並列項書換え抽象機械：
Parallel TRAM の設計と実装

近藤 勝

北陸先端科学技術大学院大学
情報科学研究科 情報システム学専攻

1997 年 2 月 14 日

キーワード： 並列項書換え抽象機械, TRAM, 並列 E-戦略 マルチプロセッサ CafeOBJ.

本研究の目的・背景

CafeOBJ をはじめとする代数仕様言語は、厳密なセマンティクスを持ち、明晰/無曖昧/無矛盾な（代数）仕様を記述できるという理由から広く注目を浴びている。また、代数仕様言語は、（順序ソート条件付き）項書換え系を計算モデルとしているため、項書換え系を処理エンジンとして使用すれば、代数仕様の種々の性質を機械的に検証・証明でき、さらに代数仕様をプログラムとして実行することが可能となる。この書換えエンジンを通常の計算機に効率よく実装する方法として、（順序ソート条件付き）項書換え系の抽象機械を定義し、この抽象機械を計算機上に写像する方法がある。”TRAM” はそのような抽象機械の一例である。

一方、ハードウェア技術の確実かつ急速な進歩により、安価かつ小型のマルチプロセッサが開発されてきた。これらマルチプロセッサは、現在広く使われている単一プロセッサのワークステーションやパーソナルコンピュータの延長線上にあり、近い将来広く使われることになるであろう。

以上のような背景から、書換えエンジンをマルチプロセッサ上に効率よく実装する方法を開発することは大いに意義があると考えられる。そこで本研究では、書換えエンジン ”TRAM” を、マルチプロセッサ上で効率よく（引数項の）並列簡約が実現できるよう拡張し、”Parallel TRAM” の設計を行なう。また、この Parallel TRAM を、マルチプロセッサ上に実装して、いくつかのベンチマークを実行させることで性能の評価を行なう。

並列 E-戦略

並列 E-戦略は、演算子ごとに引数項・全体項の簡約順序をユーザが陽に指定できる ”E-戦略” を拡張した戦略である。引数項の並列簡約を実現する戦略として、全ての引数項を自動的に並列簡約させるという戦略も考えられるが、マルチプロセッサのようなそれほどプロセッサ数を期待できない計算機では、プロセッサ数に対して、並列に簡約させる処理数の方がはるかに多くなってしまい、逆に効率の悪化を招く恐れがある。つまりマルチプロセッサの性能を最大限に活用するには、並列簡約の指定をユーザに開放し、並列性の拡大を制御する必要がある。従って並列 E-戦略では、引数項の並列簡約を明示的に指定する方針を取ることにする。(これは、E-戦略がユーザの意志を非常に尊重した戦略であるということともうまく合致する。)

<例>　`f/4 : (1 {2 3 4} 0)`

上記は、アリティ４の演算子 f に対する戦略を指定している。非零の自然数 n は n 番目の引数項簡約を表しており、零は全体項簡約を表している。”{”、”}” で囲まれた要素は並列に簡約され、その他の要素は左から逐次に簡約される。従って上記演算子を最外に持つ項を簡約する場合、まず第 1 引数を簡約し、続いて第 2〜4 引数を並列に簡約する。この並列簡約が全て終了したら最後に全体項を簡約する。

Parallel TRAM

Parallel TRAM は、先に説明した並列 E-戦略により指定された引数項の並列簡約を実現する並列項書換え系である。Parallel TRAM の構成は、基本的に TRAM の構成と変わらない。ただし、抽象命令を解釈・実行するインタープリタと、書換えの際 動的に内容が変化する 4 つの記憶領域だけはプロセッサの数だけ複製し、それぞれのプロセッサに抽象機械のレベルで仮想的に割り付けるものとする。(*印を付したものが複製される。)

<処理ユニット>

書換え規則のコンパイラ : 規則の左辺を弁別ネットに、規則の右辺を右辺の雛型にコンパイルする。

入力項のコンパイラ : 入力項をマッチングプログラムと戦略リストにコンパイルする。

抽象命令のインタープリタ* : 抽象命令を解釈・実行し、書換えを行なう。

<記憶領域>

弁別ネット : マッチする規則を効率良く検索するための木構造(弁別ネット)が格納される領域。

マッチングプログラム* : 弁別ネットとの間でパタンマッチを行なうための抽象命令(マッチングプログラム)が格納される領域。このマッチングプログラムは、Parallel TRAM(TRAM)内部において項を表現しており、書換えたい部分項に対応するマッチングプログラムを実行させると、その部分項に適用可能な規則の右辺を探し出すことができる。

戦略リスト* : 現在の項（マッチングプログラム）に対し、どのような順番でマッチングプログラムを実行していくのかを、リスト形式で格納する領域。（このデータを戦略リストと呼ぶ。）ユーザが指定した戦略は、この戦略リストに反映される。インタープリタは、この戦略リストの要素の順番にマッチングプログラムを実行し書換えを行なっていく。

