2M-4

プログラム変換器によるメモ化の適用範囲拡張

康　娜丹†　　小宮　常康‡

電気通信大学　　大学院情報システム学研究科†,‡

概要

メモ化は，重複した計算を省略することでプログラムを高速する手法である．この手法では，関数呼出しの引数と返値のペアを記憶しておき，同一の関数呼出しを再度行う際に，記憶した返値を再利用する．しかし，メモ化可能な関数は参照透過性を備えるものに限られる．本稿では参照透過性を備えない関数を，プログラム変換器によって，大域変数への参照と代入を行わない関数に変換する手法について述べる．具体的には，大域変数を局所変数として表し，その値を伝えるためのパラメータを関数へ追加する．また，大域変数へ代入された値は関数の返値の一部として返すようにする．このような変換によって，メモ化の適用範囲が広がり，自動メモ化が行いやすくなる．

1　背景

メモ化は過去の計算結果を再利用することによって，プログラム実行を高速化する手法である．メモ化された関数は計算結果を引数と一緒にテーブルに記憶しておき，その後に同じ引数で関数を呼び出す際，テーブルを検索し，記憶した結果を返す．この最適化手法によって，関数の計算時間が短縮できる．プログラマはメモ化を利用するために，ソースコードを書き直し，メモ化するためのコードを入れなければならない．このような作業はプログラマにとって煩わしい．また，複雑な関数を書き直す時に，新しいバグを導入してしまう可能性がある．このような問題を解決するために，プログラム中にメモ化のためのコードを自動的に挿入する自動メモ化技術は有益である．

しかし，自動メモ化を利用する時，メモ化できる関数とメモ化できない関数を区別する必要がある．メモ化が適用可能な関数は参照透過性を備えないといけない．即ち，メモ化の対象となる関数は同じ引数を受け取ると，必ず同じ計算結果を返す必要がある．大域変数へアクセスする場合，引数に加えその大域変数の値も関数の計算結果に影響するので，関数が同じ引数を受け取っても異なる結果を返す可能性がある．このような関数はメモ化を適用できない．従って，参照透過性を備えない関数が多いプログラムに対して自動メモ化を行う場合，最適化効果が制限される．

Extending the applicability of memoization by program transformation

† Nadan KANG, ‡ Tsuneyasu KOMIYA

†, ‡ Graduate School of Information Systems, The University of Electro-Communications.

2　提案手法

本研究では，大域変数への参照や代入がある関数に対してプログラム変換を行う．具体的には，関数が大域変数への参照を含む場合，その大域変数に相当する変数を局所変数として表すために，関数のパラメータリストにその変数を加えるように変換する．変換された関数の呼び出し元では，大域変数を引数の一部として関数に渡すように変換する．また，関数が大域変数へ代入する場合，大域変数の代わりに，対応する局所変数に値を代入する．その関数がリターンする際は，代入された値及び本来の返値からなるペアあるいはリストを返すようにする．関数がリターンした後に，返されたペアやリストから代入された値を取り出し，その値で大域変数を更新する．

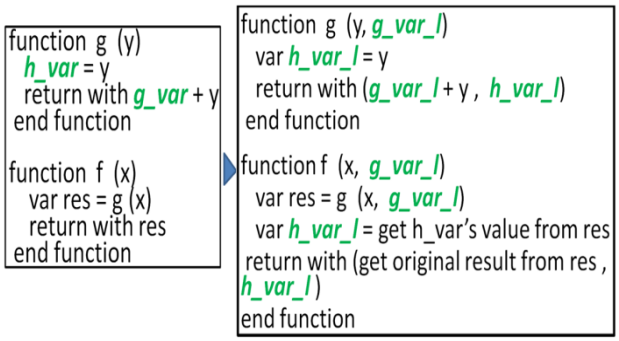

図１　プログラム変換例

一つの例を挙げる．図１の左の擬似コードは変換前のプログラムである．関数 g は大域変数 h_var へ局所変数 y の値を代入し，それから，大域変数 g_var と y の和を返す．図の右側は変換後のプログラムである．大域変数 g_var の値を関数 g に渡すために，関数 g のパラメータリストに局所変数 g_var_l を追加する．また，大域変数 h_var の代わりに，局所変数 h_var_l へ y の値を代入し，それから，h_var_l と本来の返値 g_val_l+y で作られたペアを返す．関数 g の呼び出しはいくつかの文に変換される．まず，返さ

