強連結成分ベースのグラフ分割による分散並列 LTL モデル検査の高速化

三輪 真弘† 上田 和紀‡
†早稲田大学大学院基幹理工学研究科 ‡早稲田大学理工学術院

1 研究の背景と目的

モデル検査は，状態空間の網羅的探索により，システムの検証を行う技術である．LTL モデル検査は，システムのモデルと LTL から成るオートマトングラフの到達性問題や受理サイクル探索問題に帰着できる．オートマトングラフは巨大なものになりがちで，メモリ不足の問題が起こる．そこで，メモリ資源の豊富な PC クラスタでの並列 LTL モデル検査が考えられる．しかしながら，分散環境ではグラフの分割が必要であり，分散並列 LTL モデル検査器として有名な DiVinE[1] では並列アルゴリズムの多くでハッシュ関数を用いたグラフ分割法を採用している．この方法は，各計算ノードのメモリ消費量をほぼ等しくできるものの，ノード間遷移が多数発生し，通信オーバーヘッドの増大を引き起こす．そこで本研究では，性質オートマトンの強連結成分（以下 SCC）をベースにハッシュ関数を用い，PE（Processing Element）間の遷移を減らす工夫をしたグラフ分割法（Hash On SCC）を提案・実装し，その評価を行った．

2 DiVinE

DiVinE（Distributed Verification Environment）とは，分散メモリ環境で実行できる分散検証環境であり，LTL モデル検査器として利用できる．

2.1 グラフ分割法

ここでは，DiVinE で用いられている 2 つのグラフ分割法について述べる．

2.1.1 Hash-Based 分割法

Hased-Based 分割法は，ハッシュ関数を用いて，各頂点をランダムに分割する方法である．この方法を用いると，PE（Processing Element）毎の処理する状態数をほぼ等しく出来る．しかしながら，PE 間の遷移（cross transition）が多くなり，通信オーバーヘッドの増大に繋がる．

2.1.2 Property-Driven 分割法

検証する性質（LTL）を表すオートマトン（性質オートマトン）を SCC に分解し，その SCC1 つ辺りに 1PE を割り当てるという方法である．

このグラフ分割法を用いた場合，システムと性質の同期積オートマトンに存在するサイクルが分割されることはない．なぜなら，探索グラフ（オートマトン積）はシステムと性質のオートマトンからなるので，オートマトン積グラフにおけるサイクルが存在するとき，システムおよび性質オートマトンそれぞれに対応するサイクルが存在している．したがって，性質オートマトンの SCC 分解したものに PE1 つを割り当てる方法では，同期積オートマトン積のサイクルが PE 間で分断されることはない．

このグラフ分割法の呼び出しは，実行の初期に性質オートマトンの SCC 分解の計算を行っておけば，その後は定数時間で呼び出せる．性質オートマトンはシステムのモデルのオートマトンに比べ非常に小さいので，性質オートマトンの SCC への分解に掛かる時間は僅かである．

2.2 並列 LTL モデル検査アルゴリズム

DiVinE には，複数のアルゴリズムが実装されているが，ここでは本研究で用いた OWCTY(One-Way-Catch-Them-Young) アルゴリズムについてのみ述べる．

2.2.1 One-Way-Catch-Them-Young (OWCTY)

OWCTY アルゴリズムでは，次の 2 つの削除ルールをグラフに繰り返し適用し，受理サイクルを検知する．

- 受理頂点から到達不可能な頂点を取り除く
- predecessor を持たない頂点を取り除く

取り除く頂点が存在しなくなったとき，頂点集合が空であればサイクルが存在しないことになり，空でなければそれが反例となる．

並列化は，頂点集合を分割し，適宜通信を行うことでなされている．

3 Hash On SCC 分割法

この節では，今回提案・実装したグラフ分割法 Hash On SCC（以下 HoS）の説明を行う．

HoS 分割法は，Hash-Based 分割法の問題点である PE 間の遷移の増大による通信オーバーヘッドの削減を行い，分散環境におけるより高速な並列 LTL モデル検査の実行を目的としている．

性質オートマトンを利用したグラフ分割法では，性質オートマトンの各 SCC には 1 台のみの割り当てであるが，HoS 分割法では，各 SCC に (PE 数/SCC 数)PE の割り当てを行う．

Effective graph partitioning for distributed LTL model checking
†Masahiro MIWA ‡Kazunori UEDA
†Graduate School of Fundamental Science and Engineering, Waseda University
‡Faculty of Science and Engineering, Waseda University

例えば，(N) PE で性質オートマトンの SCC 数が 2 個の場合に HoS 分割法を用いることを考える．理想的な状況を単純化して考えると，HoS グラフ分割法を用いた場合の通信オーバーヘッドは，(N/2) PE で Hash-Based 分割法を用いた場合の通信オーバーヘッドに等しくなる．

