記号処理 41－1
（1987. 6. 26）

論理式の簡単化

元吉文男・佐藤泰介

電子技術総合研究所推論システム研究室

ユニファイ可能という述語を含んだ論理式に関して、その意味まで考慮して簡単化を行うための規則と方法を紹介する。これにより、Ｐｒｏｌｏｇ等の論理型言語で記述されたプログラムを合成・変換するという操作において得られた結果から冗長な部分を削除して「簡単な」式にすることができるようになる。また数式処理においても、明示的でない集合を扱う場合等において論理式自体を操作する必要が生じてくるが、その場合においてもこのアルゴリズムを基本にすることができる。

Simplification of First Order Formulae

FUMIO MOTOYOSHI and TAISUKE SATO

ElectroTechnical Laboratory, Umezono 1-1-4, Sakura-mura,
Ibaraki-ken, Japan.

This paper presents an algorithm to simplify first order formulae. Although simplification algorithms for Boolean expressions were studied in the field of switching theory and almost established, there has been few researches on that of first order formulae inspite of many studies on theorem proving. Our algorithm simplifies the formulae on Herbrand universe, where all predicate symbols have no meanings and are differenciated from others only by their names exept the predicate 'unifiable'. Simplifications on Herbrand universe are actually necessary in the area of program transformation or program synthesis and our algorithm produces plausible results in that problem.

1．はじめに

現在、Ｐｒｏｌｏｇを始めとして論理型のプログラミング言語が多く使用されているが、これらの言語についてプログラム合成のような操作を行う場合に、論理式の処理に帰着することができるので、正当性がそのまま保証されている。しかし、論理型言語で使用する論理式は単なる記号としての意味しか持たないものでは不十分であり、基本的述語に関する意味を含んだものである必要がある。

また数式処理においても単なる数式だけでなく、より広い範囲の数学的対象を扱おうとする動きがあるが、たとえば集合を扱うにしても論理式によって示される明示的でない集合の演算においては、論理式の処理が必要になる。そこにおいても単なる記号としてではなく、意味を持った述語が含まれている。

そこでこのような論理式を処理するプログラムが必要となるが、演算の結果である論理式は冗長になるのが普通であり、その式を簡単にすることが求められることになるが、このような式の簡単化を一般に処理することは行われていなかった。限量子を含まない論理式、すなわち命題論理式の簡単化については、ブール関数の簡単化として回路設計の分野で多くの研究がなされ、ほぼ満足すべき結果がえられている。しかし、限量子を含んだ論理式の一般的な簡単化についてはほとんど行われていない。

ここで紹介するのは、単なる記号としての論理式に、レゾリューション原理において最も基本的な演算であるユニファイという意味を加えた論理式を簡単化するアルゴリズムである。

2．記法と用語

この節では3節以降で使用する記号の意味と用法を述べ。なお一般的に用いられているものについてはそれに従うことにし、ここでは、本報告において独特であるのもを中心に述べることにする。

記号の意味

この報告においてはアルファベットを次のように使いわけて使用する。1つの文脈中で同じ記号が用いられたときには、それらはすべて同じものを示しているものと考える。なお添字付きの記号については、添字まで含めて考えることにし、その表わす意味は別添字がないものと同じものとする。

ｕ、ｖ、ｗ、ｘ、ｙ、ｚ	変数
ｆ、ｇ	関数名
ｒ、ｓ、ｔ	述語名

Q	限量子（∀または∃）
Q'	Qとは異なる限量子
a、b、c、d、e	項
p、q	論理式
==	ユニファイ可能という（インフィクス）述語

ここで項というのは、変数、定数、あるいは関数適用（f(a,b,..))の形をしたものであり、論理式は述語適用（p(a,b,..))の形をしたもの）、あるいはそれらを¬（否定）、∧（連言）、∨（選言）で結び合わせたものである。

代入

論理式pの中の自由変数xにxを含んでいない項aを代入したものをp{a/x}と書くことにする。なおこの場合aの中の自由変数がxに等しくなくかつpの中に現れていないものとし、その場合に限りこの表記法を用いることにする。また複数の変数に同時に代入を行ったものは、p{a1/x1,...,an/xn}と書くことにする。またこの場合はa1...anの自由変数はx1...xnを含まずp中にも現れていないものとする。

