# アトム変数を用いた名目単一化の実装

山上 隼司　菊池 健太郎　上野 雄大　大堀 淳

本稿では，アトム変数を用いた名目単一化アルゴリズムの実装に向けての取り組みを報告する．名目単一化 (nominal unification) は項同士がアルファ同値となるような代入を見つけ出す操作であり，通常の単一化と違い束縛変数を扱えるという特徴を持つ．Schmidt-Schauß ら (2019) のアトム変数を用いた名目単一化は，置換 (permutation) の扱いに注意しなければならないが，アトム変数を用いないこれまでの名目単一化よりも記述が簡潔になる場合が多く，表現力が高い．しかし，その実装は課題として残されていた．本研究では，Schmidt-Schauß らが提示した体系の実装に向けた再構築と，再構築した体系に基づく，関数型言語 SML#を用いた実装を行っている．本稿では，その経過を報告する．

## 1　はじめに

単一化は定理証明の自動化等において用いられる重要な技術である．近年普及している高階型理論・高階述語論理に基づく定理証明支援系は，束縛変数を含む言語を対象としており，そこで用いられる単一化では束縛変数を伴う項を扱う必要がある．束縛機構として変数名を用いる場合には，項同士がアルファ同値 †1 となるような代入を求める手続きを与えることが主な目的となる．

Gabbay-Pitts による名目技法 [1] [4] に基づいて考案された名目単一化 (nominal unification) [6] [7] は，束縛変数を伴う項を対象とした単一化の一種である．そこでは，束縛される可能性のある変数はアトムとよばれ，アトムに関するアルファ同値性が構文の中で明示的に扱われるという特徴を持つ．

元々の名目単一化では，項を表す変数 (通常の意味でのメタ変数に対応する) への代入と，変数におけるアトムの出現に関する制約の組が，単一化子となる．これに対し，最近 Schmidt-Schauß ら [5] が導入したアトム変数を用いる名目単一化では，アトムを表すメタ変数に対応するアトム変数も代入の対象となる．ただし，単一化問題の可解性は，条件を満たす基底代入 (変数を含まない項を割り当てる代入) が存在するか否かによって判定される．アトム変数を持つ言語を用いることの利点については，先行研究 [2] [5] の第 1 章を参照されたい．

本稿では，先行研究 [5] で課題として残されていたアトム変数を用いる名目単一化の実装に向けての取り組みの経過を報告する．[5] では推論規則に基づく単一化手続きが提案されており，それらの推論規則は，変数やアトム変数を含む項 (名目項式とよぶ) の間の等式に関する規則と，アトムの出現制約に関する規則の 2 つのグループに分けられる．本研究でも，それらの推論規則に基づいて単一化手続きの実装を進める方針をとるが，実装を行うための基盤としてより適した形へ構文や規則の記述を洗練する．本稿では，特に，

1. 名目項式の文法の再定義と，それに沿った各種操作の実装に即した定義を与える．
2. アトム変数を用いる名目単一化において重要となるアトム変数置換に関する操作について，[5] で提案されている規則を拡張した新たな規則を導入し，詳しい説明を行う．

---

Implementation of nominal unification with atom-variables

Shunji Yamagami, Kentaro Kikuchi, Katsuhiro Ueno, Atsushi Ohori, 東北大学, Tohoku University.

†1　アルファ同値性に加え，いくつかの関係を法として項を同値とする場合もある．

上記 1 について，先行研究と異なるのは以下の点である．[5](p. 48) では，名目項式の文法が次のように与えられている．

$e ::= A \mid S \mid \pi \cdot A \mid \pi \cdot S \mid \langle f\ e_1\ \ldots\ e_{ar(f)} \rangle \mid \lambda \pi \cdot A.e$

$\pi ::= \varnothing \mid (A\ B) \cdot \pi$

しかしながら，この定義では，アトム変数置換 $\pi$ に現れるアトム変数 $A$ を，保留アトム変数 $\pi' \cdot A'$ で置き換えた場合に文法エラーとなる．一方，単一化手続きを実装する際に，単独のアトム変数 $A$(あるいは変数 $S$) と $\varnothing \cdot A$(あるいは $\varnothing \cdot S$) を同一視すべき場面でも，構文的に区別されており，実装が煩雑になる．そのため，上記の文法は，実装のための基盤としては適していないと考えられる．[5] では，アトム変数置換に現れるアトム変数に対する場合に限り，(保留アトム変数ではなく) 単独のアトム変数を代入するように注意深く制限することにより，単一化手続きを構成することが可能であるとしている．しかし，本研究では，アトム変数置換における保留アトム変数の出現を許す，より広い文法を採用することにする．

他の先行研究 [2] [3] では，本稿と同様にアトム変数置換における保留アトム変数の出現を許す形で名目項式の文法を与えている ($v ::= \pi \cdot A$ のような保留アトム変数の構文カテゴリを加えている)．しかし，本稿の定義 6 で見るような代入の再帰的な定義は与えられておらず，単一化手続きの実装についても触れられていない．

アトム変数置換に関する操作について先行研究と異なるのは以下の点である．[5](p. 53) で注意されているように，アトム変数置換を構成する列の要素は，条件を満たす場合に順番を入れ替えることが可能である．しかし，それをそのまま規則として取り入れた場合，全体の手続きの停止性を妨げてしまう．[5] では，入れ替えを行う規則は導入しないが，それに修正を加えた場合，実装におけるアトム変数置換の簡単化に効果的であり得るとしている．本研究では，そのような入れ替えを他の規則と適切に組み合わせることにより，[5] で示唆されていることを実現するいくつかの新しい規則を導入した．

本論文の構成は以下の通りである．2 章では，基底名目項と名目項式の基本事項を述べる．3 章では，アトム変数を用いた名目単一化手続きについて議論する．4 章では，現時点の実装と実行例について述べる．

## 2 名目項式

本章では，変数を含まない項 (基底名目項) と，変数やアトム変数を含む項 (名目項式) について説明する．基底名目項は関数記号を含むラムダ式で表され，アルファ同値性は基底名目項の間の関係として定義される．名目項式について説明した後，アトム変数置換に関する注意点を述べる．

### 2.1 基底名目項

関数記号の集合を $\mathcal{F}$ とする．関数記号は $f$，$g$ 等で表され，0 以上のアリティを持つ．関数記号 $f$ のアリティを $ar(f)$ と書く．また，アトムの集合を $\mathcal{A}t$ とする．アトムは $a$，$b$ などで表され，名前の異なるアトム同士は互いに異なるアトムとする．$\mathcal{F}$ と $\mathcal{A}t$ は互いに共通の要素を持たない．基底名目項はアトム，関数記号，アトムを束縛するラムダ式で構成される．

**定義 1.** 基底名目項の集合 $NL_a$ は以下の文法で定義される．

$$e \in NL_a ::= a \mid \langle f\ e_1\ \ldots\ e_{ar(f)} \rangle \mid \lambda a.e$$

