修 士 論 文

ナローイングの効率的な実現に関する研究

指導教官　酒井正彦 助教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

松野吉宏

1997 年 2 月 14 日

目 次

第 1 章

序論

1.1 研究の背景・目的

E 単一化は、等式集合 E と項 s と t が与えられたとき E を法として $s\sigma = t\sigma$ をみたす代入σを求める問題であり、これを効果的に解くためにナローイングが導入された。ナローイングは項書換えの照合を単一化に拡張した書換えである。これにより等式論理における式の解を求めることができるため、等式論理の証明や等式言語と論理型言語の融合等への応用が期待されている。その例として、ALF(Hanus [17])、BABEL(Moreno-Navarro and Rodriguez-Artalegjo [18])、EQLOG(Goguen and Meseguer [10])、K-LEAF(Giovannetti et al. [9])、SLOG(Fribourg [8]) などが挙げられる。

現在までに、Fay [7] と Hullot [13, 14] は E が合流性と停止性をみたす項書換え系である場合には、ナローイングが E 単一化の完全な解の集合が求められること、すなわち、すべての解をカバーする一般的な解の集合が求められることを示している。その後、ナローイングは合流性をみたす項書換え系のもとで完全であることが、Yamamoto [24] により示された。ナローイングは一般的に規則の探索範囲が広いため、これに対して探索範囲を狭くする目的でベーシックナローイングが Hullot [13, 14] によって導入され、合流性と停止性をみたす項書換え系の上では完全であることが示されている。また、ベーシックナローイングが合流性のみをみたす項書換え系では完全ではなく、合流性をみたす右線形の項書換え系に対して完全であることが Aart [19] によって示されている。

ナローイングの実現に関する研究としては、書換えの際に生じるバックトラックの問題を解消するために抽象機械を使ったものがある。その例として、遅延ナローイング抽象機

械 LNAM [20] があり最内ベーシックナローイングに基づく関数論理型言語 ALF [16] では抽象機会 AWAM を使って実現されている。

また、ナローイングは非決定性を持つために効率的ではないので、これを解善するためにベーシック以外にも様々な戦略が導入されてきている。その例として、inner[Fribourg 1985]、LSE[Bockmayer 1992]、need[Antoy1994]、lazy[Reddy 1985,Moreno-Navarro and Rodrigues-Artalejo 1992,Aart ,Okui and Ida 1995,Gallier and Snyder 1989] 、outer[You 1989]、approximations[Al puente,Falaschi,Ramis and Vidal 1993]、directed[Dershowitz and Siva kumar 1987] などがある。

本研究でナローイングの効率化のための要因として注目したことは、どの書換え規則がある項に対してどの出現で単一化可能であるかを、効率化することが出来ないであろうかということであった。そこで、従来の単一化アルゴリズムがどのようなものであるかについて述べる。

単一化については、Herbrand[1971] が単一化子を計算するための非決定的アルゴリズムを提案した。それに先駆けて Robinson[1965] と Guard [1964] がそれぞれ独立に単一化問題を研究していた。その後 Robinson[1971] は彼の提案したアルゴリズムが非効率的であったが 1971 年に効率化の研究を行い単一化空間の効率化を計った。それから、Boyer と Moore[1972] が空間的には効率的だが実行に指数時間を要するアルゴリズムを実現し、Venture-Zilli[1975] が実行時間を 2 の乗数程度に抑えたアルゴリズムを考えた。

また、Baxter[1973] がほぼ線形時間のアルゴリズムを見つけ、そして、1976 年に Paterson と wegman[1976,1978] が本当に線形時間のアルゴリズムを提案した。これと同時期に独立に Martelli と Montanari[1976] も線形時間のアルゴリズムを提案し、1982 年には Martelli と Montanari[1982] は真に線形時間ではないが実行時間が $O(n + m \log m), (m :$ 項の中のことなる変数の数) のアルゴリズムを、Kapur と Krishnamoorthy と Narendran も同時期 [1982] に新しい線形時間のアルゴリズムを提案したが、実行時間は線形ではなかった。それから、Corbin と Bidoit[1983] が新しいデータ構造を提案して、実行時間が指数時間で Martelli と Montanari のアルゴリズムよりもシンプルで実装向きであるアルゴリズムを提案している。

以上のように一つの項と一つのパターンに対しては線形時間で単一化可能性を調べられることが分かっている。しかし、ナローイングのように繰り返し単一化を行う場合に、繰り返し高速な単一化アルゴリズムを適用するよりもオートマトンを用いた方が適している

のではないかと考えて、Tree に対するパターンマッチがどの出現でどのパターンを適用するのかを調べるのに対し、ナローイングでの項どの出現でどの書換え規則を適用するという点でオートマトンの利用が有効に作用するであろうとしてオートマトンに注目した。

いままでの Tree におけるパターンマッチングアルゴリズムについては次のようなものである。

本研究ではナローイングの効率的実現の手法として、効率的な単一化アルゴリズムを提案する。一般の単一化アルゴリズムは項の対に対する高速な単一化を行なうのに対して、本研究で提案したアルゴリズムは項の集合 $\mathcal{P}$ と項 t が与えられたとき、任意の項 $p \in \mathcal{P}$ と t の部分項 t' の単一化を効率的に行う。

各項は線形である必要があるが、計算機実験によって従来の方法より効率的であることを示す。

次のこのアルゴリズムを ベーシックナローイングに適用して評価し、本方法の有効性を示す。

1.2 本論文の概要

本論文の構成は次の通りである。まず、２章では本論文を通して使用される必要な用語の定義する。次に３章で 高速単一化の手続きを導入し、評価する。４章ではナローイングとベーシックナローイングに対して高速単一化を利用したアルゴリズムを提案し、評価する。

第 2 章

準備

この章では、Aart[1994] に従い、以下の議論で必要な用語や概念を定義する。

2.1 項

関数記号の集合 $\mathcal{F}$=$\{F, G, H, \ldots\}$，変数の集合 $\mathcal{V}$= $\{x, y, z, \ldots\}$ とする。

項 $(t, s, \ldots)$ は写像 arity : $\mathcal{F} \to \mathcal{N}$を用いて，次のように再帰的に定義される．

定義 2.1 項

1. $x \in \mathcal{V}$は項である．
2. $F \in \mathcal{F}$, $\mathbf{arity}(F) = n$ で，$t_1, \ldots, t_n$が項のとき，$F(t_1, \ldots, t_n)$ も項である．ただし $n = 0$ のときは，Fを項とする．

$\mathrm{arity}(F) = 0$ である関数記号 Fを**定数** $(A, B, \ldots)$ と呼ぶ．$\mathcal{F}$と $\mathcal{V}$ から構成される項の集合を $\mathbf{T}(\mathcal{F}, \mathcal{V})$ で表す．ある項 t に出現している変数の集合を $\mathbf{Var}(t)$ で表す．