3．簡単化の規則

本節では論理式の簡単化において使用する変型規則を述べる。

==に関する規則

f(a1,a2,...,am)==g(b1,b2,...,bn) ⇨ false、
　　fとgが異なるか、mとnが異なるとき　（U1）

f(a1,a2,...,am)==f(b1,b2,...,bm) ⇨
　　a1==b1∧a2==b2∧...∧am==bm　（U2）

x==x ⇨ true　（U3）

x==a ⇨ false、
　　aはxとは異なり、xがa中に現れているとき　（U4）

x==a∧p ⇨ x==a∧p{a/x}　（U5）

∀x:x==a ⇨ false　（U6）

∃x:x==a ⇨ true　（U7）

$\forall x:((\neg y==a)\vee p) \Rightarrow \forall x:(\neg y==a)\vee \exists x:(y==a \wedge p)$、

xとyが同じ変数であるか、xがa中に現れているとき　　（U８）

ここでU１～U７は比較的用意に理解できるが、U８は少々複雑であり、この関係は佐藤により導かれた。

<u>冗長項の消去</u>

$p\{a1/x1,\ldots,an/xn\}\wedge(\forall x1:\ldots\forall xn:p)$

$\Rightarrow \forall x1:\ldots\forall xn:p$　　（R１）

$p\{a1/x1,\ldots,an/xn\}\wedge(\exists x1:\ldots\exists xn:p)$

$\Rightarrow p\{a1/x1,\ldots,an/xn\}$　　（R２）

$\neg p\{a1/x1,\ldots,an/xn\}\wedge(\forall x1:\ldots\forall xn:p) \Rightarrow$ false　　（R３）

$(\exists y1:\ldots\exists yn:\neg p\{y1/x1,\ldots,yn/xn\})\wedge(\exists x1:\ldots\exists xn:p)$

$\Rightarrow$ false　　（R４）

$(\exists y1:\ldots\exists yn:p\{y1/x1,\ldots,yn/xn\})\vee(\exists x1:\ldots\exists xn:p)$

$\Rightarrow \exists x1:\ldots\exists xn:p$　　（R５）

$\neg(\exists y1:\ldots\exists yn:p\{y1/x1,\ldots,yn/xn\})\vee(\exists x1:\ldots\exists x:p)$

$\Rightarrow$ true　　（R６）

以上の過程において現れるtrueやfalseなどに関する自明の規則を次に示す。

$\neg$ true　$\Rightarrow$ false　　（T１）

$\neg$ false　$\Rightarrow$ true　　（T２）

p $\wedge$ false　$\Rightarrow$ false　　（T３）

p $\wedge$ true　$\Rightarrow$ p　　（T４）

p $\vee$ false　$\Rightarrow$ p　　（T５）

p $\vee$ true　$\Rightarrow$ true　　（T６）

Q x:p　$\Rightarrow$ p、　xがp中に現れないとき　　（T７）

4．簡単化アルゴリズム

ここでは3節で述べた規則を適用するための方法を説明する。3節の規則は連言に関するものが多いので、与えられた式を和標準形、すなわち連言を選言にしたものに変型することにする。また限量子は各連言の先頭にくるようにする。このための規則を次に示す。さらに式の変型を行う前に、束縛変数間の依存関係を調べておく。変数yが変数xに依存しているというのは、xとyの限量子が異なるものであり、xの有効範囲中でyが束縛されていることである。

$\neg(p \wedge q) \quad \Rightarrow \quad \neg p \vee \neg q$ （Λ1）

$\neg(p \vee q) \quad \Rightarrow \quad \neg p \wedge \neg q$ （Λ2）

$p \wedge (q \vee r) \quad \Rightarrow \quad (p \wedge q) \vee (p \wedge r)$ （Λ3）

$\neg Q x : p \quad \Rightarrow \quad Q' x : \neg p$ （Λ4）

$(Q x : p) \wedge r \quad \Rightarrow \quad Q x : (p \wedge r)$、

xがr中に現れないとき （Λ5）

$(Q x : p) \vee r \quad \Rightarrow \quad Q x : (p \vee r)$、

xがr中に現れないとき （Λ6）

規則Λ5・Λ6においてxがr中に現れている場合には、Qx：pの中のxを他の変数で置き換えてから適用することにする。なおそのときにxに関する依存関係は新しい変数に受け継いでおく。

