CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Kyoto University Research Information Repository
KURENAI
Kyoto University Research Information Repository
KYOTO UNIVERSITY

Title	等号公理下での論理式の標準形とその一階言語への応用 (数式処理における理論と応用の研究)
Author(s)	元吉, 文男; 秋葉, 澄孝; 佐藤, 泰介
Citation	数理解析研究所講究録 (2000), 1138: 220-225
Issue Date	2000-04
URL	http://hdl.handle.net/2433/63813
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

Kyoto University

数理解析研究所講究録
1138巻 2000年 220-225

等号公理下での論理式の標準形とその一階言語への応用

電子技術総合研究所　元吉　文男(Fumio Motoyoshi) *
電子技術総合研究所　秋葉　澄孝(Sumitaka Akiba) †
東工大　佐藤　泰介(Taisuke Sato) ‡

1　はじめに

Clarkの公理系に関する論理式を、自由変数に対する代入という形の論理和として表現するアルゴリズムが知られており、それを利用して一階言語という述語言語が開発された。しかし、一階言語では、ユーザ定義述語を含む論理式から解代入を求めるために、一旦、ユーザ述語を含まない式に変形し、それに対して上記のアルゴリズムを適用しているために、解を逐次的に求める場合には、余分な操作が必要であった、

ここでは、ユーザ述語を含んだままの式を変形して、標準形と呼ぶ形に変形する方法を提案し、その標準形が従来の計算で導かれる解を含んでいると同時に、ユーザ述語を含む残りの解を表していることを示す。

2　Clarkの等号公理と標準形

Σ を関数記号の集合 (含定数) としたときに、以下の公理系のことをClarkの等号公理 $E(\Sigma)$ という。

$$
\begin{aligned}
\{\quad & f(x_1,x_2,\ldots,x_n)=f(y_1,y_2,\ldots,y_n)\rightarrow x_1=y_1\wedge x_2=y_2\wedge\cdots\wedge x_n=y_n,\\
& \neg f(x_1,x_2,\ldots,x_n)=g(y_1,y_2,\ldots,y_m),\\
& \neg x=e[x]\\
& \mid\quad f,g\in\Sigma,\quad f\neq g\}
\end{aligned}
$$

*moto@etl.go.jp
†akiba@etl.go.jp
‡sato@cs.titech.ac.jp

ここで、$e[x]$ は、x とは異なり、かつ、x を陽に自由変数として持つ任意の論理式である。

なお、本稿では $|\Sigma| = \infty$ の場合を扱う。Clark の公理系は、融合原理における Herbrand 世界の同値関係を公理化したものと考えられる。

表記法に関して、本稿では、文字の並びをボールド体の文字で表わし、それらに関する関係は対応する要素同士の関係を意味するものとする。たとえば、$\boldsymbol{a} = \boldsymbol{b}$ は $a_1 = b_1, \cdots, a_n = b_n$ を表わす。さらに、$E(\Sigma)$ での等号記号と通常の等号とは文脈で区別することができるが、一応、= とし通常の等号である $=$ とは区別して書く。

また、論理式の解釈として 3 値のモデルを使用するために、通常の二値の論理値 (真 T、偽 F) 以外に未定 U という値も使用する。これらの記号は常にその論理値をとる定数述語としても使用する。さらに、これらの並びも同じ記号で表すことにする。

$E(\Sigma)$ と任意のユーザ述語を含む論理式の標準形 DNF とは以下の形の論理式のこととする。

$$
\begin{aligned}
DNF &:= CF \vee CF \vee \cdots \vee CF \\
CF &:= XBF \wedge \neg BF \wedge \neg BF \wedge \cdots \wedge \neg BF \\
BF &:= \exists y_1, y_2, \ldots, y_m (x_1{=}e_1 \wedge x_2{=}e_2 \wedge \cdots \wedge x_n{=}e_n) \\
&\equiv \exists \boldsymbol{y}(\boldsymbol{x}{=}\boldsymbol{e}) \\
XBF &:= \exists y_1, y_2, \ldots, y_m (x_1{=}e_1 \wedge x_2{=}e_2 \wedge \cdots \wedge x_n{=}e_n \wedge P) \\
&\equiv \exists \boldsymbol{y}(\boldsymbol{x}{=}\boldsymbol{e} \wedge P)
\end{aligned}
$$