項 t と s が構文的に同じであることを $t \equiv s$ と表す。

項 t の部分項の出現 (occurrence)O(t) は以下に定義する自然数列で表す。

$$O(t) = \begin{cases} \Lambda & t \in V \text{の場合。} \\ \{\Lambda\} \cup \{u.p | 1 \leq i \leq n \text{ かつ } p \in O(t_i)\} & t \equiv f(t_1 \ldots t_n), (n \geq 0) \text{ の場合} \end{cases}$$

項 t の出現 p の部分項を $t|_p$と表し、以下のように定義する。

$$t|_p \equiv \begin{cases} t & p = \Lambda \text{の場合。} \\ (t_i)|_q & t \equiv F(t_1 \ldots t_n) \text{ でかつ } p = i \cdot q \text{の場合} \end{cases}$$

O(t) の中でも、特に変数ではない部分項の出現位置の集合を $Oc(t) = \{p \in O(t) | t|_p \notin \mathcal{V}\}$ と表し、変数の部分項の出現位置の集合を $Ov(t) = \{p \in O(t) | t|_p \equiv \mathcal{V}\}$ と表す。t の部分項 $t|_p$ $(p \in O(t))$ が出現位置 p で項 s と置き換えられるときこれを $t[s]_p$ と表す。

$$t[s]_p = \begin{cases} s & p = \Lambda\text{の場合} \\ f(t_1, \ldots, t_i[s]_q, \ldots, t_n) & t \equiv f(t_1, \ldots, t_n) \text{ かつ } p = i \cdot q\text{の場合} \end{cases}$$

2.2 代入

代入σとは、$D\sigma \equiv \{x \in \mathcal{V} | \sigma(x) \neq x\}$ が有限であるような$\mathcal{V}$から $\mathbf{T}(\mathcal{F}, \mathcal{V})$ への写像である。代入σ は 、$\sigma(F(t_1 \ldots t_n)) \equiv F(\sigma t_1 \ldots \sigma t_n)$ のように $\mathbf{T}(\mathcal{F}, \mathcal{V})$ から $\mathbf{T}(\mathcal{F}, \mathcal{V})$ への写像へ拡張される。また、$\mathcal{I}\sigma = \bigcup_{x \in \mathcal{D}\sigma} \mathbf{Var}(\sigma x)$ に含まれる変数はσにより導入されたという。

代入$\sigma(t)$ をσt と表記し、空代入をϵと表す。代入の合成は通常の写像の合成と同様であり、$\circ$を用いて表す。代入τと代入σに対し$\rho \circ \sigma = \tau$なる代入ρが存在するならば、σはτ、より一般的であるといい$\sigma \leq \tau$と表す。

2.3 項書換え系

書換え系とは、次の二つの条件をみたす方向付けされた等式 $l \to r$ である。

- $l \notin calV$
- $\mathbf{Var}(R) \subseteq \mathbf{Var}(l)$

書換え規則の集合 $\mathcal{R}$ を項書換え系もしくは TRS とよぶ。書換え規則が$l \to r$が左（右）線形とは $l(r)$ に同じ変数が二回以上出現しないということである。$\mathcal{R}$ 上での書換え関係 $\to_{\mathcal{R}}$ はポジション $p \in O(s)$、代入 つ $t = s[r\sigma]_p$ が成立することである。これを書換え、もしくはリダクションと呼ぶ。このとき s から t への書換えを、$s \to_{[p,\, l \to r]} t$ と表す。

書換えが起こる部分 $s|_p$ を簡約項 $(redex)$ と呼ぶ。項 s に簡約項が存在しないとき s を正規形と呼ぶ。

$t_0 \to_{\mathcal{R}} \to_{\mathcal{R}} \ldots$ のような無限のリダクション列が存在しないとき、$\mathcal{R}$ は停止性をみたすという。

任意の項 s , t , u に対して、$s \xrightarrow{*}_R u$ かつ $t \xrightarrow{*}_R u$ をみたす u が存在するとき $s \downarrow t$ と書く。任意の項 s , t , u において $u \xrightarrow{*}_R s$ かつ $u \xrightarrow{*}_R t$ ならば $s \downarrow t$ をみたすとき、項書換

え系は合流性あるいはチャーチ・ロッサ性をみたすという。$\mathcal{R}$ が合流性と停止性を満たすとき完全 (complete) であるという。

項 t の一番内側の簡約項を簡約してゆく戦略を最内戦略とよぶ。これとは反対に項 t の一番外側の簡約項を簡約してゆく戦略を最外戦略とよぶ。

2.4 単一化

項 t,s について、代入θが$\theta t \equiv \theta s$ を満たすとき、θを t と s との単一化代入 (unifier) とよび、項 t と s は単一化可能 (unifiable) であるという。s と t のすべての単一化代入τに対して$\sigma \leq \tau$となるようなσを最汎単一化代入 (most general unifier)、もしくは単に単一化子と呼ぶ。代入 σの $\mathcal{V}$に対する制限$\sigma|_{\mathcal{V}}$ を以下のように定義する。

$$\sigma|_{\mathcal{V}}x \equiv \begin{cases} \sigma x & x \in \mathcal{V}\text{の場合。} \\ x & \text{それ以外の場合} \end{cases}$$

変数を名前替えした代入σとθが、$t\theta \equiv s, s\sigma \equiv t$ を満たす時、s と t はお互いに他の変種 (variant) であるという。

2.5 Ω-項

定義 2.2 (**Ω-項**) $[HL79, Klo92, KM91]$ Ωを特別な定数とする。$T(\mathcal{F} \cup \{\Omega\}, \mathcal{V})$（簡単のため$T_\Omega$と表す。）におけるΩ-項によって項の順序を表現する。

(a) t_Ωは項 t の各変数をΩで置き換えて得られるΩ-項である。

(b) T_Ω上の接頭順序を次のように定義する。

- $t \succeq \Omega$ （すべての$t \in T_\Omega$に対して）
- $t \succeq t$ （各変数または定数t に対して）
- $f(t_1, \ldots, t_n) \succeq f(s_1, \ldots, s_n)$ （もし $t_i \succeq s_i\ (i = 1, \ldots, n)$）

(c) t と s が整合であるとは $t \uparrow s$ と記し、ある u に対して $u \succeq t$ かつ $u \succeq s$ が成り立つときである。さもなくば不整合であるといい、$t \sharp s$ と記す。

(d) $s \sqcup t$ は $u \succeq s$ かつ $u \succeq t$ なる最小のΩ-項 u がもし存在するならば、それを表す。

(e) Ω-項 t とΩ-項の集合 S が不整合とは、$t \sharp S$と記し、t と S の任意の元 s が不整合であるときである。

2.6 Tree オートマトン

定義 2.3 (有限 Tree **オートマトン)** 有限 *Tree* オートマトン $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \mathcal{T} \rangle$ で表される。ここで、$\mathcal{F}$は引数を持つシンボルの有限集合、$\mathcal{Q}$ は状態の有限集合、$\mathcal{Q}_f$は終了状態と呼ばれる $\mathcal{Q}$ の部分集合、そして $\mathcal{T}$は遷移規則の集合とする。　遷移規則は次のいづれかの形をとる。

- $f(q_1, \ldots, q_n) \to q$, $(f \in \mathcal{F}, q_1, \ldots, q_n, q \in \mathcal{Q})$,
- $q \to q'$, $(q, q' \in \mathcal{Q})$.

