2V-07

マルチコア CPU に対応した 汎用前向き推論エンジン FreeEnCal の開発

多賀　靖晃†　　後藤　祐一†

埼玉大学　大学院理工学研究科†

1. はじめに

前向き推論エンジンは既知の知識から新しい知識を発見予測する情報システムの実現に必要不可欠なものである．前向き推論エンジンの１つとして汎用前向き推論エンジン FreeEnCal[1]が提案，開発されている．

FreeEnCal は前向き推論を用いた自動定理発見環境の構築[2]などの応用研究に用いられており，これらの応用研究において許容できる時間内に応用に十分な量の推論結果を得る必要があるため，前向き推論エンジンの実行時間を短縮する事は重要な課題であり，処理の高速化が試みられてきた[3][4]．並列処理は処理の高速化の手法として期待されている[4]．

2. 汎用前向き推論エンジン FreeEnCal

汎用前向き推論エンジン FreeEnCal は，入力として与えられた論理式集合に同じく入力として与えられた推論規則を繰り返し適用して，任意の論理体系の論理定理の集合，あるいは，ある論理体系に基づく経験定理（ある分野において真とみなされている公理や定理）の集合を与えられた制限条件の範囲で導出するプログラムである．制限条件として様相演算子や論理結合子の入れ子の度合いを用いて，導出される論理式の数を有限集合に抑えている．論理定理集合を導出する場合は，入力として，ある論理体系の公理，推論規則，様相演算子や論理結合子の入れ子の度合いの制限，導出した論理式を簡単化するための除去規則を受取り，論理定理を導出する．経験定理の集合を導出する場合は，入力として，すでに導出された論理定理の集合，ある分野における経験定理の集合，推論規則，様相演算子や論理結合子の入れ子の度合いの制限，除去規則を受取り，経験定理を導出する．導出される論理式の数に応じて，FreeEnCal の実行時間は多項式的に長くなる[3]．

Development of a Forward Reasoning Engine FreeEnCal for Multi-core Processors

† Yasuaki Taga, Yuichi Goto, Saitama University, Graduate School of Science and Engineering

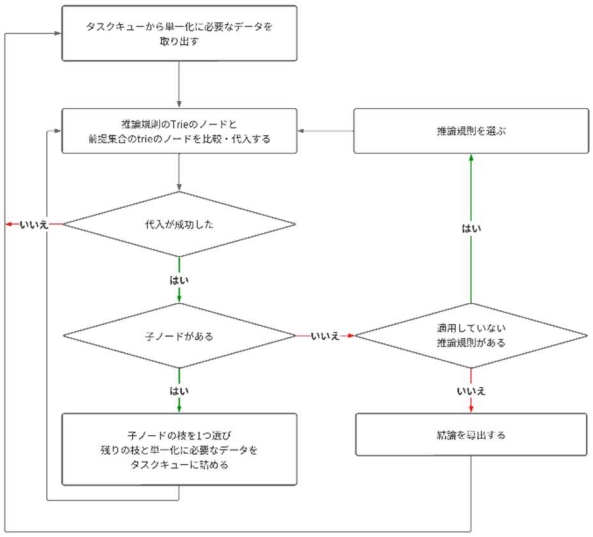

図 1：スレッドが行う処理

FreeEnCal は入力処理，導出処理，重複検査処理，追加処理，出力処理の５つの処理から成り立っている．このうち導出処理にかかる時間が総処理時間の多くを占めているため，この導出処理を並列化することで全体の処理時間を大きく短縮することが出来る．

導出処理では新たな式を導出する処理を行う．推論規則の左辺の１つと，前提として与えられた論理式およびすでに導出済みの論理式で構成される前提集合の論理式で単一化を繰り返し行い，新たな論理式を導出する．単一化とは，2 つの論理式を比較し，論理式が等しくなるように変数を置換する処理である．

3. 並列処理版 FreeEnCal

逐次処理版 FreeEnCal では格納している論理式の集合を Trie 木の形で保存している[5]．導出処理では Trie 木を深さ優先探索でたどっていき，葉までたどり着くと単一化が成功したと判定している．単一化が成功あるいは失敗した場合は直近の枝分かれしている節までの戻り，そこから単一化を行っている．