$\langle f\ e_1\ \ldots\ e_{ar(f)} \rangle$ は関数適用を表す．以下ではタプル $\langle e_1\ \ldots\ e_{ar(f)} \rangle$ も用いるが，これは関数適用 $\langle f\ e_1\ \ldots\ e_{ar(f)} \rangle$ の関数名が無い場合とみなす．

二つのアトム $a$ と $b$ を入れ替える操作を互換 (swapping) と言い，$(a\ b)$ で表す．定義域 $dom(\tau) = \{a \in \mathcal{A}t \mid \tau(a) \neq a\}$ が有限集合となるような $\mathcal{A}t$ 上の全単射 $\tau$ を，置換 (permutation) と言う．置換は，互換の有限列で表すことができる．置換 $\tau$ をアトム $a$ に適用することを $\tau \cdot a$ と表し，このとき列の右から順に互換をアトムに適用していく．例えば置換 $\tau = (a\ b)(a\ c)$ をアトム $c$ に適用する場合は，$\tau \cdot c = (a\ b)(a\ c) \cdot c = (a\ b) \cdot a = b$ となる．

**定義 2.** 置換を $NL_a$ の項へ適用する関数 $\odot$ を，以下のように定義する．

$$\tau \odot a := \tau \cdot a$$

$$\tau \odot \langle f\ e_1\ \ldots\ e_{ar(f)} \rangle\ :=\ \langle f\ \tau \odot e_1\ \ldots\ \tau \odot e_{ar(f)} \rangle$$

$$\tau \odot (\lambda a.e)\ :=\ \lambda(\tau \cdot a).(\tau \odot e)$$

アトム $a$ と基底名目項 $e$ の組 $a\#e$ を非出現関係

$$\frac{}{\vdash_{NL_a} a\#b}$$

$$\frac{\vdash_{NL_a} a\#e_1 \quad \cdots \quad \vdash_{NL_a} a\#e_{ar(f)}}{\vdash_{NL_a} a\#\langle f\ e_1\ \ldots\ e_{ar(f)}\rangle}$$

$$\frac{}{\vdash_{NL_a} a\#\lambda a.e} \qquad \frac{\vdash_{NL_a} a\#e}{\vdash_{NL_a} a\#\lambda b.e}$$

**図 1**　$NL_a$ **上の非出現関係の定義**

$$\frac{}{\vdash_{NL_a} a \sim a}$$

$$\frac{\vdash_{NL_a} e_1 \sim e'_1 \quad \cdots \quad \vdash_{NL_a} e_{ar(f)} \sim e'_{ar(f)}}{\vdash_{NL_a} \langle f\ e_1\ \ldots\ e_{ar(f)}\rangle \sim \langle f\ e'_1\ \ldots\ e'_{ar(f)}\rangle}$$

$$\frac{\vdash_{NL_a} e \sim e'}{\vdash_{NL_a} \lambda a.e \sim \lambda a.e'} \qquad \frac{\vdash_{NL_a} (a\ b) \odot e \sim e' \quad \vdash_{NL_a} b\#e}{\vdash_{NL_a} \lambda a.e \sim \lambda b.e'}$$

**図 2**　$NL_a$ **上のアルファ同値性の定義**

(freshness relation) とよぶ．これは，アトム $a$ が基底名目項 $e$ に自由アトムとして現れないことを意味する．アトムと $NL_a$ の項の関係 $\#$ を図 1 のように定義する．

また，$NL_a$ 上の関係 $\sim$ を図 2 のように定義する．これは，通常の意味でのアルファ同値性 (束縛アトムの名前替えにより到達できる関係) と一致する [1].

### 2.2　名目項式

名目項式の集合 $NL_{AS}$ を定義するために，変数とアトム変数を導入する．名目項式は，基底名目項を変数とアトム変数，アトム変数置換を用いて改変したものである．

変数の集合を $\mathcal{S}$ とする．変数は $S$ や $T$ で表される．また，アトム変数の集合を $\mathcal{A}$ とする．アトム変数は $A$ や $B$ で表される．アトム変数は $NL_a$ におけるアトムに対応する変数であり，以下で説明するアトム変数からアトムへの代入 (基底代入) によって，アトムでのみインスタンス化される変数である．$\mathcal{S}$ と $\mathcal{A}$，また $\mathcal{F}$ は互いに共通の要素を持たない．

**定義 3.** 名目項式の集合 $NL_{AS}$ を以下の文法で定義する．

$$e \in NL_{AS} ::= \pi \cdot A \mid \pi \cdot S \mid \langle f\ e_1\ \ldots\ e_{ar(f)}\rangle \mid \lambda\pi \cdot A.e$$
$$\pi ::= \varnothing \mid (\pi' \cdot A\ \pi'' \cdot B) \cdot \pi$$

$\pi \cdot A$ を保留アトム変数 (suspensions of atom-variables) とよび，$(\pi' \cdot A\ \pi'' \cdot B)$ をアトム変数互換 (atom-variable swapping) と言う．アトム変数互換の有限列 $\pi$ をアトム変数置換 (atom-variable permutation) と言う．また，$\pi \cdot S$ を保留変数 (suspensions of expression-variables) とよぶ．二つのアトム変数置換 $\pi$ と $\pi'$ の連結を $\pi \circ \pi'$，アトム変数置換 $\pi$ の逆順の列を $\pi^{-1}$ と書く．

ここで，二種類の代入を定義する．まず，写像 $\rho : (\mathcal{A} \cup \mathcal{S}) \to NL_a$ を基底代入 (ground substitution) と言う．基底代入はアトム変数にアトムを，変数に $NL_a$ の項を割り当てる写像である．

**定義 4.** 名目項式 $e \in NL_{AS}$ と基底代入 $\rho$ について，$e$ に $\rho$ を適用して得られる $NL_a$ の項を $e\rho$ と書き，以下のように定義する．

$$\begin{aligned}
(\pi \cdot A)\rho &:= \pi\rho \cdot \rho(A) \\
(\pi \cdot S)\rho &:= \pi\rho \odot \rho(S) \\
\langle f\ e_1\ \ldots\ e_{ar(f)}\rangle\rho &:= \langle f\ e_1\rho\ \ldots\ e_{ar(f)}\rho\rangle \\
(\lambda\pi \cdot A.e)\rho &:= \lambda(\pi\rho \cdot \rho(A)).(e\rho) \\
\varnothing\rho &:= \varnothing \\
((\pi' \cdot A\ \pi'' \cdot B) \cdot \pi)\rho &:= \\
&\quad (\pi'\rho \cdot \rho(A)\ \pi''\rho \cdot \rho(B)) \cdot \pi\rho
\end{aligned}$$