簡約化関係 $\to_A$は状態集合 $\mathcal{Q}$ を認識することで、$\mathbf{T}(\mathcal{F} \bigcup \mathcal{Q}) \times \mathbf{T}(\mathcal{F} \bigcup \mathcal{Q})$ 上で自然に定義される。Tree オートマトンが $t \in \mathbf{T}(\mathcal{F})$ を受理するのは、$t \to_A^+ q \in \mathcal{Q}_f$である簡約が存在するときである。ここで、$\to_A^+$ は$\to_A$ の推移律を意味する。

第 3 章

オートマトンに基づく高速単一化アルゴリズム

3.1 高速アルゴリズム

ここでの方針は、マッチングオートマトンを単一化に対応するようにすることにより、従来の線形時間の高速単一化アルゴリズムを目的項の各ポジションで各パターンに対して適用するのではなく、オートマトンを利用することで無駄な適用を無くして効率的に単一化可能なポジションとパターンを求めるものである。鍵となるアイデアは Tree におけるパターンマッチングのボトムアップパターンマッチングアルゴリズム [Hoffman and O'donell 1982] であり、優先順序付き項書換え系の逐次的クラスの決定性 [酒井 1996] で用いられている手法を基にして単一化に適用できるようにした。ただし、ここでは線形な項のみを扱う。

定義 3.1 (多規則かつすべての部分項に対する単一化問題) $\mathcal{P}$をパターンの有限集合、t を項とする。このとき、単一化問題は次の集合 $Munify(t, \mathcal{P}) = \{(u, p) | u \in \mathcal{P}, p \in O(t), u と t|_p$ は単一化可能 $\}$ を求める問題である。

線形の項を対象としているので、ここでは t と $\mathcal{P}$の変数を $\mathcal{P}$ よりあらかじめすべてΩに置き換えて議論する。したがって s と t が単一化可能であることは、$s_\Omega \uparrow t_\Omega$と表せる。

まず、単一化のための情報を含む状態として用いる項の有限集合 MR を定義する。

(a) $MR_0 = \{\ t |\ t$ は l_Ωの部分項, $l \in \mathcal{P}\}$

(b) $MR_{n+1} = MR_n \cup \{\, t \mid t \preceq (s \sqcup s'),\ s, s' \in MR_n \ s.t.\ s \uparrow s' \}$

(c) $MR_\Omega = \bigcup_{i \geq 0} MR_i$

MR_Ωの項の最大の深さは高々 $\mathcal{P}$ に現れる項の最大の深さであるため、MR_Ωは明らかに有限集合である。$\perp$を使われていない定数とする。また、MRはMR_Ωの各元に対して、出現するΩを任意の数$\perp$に置き換えることにより得られる項の集合とする。$MR_\perp$はMRの元で$\perp$を含まない項の集合とする。$t \preceq_\perp s$ はt に出現する$\perp$を（それぞれ任意の）項で置き換えることでs が得られることを意味する。

補題 3.1 [酒井 [*23*]] s を $s \preceq_\perp t$ となるMRの最大元とする。そのとき、s は一意である。

定義 3.2 (単一化のための Tree automaton) *Tree automaton* $\mathbf{A}^\omega$ は $\langle \mathcal{F}, \mathcal{Q}^\omega, \mathcal{Q}^\omega, \mathcal{T}^\omega \rangle$ とする、ただし $\mathcal{Q}^\omega = \{\langle t \rangle | t \in MR\}$、 $\mathcal{T}^\omega$ は以下の遷移規則に従う。

（a）$\Omega \to \langle \Omega \rangle$

（b）$f(\langle t_1 \rangle, \ldots, \langle t_n \rangle) \to \langle t \rangle^*$ （ある $l \in \mathcal{P}$のある元に対して $l_\Omega \uparrow f(t_1, \ldots, t_n)$ の場合、ただし t は $t \preceq_\perp f(t_1, \ldots, t_n)$ をみたす MR の最大元とする。）

（c）$f(\langle t_1 \rangle, \ldots, \langle t_n \rangle) \to \langle t \rangle$ （ある $l \in \mathcal{P}$に対して $l_\Omega \sharp f(t_n, \ldots, t_n)$ の場合、ただし t は $t \preceq_\perp f(t_1, \ldots, t_n)$ をみたす MR の最大元とする。）

$\mathcal{A}^\omega$は補題 1.1 により決定的であるので、$\mathcal{A}^\omega$の（b）の規則が適用できる場合、すなわち、$C[s_\Omega]_p \to^*_{\mathcal{A}^\omega} C[f(\langle t_1 \rangle, \ldots, \langle t_n \rangle)]_p \to_{(b)} C[\langle t \rangle]_p$の場合には s が l と単一化可能となる。したがって $\mathcal{A}^\omega$を用いて単一化問題を解くには (b) の規則が適用されたときに、(l, p) を出力すればよいことがわかる。

補題 3.2 t を Ω項として、$t \to^*_{\mathcal{A}} \langle s \rangle$ が成り立つとき、$s \leq_\perp t$ をみたすMRの最大元 $s \in MR$ が存在する。

i) $|t| = 1$ のとき

- $t \equiv \Omega$のとき、$\Omega \to \langle \Omega \rangle$ は (a) より成り立つ。
- $t \equiv C$(定数) のとき、$C \to \langle s \rangle$ は遷移規則の定義より成り立つ。

ii) $|t| > 1$ のとき

- $t \equiv f(t_1, \ldots, t_n) \rightarrow^*_{\mathcal{A}} f(\langle s_1 \rangle, \ldots, \langle s_n \rangle) \rightarrow \langle s \rangle$ が成り立つとき、帰納法の仮定より $s_i \leq_\perp t_i$ となる MRの最大元 s が存在する。これより、$f(s_1, \ldots, s_n) \leq_\perp f(t_1, \ldots, t_n)$ となる。これは、遷移規則の定義から分かる。また、s は $s \leq_\perp f(s_1, \ldots, s_n)$ をみたす MR の最大元より、$\leq_\perp$の推移律より $t \rightarrow^*_{\mathcal{A}} s$ が成り立つ。このとき、補題３．１よりこれをみたす s は一意、よって $t \rightarrow \langle s \rangle$ がなりたつ。

補題 3.3 [酒井 [23]] s を $s \leq_\perp t$ をみたす MRの最大元とすると、MRの任意の元 u に対して $u \uparrow s \Longleftrightarrow u \uparrow t$ が成立する。

