4F−02　全称子で束縛された冠頭標準形プレスブルガー文の真偽判定の分散実行

辻川 竜宏　柴田 直樹　岡野 浩三　谷口 健一

大阪大学 大学院 基礎工学研究科 情報数理専攻

1　まえがき

我々の研究グループでは, ソフトウェア・ハードウェア, 通信プロトコル等に対してプレスブルガー文真偽判定手続きを用いた形式的検証を行っている [1, 2]. 我々はプレスブルガー文真偽判定手続き高速化のための様々な工夫を提案し, 実際に真偽判定が高速化されることも確認している [3]. プレスブルガー文は整数の集合上の変数, 定数, および, $+$, $-$, $=$, $<$, $\wedge, \vee, \neg, \forall, \exists$ からなる閉論理式である [4]. とくに [3] では, 変数の出現位置に注目した判定の高速化の工夫を行ってあり, いくつかの実用例題で有用性が確認されている. しかしそれでも実用時間で解けない問題が数多く存在し, さらなる高速化が望まれる. 近年の MPU の高速化を踏まえても, 従来のように計算機一台ですべて検証する方法よりも, ネットワーク上の数十台の計算機を使い, 並行分散して処理することが次の点で望ましいと考える. 一つのプレスブルガー文を分割しその各部分式を並列に判定できれば分散実行により判定時間を削減することが期待できる. とりわけ分散化による検証時間の短縮は設計時のデバッグにおいても有効である.

そこでプレスブルガー文判定処理ルーチンの分散実行系を作成し, LAN で結ばれた 10 台の計算機を用いた評価実験を行った. 実験により分割せずに一台で判定するよりも, 使用した計算機の台数分だけ速くなることを確認した.

なお, 使用するプレスブルガー文判定ルーチンはさらにブール変数と含意を使えるように拡張している. 一方で, 限定子を全称子で束縛された冠頭標準形に限っている. 一般のプレスブルガー文判定問題は決定可能であるが, 判定する式の式数や式長が増大するとその真偽判定に必要となる計算時間が 3 重指数的に増大する. 全称子で束縛された冠頭標準形のプレスブルガー文の場合は, 計算量は指数程度であるが, 問題によっては現実的な時間内で解くことは困難となり得る.

2　分散実行システムの分散方針

ソフトウェア・ハードウェアなどの正しさの検証では, 与えられる式の形が, $\forall x_1 ... \forall x_n\ (P(x_1, ..., x_n) \rightarrow Q(x_1, ..., x_n))$ のような形をしていることが多い. ここで冠頭の限定子は全て全称子である. 我々はこの式を変形し, 和項単位で式を分割する方針を取った. 分割数はユーザーが指定できる. ここでいう分割数とは, すべての式に対して分配則を適用する回数である. 構文木の走査の方法により複数の分割方式を用意した. 代表的なものに元の式の左側を優先して分ける方式 (左優先分割) と, 右側を優先して分ける方式 (右優先分割) と, 中央で分割する方式 (中央分割) などがある. 例えば, $P_1 \wedge P_2 \wedge P_3 \vee Q_1 \wedge (Q_2 \vee Q_3)$ のような式を分割数 1 で分割する場合は, 左優先なら,

$$P_1 \vee Q_1 \wedge (Q_2 \vee Q_3) \text{ と } P_2 \wedge P_3 \vee Q_1 \wedge (Q_2 \vee Q_3),$$

右優先なら,

$$P_1 \wedge P_2 \wedge P_3 \vee Q_1 \text{と } P_1 \wedge P_2 \wedge P_3 \vee Q_2 \vee Q_3$$

のように分割する. 分割数が 2 以上の場合は得られたそれぞれの式に対して再び分配則をそれぞれ繰り返し適用する.

式により, これらの方式をうまく選ぶと検証をより速く終えることができる.

3　分散実行システムの実現方針

本処理系は式の判定を行う複数のバックエンドサーバーと, 式の分割およびサーバーへの式の送信を RPC を用いて行うユーザー側のクライアントからなる. サーバーは各計算機毎に起動され, サーバーはクライアントに判定結果を送信する. 判定結果を返してきたサーバーには即座に次の未判定の式を送信する. サーバーには従来のプレスブルガー文真偽判定ルーチン [3] を組み込んでいる. 次にユーザーがクライアントプログラムに対して指定できる引数について述べる.

ユーザーは引数としてプレスブルガー文が記述されたファイルを指定する. 現在の仕様では, 一つの入力ファイルに記述できる式の数は一つであるが, ファイルを複数指定することで複数の式を自動的に検証することができる.

式の分割数を上げていくと, 判定時間が数秒以内の式が多数出現し, 式を一つ一つサーバーに送ると通信のオーバーヘッドが大きくなる. 数個まとめてサーバーに送ることでこれをを小さくすることができる. このサーバーに送る式の数もユーザーが引数として指定できる.

その他のクライアントに対する引数は, 式の分割数, 分割方式である.

4　評価実験

評価実験として従来の処理系でも数時間以上要することが分かっている組合せ論理回路 (TTL の 4 ビットコンパレータ 7485, 2 入力 1 出力マルチプレクサ 74157,

---

Distributed Execution of Decision Procedure for Prenex Normal Form Presburger Sentences bounded only by Universal Quantifiers.