ここで、XBF における P は、常に真である T か、ユーザ述語に U という解釈をしたときに U となる論理式である。

上記の定義において、P が真 T の場合には BF と同じ形になるため、XBF は BF の拡張となっている。また、この標準形で、ユーザ述語を含む可能性のあるのは XBF 中だけである。なお、P が真 T となる XBF を含む CF は述語の解釈によらず、自由変数への解代入を表現しているため、変形する前の論理式の確定部分と呼ぶ。

3 標準形への変形

まず、BF に関しての簡約化手続きを述べる。

BF あるいは XBF 中の等号の右辺に表われる式のうち、それが変数であり、かつ、その変数が同じ BF 中の他の式に現れていないときに、その右辺を孤立変数と呼ぶ。その変

数を y_1 とし、$e_1 \equiv y_1$ とする。

$$\begin{aligned} XBF &\equiv \exists y_1, y_2, \ldots, y_m \, (x_1{=}y_1 \wedge x_2{=}e_2 \wedge \cdots \wedge x_n{=}e_n \wedge P) \\ &\Leftrightarrow (\exists y_1 \, x_1{=}y_1) \wedge \exists y_2, \ldots, y_m \, (x_2{=}e_2 \wedge \cdots \wedge x_n{=}e_n \wedge P) \\ &\Leftrightarrow \exists y_2, \ldots, y_m \, (x_2{=}e_2 \wedge \cdots \wedge x_n{=}e_n \wedge P) \end{aligned}$$

この性質を利用すると、右辺の孤立した変数は消去することが可能であり、消去すべき孤立変数が存在していない BF あるいは XBF は簡約化されているということにする。

与えられた論理式を標準形に変形する手続きは以下のいずれかの操作を内側から順に適用することによって実行する。

- 一番内側の式は、= の式かユーザ述語のいずれかであり、その場合は以下の規則を適用する。

  1. $p(\boldsymbol{x}) \quad \Rightarrow \quad \exists . p(\boldsymbol{x})$
  2. $x{=}e[\boldsymbol{y}] \quad \Rightarrow \quad \exists \boldsymbol{z} \, (\boldsymbol{y}{=}\boldsymbol{z} \wedge x{=}e[\boldsymbol{z}])$

  ユーザ述語だけの式は XBF で = の項がない特殊な形である。

- 外側に向って処理する際に、限量子以外の演算の場合にはド・モルガンの定理と分配則を使用して以下の基本演算に帰着させる。

  1. $XBF \wedge XBF$ の場合
     $\exists \boldsymbol{y_1}(\boldsymbol{x_1}{=}\boldsymbol{e_1} \wedge P_1) \wedge \exists \boldsymbol{y_2}(\boldsymbol{x_2}{=}\boldsymbol{e_2} \wedge P_2)$ の式において、$\boldsymbol{y_1}$ と $\boldsymbol{y_2}$ には共通部分がないようにして、2 つの限量子の有効範囲を式全体にかかるようにしておく。そこで、等式部分の MGU(most general unifier) を計算する。MGU が存在しないときには、全体の値は F になる。存在するときには MGU を代入して整理することによって新たな XBF を得ることができる。
  2. $\neg XBF$ の場合
     $\neg \exists \boldsymbol{y}(\boldsymbol{x}{=}\boldsymbol{e} \wedge P) \quad \Rightarrow \quad \neg \exists \boldsymbol{y} \, \boldsymbol{x}{=}\boldsymbol{e} \vee \exists \boldsymbol{y}(\boldsymbol{x}{=}\boldsymbol{e} \wedge \neg P)$

- 全称限量子 $\forall$ の場合には次の変形を行なって、存在限量子 $\exists$ の場合に帰着させる。

$$\forall \boldsymbol{x} \, P \quad \Rightarrow \quad \neg \exists \boldsymbol{x} \, \neg P$$