定理 3.1 t をΩ項、$l \in P$で、$t \rightarrow^*_{\mathcal{A}} f(\langle s_1 \rangle, \ldots, \langle s_n \rangle)$ が成り立っているとする。このとき、$l_\Omega \uparrow f(s_1, \ldots, s_n) \Longleftrightarrow l_\Omega \uparrow t$ が成り立つ。

- 任意の MR_Ωの元に対して、$l_\Omega \equiv f(l_1, \ldots, l_n)$ かつ $t \equiv f(t_1, \ldots, t_n)$ とする。そのとき、任意の i に対して $t_i \rightarrow^*_{\mathcal{A}} \langle s_i \rangle$ が成り立つ。補題3.2 から、$s_i \leq_\perp t_i$となるような MRの最大元が存在する。このとき補題 3.3 より、任意の i に対して $l_i \uparrow s_i \Longleftrightarrow l_i \uparrow t_i$

$C[s_\Omega]_p \rightarrow^*_{\mathcal{A}} C[f(\langle t_1, \ldots, \langle t_n \rangle)]_p \rightarrow_{\mathcal{A}} C[\langle t \rangle]_p$の系列において、最後の簡約に用いられて規則が (a) であれば s は変数なので如何なる項とも単一化可能になる。また、定理より (b) であれば s と単一化可能な $l \in P$ が存在し、(c) ならば s と単一化可能な $l \in P$が存在しないことが分かる。$\mathcal{A}^\omega$は決定的であるので、単一化問題が目的項のノードの数に対して線形時間で求められることがわかる。

3.2 評価

ここでは項が線形なので線形時間のアルゴリズムは提案されているが、次のアルゴリズムより速くなることはない。したがって、以下のアルゴリズムを利用して単一化問題を解く場合と本方法と比較する。

Input: 目的項 t , パターン p
Output: t と p が単一化可能ならば $true$ それ以外なら $false$
Algorithm. $unif(t, p)$

$$= \begin{cases} true & if\ t \in \mathcal{V}\ or\ p \in \mathcal{V} \\ unif(t_1, p_1) \wedge \ldots \wedge unif(t_n, p_n) & if\ t \equiv f(t_1, \ldots, t_n) \wedge p \equiv f(p_1, \ldots, p_n) \\ false & if\ t \equiv f(t_1, \ldots, t_n) \wedge p \equiv g(p_1, \ldots, p_n) \wedge f \neq g \end{cases}$$

3.2.1 遷移表の大きさに関する最悪値の評価

下記のように定めると、遷移表の最大の大きさの評価は次のようになる。

- パターン集合 $\mathcal{P}$の大きさ（総ノード数）を $|\mathcal{P}|$ とおけば、 MR_0の要素数は $|\mathcal{P}|$ で抑えられるので、MR_Ωの要素数は $|\mathcal{P}|^3$ 程度になる。よって、MRの要素数も最悪でも $|\mathcal{P}|^3$程度になる。（詳細は後に記す。）

これより、パターン集合 $\mathcal{P}$が有限集合である場合は、MRも有限集合である。また、集合 MR_0の濃度が大きい場合（パターン集合から生成される部分集合の濃度が大きい場合）、例えば $\mathcal{P} = 10$ の場合で約 10^3個の要素を持つことがある。ただし、これは特殊な例であり、実際上は最悪でも問題にならない。

- 遷移表全体に含まれる遷移規則数も最悪でも $\mathcal{P}^{3n} \approx O(2^{|\mathcal{P}|})$ である。

もし集合 MRの濃度が 10^3程度でパターン集合に出現する関数記号の $arity$の最大値が 10 程度だとすると、遷移表の大きさは 10^{30} となる。また、関数記号の数だけ遷移表はできるので、遷移規則集合全体の最大数は 10^{30}の何倍かになる。

よって、遷移表の大きさを決定する最大の因子は MRの元の多さであり、この遷移表を記憶しておくための領域が大きいことと実行時間との間のトレードオフがパターン集合によってことなるので、どのような場合に有効であるのかが重要になる。

- 遷移表の大きさの評価について
  - MR_0のサイズについては、最大でパターンに含まれるノード数 $|\mathcal{P}|$ を越えることはないので、最悪の場合は MR_nのすべての元が互いに単一化可能であるときである。よって MR_iの要素数を n とすれば、${}_nC_2$だけ増える可能性がある。ただし、複数項に対して最汎単一化子を求めるとそれは一意であるので MR_i の増加は収束する。

また、MR_iのiが何回で止まるかについては各iで項i個の最汎単一化子を作っているのにあたるので、MR_0の要素数がmとすれば、MR_{m-1}ですべての要素に対する最汎単一化子を作ることになる。よって、MR_Ωまでの増加分は最悪の場合${}_mC_2 \times (m-2)+1$になり、MR_Ωの要素数は$m + {}_mC_2 + 1$である。
つぎに、MRの要素にはMR_Ωの要素のΩの出現を任意に$\perp$に変えたものが含まれるから、パターンの最大引数をβとすると、出現には順列がつくので$\sum_{i=1}^{\beta} {}_\beta P_i$となる。よって、$MR$の最大要素数は$(m+1+{}_mC_2)\times \sum_{i=1}^{\beta} {}_\beta P_i$となる。
以上より遷移表の最大サイズはパターンに含まれる関数記号数をηとすれば、$\eta \times (m+1+{}_mC_2) \times \sum_{i=1}^{\beta} {}_\beta P_i \times \beta$となり、$O(2^{|\mathcal{P}|})$で抑えられる。

3.2.2 実行結果

次のような条件のもとで、実行・評価を行なった。

- Standard ML for New Jergy で実装し、Sun Spare Station 5（32Mbyte メモリ）上で実験した。
- 目的項はランダムに作成し、評価を行なった。
- 時間データは30回の試行を時間をおいて計測し、その平均値（繰り上げ）を求めた。
- 従来の方法として線形時間で実行できる [Martelli and Montanari 1982] を元にしたアルゴリズム使い、目的項のすべての部分項とすべてのパターンに対して実行するプログラムを作成した。

評価１

[パターン集合１]

P(0,x)	P(S(x),y)

のパターンのもとで、従来の単一化アルゴリズムの実行時間：従来 [msec] と そのときの記号の比較回数と Tree オートマトンを使った単一化アルゴリズムの実行時間：本研究 [msec] と記号の比較回数の比較。この時、遷移表を作るための事前計算時間は２０ msec で、遷移規則数は３１であった。

表１：評価１の実行結果

目的項のノード数	1223	1731	2339	2855	3671	4896	7344	8569	9795
従来 [msec]	80	100	130	210	280	300	510	600	690
記号比較回数	4278	6078	7270	9990	12846	17133	25701	29986	34176
本研究 [msec]	70	80	100	140	180	200	350	420	530
記号比較回数	1223	1731	2339	2855	3671	4896	7344	8569	9795
単一化可能数	288	392	292	672	864	1153	1729	2016	2306

この結果により分かったことは、パターンの個数が少なく、大きさも小さい場合には本研究のアルゴリズムと従来のアルゴリズムの差は殆んど生じない。但し、若干ながら目的項のサイズが大きくなってくると本研究のアルゴリズムの方が早いことが認められる。よって、繰り返し用いられるような場合には適していると考えられる。

