A-4　依存グラフの変形による並列化処理の整合性

Consistency of Parallelizing Method Based on Transformation of Dependency Graph

内田 智士†　朝倉 宏一†　渡邉 豊英†
Satoshi UCHIDA　Koichi ASAKURA　Toyohide WATANABE

1．まえがき

我々は、並列化コンパイラの構築を支援するツールとして並列化コンパイラ・ツールキットを提案している[1]。並列化コンパイラ・ツールキットは、並列化の戦略と対象とする環境を記述することで、並列化コンパイラを構成する。これにより、容易に並列化コンパイラを構築することが可能となり、並列化コンパイラ開発者は、計算機環境と応用プログラムに適した並列化戦略の考案に集中することが可能となる。

我々は、プログラムの並列化処理を依存グラフによる変形処理としてモデル化している[2]。並列化戦略は依存グラフにおける変形パターンとなり、条件部を部分グラフパターンで、操作部を結合や分割などによる基本変形操作の集合として表現している。しかし、依存グラフの変形処理には変形ルールの衝突・競合や階層依存グラフの階層間における整合性などの問題が発生する。

本稿は、依存グラフの変形処理により生じる問題の解決手法を提案する。これにより、依存グラフの変形における矛盾を解消し、一意的な変形処理を実現させる。

2．並列化処理機構

プログラム並列化の注目要素としてはプログラム要素の特性とプログラム要素間の依存関係である。依存グラフは、プログラム要素をノードとして、依存関係をエッジとして表現し、注目要素を表現するのに適する。そこで、プログラム並列化処理を依存グラフに基づいてモデル化する。ここで、FOR 文などのループ文はそれらループブロックを1ノードとし、ループボディ文による依存グラフを下位階層に保持する階層型依存グラフを取り扱う。

依存グラフにおけるノードを並列処理単位へと拡張する。ここで、並列処理単位とは並列に実行されるプログラム単位のことで、他の並列処理単位との同期、通信処理は単位の最後に行われる。依存グラフ上における相互非依存のノードが並列処理可能な部分となる。このように依存関係の解釈を拡張することで、並列化処理は依存グラフの変形における相互非依存となるノードの生成処理となる。そして並列化戦略は、相互非依存のノード群を生成する変形パターンと解釈できる。この変形パターンは、条件部を部分グラフのパターンで、操作部を基本変形操作の組合せで表現する。部分グラフパターンはプログラム要素のタイプや実行予測ステップ数などをノードの属性として、依存データ量などをエッジの特性として制約を表現する。基本操作として、結合、分割、複製、消去の4つを定義する。これらの基本操作を組み合わせることにより、任意の変形操作を表現できる。我々は、この変形ルールを並列化ポリシと呼ぶ。並列化ポリシを依存グラフに適用して、依存グラフを変形することで、並列処理単位が生成される（図1）

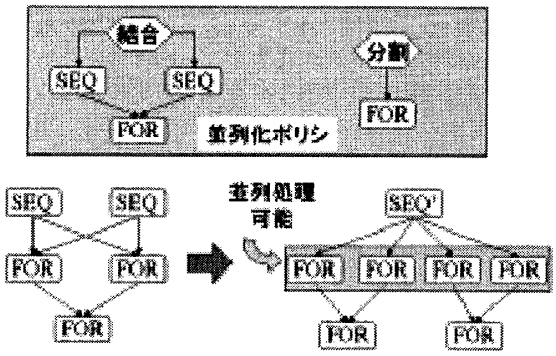

図1　依存グラフの変形による並列化処理

依存グラフの変形による並列化処理手法では、以下に挙げる問題が生じる。

- 並列化ポリシの衝突・競合
- 下位階層への並列化処理の適用による整合性

次節から、これらの問題に対する解決手法を提案する。

3．依存グラフ変形処理における整合処理

3.1　並列化ポリシの衝突・競合

前述したように、並列化ポリシは、変形対象となる部分グラフパターンとそれらに対する変形操作で表現されている。そのため、変形対象が重複する可能性がある。異なる並列化ポリシの場合を衝突と、同じ並列化ポリシの場合を競合と呼ぶ（図2）。

並列化ポリシは、並列化戦略を表現している。これらの並列化戦略は、抽出する並列性などから一定の順序が存在すると考えられる。そこで、並列化ポリシの衝突には、並列化ポリシに優先度を設定することで適用順序を一意にする。つまり、優先して適用すべき、つまり、より並列性を

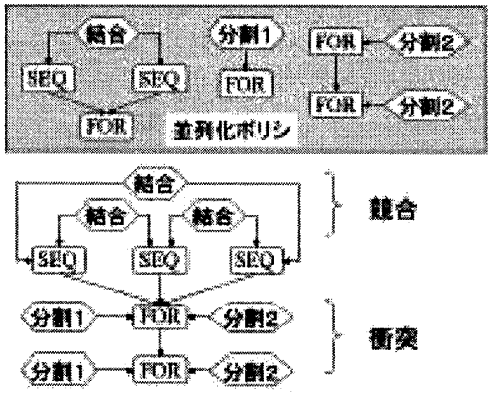

図2　並列化ポリシの衝突・競合

†名古屋大学 大学院 工学研究科 情報工学専攻
Graduate School of Engineering, Nagoya University

抽出可能な並列化ポリシほど高い優先度を設ける。

しかし、並列化ポリシの競合の場合には、同一の並列化ポリシであるため、優先度により一意に順序付けることが不可能である。そこで、我々は並列化戦略の意図に注目し、競合に対する並列化ポリシの変形操作の方法を定義する。