HoS 分割法では，Property-Driven 分割法を行った上で，Hash-Based 分割法を行う．それぞれの呼び出しコストが定数時間なので，HoS 分割法の呼び出しコストは定数時間である．

4 評価実験

分散検証環境 DiVinE（version 0.7.1）を使用し，HoS 分割法の性能評価を行った．

OWCTY アルゴリズムには 2 種類の実装があるが，今回は OWCTY-reversed を用いた．

LTL モデル検査のベンチマーク問題は BEEM のものを使用した．実験に使用した問題のうち，bakery.5.prop2 のみが invalid（受理サイクルあり）で，それ以外の問題は valid（受理サイクルなし）である．また，性質オートマトンの SCC 数は，elevator2.3.prop4 が 6 個で，それ以外は 2 個である．

実験を行ったマシンは，全 30 ノードの PC クラスタであり，各計算ノードは，Core2Duo 2.13GHz，4GB メモリが搭載され，ネットワークは GigE である．

5 回実行時間を測定し，実行時間の短いもの 3 つの平均を取った．なお，実行は 1200 秒で打ち切っている．

4.1 実験結果

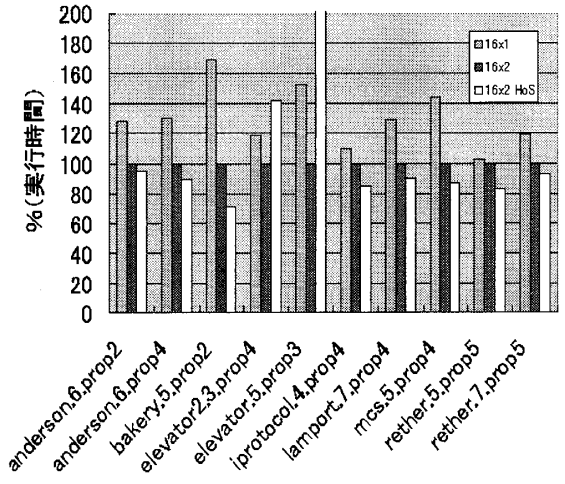

図 1: HoS 分割法の性能評価

実験の結果を図 1 に示した．縦軸の値は，32PE（16x2）で Hash-Based 分割を用いたときの実行時間を 100 として正規化した値である．全問題を通して，16PE（16x1）時より 32PE の方が実行時間は短くなっており，台数効果があることが分かる．HoS 関数を用いて 32PE で実行したとき（16x2 HoS）では，Hash-Based 分割法の場合より実行時間が短くなっていることが分かる．しかし，elevator2.3.prop4 と elevator.5.prop3 では，大幅に遅くなるという結果になっている．この原因は共に SCC サイズに見合った PE 数の割り当てがうまくできなかったためである．適度な PE 数の割り当てができなかったのは，各 SCC のサイズに大きな差があり，等しい PE 数の割り当てではうまく負荷分散ができなかったためである．

4.2 最適化

HoS 分割法において，各 SCC に割り当てる PE 数を等しくした場合が，最適な割り当てでない場合がある．実験で用いた問題のうち，elevator.5.prop3，lamport.7.prop4，rether.7.prop5 において，実験的に調べ，最速値を出す PE 数の割り当てを行ったときの結果を表 1 に示す．

SCC へ割り当てる PE 数を等しくする場合（16x2 HoS）

表 1: HoS 分割法の最適化

	16x2	16x2 HoS	16x2 HoS Opt
elevator.5.prop4	100 (%)	473.2	98.8
lamport.7.prop4	100	89.9	82.2
rether.7.prop5	100	92.8	88.7

より，さらに実行時間の短縮を行うことができたことが確認できる（16x2 HoS Opt）．

適切な PE 数の割り当て方法としては，OWCTY アルゴリズムは，最初にグラフの展開を行うので（到達性），その際に SCC サイズの差異をプロファイリングし，適切な PE 数を知るという方法などが考えられる．

また，HoS 分割法の応用として，Property-Driven 分割をベースにすると，SCC に受理頂点が一切含まれていないことを事前に知ることができ，そのような SCC を担当する PE は，コストの高い受理サイクル探索をする代わりに，到達性のみを行えばよい．今回実験に用いた 10 問中 9 問が受理頂点を含まない SCC を含んでおり，さらなる最適化の可能性がある．

参考文献

[1] J. Barnat, L. Brim, and I. Cerna.: Cluster-Based LTL Model Checking of Large Systems. In Formal Methods for Components and Objects, pp.259-279, 2005.

[2] J. Barnat, L. Brim and I. Cerna.：Property Driven Distribution of Nested DFS. in Proc. of 3rd Int'l Workshop on Verification and Computational Logic (VCL '02), 2002.

[3] 渋谷 健介，上田 和紀：分散検証環境 DiVinE を用いた分散 LTL モデル検査アルゴリズムの性能評価. 第 70 回情報処理学会全国大会, 2008.