例えば，$e = (\varnothing \cdot A\ \varnothing \cdot B) \cdot C$，$\rho = \{A \mapsto a, B \mapsto b, C \mapsto a\}$ のとき，$e\rho = (\varnothing\rho \cdot \rho(A)\ \varnothing\rho \cdot \rho(B)) \cdot \rho(C) = (a\ b) \cdot a = b \in NL_a$ となる．空列 $\varnothing$ は恒等置換を表すことに注意する．

また，写像 $\sigma : (\mathcal{A} \cup \mathcal{S}) \to NL_{AS}$ を単に代入 (substitution) と言う．この代入はアトム変数に保留アトム変数を，変数に $NL_{AS}$ の項を割り当てる写像である．

**定義 5.** アトム変数置換を $NL_{AS}$ の項へ適用する関数 $\bullet$ を，以下のように定義する．

$$\begin{aligned}
\pi \bullet (\pi' \cdot A) &:= (\pi \circ \pi') \cdot A \\
\pi \bullet (\pi' \cdot S) &:= (\pi \circ \pi') \cdot S \\
\pi \bullet \langle f\ e_1\ \ldots\ e_{ar(f)}\rangle &:= \langle f\ \pi \bullet e_1\ \ldots\ \pi \bullet e_{ar(f)}\rangle \\
\pi \bullet (\lambda\pi' \cdot A.e) &:= \lambda(\pi \circ \pi') \cdot A.(\pi \bullet e)
\end{aligned}$$

**定義 6.** 名目項式 $e \in NL_{AS}$ と代入 $\sigma$ について，$e$ に

$\sigma$ を適用して得られる項 $e\sigma \in NL_{AS}$ を以下のように定義する．

$$
\begin{aligned}
(\pi \cdot A)\sigma &:= \pi\sigma \bullet \sigma(A) \\
(\pi \cdot S)\sigma &:= \pi\sigma \bullet \sigma(S) \\
\langle f\ e_1\ \ldots\ e_{ar(f)}\rangle\sigma &:= \langle f\ e_1\sigma\ \ldots\ e_{ar(f)}\sigma\rangle \\
(\lambda\pi \cdot A.e)\sigma &:= \lambda(\pi\sigma \bullet \sigma(A)).(e\sigma) \\
\varnothing\sigma &:= \varnothing \\
((\pi' \cdot A\ \pi'' \cdot B) \cdot \pi)\sigma &:= \\
&\quad (\pi'\sigma \bullet \sigma(A)\ \pi''\sigma \bullet \sigma(B)) \cdot \pi\sigma
\end{aligned}
$$

例えば，$e = \pi \cdot S$，$\sigma = \{S \mapsto \lambda(\pi' \cdot A).(\varnothing \cdot T)\}$ のとき，$e\sigma = \pi\sigma \bullet \sigma(S) = \pi\sigma \bullet \lambda(\pi' \cdot A).(\varnothing \cdot T) = \lambda((\pi\sigma \circ \pi') \cdot A).(\pi\sigma \cdot T) \in NL_{AS}$ となる．

以下では二つの代入 $\sigma$，$\sigma'$ の合成を $\sigma \circ \sigma'$ と書く．このとき，$e(\sigma \circ \sigma') = (e\sigma)\sigma'$ であることに注意する．同様に代入 $\sigma$ と基底代入 $\gamma$ の合成も $\sigma \circ \gamma$ と書き，このとき，$e(\sigma \circ \gamma) = (e\sigma)\gamma$ である．

また，アトム変数 $A$ と名目項式 $e$ の組 $A\#e$ を非出現制約といい，非出現制約の集合 $\nabla$ を非出現制約集合という．非出現制約集合 $\nabla$ が $A\#\varnothing \cdot B$ と $A\#\varnothing \cdot S$ の形の非出現制約のみからなるとき，その $\nabla$ は標準化 (standardized) されているという．

### 2.3 アトム変数置換

アトム変数置換は，基底代入後のアトムについての置換を表すものであり，アトム変数同士の置換を表すものではないことに注意する．例えばアトム変数置換 $\pi = (\varnothing \cdot A\ \varnothing \cdot B)$ は，$\pi$ に任意の基底代入 $\rho$ を適用した後，$(\varnothing \cdot A)\rho$ と $(\varnothing \cdot B)\rho$ の置換を行うことを表す．よってアトム変数同士の基底代入後の関係が明らかになっていないうちは，置換をアトムに適用するかのように，アトム変数置換をアトム変数に適用することはできない．例えば，$(\varnothing \cdot A\ \varnothing \cdot B) \cdot C$ は，基底代入 $\rho$ について，$\rho(A) = \rho(C)$ の場合と $\rho(B) = \rho(C)$ の場合，またこれらのどちらでもない場合とで適用結果が変わる．

しかし逆に，アトム変数同士の基底代入後の関係が明らかになっていれば，置換をアトムに適用するのと同じように，アトム変数置換をアトム変数に適用できる場合もある．例えば，$(\varnothing \cdot A\ \varnothing \cdot B) \cdot A$ は，任意の基底代入 $\rho$ について，$\rho(A)$ と $\rho(B)$ の関係にかかわらず，$((\varnothing \cdot A\ \varnothing \cdot B) \cdot A)\rho = (\rho(A)\ \rho(B)) \cdot \rho(A) = \rho(B)$ となる．よってこれは，基底代入前に $(\varnothing \cdot A\ \varnothing \cdot B) \cdot A = \varnothing \cdot B$ と計算しておいても結果は変わらない．アトム変数を用いた名目単一化の手続きでは，このような $NL_{AS}$ 上でのアトム変数置換の計算を多く用いる．

アトム変数置換のアトム変数への適用が保留になっている状態を表すのが，保留アトム変数である．なお，単なるアトム変数は $\varnothing \cdot A$ で表すことができるため，アトム変数を保留アトム変数と別で文法に取り入れる必要はない．

また，アトム変数置換 $\pi$ の変数 $S$ への適用 $\pi \cdot S$ はこれ以上計算することはできない．このようなアトム変数置換の変数への適用が保留になっている状態を表すのが，保留変数である．なお，単なる変数も $\varnothing \cdot S$ で表すことができるため，変数を保留変数と別で文法に取り入れる必要はない．

## 3 アトム変数を用いた名目単一化

本章では，アトム変数を用いた名目単一化の手続きについて議論する．以下では，[5] で提案されている推論規則に基づく非決定的な手続きを，前章の構文に対応させたものを考える．推論規則は，[5] と同様に，$NL_{AS}$ 上の等式に関する規則群 $\mathrm{AVN_{OM}U_{NIFY}}$ と非出現制約集合に関する規則群 $\mathrm{AVS_{OL}N_{ABLA}}$ から成る．$\mathrm{AVS_{OL}N_{ABLA}}$ は，$\mathrm{AVN_{OM}U_{NIFY}}$ で等式の集合が空になった場合に呼び出される副手続きである．ここでは，本研究で一部実装を終えている $\mathrm{AVS_{OL}N_{ABLA}}$ のみ扱う．$\mathrm{AVN_{OM}U_{NIFY}}$ については[5](§4.3) を参照されたい．