結合処理は、細かい粒度のノードを纏めて１つの粗い粒度のノードを生成するときに使用される。そのため、例えば、競合しているノードに対する結合対象すべてを纏めて、１つの粗い粒度のノードを作成することが目的となる。したがって、結合処理では対象となるノード全てを纏めて１つの粗い粒度のノードを作成する。分割処理では、粗粒度ノードを複数の細かい粒度のノードへ分割することで、並列処理可能な部分を増やすことができる。これらは、ただ細かい粒度のノードを生成するのではなく、他の並列化ポリシ適用箇所と併せて適度な粒度へ分解する必要がある。そのため、一度に競合した分割操作を適用するのではなく、一度の並列化ポリシ適用時には競合した分割操作のどれかひとつを適用すべきである。ここで、どの分割操作が選択されるべきかという問題が生じる。これらに対しては、他の適用箇所と合わせた分割操作が選択されるのが望ましい。そのため、分割方法の一番多い方法で分割操作を適用する。複製操作は、複製したノードと他のノードに対して結合するなどして、データのローカル化や細粒殿ノードをなくすために用いられる。つまり、複製操作分だけノードを複製する必要がでてくる。そのため、複製操作は競合分だけ適用し、複製ノードを作成する。最後に、消去操作は、対象ノードが不要となるため、依存グラフ上でノードを除去することが目的となる。この操作は一度消去すれば依存グラフ上に対象ノードが残らないため、任意の消去操作を１度だけ適用する。

3.2 下位階層へ並列化処理の適用

上記により、通常の依存グラフにおける変形に関する問題は解決される。しかし、並列化ポリシは一般に並列性抽出の概念をルール化しており、下位階層においてもその並列性を抽出するために適用する必要がある。しかし、並列処理単位はその並列性を抽出し、利用されるためにフラットに構成されるほうがよい。そこで、我々は階層間における並列処理単位の整合を保持しながら、並列処理のフラット化を図る。次から、これらを対処するため２つの手法を述べる。

最初に、ループ文の分割後の反復数からフラット化する。ループ文を分割後、反復が１回のプログラム片は逐次文で構成されるプログラム片と何ら変わらない。そのため、ループボディ文に反復変数の値を代入したプログラム片を上位階層における依存グラフと統合する。これにより、階層

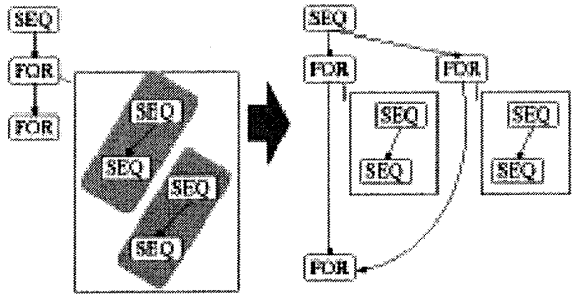

図 3 階層間の並列処理単位の整合処理

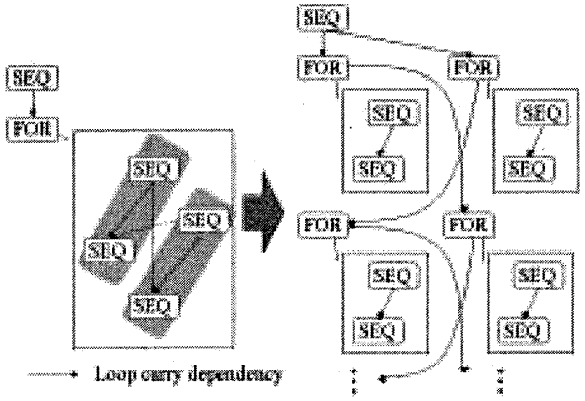

図 4 環状依存が存在する場合の整合処理

を上位階層へ上げることができ、フラット化を図ることができる。また、それにより、他のプログラム要素と対応した並列化ポリシの適用が可能となる。

２つ目に、下位階層の並列処理単位の関係からフラット化を図る。ここで、下位階層の依存グラフの解釈を次のように捕らえる。依存グラフで表現されているノードはループボディ部における文を表現しているが、これらの文はループの反復数分実行されるため、そのノードに反復数分だけの文が表現されていると考えることができる。そのため、下位階層で生成された並列処理単位は、上位階層のループ分を分割することを意味しているため、ループボディ部を分割したループ文であるノードを上位階層で構成する（図3）。これにより、下位階層で生成された並列処理単位の意図を反映することができる。最上位階層で全ての並列処理端が表現され、並列処理単位のフラット化が成される。

しかし、下位階層にデッドロックの要因となる環状の依存関係が存在するときは、ただループ文を分割しても上位階層でもその問題が保持される。我々は、この依存関係に対して環を分割するように、並列処理単位を分割することで対処している。これらの並列処理単位を上位階層に対応付けるときは、あらかじめループ依存距離で分割することにより、環状の依存関係を解消する（図4）。

4. おわりに

本稿では、依存グラフの変形における並列化処理における整合処理について述べた。これにより、並列化ポリシ適用における矛盾を解消し，一意的に処理が定義される。整合処理の実装とその評価が今後の課題である。

参考文献

[1] 朝倉宏一，渡邉豊英：“並列化コンパイラ構築のためのツールキットの構成”，信学技報 COMP，Vol.100, No.144，pp.97-104 (2000).

[2] 内田智士，朝倉宏一，渡邉豊英：“並列化コンパイラ・ツールキットにおける並列化ポリシによる並列処理単位生成手法”，電気関係学会東海支部連合大会講演論文集、p.296 (2001).