), g(a,a)] \end{cases}$$

多重集合の定義より $cf(St(C[s])) \gg_{SDL} cf(St(C[t]))$ が成立するので $C[s] >_{SDL} C[t]$。

4.3.4 SDL 順序の反例

SDL 順序の、単調性に関して以下のような反例が存在する。

例 62 (SDL 順序の単調性に関する反例) $a \in \mathcal{F}$, $f, g \in \mathcal{F}_{\mathcal{AC}}$、$a >_{\mathcal{F}} g >_{\mathcal{F}} f$とし、$s = f(g(a,a), a)$、$t = g(f(a,a), a), C[\ ] = f(\Box, a)$ とする。
このとき

$$\begin{cases} cf(St(s)) = [f(\bot, \bot), g(a,a), a] \\ cf(St(t)) = [g(\bot, \bot), f(a,a), a] \\ \\ cf(g(a,a)) = [g(\bot, \bot), a, a] \\ cf(f(a,a) = [f(\bot, \bot), a, a] \end{cases}$$

より $g(a,a) >_{SDL} f(a,a)$ が成立するので $cf(s) \gg_{SDL} cf(t)$。
しかし、

$$\begin{cases} cf(St(C[s])) = cf(f^2(g(a,a), a, a)) = [f^2(\bot, \bot, \bot), g(a,a), a, a] \\ cf(St(C[t])) = [f(\bot, \bot), g(f(a,a), a), a] \\ \\ cf(g(a,a)) = [g(\bot, \bot), a, a] \\ cf(g(f(a,a), a)) = [g(\bot, \bot), f(a,a), a] \end{cases}$$

より $g(f(a,a), a) >_{SDL} g(a,a)$ が成立しているので $cf(C[t]) \gg_{SDL} cf(S[s])$ となり、単調性に反する。

反例 62 の検討は 4.4 節で行なう。

4.4 積み上げ化に関する問題点について

本節ではこれまでに提案した順序の問題点について考察を行なう。

4.4.1 これまでに提案した順序

再編化順序、SDL 順序以外にこれまでに提案した順序の定義を以下に示す。

定義 63 (分配順序 (DRPO)) $f \in \mathcal{F}_{\mathcal{AC}},\ g \in \mathcal{F},\ g >_{\mathcal{F}} f$として、分配規則を
$f(g(x,y),z) \rightarrow_{Di} g(f(x,z),f(y,z))$ とし、$Di(t)$ を t の分配規則に関する正規形とする。
このとき、$s >_{DRPO} t \quad \Leftrightarrow \quad Di(St(s)) >_{RPO} Di(St(t))$
ただし、AC 関数記号は関数記号の中で最大の順序か、または、ある AC 関数記号より大きい関数記号は、最大の順序をもつ AC 関数記号である。

DRPO は、Bachmair による平坦化を基礎とした APO を積み上げ化に置き換えた方法である。

定義 64 (差分順序 (DO)) 任意の項 s,t の積み上げ化に関する正規形を、それぞれ $s' = h^n(S),\ t' = g^m(T)$ とする。また、それぞれの部分項集合について $ST = S \cap T$とする。このとき、
$s >_{DO} t \quad \Leftrightarrow$

1. $top(s) = top(t) \in \mathcal{F}_{\mathcal{AC}},\ n \neq m$ の場合。

   $s'' = h^n(S - ST) >_{DO} g^m(T - ST) = t''$
2. その他の場合。

   (a) $h >_{\mathcal{F}} g$の場合、$[s] \gg_{DO} T$。
   (b) $h = g$の場合、 $S \gg_{DO} T$。
   (c) $h <_{\mathcal{F}} g$の場合、$S \gg_{DO} [t]$。

再編化と同様に、単調性の反例の原因となる項が含まれる共通部分に注目し、両項の共通部分を削除することでそのような項を排除している。

定義 65 (分離順序 (DPO)) $f \in \mathcal{F}_{\mathcal{AC}},\ g \in \mathcal{F},\ g >_{\mathcal{F}} f,\ C[\ ]$ を $top(C[\ ]) = f$を満たす文脈としたとき、以下のような分離化を定義する。
$C[g(T)] \rightarrow_{dep} [C[\bot],\ g(T)]$
文脈 $C[\ ]$ で、最外の関数記号 fから最初に現れる fよりも大きい関数記号 g を最外の関数記号とする部分項 $g(T)$ を、文脈から抽出する。このとき、$dep(t)$ を t の分離化に関する正規形とする。
このとき、
$s >_{DPO} t \quad \Leftrightarrow \quad dep(s) \gg_{RPO} dep(t)$

SDL 順序の分離操作は、分離順序の分離化の発展である。

4.4.2 積み上げ化を基礎とした順序

再編化順序、SDL 順序が満たす AC 適合性をもつ単純化順序の性質を、以下のように表にした。

	AC 適合性	非反射律	推移律	部分項性	単調性	備考
分配順序 (DRPO)	○	○	○	○	×	関数記号の順序に制限
差分順序 (DO)	○	○	×	○	○	分離操作の原型
分離順序 (DPO)	○	○	○	○	×	正規形は多重集合
再編化順序 (RCO)	○	○	?	○	○	差分順序の発展
SDL 順序 (SDLO)	○	○	○	○	×	分離順序の発展