### 3.1 名目単一化問題

まず，名目単一化問題とその解，単一化子を定義する．

**定義 7.** $NL_{AS}$ 上の等式 $e_1 \doteq e_2$ $(e_1, e_2 \in NL_{AS})$ の集合を $\Gamma$，非出現制約集合を $\nabla$ としたとき，アトム変数を用いた名目単一化問題 (a variable-atom nominal unification problem，以下 VANUP) を $(\Gamma, \nabla)$ と定義する．

**定義 8.** ある基底代入 $\rho$ が VANUP$(\Gamma, \nabla)$ の解 (so-

lution) であるとは，すべての等式 $e_1 \doteq e_2 \in \Gamma$ に対して $\vdash_{NL_a} e_1\rho \sim e_2\rho$ であり，かつ，すべての非出現制約 $A\#e \in \nabla$ に対して $\vdash_{NL_a} \rho(A)\#e\rho$ となるときをいう．VANUP$(\Gamma, \nabla)$ が解を持つとき，その $(\Gamma, \nabla)$ は可解 (solvable) であるという．また $(\varnothing, \nabla)$ が可解であるとき，その $\nabla$ は可解であるという．

**定義 9.** VANUP$(\Gamma, \nabla)$ について，ある代入 $\sigma$ と非出現制約集合 $\nabla'$ の組 $(\sigma, \nabla')$ が以下を満たすとき，その組 $(\sigma, \nabla')$ を VANUP$(\Gamma, \nabla)$ に対する単一化子 (unifier) であるという．

$\nabla'$ が可解であり，かつ，任意の基底代入 $\gamma$ について，$(\forall A\#e \in \nabla'. \vdash_{NL_a} (\sigma(A))\gamma\#e\sigma\gamma)$ ならば，$\sigma \circ \gamma$ が $(\Gamma, \nabla)$ に対する解となる．

### 3.2 名目単一化手続き

本節では，前章の構文に適合させ，新たな規則を取り入れた $\mathrm{AVS_{OL}N_{ABLA}}$ について詳しく述べる．まず，以下で用いる表記や記号を説明する．

非出現制約集合 $\nabla$ について，$A\#(\pi^{-1} \circ \pi') \cdot B \in \nabla$ または $B\#(\pi'^{-1} \circ \pi) \cdot A \in \nabla$ であるとき，$\nabla \vdash \pi \cdot A \neq \pi' \cdot B$ と書く．特に，$\nabla \vdash \varnothing \cdot A \neq \pi' \cdot B$ の場合は空列 $\varnothing$ を省略し，$\nabla \vdash A \neq \pi' \cdot B$ と書く．

保留アトム変数の集合 $M_1$，$M_2$ について，$M_1$ と $M_2$ に含まれるすべての保留アトム変数の組 $\pi \cdot A \in M_1$，$\pi' \cdot B \in M_2$ に対して $\nabla \vdash \pi \cdot A \neq \pi' \cdot B$ であるとき，$\nabla \vdash M_1\#M_2$ と書く．

保留アトム変数の集合 $M$ について，$M$ に含まれるすべての保留アトム変数の組 $\pi \cdot A$，$\pi' \cdot B \in M$ に対して $\nabla \vdash \pi \cdot A \neq \pi' \cdot B$ であるとき，$\nabla \vdash \#M$ と書く．

**定義 10.** アトム変数置換 $\pi$ について，保留アトム変数の集合 $SusAt(\pi)$ を以下のように定義する．

$$SusAt(\varnothing) := \varnothing$$
$$SusAt((\pi' \cdot A\ \pi'' \cdot B) \cdot \pi) := \{\pi' \cdot A, \pi'' \cdot B\} \cup SusAt(\pi)$$

例えば，$\pi = (\varnothing \cdot A\ \varnothing \cdot B)((\varnothing \cdot C\ \varnothing \cdot D) \cdot A\ \varnothing \cdot B)$ のとき，$SusAt(\pi) = \{\varnothing \cdot A, \varnothing \cdot B, (\varnothing \cdot C\ \varnothing \cdot D) \cdot A\}$ となる．

$\mathrm{AVS_{OL}N_{ABLA}}$ は，非出現制約集合 $\nabla$ と代入 $\theta$ の組 $(\nabla, \theta)$ を入力とし，非出現制約集合 $\nabla$ に代入 $\theta$ を適用した結果得られる非出現制約集合が可解かどうかを判定するアルゴリズムである．$\mathrm{AVS_{OL}N_{ABLA}}$ は後述する九つのルールから成っており，ここではまず重要な三つのルール (Rew1)，(Rew2)，(Simp) について説明する．

アトム変数置換 $\pi$ と保留アトム変数 $\pi \cdot A$ を簡単化するルール (Rew1)，(Rew2) を図 3 に示す．これらのルールは，アトム変数置換をより短い形に変形したり，アトム変数置換をアトム変数に適用することで，アトム変数置換や保留アトム変数を簡単化する．

ルール 1 は，同じ保留アトム変数同士のアトム変数互換を削除するルールである．同じ保留アトム変数同士のアトム変数互換は恒等置換となるため，空列 $\varnothing$ と同義である．

ルール 2 は，アトム変数置換を最小限の数のアトム変数互換で表現するルールである．アトムについての置換 $\tau$ は，($(\tau$に含まれるアトムの数 ) – 1) 個以下の互換の列で表すことができる．一定の条件を満たせば，アトム変数置換においてもこの性質を当てはめることができる．ルール 2 ではこの性質を利用して，アトム変数置換に含まれるアトム変数互換の数を最小限にする．詳しくは実装の章で説明する．

ルール 3 は，アトム変数置換を保留アトム変数に適用するルールである．ある保留アトム変数について，アトム変数置換の中に $(\pi \cdot A\ \pi' \cdot B)$ が含まれており，かつその右側のアトム変数置換が $\pi$，アトム変数が $A$ となっているとき，$(\pi \cdot A\ \pi' \cdot B)$ を右側の $\pi \cdot A$ の部分に適用することができる．このときの適用結果は $\pi' \cdot B$ である．

ルール 4 は，保留アトム変数 $\pi \cdot A$ について，$\pi$ に含まれる $A$ と関係のないアトム変数置換を削除するルールである．例えば $(\varnothing \cdot C\ \varnothing \cdot D) \cdot A$ について，$\nabla \vdash \{A\}\#\{\varnothing \cdot C, \varnothing \cdot D\}$ のとき，どんな基底代入 $\rho$ に対しても，$\rho(A) \neq \rho(C)$ かつ $\rho(A) \neq \rho(D)$ である．よって $(\varnothing \cdot C\ \varnothing \cdot D) \cdot A$ は，どんな基底代入 $\rho$ に対しても結果は $\rho(A)$ となる．このような場合，アトム変数置換 $(\varnothing \cdot C\ \varnothing \cdot D)$ は基底代入前に削除しても結果は変わらない．このようなアトム変数置換を削除するルールが，ルール 4 である．またこのルールは $((\varnothing \cdot C\ \varnothing \cdot D) \circ \pi) \cdot A$ のように，削除対象となるアトム変数置換とアトム変数との間に別のアトム変数置換 $\pi$ が挟まっていて