並列処理版 FreeEnCal では Trie 木を深さ優先探索でたどっている際に，枝分かれが発生したならば，現在のスレッドでたどる枝を１つ選択し，残りの枝を他のスレッドにタスクとして渡

すことで並列化を行っている．

図1に導出処理で1つのスレッドが行う処理を示す．初めに，スレッドはタスクキューから前提集合と推論規則の Trie の枝とノード，変数に代入した記録，使用した論理式の ID 等の導出処理に必要なデータを取り出す．スレッドは取り出した 2 つの Trie の枝を比較し，代入が可能であれば代入する．代入が失敗した場合，新たにタスクキューから単一化に必要なデータを取り出し，再び単一化を進める．代入が成功し，取り出した前提集合の Trie ノードに子ノードが存在した場合，1 つをそのスレッドで処理し，残りの枝をここまで変数に代入してきた記録などと共にタスクキューに詰める．子ノードが存在しない場合，推論規則と前提集合の論理式との単一化が成功したことを示しているため，次の推論規則を選択する．すべての推論規則に単一化を適用し終えた場合，新たな論理式を導出し，再びタスクキューから単一化に必要なデータを取り出す．

並列処理版 FreeEnCal は並列処理に強いプログラミング言語である Rust を用いて実装した．Rust では所有権という機構によってデータ競合が起きない形でしか複数のスレッドがメモリを共有できないようにしてくれる[5] ため信頼できる並列化を行うことができる．

4. 実験

PC （Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz，48GB）上で,連言と帰結関係の入れ子の度合いを 2 とした強相関論理 Tc2 と帰結関係の入れ子の度合いを 4 とした相関論理 Te4 実行し，実行時間（秒）を計測した．

実験結果を表 1 に示す．列見出しの「逐次」は逐次処理版 FreeEnCal，数字は並列処理版 FreeEnCal において使用したスレッド数を表している，並列処理版で 1 スレッドのみ使用した場合は逐次処理よりも実行時間がかかってしまった．しかし，Te4 の場合 2 スレッド使用時には約 0.63 倍，4 スレッド使用時には約 0.39 倍の実行時間，Tc2 の場合，2 スレッド使用時には約 0.60 倍，4 スレッド使用時には 0.34 倍の実行時間となった．

1 スレッド使用時に逐次処理よりも遅くなってしまった理由として，Trie の枝等を複製してタスクとしてまとめる操作や，タスクキューの操作がオーバヘッドとして発生してしまった事が考えられる．

また，スレッドを増やした際に線形に処理時間が短縮されなかった理由としては，タスクキューへの操作や導出した論理式を Trie 木に格納する際に発生する共有ロックがネックとなってしまったと考えられる．

表 1: 実行時間(s)

	逐次	1	2	3	4
Te4	970.4	1120.6	613.1	441.7	385.0
Tc2	18.7	21.0	11.2	8.0	6.4

5. おわりに

本研究では FreeEnCal の導出処理を並列処理によってマルチコア CPU に対応させることで，実行時間の短縮に成功した．

今後の課題として，オーバヘッドの短縮やネックとなってしまっている処理の改善を行い，スレッド数に対して線形の高速化を達成する事が挙げられる．

参考文献

[1] J. Chang, S. Nara, and Y. Goto: FreeEnCal: A Forward Reasoning Engine with General-Purpose, LNAI (Subseries of LNCS), Vol. 4693, pp. 444-452, Springer-Verlag, September 2007.

[2] J. Cheng: Entailment Calculus as the Logical Basis of Automated Theorem Finding in Scientific Discovery, AAAI Technical Report SS-95- 03, pp. 105–110, AAAI Press, March 1995.

[3] Y. Goto, S. Nara, and J. Cheng: Efficient Anticipatory Reasoning for Anticipatory Systems with Requirements of High Reliability and High Security, IJCAS, Vol. 14, pp. 156-171, CHAOS, December 2004.

[4] T. Koh, Y. Goto, and J. Cheng: A Fast Algorithm for Derivation Process in Forward Reasoning Engines, International Journal of Computational Science, Vol. 4, No. 3, pp. 219-231, Global Information Publisher, June 2010.

[5] J. Blandy, J. Orendorff: Programming Rust, O'Reilly Media, Inc. , 2018(ジム・ブランディ，ジェイソン・オレンドルフ著，中田 秀基 訳: プログラミング Rust，オライリー・ジャパン，2018 年. ）