評価２

[パターン集合２]

F(F(A,x),B)	F(F(B,x),B)	F(F(C,B),x)	F(A,x)	F(A,A)
F(A,B)	F(B,x)	F(B,B)	F(x,A)	F(C,B)
F(D,B)	F(C,B)	F(B,C)	F(D,C)	F(C,D)
G(C)	G(A)	G(B)	H(A)	K(A)
K(B)	K(C)	L(B)	M(A)	N(A)

のパターンのもとで、従来の単一化アルゴリズムの実行時間：従来 [msec] とそのときの記号の比較回数と Tree オートマトンを使った単一化アルゴリズムの実行時間：本研究 [msec] とそのときの記号の比較回数の比較を行なった。この時、遷移表を作るための事前計算時間は１３６０ msec、遷移規則数は５７０であった。

表２：評価２の実行結果

目的項のノード数	820	1633	2263	4048	4909	5515	6775	7320	8585
従来 [msec]	510	1050	1400	2390	3090	3380	4830	5210	6110
本研究 [msec]	50	90	130	240	310	370	440	450	550
単一化可能数	272	540	631	1360	1633	1324	1498	1658	1832

この結果より分かったことは、パターンのヘッドシンボルが同一であるものが多い場合は従来の単一化アルゴリズムの場合には比較回数が各ノードで多くなるのに対して、本研究のアルゴリズムの場合には一度の比較で良いので両者の間に大きい差となる。よって、同一ヘッドシンボルのパターンが存在する場合に有効である。

評価３

[パターン集合３]

F(x,A,B)	G(x,y,z)	L(x,C,z)	Q(x,A,C)	R(x,A,z)	U(x,y,A)	P(x,y,z)

のパターンのもとで、従来の単一化アルゴリズムの実行時間：従来 [msec] と記号比較回数、Tree オートマトンを使った単一化アルゴリズムの実行時間：本研究 [msec] と記号比較回数の比較を行なった。このとき、遷移表を作るための事前計算時間は３４０ msec 、遷移規則数は８７８であった。

表３：評価３の実行結果

目的項のノード数	283	850	1192	2047	3577	4771	5965	8347	10732
従来 [msec]	20	70	110	190	330	520	570	960	1130
記号比較回数	2263	6799	9535	16375	28615	38167	47719	66775	85855
本研究 [msec]	10	50	70	110	210	250	320	490	560
記号比較回数	283	850	1192	2047	3577	4771	5965	8347	10732
単一化可能数	63	190	265	455	796	1410	1061	1326	2385

この結果より分かったことは、パターンのヘッドシンボルがすべて異なるような場合は両者の実行結果には差が殆んど生じなかったが、目的項が大きくなると事前時間の分程度の差は生じている。

評価４

[パターン集合４]

F(x,A)	F(F(A,C))	F(F(F(x,B),x),C)	G(x,A)	G(G(x,C),x)	G(G(G(B,A),C),B)

のパターンのもとで、従来の単一化アルゴリズムの実行時間：従来 [msec] とその記号比較回数、また Tree オートマトンを使った単一化アルゴリズムの実行時間：本研究 [msec] とその記号比較回数の比較を行なった。このとき、遷移表を作るための事前計算時間は２４５９０ msec で、表の遷移規則数は１５８４５であった。

表４：評価４の実行結果

目的項のノード数	419	1259	2087	2939	4535	6191	7421	9791
従来 [msec]	80	220	360	670	710	1130	1340	1950
記号比較回数	4926	14814	21822	34590	47950	66158	79194	104494
本研究 [msec]	0	60	110	170	230	320	380	530
記号比較回数	419	1259	2087	2939	4535	6191	7421	9791
単一化可能数	204	612	720	1428	1618	2280	2727	3592

この結果より分かったことは、繰り返しの多いパターンの場合にはノード数が大きくなるに連れて実行時間の差は開くので、事前計算時間はかなりかかるが繰り返し使われる場合などに適している。

3.2.3 アルゴリズムの特徴について

本研究アルゴリズムと従来の高速単一化アルゴリズムに対する実装・評価より分かった特徴は次のようなものである。

- 利点としては、書換え規則が多い場合、トップシンボルが等しいパターンが多い場合、パターンが大きい場合、目的項が大きい場合において高速な探索が可能である。
- 不得手なのはパターンの数が少ない場合、目的項が小さい場合には遷移表を作るためのオーバーヘッドが大きい。また、非線形の場合には直接の適用が出来ない。ただし、探索に必要となる時間は目的項が大きくなるに従って従来のアルゴリズムに比べて短時間で可能になる。

- このような不得手が生じる理由は本研究アルゴリズムには遷移表を作るための事前計算が必要であるのに対して、従来の高速単一化アルゴリズムには必要ではないためである。
- パターンの数が多い場合、集合MRが大きくなる可能性があり遷移表のために大きい記憶領域が必要になるが、探索時間は速いので繰り返し遷移表が使われることによって効率は上がる。

第 4 章

ベーシックナローイングへの適用

4.1 ナローイング

ここではナローイングの定義について述べる。

定義 4.1 (ナローイング) t と s を項、$\mathcal{R}$ を書換え規則集合とする。t が s に出現 $p \in O_c(t)$ と代入 σ でナローイングされるとは、規則 $l \to r$ が存在して、次の条件が成り立つときである。

- $t \equiv t[u]_p$ なる u について、σ は u と l の最汎単一化子
- $s \equiv (t[r]_p)\sigma$

このとき $t \rightsquigarrow_{p,\sigma} s$ と書き、関係 $\rightsquigarrow$ をナローイングという。また、$t \equiv t_1 \rightsquigarrow_{\sigma_1} t_2 \rightsquigarrow_{\sigma_2} \ldots \rightsquigarrow_{n-1} t_n \equiv t'$ が成り立つとき、$t \rightsquigarrow^*_{\sigma} t'$ と表す。ここで、$\sigma = \sigma_{n-1} \circ \ldots \circ \sigma_2 \circ \sigma_1$。

書換え $s \to_{[p,\, l \to r,\, \sigma]} t$ において適用される書換え規則は s と共通の変数と持たないこと、代入σは l の中に出現する変数に制限されることが常に仮定されるので、結局、σ は $s|_p$ と lの最汎単一化子であり、$t \equiv s[\sigma r]_p \equiv (s[r]_p)\sigma$をみたしている。これより、書換えはナローイングの特別な場合とみることができる。

説明の簡便化のために、あたらしく二項関数記号$=^?$と定数 $true$を導入する。さらに、書換え規則集合 $\mathcal{R}$ は規則 $x =^? x \to true$を含むものとし、ナローイングで使われる項は次のような項のみであるとする。