(Rew1)，(Rew2)

1. $(\pi \cdot A\ \pi \cdot A) \overset{\nabla}{\rightarrow} \varnothing$
2. $\pi \overset{\nabla}{\rightarrow} \pi'$　$\pi$ に含まれるすべての保留アトム変数のアトム変数置換が $\varnothing$ であり，かつ $\nabla \vdash \#SusAt(\pi)$ であり，かつ $\pi$ に含まれるアトム変数互換の数が $|SusAt(\pi)|$ 以上であり，かつ $\pi'$ が $\pi$ と同じアトム変数置換を表すもので，$\pi'$ に含まれるアトム変数互換の数が $|SusAt(\pi)|$ 未満のとき
3. $(\pi'' \circ (\pi \cdot A\ \pi' \cdot B) \circ \pi) \cdot A \overset{\nabla}{\rightarrow} (\pi'' \circ \pi') \cdot B$
4. $(\pi \circ (\pi' \cdot C\ \pi'' \cdot D) \circ \pi''') \cdot A \overset{\nabla}{\rightarrow} (\pi \circ \pi''') \cdot A$
   $(\nabla \vdash \{A\}\#\{\pi' \cdot C, \pi'' \cdot D\}) \wedge (\nabla \vdash \{\pi' \cdot C, \pi'' \cdot D\}\#SusAt(\pi'''))$ のとき．
5. $\pi'' \circ (\pi \cdot A\ \pi' \cdot B) \circ \pi''' \circ (\pi \cdot A\ \pi' \cdot B) \circ \pi'''' \overset{\nabla}{\rightarrow} \pi'' \circ \pi''' \circ \pi''''$　$\nabla \vdash \{\pi \cdot A, \pi' \cdot B\}\#SusAt(\pi''')$ のとき

**図 3　(Rew1)，(Rew2) の定義**

も，条件を満たせば削除することができる．例えば $((\varnothing \cdot C\ \varnothing \cdot D) \circ \pi) \cdot A$ について $\nabla \vdash \{A\}\#\{\varnothing \cdot C, \varnothing \cdot D\}$ であり，かつ $\nabla \vdash \{\varnothing \cdot C, \varnothing \cdot D\}\#SusAt(\pi)$ が成り立っているとき，$(\varnothing \cdot C\ \varnothing \cdot D)$ と $\pi$ は互いに異なる保留アトム変数に対する操作であるため，順番を入れ替えても適用結果に影響はない．つまり $((\varnothing \cdot C\ \varnothing \cdot D) \circ \pi) \cdot A$ と $(\pi \circ (\varnothing \cdot C\ \varnothing \cdot D)) \cdot A$ は同義である．よってこの場合も，$A$ と関係のないアトム変数置換 $(\varnothing \cdot C\ \varnothing \cdot D)$ を削除することができる．このようにルール 4 では条件を満たしていればアトム変数置換の順番を入れ替えることができるため，削除対象のアトム変数置換とアトム変数との間に別のアトム変数置換が挟まっている場合でも，対象のアトム変数置換を削除する．

ルール 5 は，同一のアトム変数置換を打ち消しあって削除するルールである．例えば $(\pi \cdot A\ \pi' \cdot B)(\pi \cdot A\ \pi' \cdot B)$ は同一のアトム変数置換が連続しているため，全体として恒等置換になっている．ルール 5 はこのようなアトム変数置換を削除する．またルール 4 と同様の理由から，ルール 5 でも条件を満たしていればアトム変数置換の順番を入れ替えることができるため，間に別のアトム変数置換が挟まっている場合でも，条件を満たしていれば同一のアトム変数置換同士を打ち消しあって削除する．

次に，非出現制約 $A\#e$ を簡単化するルール (Simp) を図 4 に示す．このルールは，非出現制約に含まれる名目項式 $e$ を簡単化したり，非出現制約の右辺に含まれるアトム変数置換を左辺のアトム変数に適用することで，非出現制約を簡単化する．

ルール 1 は，関数適用に対する非出現制約を，関数適用の中に含まれる各名目項式に対する非出現制約に分解するルールである．このとき，関数記号のみについての非出現制約は生成しないことに注意する．

ルール 2 は，ラムダ式の中に関数適用が含まれている非出現制約を，ラムダ式ごと関数適用の中の各名目項式に対する非出現制約に分解するルールである．これもルール 1 同様，関数記号のみについての非出現制約は生成しないことに注意する．

ルール 3 は，非出現制約の左辺のアトム変数が，右辺でラムダ式によって束縛されているとき，その非出現制約を削除するルールである．非出現制約 $A\#e$ は，基底代入 $\rho$ が非出現関係 $\vdash_{NL_a} \rho(A)\#e\rho$ を満たすという制約を示すものである．しかし $A\#\lambda\varnothing \cdot A.e$ は，任意の基底代入 $\rho$ に対して $\vdash_{NL_a} \rho(A)\#\lambda\rho(A).e\rho$ となり，どんな基底代入を適用しても非出現関係が成立する，意味の無い制約である．ルール 3 はこのような非出現制約を削除する．

ルール 4 は，右辺にアトム変数と変数を含まない非出現制約を削除するルールである．右辺にアトム変数と数を含んでいない非出現制約は，ルール 3 で削除する非出現制約同様，任意の基底代入に対して非出現関係が成立する，意味の無い制約である．ここで，ラムダ式のラムダの直後のアトム変数は，このルールの条件に関わるアトム変数として数えないことに注意する．ラムダ式のラムダの直後のアトム変数が存在しても，それ以降に続く名目項式の中にアトム変数や変数が存在しなければ，その制約は意味の無い制約で

(Simp)

1. $\{A\#\langle f\ e_1\ \ldots\ e_{ar(f)}\rangle\} \uplus \nabla \overset{\#}{\rightarrow} \{A\#e_1,\ \ldots\ , A\#e_{ar(f)}\} \uplus \nabla$
2. $\{A\#\lambda\pi\cdot B.\langle f\ e_1\ \ldots\ e_{ar(f)}\rangle\} \uplus \nabla \overset{\#}{\rightarrow} \{A\#\lambda\pi\cdot B.e_1,\ \ldots\ , A\#\lambda\pi\cdot B.e_{ar(f)}\} \uplus \nabla$
3. $\{A\#\lambda\varnothing\cdot A.e\} \uplus \nabla \overset{\#}{\rightarrow} \nabla$
4. $\{A\#e\} \uplus \nabla \overset{\#}{\rightarrow} \nabla$ 　$e$ が (ラムダ抽象以外の) アトム変数と変数をどちらも含まないとき
5. $\{A\#\lambda\pi\cdot B.e\} \uplus \nabla \overset{\#}{\rightarrow} \{A\#e\} \uplus \nabla$ 　$\nabla \vdash A \neq \pi\cdot B$