右辺の雛型 : 書換えが行なわれ、入力項の構造が変化する（部分項がマッチした規則の右辺で置き換えられる）と、それにともないマッチングプログラムも変化させなければならない。また同様に戦略リストも再構成しなければならない。この領域には、その置き換えられるマッチングプログラムと、再構成する戦略リストの雛型が格納される。

スタック* : 抽象機械の作業領域。

変数バインディング* : 変数束縛の情報を格納する領域。

Parallel TRAM において、規則のコンパイルと入力項のコンパイルは、メインとなるプロセッサ上で逐次に行なわれるものとする。

また、並列簡約を実現するために、次の４つの命令を新たに追加する。引数項の並列簡約を戦略として指定すると、これら命令を呼び出すためのラベルが戦略リスト中の適切な位置に埋め込まれる。インタープリタは、これら命令に制御が移されると次のように動作する。

FORK : 戦略リストの次に続く簡約をアイドル状態のプロセッサに割り付ける。アイドル状態のプロセッサが無かった場合は、自分自身でこの簡約を処理する。

JOIN : FORK 命令で他プロセッサに割り付けられた簡約が全て終了するまで同期を取って待機する。待機中は自分自身の状態をアイドルにし、他プロセッサの簡約を受け付けられるようにする。

EXIT : FORK 命令で割り付けられた簡約が終了したことを親プロセッサに伝える。この命令は、FORK 命令が実行された時に割り付けられる。

SLEEP : 自プロセッサの状態をアイドル状態（他簡約受け付け可能状態）に移行させる。この命令は、自プロセッサ上で処理すべき簡約が全て無くなった時に実行される。つまりメイン以外のプロセッサには、デフォルトでこの命令のラベルが戦略リスト末尾に置かれている。

例えば、`f(A,B,C)` が入力項として与えられ、演算子 `f` に`({1 2 3} 0)` なる戦略が指定されていた場合に生成される戦略リストは次のようになる。

```
[<FORK>, <matching program for "A">, <FORK>, <matching program for "B">,
 <matching program for "C">, <JOIN>, <matching program for "f">,
 <return result>]
```

この戦略リストはメインとなるプロセッサ上に置かれ、次のようなステップで並列簡約を行なう。

1. メインプロセッサ上で戦略リストの要素を 1 つずつ取り出し簡約が始まる。この時サブプロセッサ 1 とサブプロセッサ 2 は、デフォルトで挿入された SLEEP 命令により待機している。

```
SL main = [<FORK>, <m.p. for "A">, ···  , <return result>]
SL sub1 = [<SLEEP>]
SL sub2 = [<SLEEP>]
```

2. FORK 命令により、引数項 A の簡約と EXIT 命令がアイドル状態のプロセッサ（サブプロセッサ 1）に割り付けられる。同様に引数項 B の簡約と EXIT 命令がサブプロセッサ 2 に割り付けられる。

```
SL main = [<m.p. for "C">, <JOIN>, <m.p. for "f">, <return result>]
SL sub1 = [<m.p. for "A">, <EXIT>, <SLEEP>]
SL sub2 = [<m.p. for "B">, <EXIT>, <SLEEP>]
```

3. メインプロセッサで引数項 C の簡約、サブプロセッサ 1 で引数項 A の簡約、サブプロセッサ 2 で引数項 B の簡約が並列に行なわれる。

4. メインプロセッサで C の簡約が終了し、なおかつサブプロセッサ 1、2 上の A、B の簡約が終了すると、EXIT 命令によりその終了が JOIN 命令に伝えられプロセッサ 1 上の同期が解除される。

5. メインプロセッサ上で全体項 f の簡約が行なわれ結果が出力される。サブプロセッサ 1、2 は SLEEP 命令により待機状態に入る。

研究成果と結論

本研究では、TRAM に引数項の並列簡約を行なうためのメカニズムを組み込み、Parallel TRAM の設計を完成させた。また、この Parallel TRAM を LUNA-88K2（MC88100 × 4 台搭載、マルチプロセッサ）上に C 言語を用いて実装し、性能の評価を行なった。得られた成果は次のようになる。

- 並列 E-戦略を新たに定義したことにより、ユーザーは引数項の並列簡約を明示的に指定することができ、結果として過度の並列性拡大を制限できるようになった。
- 戦略リストに並列簡約のメカニズムを組み込んだため、TRAM の持つ長所をほとんど損なうことなく並列化を実現することができた。
- フィボナッチ数列で 1.97 倍、並列 Lisp（並列処理が行なえるように拡張した Lisp インタープリタ）で 1.98 倍、クイックソートで 1.67 倍という速度向上を確認することができた。