(1) $=^?$と $true$ を含まない項。

(2) 条件 (1) をみたす項 s と t からなる項 $s =^? t$。

(3) 定数 $true$。

(2) の形の項をゴールと呼び、書換え規則集合 $\mathcal{R}$ に規則 $x =^? x \to true$ が加えられるときには、$\mathcal{R}$ は合流性と完全性をみたしているものとする。

4.2 ベーシックナローイング

ナローイングの探索範囲はとても広いので、解の探索において停止することは稀である。そこで、Hullot[1980] は停止性と合流性をみたす項書換え系に対して完全である、つまり、すべての解が見つかるようなナローイングの制限型を導入した。それが、ベーシックナローイングである。

定義 4.2 (ベーシックナローイング)

(1) $t_1 \leadsto_{[p_1, l_1 \to r_1, \sigma_1]} t_2 \leadsto_{[p_2, l_2 \to r_2, \sigma_2]} \cdots \leadsto_{[p_{n-1}, l_{n-1} \to r_{n-1}, \sigma_{n-1}]} t_n$ をナローイング列とし、出現の集合を次のように再帰的に定義する。

- $B_1 = O_c(t_1)$
- $B_{i+1} = \mathcal{B}(B_i, p_i, r_i) \quad (1 \leq i < n)$

ここで、$\mathcal{B}(B_i, p_i, r_i) = (B_i - \{q \in B_i \mid p_i \leq q\}) \cup \{p_i \cdot q \mid q \in O_c(r_i)\}$ とおく。

(2) 書換え列 $t_1 \to_{[p_1, l_1 \to r_1, \sigma_1]} t_2 \to_{[p_2, l_2 \to r_2, \sigma_2]} \cdots \to_{[p_{n-1}, l_{n-1} \to r_{n-1}, \sigma_{n-1}]} t_n$ は、上で定義された $B_2, \ldots, B_{n-1}$ に対して $1 \leq i < n$ であるとき $p_i \in B_i$ ならば、出現集合 $B_1 \subseteq O_c(t_1)$ の上で $based$ である。

また、$1 \leq i \leq n$ において B_i の出現を ベーシック出現 と呼び、$O_c(t_i) - B_i$ の出現を ノンベーシック出現 と呼ぶ。もし、$1 \leq i < n$ において $p_i \in B_i$ をみたすならば 上記のナローイング列を ベーシック という。

つまり、ベーシックナローイング列において代入によって生じた部分項には、決してナローイングは適用されないのである。

Hullot が示したことは、書換え規則の右辺から始めたすべてのベーシックナローイング列が停止するとき、ベーシックナローイングの探索範囲は任意の項に対して有限ということである。近年では、Cabin と Rety[1991] がベーシックナローイングの停止性の振舞いについて項書換え系と求解可能なゴールに対してグラフ表現を用いて、さらなる改良を行なっている。

Herold [11] は完全性を失うことなく $left-to-right$ ベーシックナローイングを用いて集合 $\mathcal{B}_i$を小さくできる、つまり、探索範囲を狭くできることを示している。探索範囲はBosco と Giovannetti と Moiso [4] のいわゆる $selection$ ナローイングの適用によっても縮小できる。Kischer と Bockmayer [1] はベーシックナローイング列を見つける様々な判定法について述べている。

次の定理ではベーシックナローイングは合流性と停止性をみたす書換え系に対しては、すべての解を求めることが可能であることを述べている。

定理 4.1 $[Hullot\ 1980]$ ベーシックナローイングは完全な項書換え系に対して、完全である。

4.2.1 ベーシックナローイングの効率的な表現法

ベーシックナローイングのより効率的な表現法として、Nutt と Rety[1989] と Holldobler[1989] によってゴールを $skeleton$ と $environment$ に分ける方法が提案されている。そのような表現法において、Aart [19] ではナローイングは項 t $(skelton)$ と代入 θ $(environment)$ として $\langle t, \theta \rangle$ の対の上で定義され、$p \in O_c(t)$ かつσが $(\theta t)_p$とある書換え規則 $l \to r$ の l とθt の最汎単一化子として、$\langle t, \theta \rangle \rightsquigarrow_\sigma \langle t[r]_p, \sigma \circ \theta \rangle$ と定義できるとして提案されている。

定義 4.3 (対表現によるベーシックナローイング) $(t, \theta) \rightsquigarrow_{p,\sigma} (t', \theta') \overset{\text{def}}{\Leftrightarrow} t \equiv t[u]_p, t' \equiv t[r]_p, \theta' = \sigma \circ \theta$ $(\exists l \to r \in R,\ \sigma : l$と $u\theta$の最汎単一化子)

次に、項と代入による表現法が正しく機能していることを証明する。

項と代入の対による表現法

1. $t_0 \rightsquigarrow_{p_0,\,\sigma_0} \ldots \rightsquigarrow_{p_{j-1},\,\sigma_{j-1}} t_j \rightsquigarrow_{p_j,\,\sigma_j} t_{j+1} \equiv (t_j[r_j]_{p_j})\sigma_j$

2. $(t_0, \phi) \rightsquigarrow_{p_0,\,\sigma_0} \ldots \rightsquigarrow_{p_{j-1},\,\sigma_{j-1}} (t'[u']_{p_j}, \theta_j) \rightsquigarrow_{p_j,\,\sigma_j} (t'[r_j]_{p_j}, \sigma_j \circ \theta_j)$ (ここで、$t_0 = t'$)

- 完全性: 任意の 1 のベーシックナローイング列があれば、 ある 2 の列があって $t_j \equiv t_j\theta_j$ が成り立つ。
- 健全性: 任意の 2 の列に対して、ある 1 のベーシック列があって $t_j \equiv t'_j\theta_j$ が成り立つ。

ここでは完全性と健全性をを示すことで、項と代入の対による表現法が正しく機能していることを証明する。つまり、$t_{j+1} \equiv (t'_j[r_j]_{p_j})(\sigma_j \circ \theta_j)$ が成り立つことを示す。

- $j = 0$ のとき、書換え規則 $i_0 \rightarrow r_0$ との間の関係と$\theta_0 \equiv [\ ]$ より $t_0 \equiv t'_0$なので、$l_0\sigma_0 \equiv t_0|_{p_0}\sigma_0 \equiv (t'_0|_{p_0}\theta_0)\sigma_0$ で、$t_1 \equiv (t_0[r_0]_{p_0})\sigma_0 \equiv (t_0[r_0]_{p_0})(\sigma_0 \circ [\ ]) \equiv (t_0[r_0]_{p_0})(\sigma_0 \circ \theta_0) \equiv (t'_0[r_0]_{p_0})(\sigma_0 \circ \theta_0)$ が成り立つ。よって $j = 0$ のとき成り立つ。
- $j = n$ のとき、$t_{n+1} \equiv (t_n[r_n]_{p_n})\sigma_n \equiv (t'_n[r_n]_{p_n})(\sigma_n \circ \theta_n)$ が成り立ことを示したい。ここで、書換え規則を部分項を単一化するときは両者に同一の変数の出現はないものとするので、必ず $r_j\theta_j \equiv r_j$が成立し、$t_n = (t'_n[u']_{p_n})\theta_n \equiv t'_n[u'\theta_n]_{p_n}\theta_n$ より、$t_n[\ ]_{p_n} \equiv t'_n[\ ]_{p_n}\theta_n$ が成り立する。これより、$(t'_n[r_n]_{p_n})(\sigma_n \circ \theta_n) \equiv (t'_n[r_n\theta_n]_{p_n}\theta_n)\sigma_n \equiv (t'_n[r_n]_{p_n}\theta_n)\sigma_n \equiv (t_n[r_n]_{p_n})\sigma_n \equiv t_{n+1}$で、$j = n$ のときも成り立つことが分かった。