- 存在限量子 $\exists$ の場合 $\exists y(\exists \boldsymbol{z}_0 . (\boldsymbol{x}_0{=}\boldsymbol{e} \wedge p) \wedge \neg \exists \boldsymbol{z}_1 . \boldsymbol{x}_1{=}\boldsymbol{e} \wedge \cdots \wedge \neg \exists \boldsymbol{z}_n . \boldsymbol{x}_n{=}\boldsymbol{e})$ から y を消去する

  1. 各 i $(1 \geq i \geq n)$ に対して、$(\exists \boldsymbol{z}_i . \boldsymbol{x}_i{=}\boldsymbol{e}_i)\{\boldsymbol{e}/\boldsymbol{x}_0\}$ の代入を行ない、簡約化して、それを BF_i とする。

2. – $y \in \boldsymbol{x}_0$ のとき
   $\boldsymbol{u} = (\boldsymbol{x}_0\text{-}y$ に対応する右辺の自由変数)
   $\boldsymbol{v} = \boldsymbol{z}_0\text{-}\boldsymbol{u}$

   – $y \notin \boldsymbol{x}_0$ のとき
   $\boldsymbol{u} = \boldsymbol{z}_0$
   $\boldsymbol{v} = \{y\}$

   とする。

3. BF_i のうち、その自由変数と v が共通部分を持つときはそれを $\boldsymbol{BFN}$ に入れ、持たないときは $\boldsymbol{BF}$ に入れる。

4. p の自由変数と v が共通部分を持ち、かつ、$\boldsymbol{BFN}$ が空ではないときには $p := \exists \boldsymbol{v}.(p \wedge \boldsymbol{BFN})$ とする

5. 答は $\exists \boldsymbol{z}'_0.(\boldsymbol{x}_0\text{-}y\text{=}\boldsymbol{e}'_0 \wedge p) \wedge \boldsymbol{BF}$ となる。

上記の 4. の変形において、BFN が空でないときには $\exists \boldsymbol{v}\, BFN$ は真になることが知られているので、ユーザ述語を U に解釈したときには、p は T か U なので、$\exists \boldsymbol{v}.(p \wedge \boldsymbol{BFN})$ も T か U になる。

4 応用

著者等が提案している「一階言語」の実行アルゴリズムとしての使用法を簡単に説明する。

一階言語は同値関係として定義された述語から、ゴールとして与えられた論理式中にある自由変数への解代入を求める言語である。論理型言語として知られている Prolog は融合原理に基づいているが、一階言語は 3 値での同値変換に理論を置いている。その原理とは、次の二つの式が同値であることに依存している:

- $E(\Sigma), \forall \boldsymbol{x}\, (\boldsymbol{p}(\boldsymbol{x}) \Leftrightarrow \boldsymbol{P}[\boldsymbol{x}]) \models_3 G$
- ある n があって
  $E(\Sigma) \models_2 G\{\boldsymbol{P}/\boldsymbol{p}\}^n\{\mathtt{F};\mathtt{T}/\boldsymbol{p}\}$

ここで、$\models_3$ ($\models_2$) の意味は、左辺を真にする任意の 3 (2) 値の解釈おいて右辺が真となることを表わす。また、$\{\boldsymbol{P}/\boldsymbol{p}\}$ は、その左の論理式中に出現する全ての述語 $\boldsymbol{p}$ をその定義 $\boldsymbol{P}$ で展開する操作を表わし、$\{\boldsymbol{P}^+;\boldsymbol{P}^-/\boldsymbol{p}\}$ は、正に出現する $\boldsymbol{p}$ は $\boldsymbol{P}^+$ で、負に出現する $\boldsymbol{p}$ は $\boldsymbol{P}^-$ で展開することを表わすものとする。さらに、ある述語が正 (負) に出現するとは、

