1L－7

プログラム検証支援のための プレスブルガー文真偽判定ルーチンの高速化

森岡 澄夫　東野 輝夫　谷口 健一

大阪大学 基礎工学部 情報工学科

1 まえがき

加算を持つ整数の理論（整数の集合Z上の変数，定数，加減算，大小比較演算，$\wedge, \vee, \neg, \forall, \exists$ からなる理論）はプレスブルガー（Presburger）算術と呼ばれている．その上の閉論理式（以下，プレスブルガー文，またはP文と呼ぶ）の真偽は決定可能である [1]．

P文の真偽判定は，プログラムや同期式順序回路の正当性証明（プログラムや回路が想定している任意の入力に対して正しい出力を出すことの証明）や [2]，ソフトウェアのInfeasible Path検出 [3] などに使われている．

従来より我々は，判定するP文が全ての変数が全称記号$\forall$で束縛された冠頭標準形（以下，$\forall$冠頭形と呼ぶ）になるような，プログラムや回路の記述法・検証法について，研究してきた [4,5]．

その方法で真であることを判定する$\forall$冠頭形P文（の母式）は，次のような形をしている場合が多い：

[前提条件1 $\wedge$ ..... $\wedge$ 前提条件p]
$\supset$ [結論1 $\wedge$ ..... $\wedge$ 結論q]

ここで，各前提条件や各結論は，それぞれ幾つかの(不)等式の論理積ないし論理和である．前提条件は比較的簡単な式，結論は複雑な式であることが多い．

また，それらの$\forall$冠頭形P文では，複雑な性質の証明を行う場合など，式長（式中のシンボル数）が，1000～2000程度と，かなり大きくなることがある．

今回，検証作業の能率化を目指し，$\forall$冠頭形P文のうち，上述のような形をしたものを対象に，大きな式であっても高速に真偽判定を行う方法を考え，プログラムを作成した．本稿では，その高速化の方法と評価実験の結果について述べる．

2 Cooperの真偽判定アルゴリズム

本ルーチンは，現在知られているうち最も速い一般のP文に対する真偽判定アルゴリズムであるCooperのアルゴリズム [1] に基づく．ここでは，Cooperのアルゴリズムについて簡単に説明する．

Cooperのアルゴリズムは，内側の束縛変数から順に，等価性を保存しながら消去し（変数消去），最終的に定数だけからなる式を得て真偽を決定する方法である．

具体的には，真であることを示したい$\forall$冠頭形P文 $\forall x_1 ... \forall x_n P(x_1, ..., x_n)$ があるとき，次のように判定を行う：まず，$\exists$冠頭形のP文 $\phi \triangleq \exists x_1 ... \exists x_n \neg P(x_1, ..., x_n)$ を作る（以下，母式$\neg P(x_1, ..., x_n)$ を Q と呼ぶ）．続いて，ϕ中の各変数を消去し，ϕが偽であることを示す．

ϕの変数消去は，次のように行う：例えば，最も内側の変数 x_n を消去するとする．そのとき，x_n の幾つかの有限個の値 $exp_1 \sim exp_k$（それらは定数と x_n 以外の変数との一次結合であり，Q から定まる）に対して，それぞれ，$\exists x_1 ... \exists x_{n-1} Q(x_1, ..., x_{n-1}, exp_i)$ $(1 \leq i \leq k)$ が偽であることを，さらに変数消去を行って調べる（全て偽であればϕは偽．一つでも真になれば，ϕは真）．

本ルーチンでは，各 $exp_1 \sim exp_k$ の Q への代入，および代入後の式の変数消去を，深さ優先で行うことによって，判定に必要なメモリの量を減らしている．代入による式変形の様子は，図1の（例えば左側の）木で表され，変数消去は，その木を深さ優先で辿る形で進められる．プログラム検証では$\forall$冠頭形P文が真であることを示す（ϕが偽であることを示す）が，その場合，変数への全ての代入値について調べるので，木全体を辿ることになる．

3 変数の消去順の工夫による高速化

Cooperのアルゴリズムでは，一般のP文については，内側の変数から消去しなければならない．しかし，$\exists$冠頭形のP文の真偽判定の場合は，どのような順番で変数を消去してもよい．