このように変型した後で3節の規則を適用することになるが、その規則では限量子が項の直前に付いた形のものがあり、直ちに適用できない。そこで、限量子をその項の直前まで持ってこれるかどうかを判断して、その場合に規則を適用することにする。ここで式変型の際に限量子の有効範囲をできるだけ狭くしておけばよいと考えられるが、そうしない理由はその方法がユニークではないことと、有効範囲を拡げておかないと適用できない規則もあるからである。限量子Qx：を式pの前までもってこれるということをQx：··pと書くことにするとQx：··pは次のときに成り立つ：

Q1．Qx：p

Q2．Q'x：··¬p

Q3．∀x：··(p∧q)

Q4．∀x：··(p∨q) かつ xがq中に現れない

Q5．∃x：··(p∧q) かつ xがq中に現れない

Q6．∃x：··(p∨q)

Q７．Ｑｘ：‥(Ｑｙ：‥ｐ)

Q８．Ｑｘ：‥(Ｑ'ｙ：‥ｐ)

　　かつ　(ｙがｐ中に現れない　または　'ｙがｘに依存しない)

Q９．∀ｘ：‥(∃ｙ：‥ｐ∧ｑ)　かつ　ｙがｐ中に現れない

Q１０．∃ｘ：‥(∀ｙ：‥ｐ∨ｑ)　かつ　ｙがｐ中に現れない

これを用いて規則Ｕ1-8、Ｒ1-6を適用するが、その結果にＴ1-7を適用する。この操作を、それ以上規則が適用できなくなるまで繰り返し、得られた式が簡単化された式であるとする。

５．例

全称限量子∀を含んだ次の定義を考える：

max(ｘ,ｕ):=mem(ｘ,ｕ)∧∀ｙ:(mem(ｙ,ｕ)→ｘ〉=ｙ).

mem(ｘ,ｕ):=(∃ｚ:ｕ==〔ｘ|ｚ〕)∨(∃ｖ:∃ｗ:(ｕ==〔ｗ|ｖ〕∧mem(ｘ,ｖ))).

これから∀を含んでいない論理式、すなわちPrologで実行できる式に変型すると

max(ｘ,ｕ):=mem(ｘ,ｕ)∧ｍ(ｘ,ｕ).

ｍ(ｘ,ｕ)　:=((∀ｙ:∀ｚ:¬ｕ==〔ｙ|ｚ〕)∨(∃ｙ:∃ｚ:ｕ==〔ｙ|ｚ〕∧ｘ〉=ｙ))

　　∧((∀ｖ:∀ｗ:¬ｕ==〔ｗ|ｖ〕)∨(∃ｖ:∃ｗ:ｕ==〔ｗ|ｖ〕∧ｍ(ｘ,ｖ))).

このままでも実行できる形ではあるが、この式は冗長な部分があるのでこれを簡単化することにする。

まずこの式を和標準形にして各連言の先頭に限量子をもってくる。なおこのときに必要があれば名前の付け換えを行うが、この場合は必要がない。また変数の依存関係はこの場合にはない。変型の結果は

(∀ｙ:∀ｚ:∀ｖ:∀ｗ:¬ｕ==〔ｙ|ｚ〕∧¬ｕ==〔ｗ|ｖ〕)　∨

(∀ｙ:∀ｚ:∃ｖ:∃ｗ:¬ｕ==〔ｙ|ｚ〕∧ｕ==〔ｗ|ｖ〕∧ｍ(ｘ,ｖ))　∨

(∃ｙ:∃ｚ:∀ｖ:∀ｗ:ｕ==〔ｙ|ｚ〕∧ｘ〉=ｙ∧¬ｕ==〔ｗ|ｖ〕)　∨

(∃ｙ:∃ｚ:∃ｖ:∃ｗ:ｕ==〔ｙ|ｚ〕∧ｘ〉=ｙ∧ｕ==〔ｗ|ｖ〕∧ｍ(ｘ,ｖ))