6-a. $\{A\#((\varnothing\cdot A\ \pi\cdot B)\cdot\pi')\cdot X\} \uplus \nabla \overset{\#}{\rightarrow} \{B\#(\pi^{-1}\cdot\pi')\cdot X\} \uplus \nabla$ 　($X$ はアトム変数または変数)

6-b. $\{A\#\lambda((\varnothing\cdot A\ \pi\cdot B)\cdot\pi')\cdot C.e\} \uplus \nabla \overset{\#}{\rightarrow} \{B\#\lambda(\pi^{-1}\cdot\pi')\cdot C.((\pi^{-1}\cdot(\varnothing\cdot A\ \pi\cdot B))\cdot e)\} \uplus \nabla$

7-a. $\{A\#(\pi''\circ(\pi\cdot D\ \pi'\cdot E)\circ\pi''')\cdot X\} \uplus \nabla \overset{\#}{\rightarrow} \{A\#(\pi''\circ\pi''')\cdot X\} \uplus \nabla$ 　($X$ はアトム変数または変数)
$(\nabla \vdash \{A\}\#\{\pi\cdot D, \pi'\cdot E\})\ \wedge\ (\nabla \vdash \{\pi\cdot D, \pi'\cdot E\}\#SusAt(\pi''))$ のとき

7-b. $\{A\#\lambda(\pi''\circ(\pi\cdot D\ \pi'\cdot E)\circ\pi''')\cdot F.e\} \uplus \nabla \overset{\#}{\rightarrow} \{A\#\lambda(\pi''\circ\pi''')\cdot F.((\pi\cdot D\ \pi'\cdot E)\cdot e)\} \uplus \nabla$
$(\nabla \vdash \{A\}\#\{\pi\cdot D, \pi'\cdot E\})\ \wedge\ (\nabla \vdash \{\pi\cdot D, \pi'\cdot E\}\#SusAt(\pi''))$ のとき

**図 4　(Simp) の定義**

ある．

ルール 5 は，ラムダ式で束縛されているアトム変数が，左辺のアトム変数と非出現制約で結ばれている場合，そのラムダ式に対する非出現制約を，ラムダ式の中の名目項式に対する非出現制約に変更するルールである．例えば $A\#\lambda\pi\cdot B.e$ について，$\nabla \vdash A\#\pi\cdot B$ のとき，任意の基底代入 $\rho$ について $\rho(A) \neq (\pi\cdot B)\rho$ である．よって $\vdash_{NL_a} \rho(A)\#\lambda(\pi\cdot B)\rho.e\rho$ は図 1 の定義より，$\vdash_{NL_a} \rho(A)\#e\rho$ に置き換えられる．つまりこの場合，$A\#\lambda\pi\cdot B.e$ は $A\#e$ と同義である．ルール 5 はこのように，ラムダ式に対する非出現制約を変形する．

ルール 6-a，6-b は，右辺のアトム変数置換を左辺のアトム変数に適用するルールである．例えば 6-a は，右辺が保留アトム変数または保留変数の場合であり，アトム変数置換の一番左のアトム変数互換に，空列と左辺のアトム変数の組の保留変数が含まれているならば，そのアトム変数互換を右辺から削除し，左辺に適用する．このとき左辺の適用結果の保留アトム変数のアトム変数置換は，逆順にして右辺に移項する．ルール 6-b はルール 6-a の右辺がラムダ式の場合であり，ラムダの直後の保留アトム変数のアトム変数置換について 6-a と同様の操作を行う．このとき，右辺のラムダ式のアトム変数の束縛の関係を保つために，ラムダ式に含まれる名目項式にもアトム変数互換を適用することに注意する．

ルール 7-a，7-b は，右辺に含まれる，左辺のアトム変数に関係のないアトム変数置換を削除するルールである．これらは (Rew1)，(Rew2) のルール 4 とよく似ている．(Rew1)，(Rew2) のルール 4 は保留アトム変数 $\pi\cdot A$ について $A$ と関係のないアトム変数置換を削除したが，(Simp) のルール 7-a，7-b は非出現制約 $A\#e$ について，$e$ に含まれる $A$ と関係のないアトム変数置換を削除する．7-a は右辺が保留アトム変数または保留変数の場合であり，7-b は右辺がラムダ式の場合である．7-b も 6-b と同様に，右辺のラムダ式のアトム変数の束縛の関係を保つために，適用後のラムダ式に含まれる名目項式にも，削除するアトム変数置換を適用することに注意する．また (Rew1)，(Rew2) のルール 4 やルール 5 同様，条件を満たせばアトム変数置換の順番を入れ替えることができるため，削除対象のアトム変数置換の左側に別のアトム変数置換が挟まっている場合でも，条件を満たしていれば対象のアトム変数置換を削除する．

$\mathrm{AVS_{OL}N_{ABLA}}$ は以上の (Rew1)，(Rew2)，(Simp) の三つのルールを含む，計九つのルールからなる．$\mathrm{AVS_{OL}N_{ABLA}}$ を図 5 に示す．図中では $[\pi\cdot B/A]$ という表記を用いているが，これはアトム変数 $A$ への保留アトム変数 $\pi\cdot B$ の代入操作を表す．また，$[\pi'/\pi]$ はアトム変数置換 $\pi$ をアトム変数置換 $\pi'$ で置き換え

$\mathrm{AVS_{OL}N_{ABLA}}$

$$\text{(Rew1)}\ \frac{(\nabla,\theta)}{(\nabla,\theta)[\pi'/\pi]}\ \pi \overset{\nabla}{\to} \pi' \text{ のとき}$$

$$\text{(Rew2)}\ \frac{(\nabla,\theta)}{(\nabla,\theta)[\pi'\cdot A'/\pi\cdot A]}\ \pi\cdot A \overset{\nabla}{\to} \pi'\cdot A' \text{ のとき}$$

$$\text{(Simp)}\ \frac{(\nabla,\theta)}{(\nabla',\theta)}\ \nabla \overset{\#}{\to} \nabla' \text{ のとき}$$

$$\text{(FreshnessFail)}\ \frac{(\Gamma,\nabla\uplus\{A\#\varnothing\cdot A\},\theta)}{\mathit{Fail}}$$

$$\text{(GuessEQFC)}\ \frac{(\nabla,\theta)}{(\nabla[\varnothing\cdot A_2/A_1],\theta[\varnothing\cdot A_2/A_1])\mid(\nabla\uplus\{A_1\#\varnothing\cdot A_2\},\theta)}\ \begin{array}{l}\text{他のどのルールも適用できず,}\\ \text{かつ } A_1,\ A_2 \text{ が } \nabla \text{ か}\\ \theta \text{ の定義域以外に現れているとき}\end{array}$$