論理式のトップレベルからその述語に到達するまでに偶 (奇) 数回の否定がかかっているものをいう。

$G(y)\{\boldsymbol{P}/\boldsymbol{p}\}^n$ に標準形への変形規則を適用することによって、$\{\mathtt{U}/\boldsymbol{p}\}$ の代入を行なわずに以下のように計算が継続できる。

1. $G_0 \equiv G(y)$ とする。
2. G_i から $Gi+1$ を以下のように計算する。
   G_i を標準形に変形し、そのうち、確定部分を A_i、残りの部分を G_{i+1} とする。

このように計算すると、標準形のうち、非確定部分 G_{i+1} の XBF はユーザ述語を含んでおり、$\{\mathtt{U}/\boldsymbol{p}\}$ の代入を行なうと U になる (そのように標準形を定めた) ので、$\{\mathtt{F};\mathtt{T}/\boldsymbol{p}\}$ の代入を行なうと F になり、残るのは A_i だけである。したがって以下の関係が成り立つ。

$$G(y)\{\boldsymbol{P}/\boldsymbol{p}\}^n\{\mathtt{F};\mathtt{T}/\boldsymbol{p}\} \quad \Leftrightarrow \quad A_0 \vee A_1 \vee \cdots \vee A_n$$

このようにして、$G(y)$ の解を逐次的に求めることができる。

例

$e(x) \Leftrightarrow x{=}0 \vee \exists z\,(x{=}s(z) \wedge \neg e(z))$ のとき

$$\begin{aligned}
& \neg e(x)\{P/e\}^2 \\
\Rightarrow\ & \neg(x{=}0 \vee \exists z\,(x{=}s(z) \wedge \neg e(z)))\{P/e\} \\
\Rightarrow\ & \neg(x{=}0 \vee \exists z\,(x{=}s(z) \wedge \neg(z{=}0 \vee \exists u\,(z{=}s(u) \wedge \neg e(u))))) \\
\Rightarrow\ & \neg x{=}0 \wedge \neg\exists z\, x{=}s(z) \ \vee \\
& x{=}s(0) \ \vee \\
& \exists u\,(x{=}s(s(u)) \vee \neg e(u))
\end{aligned}$$

5 まとめ

本稿では、Clark の等号公理で任意の述語を含む式を解代入の論理和という形の標準形に変形する方法を示した。この解は、論理型言語 Prolog の解を一般化して制約を付加した形になっているものである。また応用として、一階言語における変形規則として使用する方法を示し、逐次的に解を求められることを示した。

また、標準形への変形はClarkの公理系に関する論理式を簡単化するためのアルゴリズムとしても、標準形への変形という強力な手続きとして使用が可能であり、プログラム変換などでの利用が可能であると思われる。

本稿の変形方法では、全称限量子の場合には $\forall \boldsymbol{x}\, P \Rightarrow \neg\exists \boldsymbol{x}\, \neg P$ という規則を使用しているが、否定の演算には計算量が多くかかるため、この部分の効率化が必要であり、本稿での標準形と否定をとった論理積の形との2つの標準形を使用するアルゴリズムを開発中である。

参 考 文 献

[1] Clark, K.L: Negation as Failure, *Logic and Databases* (Gallaire, H. and Minker,J. (eds.)), Plenum Press, New York, pp.293–322 (1978).

[2] Kunen, K: Negation in Logic Programming, *Journal of Logic Programming*, Vol. 4, pp.289–308 (1987).

[3] LLoyd, J.W.: *Foundations of Logic Programming*, Springer-Verlag, 2nd, extended edition (1987).

[4] Motoyoshi, F and Sato, T: Implementation of Augmented Logic Language (ALL), *Advances in Software Science and Technology*, Vol. 5, pp.91–106 (1993).

[5] Sato, T.: Quantifier Elimination for Finite and Infinite Trees, *Technical Report TR-89-25*, Electrotechnical Laboratory, Tsukuba (1989).

[6] Shepherdson, J.C.: Nagation in Logic Programming, *Foundations of Deductive Databases and Logic Programming* (Minker,J. (ed.)), Morgan kaufmann, pp.19–88 (1988).