となるが、この式をＳと呼ぶことにする。

式Ｓの１行目は規則Ｒ1において〔ｙ/ｗ，ｚ/ｖ〕という代入がされたものとみることができて

∀ｙ:∀ｚ:∀ｖ:∀ｗ:¬ｕ==〔ｗ|ｖ〕

となり、これにＴ7を適用すると

∀ｖ:∀ｗ:¬ｕ==〔ｗ|ｖ〕

になる。式Ｓの２行目はＵ3を適用してｕに〔ｗ|ｖ〕を代入すると

(∀ｙ:∀ｚ:∃ｖ:∃ｗ:¬〔ｗ|ｖ〕==〔ｙ|ｚ〕∧ｕ==〔ｗ|ｖ〕∧ｍ(ｘ,ｖ))

となるが、ここでＱ７の条件から∀ｙ:∀ｚ:を最初の項の前に持ってくることができて、その項は

∀ｙ:∀ｚ:¬〔ｗ|ｖ〕==〔ｙ|ｚ〕

となる。この式は規則Ｕ２を用いると

∀ｙ:∀ｚ:¬(ｗ==ｙ∧ｖ==ｚ)

になり、これはΛ１・Ｑ３・Λ４より

(¬∃ｙ:ｗ==ｙ)∨(¬∃ｚ:ｖ==ｚ)

となる。ここでＵ６より

(¬true)∨(¬true)

となって、結局

false

になる。これは式Ｓの２行目において連言の１つがfalseになることであり、２行目はfalseになる。

式Ｓの３行目も２行目と同じことを行うと結局falseになる。最後の行は、まず規則Ｕ３によって最初の項をそれ以降に代入すると

(∃ｙ:∃ｚ:∃ｖ:∃ｗ:ｕ==〔ｙ|ｚ〕∧ｘ)=ｙ∧〔ｙ|ｚ〕==〔ｗ|ｖ〕∧ｍ(ｘ,ｖ))

となるが、この式の第３項はＵ２によって

ｙ==ｗ∧ｚ==ｖ

となるので、再びＵ３によってこれらを代入すると

(∃ｙ:∃ｚ:∃ｖ:∃ｗ:ｕ==〔ｗ|ｖ〕∧ｘ)=ｗ∧ｙ==ｗ∧ｚ==ｖ∧ｍ(ｘ,ｖ))

となるが、この式でｙは第3項にｚは第4項にしか現れていないのでＱ5よりヨｙ:と

∃ｚ：はそれぞれの前に持ってくることができて

(∃ｖ：∃ｗ：ｕ＝＝〔ｗ｜ｖ〕∧ｘ〉＝ｗ∧(∃ｙ：ｙ＝＝ｗ)∧(∃ｚ：ｚ＝＝ｖ)∧ｍ(ｘ，ｖ))

となる。ここでＵ６より第３・４項はtrueとなるので式Ｓの４行目は

(∃ｖ：∃ｗ：ｕ＝＝〔ｗ｜ｖ〕∧ｘ〉＝ｗ∧ｍ(ｘ，ｖ))

となり、ｍの定義は

ｍ(ｘ，ｕ)：＝(∀ｖ：∀ｗ：¬ｕ＝＝〔ｗ｜ｖ〕) ∨

(∃ｖ：∃ｗ：ｕ＝＝〔ｗ｜ｖ〕∧ｘ〉＝ｗ∧ｍ(ｘ，ｖ))

となる。これはほぼ等価なＰｒｏｌｏｇのプログラム

```
m(X,U):-not(U=[W|V]).
m(X,[W|V]):-X>=W,m(X,V).
```

で書くことができる。

６．おわりに

単なる記号としての論理式にユニファイ可能という意味を加えた式についての簡単化のアルゴリズムを説明したが、実際に応用するにはこれだけでは不十分である。今後の課題は基本的述語の数を増やして、等号や不等号というものについての簡単化を行うことであるが、これは本報告で述べたようには容易には実現できない問題である。

参考文献

佐藤・玉木：第一階コンパイラ――決定性の論理プログラム合成、日本ソフトウェア科学会第３回大会論文集、１９８６。

Chang, C.L.: 'Symbolic Logic and Mechanical Theorem Proving,' Academic Press, New York, 1973.

McCluskey, E.J.: 'Minimization of Boolean Functions,' Bell System Tech. J. Vol 35 (1956), pp. 1417-1444.

Quine, W.V.: 'The Problem of Simplifying Truth Functions,' Am. Math Monthly, vol 59 (1952), pp. 521-531.