$$\text{(NSubst1)}\ \frac{(\nabla\uplus\{A\#\pi\cdot S\},\theta\cup\{S\mapsto e\})}{(\nabla\uplus\{A\#\pi\bullet e\},\theta\cup\{S\mapsto e\})}\ S \text{ が } \theta \text{ の中に現れていないとき}$$

$$\text{(NSubst2)}\ \frac{(\nabla,\theta\uplus\{A\mapsto\varnothing\cdot B\})}{(\nabla[\varnothing\cdot B/A],\theta[\varnothing\cdot B/A])}$$

$$\text{(NSubst3)}\ \frac{(\nabla,\theta\uplus\{V\mapsto e\})}{(\nabla,\theta)}\ V(A \text{ または } S) \text{ が } \nabla \text{ と } \theta \text{ のどちらにも現れないとき}$$

$$\text{(Sat)}\ \frac{(\nabla,\theta)}{\text{"satisfiable"}}\ \begin{array}{l}\nabla \text{ が標準化されていて，かつ } \theta \text{ が空であり,}\\ \text{かつ (FreshnessFail) が適用できないとき}\end{array}$$

**図 5　$\mathbf{AVS_{OL}N_{ABLA}}$ の定義**

る操作，$[\pi'\cdot B/\pi\cdot A]$ は保留アトム変数 $\pi\cdot A$ を保留アトム変数 $\pi'\cdot B$ で置き換える操作を表す．

(FreshnessFail) は，非出現制約集合 $\nabla$ が可解でない場合に *Fail* を返し，アルゴリズムを終了するルールである．具体的には，$\nabla$ が $A\#\varnothing\cdot A$ の形の非出現制約を含んでるときに，この非出現制約集合は可解ではないと判断し，アルゴリズムを終了する．

(GuessEQFC) は，$\mathrm{AVS_{OL}N_{ABLA}}$ の他のどのルールも適用できないときに，$(\nabla,\theta)$ に含まれる二つのアトム変数同士の基底代入後の関係を仮定することで，アルゴリズムの適用を進めるためのルールである．例えば入力 $(\nabla,\theta)$ について，$\nabla=\{A\#\lambda\varnothing\cdot B.e\}$，$\theta=\varnothing$ のとき，この $(\nabla,\theta)$ にはこれ以上 (GuessEQFE) 以外の $\mathrm{AVS_{OL}N_{ABLA}}$ のルールを適用することはできない．このようなときに (GuessEQFC) を適用すると，任意の基底代入 $\rho$ について $\rho(A)=\rho(B)$ と仮定した場合の $(\nabla[\varnothing\cdot B/A],\theta[\varnothing\cdot B/A])$ と，$\rho(A)\neq\rho(B)$ と仮定した場合の $(\nabla\uplus\{A\#\varnothing\cdot B\},\theta)$ の両方を出力する．これにより，それぞれの分岐について再び $\mathrm{AVS_{OL}N_{ABLA}}$ の適用を進めていくことができるようになり，$\mathrm{AVS_{OL}N_{ABLA}}$ は最終的にそれぞれの分岐についての結果を出力する．$\mathrm{AVS_{OL}N_{ABLA}}$ の他のどのルールも適用できないときにのみ，(GuessEQFC) が適用できることに注意する．

(NSubst1) と (NSubst2) は，$\theta$ を $\nabla$ に適用するルールである．(Nsubst1) は変数への名目項式の項の代入，(NSubst2) はアトム変数への保留アトム変数の代入を行う．(NSubst2) は，代入後の保留アトム変数のアトム変数置換が空列 $\varnothing$ の場合にのみ適用することに注意する．(NSubst3) は，適用先のない代入を削

除するルールである．

(Sat) は，$(\nabla, \theta)$ が可解だった場合に “satisfiable” を返し，アルゴリズムを終了するルールである．$\nabla$ が標準化されていて $\theta$ が空のとき，(FreshnessFail) が適用できなければ，その $(\nabla, \theta)$ は可解であると判断し，アルゴリズムを終了する．

$\mathrm{AVS_{OL}N_{ABLA}}$ は以上のルールを，*Fail* か “satisfiable” か出力されるまで繰り返し適用する．次章では，この $\mathrm{AVS_{OL}N_{ABLA}}$ のルールの一部を実装した結果について報告する．

## 4 実装と実行例

### 4.1 実装

本研究では関数型言語 SML#を用いて実装を行っている．実装の計画としてはまず $\mathrm{AVS_{OL}N_{ABLA}}$ の実装を行い，その後 $\mathrm{AVN_{OM}U_{NIFY}}$ の実装を行う．$\mathrm{AVS_{OL}N_{ABLA}}$ の方針は以下の通りである．1. 入力に対して (Rew1)，(Rew2)，(Simp)，(FreshnessFail) を，出力が変化しなくなるか，FreshnessFail の例外が返ってくるまで繰り返し適用する．2. (Sat) を適用できるか確かめる．適用できたら終わり．適用できなければ 3 に続く．3. (NSubst1)，(NSubst2)，(NSubst3) のいずれかを適用する．どれも適用できない場合は (GuessEQFC) を適用する．4. 1 に戻る．現段階では，非出現制約集合 $\nabla$ を入力とし，$\nabla$ に対して 1 を行う関数 smpN の実装と動作確認が終了している．

ここで，(Rew1)，(Rew2) のルール 2 について説明する．アトムについての置換 $\tau$ は，定義域 $dom(\tau)$ に含まれる任意のアトムについて，$\tau$ を適用し，適用後のアトムにまた $\tau$ を適用し， ... と $\tau$ 自身の適用先のアトムを辿っていくと，その道筋は循環し，最終的に必ず元のアトムにたどり着く．例えば $(a\ b)(a\ c)(b\ c)(a\ b)$ という置換の関数としての振る舞いは $\{a \mapsto b, b \mapsto c, c \mapsto a\}$ であり，$a$ から順にこの置換を適用した先のアトムを辿っていくと，$a \to b \to c \to a \to$ ... と循環する．この循環は一つの置換につき一つとは限らず，二つ以上に分かれることもある．例えば $(a\ b)(b\ c)(c\ d)(b\ d)$ の関数としての振る舞いは $\{a \mapsto b, b \mapsto a, c \mapsto d, d \mapsto c\}$ となり，$a \to b \to a \to$ ... と $c \to d \to c \to$ ... という二つの循環に分かれる．いずれにせよ，置換 $\tau$ の関数としての振る舞いは必ずこのような一つ以上の循環で表すことができ，この循環は (( 循環に含まれるアトムの数 ) – 1) 個の互換の列で表すことができる．例えば $a \to b \to c \to a \to$ ... は $(a\ c)(a\ b)$ で，$a \to b \to a \to$ ... は $(a\ b)$ で，$c \to d \to c \to$ ... は $(c\ d)$ で表せる．これを利用すると，アトムについての置換 $\tau$ は必ず，($\tau$に含まれるアトムの数 ) – 1) 個以下の互換の列で表すことができる．例えば，これまでの議論より，$(a\ b)(a\ c)(b\ c)(a\ b)$ は $(a\ c)(a\ b)$ で，$(a\ b)(b\ c)(c\ d)(b\ d)$ は $(a\ b)(c\ d)$ で表せる．