Tatsuhiro TSUJIKAWA, Naoki SHIBATA,
Kozo OKANO and Kenichi TANIGUCHI
Deperment of Informatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University
Toyonaka-shi, Osaka 560 Japan

4 ビット 2 進フルアダー 74283) の実現の正しさの証明行った. これらのプレスブルガー文のトークン数はそれぞれ, 100, 106, 162 である. また変数の個数はそれぞれ, 36, 34, 35 である. 分割数, 分割方法を変えてそれぞれ実験を行った. サーバーに一度に送る式の数は 20 個に固定して実験した. 実験環境は, 研究室のイーサネット LAN(100Mbps, 一部 10Mbps) で結ばれた計算機 10 台である. 表 1~3 の値は, 最初にサーバーに式を渡してから, 最終の判定結果が出るまでの時間である. 現実の使用を考慮し, サーバープロセスの優先度は最低に設定している.

表 1: 74157 の検証時間 (秒)

	式の数				
	2^7	2^8	2^9	2^{10}	2^{11}
右優先	437	396	492	490	528
左優先	1,832	2,027	1,046	1,083	1,040

Alpha 500MHz 256MB RAM, Pentium III 500MHz 128MB RAM × 4 台, Pentium II 450MHz 512MB RAM, Pentium II 330MHz 128MB RAM, Pentium II 266MHz 64MB RAM, Pentium Pro 200MHz 128MB RAM, Pentium Pro 200MHz 80MB RAM の計 10 台の計算機を使用した.

表 2: 7485 の検証時間 (秒)

	式の数			
	2^{12}	2^{13}	2^{14}	2^{15}
右優先	6,300	5,865	5,124	7,178
左優先	6,432	6,664	6,822	11,502

74157 で用いた計算機の内, Pentium III 500MHz 128MB RAM 一台を Pentium II 266MHz 128MB RAM に替えた計 10 台を使用した. 表 4 も同じ環境である.

表 3: 74283 の検証時間 (秒)

	式の数			
	2^9	2^{10}	2^{11}	2^{12}
右優先	NA	NA	NA	NA
左優先	3,952	4,011	3,654	4,253

なお, 74283 の右優先分割は, 2 時間を越えても判定が終らなかったので判定を打ち切った.

参考までに表 4 に分割せずに一台の計算機 (PentiumIII 500MHz, 128MB RAM) で検証した場合の判定時間を示す.

5 考察

今回の実験では, 分割して検証した場合, 使用した計算機の台数分だけの効果があることが分かった. 例えば 7485 の検証式を 2^{14} 個に分割した場合, 10 倍強の速度向上がみられた. 分散実行の有用性が確かめられる結果となった.

右優先を使うか, 左優先を使うかにより結果が大きく違った. 例えば 74157 の 2^8 個分割の場合, 7 倍近い差が出ている. これは, 全体的には分割の方式の選択のほうが分割数の選択よりも重要であることを示している.

右優先, 左優先のどちらが良いかは対象となる式に依存する. 例えば, 74157 の式では右優先が速かったが, 74283 では逆に左優先が速かった. もちろん, 依存するのは式の形そのものであり, 74157 でも別の形で式を記述すれば別の結果がでたものと思われる. 式の形は変数の出現順に関連することであり, このことが検証時間に与える影響は [3] で確認されている.

表 4: 分割せずに検証した場合の検証時間 (秒)

	所要時間
7485	55,959
74157	8,309
74283	32,623

分割数を単純に上げることは個々の式の検証時間短縮につながるが, 式の数が指数的に増加し, また分割することによる速度向上も頭打ちになってくるので, 全体的な性能は落ちることが分かった.

なお, 当初は式の形に着目し, 含意の後部の積項を分割する方針を考えていたが, 予備実験の結果, あまり効果がないことが分かった. 理由としては, 分割できる数が限られることが挙げられる. この予備実験の結果は, 今回のように単純に分割していく方が分散実行としては効率が良いことを示唆している.

式のサイズと計算機の台数にも依存するが, 実験の例題では 2^{10} 程度の数に式を分割することで十分な粒度が得られ計算機の負荷分散が適切に行われることが分かった.

各サーバーレベルでは従来研究してきた高速化の機能を組み込んでいるため, このレベルでの高速化はこれ以上あまり期待できないと思われる.

6 あとがき

プレスブルガー文判定処理ルーチンの分散実行系の作成とその評価実験を行った. 実験結果から, 式の分割と分散実行により検証時間が削減できることが分かった. しかし, 現在の仕様ではユーザーが分割数や分割の方式を指定しなければならず, 実際の検証環境を考えると改良の余地はある. よって今後の課題としては, 自動的に最適な分割数・分割の方式の導出, および他の検証例についての実験や分割方法の改良などが挙げられる.

参考文献

[1] 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, Vol.36, No.5, pp.1091-1103, 1995.

[2] Junji Kitamichi, Sumio Morioka, Teruo Higashino and Kenichi Taniguchi, "Automatic Correctness Proof of Implementation of Synchronous Sequential Circuits Using Algebraic Approach", Proceedings of the Second International Conference on Theorem Provers in Circuit Design (TPCD'94), Lecture Notes in Computer Science, Springer Verlag, Vol. 901, pp.165-184, Sept. 1994.

[3] 森岡 澄夫, 柴田 直樹, 東野 輝夫, 谷口 健一, "すべての変数が存在記号で束縛された冠頭標準形プレスブルガー文の真偽判定の高速化手法", 情報処理学会論文誌, Vol. 38, No. 12, pp.2419-2426, 1997.

[4] 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.