ただし、○：成立する×：成立しない（反例が存在する）?: 未判定。

これらの順序は以下のような特徴がある。

	変形の正規形	RPO との関係	重み付けの回避手段	変形の簡便性
DRPO	存在	変形＋ RPO	分配規則	容易
DO	逐次的に存在	RPO の拡張	該当する項を消去	容易
DPO	存在	変形＋ RPO	該当する項を分離	やや難
RCO	逐次的に存在	RPO の拡張	AC 関数記号を再分配	やや難
SDLO	逐次的に存在	RPO の拡張	該当する項の分離と変形	難

変形の正規形は、ある項に対し一意に定まる場合が「存在」、比較の各段階で正規形が求まるものを「逐次的に存在」とした。

以下では、前節で取り上げた再編化順序と SDL 順序の問題点について考察する。

4.4.3 再編化順序の問題点

再編化順序は明確な反例をもたないが、表のように推移律について証明が困難である。再編化順序が積み上げ化による AC 関数記号の次数と、その引数について微妙な操作を加える。そのため、項 $s = f(S), t = g(T), u = h(U)$ が与えられ、それぞれ $s >_{RCO} t$,$t >_{RCO} u$ という関係が成立しているとする。例えば、$top(s) = top(t) = top(u) = f \in \mathcal{F}_{\mathcal{AC}}$で、各 s, t, u の fの次数が m, n, lであった場合を考える。このとき、m, n, lの大小関係に注目すると、

1. $m > n$ 場合。

   (a) $m > n > l$の場合。
   (b) $m > l > n$ の場合。

(c) $l > m > n$ の場合。
(d) $m > n = l$の場合。
(e) $l = m > n$ の場合。

2. $n > m$ の場合。

(a) $n > m > l$の場合。
(b) $n > l > m$ の場合。
(c) $l > n > m$ の場合。
(d) $n > m = l$の場合。
(e) $l = n > m$ の場合。

3. $n = m$ の場合。

(a) $n = m > l$の場合。
(b) $l > n = m$ の場合。
(c) $l = n = m$ の場合。

次数の場合分けが複雑なのは、次数が一致したときには再編化が起きないので、その分だけ場合分けの組合せが増加するためである。

また差分戦略では、再編化による変形を $f^{n+m}(S) \rightarrow_{Re} f^n(f^m(S_1), S_2)$ とすると、

1. ２項の部分項の集合に共通部分 STが存在しない。
2. 共通部分 STが存在する。

(a) STの要素数が n 個より少ない（STは全て S_2に集められる)。
(b) STの要素数が n 個である。
(c) STの要素数が n こより多い（STのうち、S_1に集められるものが存在する)。

という場合分けが考えられる。3 項を比較する推移律の証明では、s と t、t と u、s と u のそれぞれについて、上記の場合分けの組合せを考えなくてはならない。

さらに、次数の場合分けと差分戦略に関する場合分けは、ほぼ独立な条件で行なわれる。推移律の証明が困難なのは、少なくともこの２つの条件の組合せの数だけ存在するためである。実質的には、条件の重なる場合分けや、意味のない場合分けの組合せなども存在するため、単純に上記の条件の組合せの数の場合分けが必要となるわけではない。しかし、省略できる場合分けを見つけることはそれほど容易ではない。

推移律の証明の困難さを解決するには、再編化の操作をもっと単純化なものに改良して証明を行ない易くするか、再編化の性質を考察して概括的な証明方法を考案する必要がある。

4.4.4 SDL 順序の問題点

例 61 における反例では、単調性に問題が生じている。分離操作や繰り上げ化は、項の内部に例 40,41 のような反例の原因となる X を含んでいる場合に対処する操作である。一方、この反例で s と t の順序に注目すると、例 40 の積み上げ化の反例の前提条件に合致しているが、s, t 自身には積み上げ化が起きていない。よって、s, t 内部に反例の原因となる X を含んでいる可能性はない。よって、このような 2 項には cf による変形を行なわず、SDL 順序の定義 1 に従ってによって変形を行なうべきである。定義 1 の比較の動作は RPO と同一である。そこで s, t を RPO で比較すると

$t = \underline{g}(f(a,a),a) >_{RPO} f(g(a,a),a) = s$。

この場合、$g >_{\mathcal{F}} f$ なので、RPO の定義から 2 項の比較は $[\underline{g}(f(a,a),a)] \gg_{RPO} [g(a,a),a]$ となる。s 側の最外の関数記号 f は、g との比較で消去されるので順序には関与せず、また t 側に現れる g の部分項としての $f(a,a)$ も、順序の決定に影響している。