変数の消去順を変えることで，変数への値の代入の回数（変数消去時に辿る木の枝の数）が，大きく変わることがある（例えば図1参照）．

本ルーチンでは，判定する$\forall$冠頭形P文の母式が，まえがきで述べたような形をしていることを利用し，次の方針で代入の回数を減らし，高速化を図ることにした：「（もとの$\forall$冠頭形P文における）前提条件を偽にしない（なるべく真にする）ような変数への代入値についてだけ，結論部が成立するか調べる．」

このためには，式の構文木の根からの深さが最も浅い変数から，消去するのが良いと思われる．これは，まえがきで述べたように，前提条件は結論部よりも簡単な式であることが多いので，（偽であることを示す$\exists$冠頭形P文ϕにおいても）前提条件中の変数は，結論中の変数よりも，構文木中の浅い位置に出現することが多いからである．

また，構文木の根からの深さが最も浅い変数は，複数存在することがあるが，その場合，なるべく代入値の個数が少ないものを選ぶのが良いと思われる．これは，変数消去で辿る木の幅が，その根に近い位置で広がると，代入の総回数（木の枝の総本数）が増えることが経験的には多いからである（例えば図1の右上と右下を参照）．

以上より，本ルーチンでは，消去する変数を，次の2つのルールにより，動的に決定することにした：

ルール1：（その時点の）式の構文木の根からの深さが，最も浅い位置に出現する変数を選ぶ．

ルール2：ルール1で複数の候補が出た場合，それらのうち，

A Technique for Reducing Computation Time in Decision Procedure for Presburger Sentences used in Program Verification

Sumio MORIOKA, Teruo HIGASHINO and Kenichi TANIGUCHI
Deperment of Information and Computer Sciences, Faculty of Engineering Science, Osaka University
Toyonaka-shi, Osaka 560 Japan

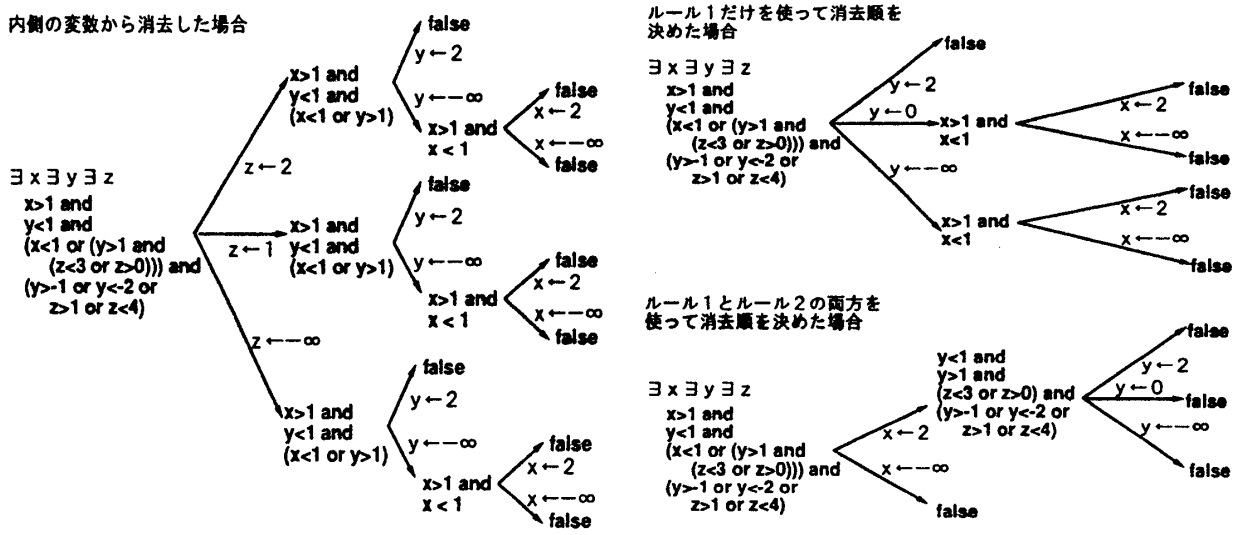

図 1: 変数の消去順の違いによる変数消去の過程の変化

代入値の個数が最も少ない変数を選ぶ.

4 高速化の効果測定実験

クイックソートプログラムの正当性証明 [6] で真であることを示した最も大きい∀冠頭形 P 文 (qsort1 と呼ぶ) と, マックスソート回路の正当性証明 [4] において, 真であることを示した最も大きい∀冠頭形 P 文二つ (msort1, msort2 と呼ぶ) を用いて, 高速化の効果測定実験を行った.