以上より $0 \leq i$ に対して、$t_{j+1} \equiv (t'_j[r_j]_{p_j})(\sigma_j \circ \theta_j)$ が成り立つ。つまり、1 と 2 のベーシックナローイング列は一致し、項と代入の対による表現法が正しく機能することが示された。

4.2.2 ベーシックナローイングアルゴリズム

次に、この表現法を用いたベーシックナローイングのアルゴリズムを提案する。

ベーシックナローイングアルゴリズム

Input $BasicNarrow(t, \phi)$: $(where\ t$: 目的項)
Output $(s, \theta)\ s.t.\ t \rightsquigarrow^*\ s\theta$
function $BasicNarrow\ (t,\ \theta_0)$;
var t: 項 $(where\ t := f(t_1, \ldots, t_n))$;

θ_0 : 代入;
if $t \in \mathcal{V}ar$ **then** $return\ (t,\ \theta_0)$
else
for $i = 1$ **to** n **do**
$(s_i,\ \theta_i) := BasicNarrow\ (t_i,\ \theta_{i-1})$
select (1) **or** (2)
(1) $\sigma := mgu\ (f(s_1, \ldots, s_n)\theta,\ l)$ $for\ some\ \ l \to r \in \mathcal{R}$;
$return\ BasicNarrow\ (r,\ \sigma \circ \theta_n)$
(2) $return\ (f(s_1, \ldots, s_n),\ \theta_n)$

- ここで、$mgu(u, v)$ は u と v の最汎単一化子を求める関数。

このアルゴリズムは項に対して最内戦略を適用し、**select** で非決定的選択を行なうことによってベーシックナローイングを実現し、完全性を保っている。また、$BasicNarrow$を部分項に対して再帰呼出しすることで、探索範囲を縮小している。

4.2.3 従来の表現法と項と代入の対による表現法の比較

ここでは、項にマークする表現方法と項と代入の対による表現方法についてベーシックナローイングを行ない、一つの解を見つけるまでに単一化を試みた書換え規則数を調べて、どの程度の効率が上がるかを調査する。そこで、以下のような条件下で ML を使って実験を行なう。

- 項へのマークの付け方は、項を $NODE(s, clds, st) : Symbol * (Term\ list) * int$ と表現して、代入を受けた部分項には $st := 1$、受けてないと部分項には $st := 0$ とする。
- 最内探索を行う。
- 両表現に対して用いる手続きは同一とし、解はどちらの表現を用いても同じものが得られるとする。

つぎの書換え規則を用いたとき、各表現法に対して単一化可能である出現を最内探索で調べたときに適用を試みた規則の回数を示す。

$$\mathcal{R} = \begin{cases} P(0, x) & \to & x \\ P(S(x), y) & \to & S(P(x, y)) \end{cases}$$

この規則を用いてベーシックナローイングを行った結果、解が得られるまでに行った単一化可能な出現を見つけるために規則を適用した回数に関する表を” 総ノード数 [個]” マークを代入によって生じる項につける方法” マークつき [回]” と項と代入の対による方法” 対表現 [回]” として示す。

表１：マークつき表現と対表現に対する結果

総ノード数	22	168	337	675	1351	2027	2707	3379	5407
マークつき [回]	101	2224	4784	10239	21819	34739	48999	64599	119439
対表現 [回]	40	896	1927	4123	8783	13979	19711	25979	47999

この結果より、項と代入による表現法が解の探索効率を良くするものと考えられる。

- この結果を評価するため、つきのように記号をおく。
  - t : 項 に対して、$|t|$ ： t の総ノード数。
  - $\sigma = [x_1 \mapsto t_1,\ x_2 \mapsto t_2, \ldots, x_n \mapsto t_n]$ に対して、$|\sigma| \stackrel{\text{def}}{=} |t_1| + |t_2| + \ldots + |t_n|$。
  - t : 項 に対して、$\mathcal{V}(t) : t$ に含まれる変数集合。
  - $\sigma|_{\mathcal{V}(t)}$ を代入σの項t に関する代入。
- ここで、以下のようなベーシックナローイング列が存在するときの対表現によって縮小されるされる探索範囲を評価する。
  - $t_0 \leadsto_{[\sigma_1,\, l_1 \to r_1]} t_1 \leadsto_{[\sigma_2,\, l_2 \to r_2]} t_2 \leadsto_{[\sigma_3,\, l_3 \to r_3]} t_3 \leadsto_{[\sigma_4,\, l_4 \to r_4]} \cdots \leadsto_{[\sigma_n,\, l_n \to r_n]} t_n$ に対して、項と代入の対による表現によって縮小される探索範囲（ノード数）を N とすると、$N = \sum_{i=1}^{n} \left|\sigma_i|_{\mathcal{V}(r_i) \cup \mathcal{V}(t_i)}\right|$ となる。
- この結果からも、項と代入の対による表現法は書換え規則集合 $\mathcal{R}$ によらず、従来の表現法よりもベーシックナローイングの解の探索を簡単にできることが分かった。

4.3 Tree オートマトンのベーシックナローイングへの適用

ベーシックナローイングは単一化を繰り返し行い解を求めるものであるので、本研究で提案された Tree オートマトンがナローイングアルゴリズムに適していると考えられる。ここでは本研究のオートマトンがナローイングに適していることを、実装・実現することで比較・検討するとともに、その手順について示す。

4.3.1 単一化のための Tree オートマトンを組み込んだベーシックナローイングアルゴリズム

Input $Basic\ (t, \phi)$
Output $(s, \theta)\ s.t.\ t \rightsquigarrow^* s\theta$
function. $Basic(t,\ \theta_0)$
var t: 項 $(where\ t := f(t_1, \ldots, t_n))$;
　　θ_0: 代入
　if $t \in \mathcal{V}ar$ **then** $return(t, \theta_0)$
　else
　　for $i = 1$ **to** n **do**
　　$(s_i,\ \theta_i) := Basic(t_i,\ \theta_{i-1})$
select (1) **or** (2)
(1) $\mathcal{R}_i := Automaton(state(s_1), \ldots, state(s_n))$
　　$\sigma := mgu(f(s_1, \ldots, s_n)\theta,\ l)\ \ for\ some\ l \to r \in R_i$
　　$return\ Basic(r,\ \sigma \circ \theta_n)$
(2) $return\ (f(s_1, \ldots, s_n),\ \theta_n)$