ところが、SDL 順序の定義に従い、s, t を cf により変形して比較を行なうと、

$$\begin{cases} cf(St(s)) = [f(\bot,\bot), g(a,a), a] \\ cf(St(t)) = [g(\bot,\bot), f(a,a), a] \\ \\ cf(g(a,a)) = [g(\bot,\bot), a, a] \\ cf(f(a,a) = [f(\bot,\bot), a, a] \end{cases}$$

s 側では f が $f(\bot,\bot)$, という形で多重集合に残り、t 側では $f(a,a)$ が分離されて g の部分項であるという情報がなくなってしまい、RPO のみによる順序とは逆の結果になる。

そこで、AC 関数記号を含むが積み上げ化が起こらない項 s, t を考える。cf は、単調性を乱している可能性のある項に対して補正を行なう事を目的とした操作である。s, t は 4.3.1 節で示した反例の前提条件を満たしているので、この 2 項を文脈に適用すると単調性を乱す可能性がある。しかし、積み上げ化が起こらないということは、変形による AC 関数記号の影響が項 s, t にはない。よって、AC 関数記号は項に現れる関数記号の内の 1 つとしてしか意味を持たないので、s, t 自身が単調性を乱す項とはならない。それにも関わらず SDL 順序では, 補正の必要のない項に対しても同様な変形を加えたために、整合性がとれなくなってしまっている。

そこで、cf の変形の条件として、積み上げ化が起こっていない項 t では、$top(t) \in \mathcal{F}_{\mathcal{AC}}$ であっても $cf(t) = [t]$ とすれば、このような状況が回避することが期待できる。

第 5 章

まとめ

本論文では、従来まで AC 適合性を満たす単純化順序を構成するために用いられてきた平坦化の問題点について考察を行ない、新しく積み上げ化を提案しその定義を行なった。

また、積み上げ化について検討を行ない、いくつかの性質を明らかにした。その結果、平坦化で指摘されている問題点を積み上げ化が補っていることが明らかになった。また、積み上げ化には平坦化とは別の問題点が存在することも明らかになった。そこで、AC 適合性を満たす単純化順序を構成するために積み上げ化に加えて、他の変形と組合わせることを考え、いくつかの手法を提案した。

積み上げ化が満たさない単調性について注目し、単調性を満たさない原因について検討を行ない、その条件を明らかにした。そしてその条件を基にして、再編化、分離操作および繰り上げ化という変形手法を提案し、それらの特質を検討した。

そして、再編化を基にした再編化順序と、分離操作、繰り上げ化を基にした SDL 順序をを提案した。これらの順序が AC 適合性を満たす単純化順序であるかどうか確認するため、それぞれの順序の性質の検討を行なった。そして、それらの性質の問題点について分析を行ない、その問題点について明らかにした。

今後、SDL 順序の単調性に関して、変形の適用方法について改善して AC 適合性を満たす単純化順序を構築する必要がある。

謝辞

本研究を行なうにあたり、終始変わらぬ御指導を賜わりました外山 芳人教授に心から感謝致します。また、適切な助言を戴いた酒井 正彦助教授に深く感謝致します。さらに、本研究に関する議論の場を与えて下さった外山研究室、酒井研究室の皆様へ心から御礼申し上げます。

参考文献

[BP85] L.Bachmair and D.A.Plaisted. Termination Ordering for Associative-commutative Rewriting System. *Symbolic Computation*(1985) Vol.1 329-349.

[D79] N.Dershowiz. Ordering for Term-Rewriting Systems. *20th Annual Symposium on Foundations of computer Science*(1978) 123-131.

[D87] N.Dershowiz. Termination of Rewriting. *JSC 3*(1987) 69-116.

[DM92] N.Dershowiz and S.Mitra. Path Ordrings for Termination of Associative-Commutative Rewriting. *LNCS 655*(1992) 168-174.

[H80] G.Huet, Confluent Reduction : Abstract Properties and Applications to Term Rewriting System. *J.ACM 27*(1980) 797-821.

[KSZ90] D.Kapur,G.Sivakumar and H.Zhang. A New Mthod for Proving Termination of AC-Rewrite System. *LNCS 472*(1990) 133-148.

[NST95] 中野 賢司, 酒井 正彦, 外山 芳人. AC 規則を含む項書換え系の研究. 平成7年度電気関係学会北陸支部連合大会 講演論文集 (1995) 315.

[NST96] 中野 賢司, 酒井 正彦, 外山 芳人. 項書換え系の AC 停止性について. 信学技報 TECHNICAL REPORT OF IEICE.COMP95-104 (1996)69-77.

[NST96b] 中野 賢司, 酒井 正彦, 外山 芳人. 項書換え系の AC 停止性について. 夏の L.A シンポジウム 論文集 (1996)61-66.

[RN93] A.Rubio and R.Nieuwenhuis. A Precedence-Based Total AC-Compatible Ordering. *LNCS 690*(1993) 374-388.

[L81] P.Lescanne. Decomposition Ordering As A Tool to Prove The Termination of Term Rewriting System. *7th IJCAI, Van Courver, British Colubia*(1981) 548-550.

[P78] D.Plaisted. A Recursively Defined Ordering for Proving Termination of Term Rewriting Systems. *Report UIUCDCS-R-78-943,Univ.of.Illinois*1978.