LMNtal処理系SLIMのモデル検査機能の並列化

小林 史佳† 後町 将人‡ 上田 和紀§
†早稲田大学大学院理工学研究科 ‡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

SLIM[1]は階層グラフ書換え言語LMNtalで記述されたプログラムを実行するための処理系であり，プログラムの検証や挙動の把握を目的としてモデル検査機能が実装されている．しかし，LMNtalのグラフ書換え処理の複雑さから，他のモデル検査ツールと比べて状態展開や等価性判定処理の時間がかかりやすく，モデル検査の抱える状態爆発問題と組み合わさって実行速度がボトルネックとなっている．

そこで，我々は共有メモリ環境下でのSLIMの持つモデル検査機能の高速化を目的として，並列モデル検査アルゴリズムであるOWCTYを用いて並列化し，その性能について評価を行った．

2 SLIMの検証機能

SLIMはLMNtalのグラフ構造を「状態」，グラフの書換えを「遷移」とみなして網羅的な状態空間探索を行うことで，プログラムの検証が可能である．しかし，状態展開や状態保存にグラフ書換えの処理を利用するため，実行時間が長くなりやすく，この改善が求められている．

2.1 LTLモデル検査機能

LTLモデル検査は，システムが線形時相論理LTLで記述された性質を満たすか否かを調べる検証手法であり，受理状態を含んだ閉路(**受理サイクル**)の探索問題に帰着して解くことが出来る．一般的に，受理サイクル探索問題は閉路を1つ発見すれば終了するが，SLIMでは閉路発見後も探索を継続する機能(ltl_all mode)があり，本論文の実験はこの機能を用いている．SLIMは逐次で最適なアルゴリズムであるNested Depth First Search (Nested DFS)を使用して探索を行っている．

2.2 非決定実行機能

非決定実行機能はプログラムの全振舞いを取得する機能であり，プログラムがユーザの意図したとおりの状態を取り得るか調べることが可能である．SLIMはDepth First Search (DFS)を用いて全状態の探索を行っており，我々は文献[2]において，Stack-Slicingアルゴリズムを使用して状態展開を並列化している．

Parallel LTL Model Checking in LMNtal Model Checker SLIM
†Fumiyoshi KOBAYASHI ‡Masato GOCHO * Kazunori UEDA
†Graduate School of Science and Engineering, Waseda University
‡Dept. of Computer Science and Engineering, Waseda University
§Faculty of Science and Engineering, Waseda University

```
proc OWCTY(V,E,s,A)
  S := Reachability(A)
  old := {}
  while S =/= old do
    old := S
    S   := Reachability(S ∩ A)
    S   := Elimination(S)
  od
  if S=={} then report no cycle
           else report cycle
end
```

図1: OWCTYアルゴリズム

3 並列モデル検査機能の設計と実装

SLIMで使用しているモデル検査アルゴリズムNested DFSは，P-完全であり並列化には不向きである．そこで，今回は並列化が容易なBreadth First Search (BFS)ベースのアルゴリズムであるOWCTY[3]を探索アルゴリズムに選んだ．また，それに伴って，ワークプール方式を用いた状態展開の並列処理を新たに実装した．実装は，Google Code上に公開されているSLIM*のRevision.322をベースとし，並列処理部分の実装はPOSIXスレッドライブラリpthreadを用いている．

3.1 OWCTYアルゴリズム

OWCTYは，到達可能な全ての受理状態を求めて，受理サイクルに含まれない状態を取り除く，という処理を反復することで受理サイクルを探索する手法である．このアルゴリズムの疑似コードを図1に示す．OWCTYは状態空間の特徴によって反復の回数が変わり，受理サイクルが無い場合は反復の回数が小さくなる傾向がある．

3.2 ワークプール方式を用いた並列化

状態展開処理の並列化は，以下のように行った．まず，各状態の状態展開処理をタスクとして捉え，そのタスクを共有キューにプールさせる．各スレッドはそのタスクをそれぞれ取り出して状態展開処理を行い，展開されて得られた状態をキューに入れる．この手法は動的なタスク分割によって均一な負荷分散が実現できるという利点があるものの，共有キューへのアクセスで各プロセスが競合を起こしやすいという欠点がある．

*URL : http://code.google.com/p/slim-runtime/

表 1: 実験環境

CPU	AMD Opteron 8222
CPU 周波数	3.0GHz
コア数 (全コア数)	Dual-core × 8 (16)
メモリ容量	256 Gbyte
OS	Debian Linux 5.0 (Lenny)

表 2: 実行時間 (sec)