これらの P 文に対し, 式 (文字列) 中に出現した順で変数を消去した場合の判定時間を表 1に示す (SONY NEWS−5000, 100MIPS, 64MB メモリ使用). 式長や変数の数が増加すると, 判定時間も急激に増加し, 実用時間で判定することができなくなってしまう.

これに対し, ルール 1 だけを使って (ルール 2 を使わずに) 消去順を決めた場合の判定時間を表 2に示す. 特別な場合, やや判定時間が増加することもあるが, 経験的には, 方法 1 に従うことで, 判定時間が数十〜数百倍程度高速になる.

さらに, ルール 1 とルール 2 の両方を使って消去順を決めた場合の判定時間を表 3に示す. いずれの P 文も, 実用的と思われる時間で判定できた. 経験的には, 方法 1 だけを使う場合と比べ, 数倍程度, 判定時間が高速になる.

表 1: 真偽判定にかかった時間：式中に出現した順番で消去 (SONY NEWS−5000 使用, 100MIPS)

式	式長	変数の数	CPU 時間 (秒)
qsort1	275	18	25.8
msort1	950	39	146.4
msort2	1613	65	15000 以上

表 2: 真偽判定にかかった時間：ルール 1 だけを使って消去順を決定 (SONY NEWS−5000 使用)

式	CPU 時間 (秒)
qsort1	37.1
msort1	3.8
msort2	22.9

表 3: 真偽判定にかかった時間：ルール 1 と 2 を使って消去順を決定 (SONY NEWS−5000 使用)

式	CPU 時間 (秒)
qsort1	2.4
msort1	1.7
msort2	3.7

5 あとがき

現在, このルーチンは, 我々の研究グループで開発を進めている代数的手法によるプログラム開発システムの ASL 検証支援系 [6] 等に使用されている. ASL 検証支援系では従来, P 文の判定に時間がかかったため, 複雑な性質の証明で式が大きくなったときは, 検証者が, 式中から必要な条件のみ抜き出して式を小さくする等の工夫を行いつつ検証を行っていた. しかし本ルーチンを用いることで, そのような工夫をしなくても, かなり実用的な時間で証明が行えるようになり, 検証作業が非常に能率化された [4,5].

文献 [4,5] のプログラムのように, 基本的に条件判定が大小比較のみ, 変数の係数が 1 であれば, 判定途中で現れる係数が小さく, 本ルーチンにより高速に判定できる.

今後の課題としては, 現在の判定ルーチンでは, 変数の係数の値が大きいと, 変数消去の際に代入値の個数が急激に増えるため, 判定に時間がかかるので, 本稿で述べた高速化法に文献 [3] での高速化法 (代入値の個数を削減する方法) を組み合わせ, より判定を高速化することが考えられる.

参考文献

[1] D. C. Cooper: "Theorem Proving in Arithmetic without Multiplication", Machine Intelligence, No.7, pp.91−99 (1972).

[2] 東野輝夫, 北道淳司, 谷口健一: "整数上の線形制約の処理と応用", コンピュータソフトウェア, Vol.9, No.6, pp.31−39 (1992−11).

[3] 直井邦彰, 高橋直久: "プレスブルガー算術を用いた Infeasible Path 検出の高速化技法", 信学技報, SS95−19, pp.71−78(1995−07).

[4] Junji Kitamichi, Sumio Morioka, Teruo Higashino and Kenichi Taniguchi: "Automatic Correctness Proof of Implementation of Synchronous Sequential Circuits Using Algebraic Approach", T. Kropf and R. Kumar (eds.), Vol.901 of LNCS, pp.165−184, Springer Verlag(1995).

[5] 森岡澄夫, 岡野浩三, 東野輝夫, 谷口健一: "関係データベースを用いた在庫管理プログラムの記述とその詳細化の正しさの証明", 情報処理学会論文誌, 第 36 巻, 第 5 号, pp.1091−1103 (1995−05).

[6] 東野輝夫, 関浩之, 谷口健一: "代数的仕様から関数型プログラムの導出とその実行", 情報処理, 第 29 巻, 第 8 号, pp.881−896 (昭 63−08).