ここで、$automaton(st_1, \ldots, st_n)$ は状態 $st_i\ (1 \leq i \leq n)$ を入力にもち、書換え規則集合を出力にもつオートマトンで、$mgu(u,\ v)$ は u と vの最汎単一化子である。

このアルゴリズムをにおいて入力される項は事前に各ノードに対して状態を与え、書換え規則の左辺にも状態を与えてから単一化を試みると代入によって生じる項に対しては状態がつくことになり、代入によって生じる項に対してはナローイングを行わないベーシックナローイングに対して適したアルゴリズムであることが分かる。

例

$$\mathcal{R} = \begin{cases} P(0, x) & \to & x \\ P(S(x), y) & \to & S(P(x, y)) \end{cases}$$

の規則に対して、ある項 t に対して付加される状態を $\langle t \rangle$ として、$(t,\ \langle t \rangle)$ と表すと次のようになる。

$(P(P(S(z), S(0)), z), \phi) \rightsquigarrow (P(S(P(x, y)), z), [x \mapsto (z, \langle \Omega \rangle),\ y \mapsto (S(0), \langle S(\bot) \rangle)])$

このとき、x に状態 $\langle \Omega \rangle$、yに状態 $\langle S(\bot) \rangle$ がついていることになり、代入部分への評価は不要になる。

第 5 章

結論

5.1 研究内容のまとめ

本研究では単一化可能な出現を効率的に見つけることに力点を置いてきた。始めに、パターンマッチングの考え方を取り入れた Tree オートマトンの改良により単一化のためのオートマトンの定義を与え、線形なパターンと目的項に対して高速である従来の単一化アルゴリズムとの比較を行った。

それから、ベーシックナローイングに適した表現法について考察し、Tree オートマトンを組み込んだベーシックナローイングの効率的実現を図った。

5.2 今後の課題

今回の実装において、遷移表の大きさはかなり大きい場合もあることが分かっているが、実際にはどの程度の大きさで十分なのかは不明のままである。よって、遷移表の構成に関してさらに考慮の余地がある。

謝辞

遅々として研究が進まない著者に、暖かく、時に厳しく学問上のことから私事に至るまで終始変わらぬ多くの有益な御指導をくださった酒井助教授、外山教授には深く感謝いたします。

さらに本研究に関する議論の場を与えて下さり外山研究室・酒井研究室の皆様に心から御礼申し上げます。

参考文献

[1] A.Bockmayr, Beitrage zur Theorie des Logisch-Funktionalen Programmierens,Ph.D.thesis, *Universitat Karlsrule,1990.(in German.)*

[2] G.Becher and U.Petermann Rigid Unification by Completion and Rigid Paramodulation *'KI-94:Advances in Artificial Intrelligence' LNAI 861(1994) 319-330*

[3] B.Beckert, A Completion-Based Method for Mixed Universal and Rigid E-Unification, *In Proc.12th Internat. Conference on Automated Deduction (CADE),LNAI 814(1994) 678-692*

[4] P.G.Bosco,E.Giovannetti,and C.Moiso, Narrowing vs. SLD-Resolution, *Theoretical Computer Science 59,pp.3-23,1988.*

[5] P.J.Downey,R.Sethi and R.E.Tarjan Variations on the Common Subexpression Problem *Journal of the ACM,27(4),758-771(1980)*

[6] J.Gallier, Logic for Computer Science – Foundations of Automatic Theorem Proving. *John Wiley and Sons,1987*

[7] M. Fay, First-Order Unification in Equational Theories, *Proceedings of the 4th International Workshop on Automated Deduction,Austin,1979,161-167.*

[8] L.Fribourg, SLOG: A Logic Programming Language Interpreter based on Clausal Superposition and Rewriting, *Proceeding of the 2nd IEEE Symposium on Logic Programming, Boston,1985,172-184*

[9] E.Giovannetti,G.Levi,C.Moiso,and C.Palamidessi, Kernel-LEAF:A Logic plus Functional Language, *Journal of Computer and System Science 42,1991,139-185.*

[10] J.A. Goguen and J. Meseguer, EQLOG: Equality, Types and Generic Modules for Logic Programming,in:"Logic Programming: Functions, Relations and Equations",(eds.D.DeGroot and G.Lindstrom), *Printice Hall,1986,295-363.*

[11] A. Herold, Combination of Unification Algorithms in Equational Theories,Ph.D.thesis, *Universitat Kaiserslautern,1987.*

[12] Christoph M. Hoffmann and Michael J. O'Donnell, Pattern Matching in Trees, *Journal of the Association for Computing Machinery, vol.29,No.1,January1982,pp.68-95.*

[13] J.-M. Hullot, Canonical Forms and Unification, *Proceedings of the 5th Conference on Automated Deduction,Lecture Notes in Computer Science 87,1980,318-334.*

[14] J.-M. Hullot, Compilation de Formes Canoniques dans les Theories Equationelles,These de troisieme cycle, *Universitee de Paris Sud,Orsay,1980.(In French.)*

[15] E. Kogel Rigid E-Unification Simplified *In Theorem Proving with Analytic Tableaux and Related Methods,LNAI 918 (1995),17-30*

[16] M. Hanus Computing Logic Programs with Equality,Proceedings of the International Workshop on Language Implementation and Logic Programming,Linkoping, *Lecture Notes in Computer Science 456,1990,381-401*

[17] M. Hanus, Efficientt Implementation of Narrowing and Rewriting , *in Proc. International Workshop on Processing Declarative Knowledge,Lecture Notes in Comuter Science, Artificial Inteerigence 567,1991,344-365.*

[18] J.J. Moreno-Navarro and M. Rodriguez-Artalejo, BABEL: A Functional and Logic Programming Language based on COnstructor Discipline and Narrowing, *Proceedings of the 1st International Conference on Algebraic and Logic Programming,Gausig,Lecture Notes in Conputer Science 343,,1989,223-232*

[19] A. Middeldorp and E. Hamoen, Completeness Results for Basic Narrowing, *Applicable Algebra in Engineering, Communication and Computing 5,1994,213-253.*

[20] 中川康二, 中原鉱一, 鈴木太郎, 井田哲雄, 遅延ナローイング抽象機会, 電子情報通信学会論文誌 *D-1 Vol.J78-D-1 No.5,1995,467-477.*

[21] L.Paulson ML for Working Programmer *Cambridge University Press(1991)*

[22] A.Robinson, A machine-oriented logic based on the resolution principle, *Journal of the ACM,12,1965,23-41*

[23] Masahiko Sakai, Yoshihito Toyama, Semantics and Strong Sequentiallity of Property Term Rewritng Systems, *Proc.or RTA96, LNCS1103,pp.377-391.1996.*

[24] A. Yamamoto, Completeness of Exttended Unification Based on Basic Narrowing, *Proceedings of the 7th Logic Programming Conference, Jerusalem,1988,1-10.*