れたペアから g_var_1+y の結果と h_var_1 の値を取り出す．それから，大域変数 h_var を h_var_1 の値で更新する．このような変換で，その後に呼び出された関数は大域変数 h_var へ代入された値を得られる．

以上の変換は，大域変数へ直接アクセスしない関数に対しても影響するので注意する必要がある．例えば，図1の関数 f は関数 g を呼出す．関数 f は直接大域変数へアクセスしない．しかし，関数 g が大域変数 g_var と h_var にアクセスするので，関数 f は間接的に g_var と h_var へアクセスすることになる．間接的にアクセスされた大域変数のために，関数 f に対して，以下の変換を行う．関数 f のパラメータリストに局所変数 g_var_1 を追加し，関数 g の返値から h_var に代入される値を取り出す．その値を h_var_1 に代入し，それから，h_var_1 と本来の返値 res によって作られたペアを返す．このような変換で，大域変数 g_var と h_var の値は各関数の間に伝わる．

変換後の各関数は大域変数へ直接アクセスしない．しかし，局所変数の導入によって，大域変数への参照と代入は変換前の関数と同じように行われる．メモ化する際，大域変数の値は引数として関数に渡されるので，テーブルに記憶される．また，大域変数の更新値は返されるので，テーブルに記憶する必要もある．提案した変換方法がプログラムのセマンティクスを守っているので，変換後の関数は安全に自動メモ化を利用できる．

3 実装

本研究では，プログラム変換を行う変換器は Scheme 言語で実装した．変換器は入力として受け取った各関数をメモ化できるように変換する．実装はプログラム解析とプログラム変換の部分を含む．プログラム解析によって，各関数を解析し，関数の中で参照される大域変数集合と代入される大域変数の集合等の情報を集める．解析は，不動点が求めるまで繰り返して行う．こうして求めた大域変数情報を用いて，前節で述べた変換を行う．

4 考察

大域変数へアクセスする関数に対して，変換の実験を行うために，fib と tak 等の関数に大域変数を導入した．

変換後の tak 関数にメモ化を適用した結果を表 1 に示す．この実験では，一個の大域変数を含む tak 関数に対して，四つの場合に分け，メモ化をしない版とメモ化をした版の実行時間を測定した．参照あるいは代入のどちらかのみ場合のメモ化効果は明らかである．これはテーブルに記憶されたキーのパターンが少ないからである．関数が大域変数への参照と代入を両方含む場合,呼出すたびに大域変数の値を更新するようにした．従って，テープルに記憶されるエントリ数は呼出しの回数と同じである．このような関数はメモリを多く消費する．また，計算結果を再利用できなく，メモ化によって最適化することができない．

表1 tak(12, 6, 0)の測定結果

	実行時間(ms)（メモ化なし）	実行時間(ms)（メモ化あり）	エントリ数
大域変数なし	2142	0	255
大域変数へ参照	2327	0	255
大域変数へ代入	2463	0	255
参照と代入両方	3667	メモリ使い切り	12604861

5 関連研究

Rito ら[1]はメモ化手法を拡張するためにソフトウェアトランザクショナルメモリ(STM)を利用し，ATOM システムを実装した．このシステムはプログラムからメモリへのアクセスを遮断し，読み込まれたロケーションと値を read-set に，書き込まれたロケーションと値を write-set にログする．ATOM は read-set と write-set をキャッシュに記憶し，それによって，内部状態があるプログラムに対してメモ化する．このような方法は STM ベースでメモ化するので，ランタイムシステムが STM をサポートする必要がある．

6 結論

参照透過性を備えない関数はメモ化を適用できない．本研究では，プログラム変換によって，大域変数への参照や代入がある関数をメモ化できる関数に変換する手法を開発した．変換後の関数に対してメモ化を適用した効果はプログラムのパターンによって異なる．本手法の活用例としては，イベントのログによるプロセス再演を高速化するためにメモ化を利用することを考えている．今後，メモ化の最適化効果を上げるために，大域変数へアクセスする関数をさらに区別してメモ化する必要があると考えている．

参考文献

[1]Hugo Rito et al., Memoization of Methods Using Software Transactional Memory to Track Internal State Dependencies, in Proc. 2010 ACM, pp.89-98, 2010.