アトム変数置換 $\pi$ に含まれるすべての保留アトム変数のアトム変数置換が $\varnothing$ であり，かつ $\nabla \vdash \#SusAt(\pi)$ であれば，この性質をアトム変数置換 $\pi$ にも当てはめることができる．(Rew1)，(Rew2) のルール 2 では上の条件に加えて，アトム変数置換 $\pi$ に含まれるアトム変数互換の数が $|SusAt(\pi)|$ 以上である場合に，この性質を利用してアトム変数互換の数を最小限にする．実装においては，アトム変数置換 $\pi$ に含まれるすべての保留アトム変数に $\pi$ を適用してすべての循環を見つけ，それぞれの循環を (( 循環に含まれるアトム変数の数 ) – 1) 個のアトム変数互換の列で表し，最後にそれらを結合して新たなアトム変数置換 $\pi'$ を作り $\pi$ と置き換えることで，このルールを実現した．このとき $\pi'$ の関数としての振る舞いは $\pi$ と同じになっており，かつ $\pi$ に含まれるアトム変数互換の数は最小限になっている．

### 4.2 実行例

以下では，実装を行った関数 simpN の入出力例を示す．simpN は非出現制約集合 $\nabla$ を入力とし，$\nabla$ に対して (Rew1)，(Rew2)，(Simp) を出力が変化しなくなるまで繰り返し適用して，適用後の非出現制約集合 $\nabla'$ を出力する．このとき，同時に (FreshnessFial) も繰り返し適用し，FreshnessFail の例外が返ってきた場合は即座に処理を中止する．

実装上の表記には，これまでの表記と異なる点があることに注意する．アトム変数はこれまで同様 `A` や `B` で表すが，変数は`<S>`のように`<>`で囲む．保留アト

(入力)

```
X#(A B) (C D)(C A)(C A)(A B)Y;
A#C;
A#D;
B#C;
B#D;
```

(出力)

```
X#(C D)Y;
A#C;
A#D;
B#C;
B#D;
```

**図 6　simpN 関数の入出力例**

ム変数や保留変数は · を使わず (A B)C や (A B)<S> のように表記する．またアトム変数置換が空列 $\varnothing$ となっている保留アトム変数 $\varnothing \cdot A$ や変数 $\varnothing \cdot S$ は，空列 $\varnothing$ を省略して A や<S>と表記する．関数記号はこれまで同様 f や g で表すが，関数適用式は f A (B C)<S> のように，関数記号の後ろに引数式を空白で区切って書く．ラムダ式は $\lambda$ を\で表す．例えば $\lambda\varnothing \cdot A.\varnothing \cdot S$ は\A.<S>となる．

図 6 に simpN の入出力例を示す．図 6 の入力と出力はともに非出現制約集合を表しており，集合に含まれている各非出現制約が; で区切られて並んでいる．この例は[5](Example 4.3) を実行した例であり，正しく動作していることが分かる．

以下に他の三つの実行例も示す．一つ目は前節で述べた (Rew1)，(Rew2) のルール 2 の実行例である．想定通りに動作している．二つ目は (Rew1)，(Rew2) のルール 3 の実行例である．アトム変数置換 $((\varnothing \cdot B\ \varnothing \cdot C) \cdot D\ \varnothing \cdot E)$ が右側の保留アトム変数 $(\varnothing \cdot B\ \varnothing \cdot C) \cdot D$ に適用され，保留アトム変数 $\varnothing \cdot E$ に置き換わっている．三つ目は (Simp) のルール 7-b の実行例である．左辺のアトム変数 $A$ と関係のない右辺のラムダの直後の保留アトム変数に含まれるアトム変数置換 $(\varnothing \cdot B\ \varnothing \cdot C)$ が，ラムダの直後の保留アトム変数の中から削除され，ラムダ式の中の保留変数 $\varnothing \cdot < S >$ に適用されている．

(入力)

```
X#(A B)(A C)(B C)(A B)Y;
A#B;
A#C;
B#C;
```

(出力)

```
X#(A C)(A B)Y;
A#B;
A#C;
B#C;
```

(入力)

```
A#((B C)D E)(B C)D;
```

(出力)

```
A#E;
```

(入力)

```
A#\(D E)(B C)(F G)H. <S>;
A#B;
A#C;
B#D;
B#E;
C#D;
C#E;
```

(出力)

```
A#\(D E)(F G)H. (B C)<S>;
A#B;
A#C;
B#D;
B#E;
C#D;
C#E;
```

## 5　まとめ

本稿では，アトム変数を用いた名目単一化アルゴリズムの実装に向けて，その取り組みを報告した．[5] で提案された推論規則に基づいて，それらの構文や規則のより実装に適した形への再定義や，新たな規則の導入を行い，その手続きの一部を実装した．

今後の課題としては，$\mathrm{AVS_{OL}N_{ABLA}}$ の残りの規則

の実装と，その後の $\mathrm{AVN_{OM}U_{NIFY}}$ の実装が挙げられる．

**謝辞** 本研究の一部は JSPS 科研費 JP18K11233, JP19K11891, JP19K11893 の助成を受けて行われた．

## 参 考 文 献

[1] Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. Formal Aspects Comput. 13, 341–363 (2002)

[2] Kikuchi, K., Aoto, T.: Confluence and commutation for nominal rewriting systems with atom-variables. In: Proceedings of the 30th LOPSTR. LNCS, vol. 12561, pp. 56–73. Springer-Verlag (2021)

[3] Kutz, Y., Schmidt-Schauß, M.: Rewriting with generalized nominal unification. Math. Struct. Comput. Sci. 30, 710–735 (2020)

[4] Pitts, A.M.: Nominal logic, a first order theory of names and binding. Inform. Comput. 186, 165–193 (2003)

[5] Schmidt-Schauß, M., Sabel, D., Kutz, Y.D.K.: Nominal unification with atom-variables. J. Symb. Comput. 90, 42–64 (2019)

[6] Urban, C., Pitts, A.M., Gabbay, M.J.: Nominal unification. In: Proceedings of the 17th CSL. LNCS, vol. 2803, pp. 513–527. Springer-Verlag (2003)

[7] Urban, C., Pitts, A.M., Gabbay, M.J.: Nominal unification. Theoret. Comput. Sci. 323, 473–497 (2004)