モデル名	Nested DFS	OWCTY(1)	OWCTY(4)
swp_ws6	20.21	20.65	11.48
swp_ws7	53.27	55.11	28.75
swp_ws8	140.08	145.81	70.61
swp_ws9	364.44	378.02	182.61
swp_ws10	894.73	937.88	443.68
mutex13	89.81	80.47	38.72
mutex14	212.23	190.45	88.12
mutex15	500.56	449.1	202.04
mutex16	1156.61	1049.14	461.85
mutex17	2726.37	2467.67	1118.88

4 性能実験と考察

SLIM 上に実装した OWCTY アルゴリズムの性能を評価するために，表 1 の環境で実験を行った．

実験で扱ったモデルは SLIM のテストベンチマークセットから，OWCTY で反復の起きた swp_ws (プロトコル) と反復の起きなかった mutex (排他制御) を選び，それらに対してパラメータを変化させ，並列実行時の性能やスケーラビリティを調査した．

表 2 は各モデルの実行時間 (sec) を表わしている．Nested DFS, OWCTY の逐次での実行時間と OWCTY の並列実行で安定して性能が良かった 4 スレッドでの実行時間を載せている．モデル swp_ws は反復が 1 回起きたモデルであり，逐次での実行は Nested DFS の方が高速に動作しているが，4 スレッドで並列実行した場合は OWCTY の方が高速に探索できている．モデル mutex に対しては，OWCTY は反復が無く探索を終えることが出来た．この場合，逐次でも OWCTY の方が高速に動作しており，4 スレッドでの実行では，2.27 倍の速度向上が得られている．

図 2，3 は，スレッド数を変化させたときの台数効果を表している．各モデルは 4 スレッドで 2 倍強の速度向上が得られており，4 スレッドでの並列実行では効率のよい探索が行えているといえる．しかし，それより多くのスレッド数で実行した場合は，4 スレッドでの実行時よりも速度が低下してしまっている．これは，各プロセスが排他制御や共有キューにアクセスが集中したことによって競合が起きていることが原因と考えられる．今回の実装では，各状態の内部変数や共有キューに多くの排他制御を用いており，これらを極力使わないように最適化することによって，スケーラビリティは改善できると考えられる．

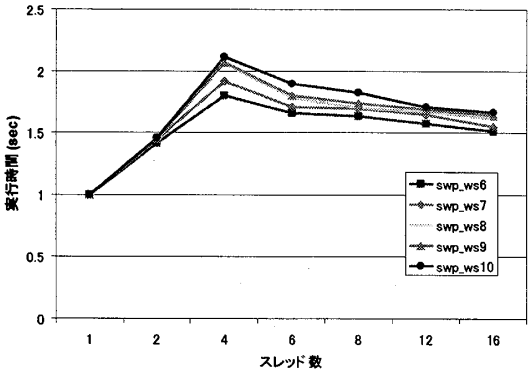

図 2: swp_ws を扱った場合の実行速度比

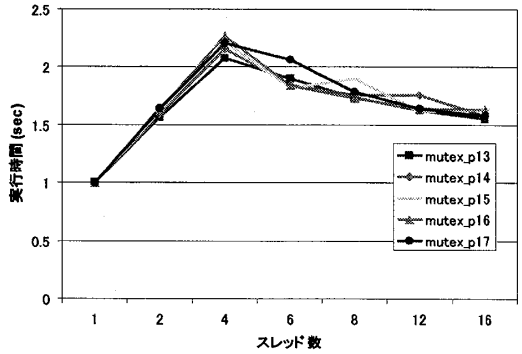

図 3: mutex を扱った場合の実行速度比

5 まとめと今後の課題

本研究では，SLIM の持つ LTL モデル検査機能を OWCTY アルゴリズムを用いて並列化し，4 スレッドでの実行で最大 2.27 倍の高速化が得られており，小規模での実行環境では効率のよく動作することが確認できた．しかし，より多くのスレッド数での実行は台数効果が出ているとはいえず，スケーラビリティの面でまだ改善の余地があると言える．この原因を詳細に調べ，大規模での実行時に対応するように最適化を施すことは今後の課題である．

謝辞 研究の一部は，科学研究費補助金（特定 18049015）の補助を得て行った．本研究では，独立行政法人産業技術総合研究所連携検証施設さつきの検証クラスタを利用した．

参考文献

[1] 堀泰祐, 佐々木隆之, 綾野貴之, 岡部亮, 上田和紀: LMNtal に基づくモデル検査環境, 第 5 回システム検証の科学技術シンポジウム, No. 59, pp. 21–32, 2008.

[2] 後町将人, 上田和紀: LMNtal モデル検査器における状態展開処理の並列化, 第 6 回ディペンダブルシステムシンポジウム, No. 64, pp. 169–178, 2009.

[3] Barnat, J., Brim, L. and Černá, I.: Cluster-Based LTL Model Checking of Large Systems, in Proc. FMCO 2005, LNCS, Vol. 4111, pp. 259–279, 2006.