共有データスペースモデルを用いた並列プログラムのデバッグ法

7M−2

河本達哉　井上謙蔵
(東京理科大学)

1．はじめに

並列プログラム（PP）の実現に対して共有データスペース（タプル空間）モデルが提案された[1][2]．このモデルの上で，既存の大部分の型のPPが簡明に記述できることが示されたが[1]，本モデルはPPのデバッグにも極めて有利な環境を提供するものである．そこで本モデルの一つであるC-Linda[1]を用いてPPのデバッグ法を提案する．

逐次型の場合とは異なり，PPのデバッグにはシステム全体を包括的にとらえるための特殊な機構が必要になる．これは既存のPPに対するデバッグ法に欠けており，本モデルではそれが可能になる．しかしデバッグ情報を得るためにモデルの拡張が必要である．

2．Lindaモデルの拡張とデバッグ法概略

Lindaモデルでは，タプル空間から見てプロセスはブラックボックスとして扱われる．そこでプロセスがタプル空間にプロセス間で授受されるデータ、タプルを挿入する時，そのプロセスの情報と後述のオペレーション情報を付加するように拡張する．また，共有データスペースに対するオペレーションの履歴をプロセスとプロセス内のコード情報とともに全て記録する．

デバッグは図1の流れに沿って，状態分布解析及びプロセスの再現を目標として進められる．

3．C-Lindaコンパイラと静的な解析

C言語のパーサにタプル授受の演算，Lindaオペレーションを受理するように改良したC-Lindaコンパイラを実現した．このコンパイラにおいて，特にLindaオペレーションに着目し，その全てにオペレーションIDを付加する．また共有データスペースに対してあるプログラムブロックがどのようなパターンのタプルを生成することができるか，どのようなパターンのタプルを取り出そうと試み，処理を中断させることがあるか．どのようなプロセスを生成することがあるかに関して解析を行なう．この情報を用いて並列プログラム内及び並列プログラム間の依存グラフを調べることができる．また，コード生成時にデバッグ情報の埋め込みも行ない，後述のプロセスの再現時に利用する．

4．タプル空間の時間的状態遷移解析

タプル空間の時間的状態遷移の解析は，以下の方法で行なう．（図1）

プロセス主体解析
パターン主体解析
経験則を用いた解析
状態分布解析

① プロセス主体解析

ある一時点から一定時間内の並行プロセスの協調作業に着目する．このときタプル空間の時間的変移をそこに存在するプロセスに焦点を当てて解析を行なう．どのプロセスがどのプログラムコードから生成されたものであるか．どのプロセスによって生成されたか等のプロセス発生の時間的経緯を明白にする．またその一定時間内にタプルの生成と取り出しがどのプロセスとどのプロセスの間に起こったかを解析する．この解析によってプロセス間の依存関係が明白になる．

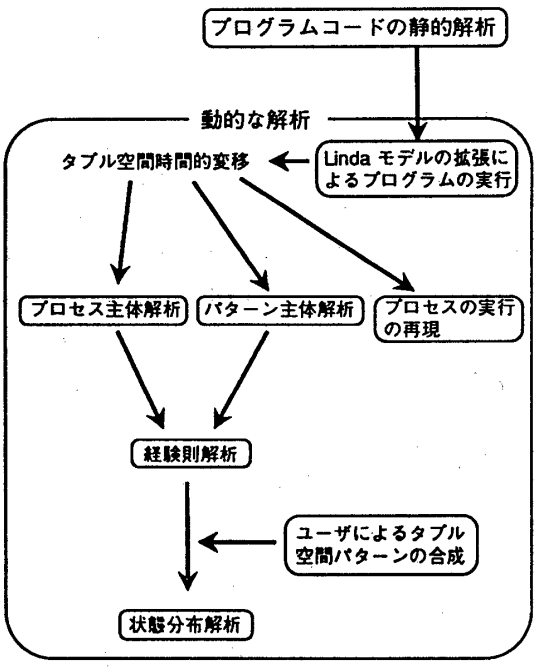

図1：解析戦略

Debugging Method for Parallel Programming Using Shared Dataspace Model
Tatsuya KAWAMOTO, Kenzou INOUE
Science University of Tokyo

② パターン主体解析

タプルを並行プロセス間の共有資源としてとらえると，その資源の使用状況に注目して情報を得ることができるであろう．共有資源の分類方法はパターンマッチングによる．資源が確保されているときはタプル空間から取り出されており，開放されているときはタプル空間に留まる．あるパターンにマッチするタプル数を資源量とし，時間的な資源量の変化を解析する．解析結果よりプロセスのLindaオペレーションと資源量の変化をシナリオとして自動的に生成する．

③ 経験則を用いた解析

より複雑で見通しの立たない様な並列プログラムのデバッグをするため，タプル空間の時間的状態遷移解析に経験的知識を用いる．タプル空間の用い方，誤りが発生する可能性があるかどうか等の解析を知識を用いてよりフレキシブルに解析する．知識として以下のようなルールを用いる．

変更を加えられずに出し入れされるタプル
　→ 特定の資源の使用権を表現している
使用権が長時間回復しない
　→ 使用権保持プロセスに誤りの可能性
複数の使用権を同時に取るプロセス
　→ 進歩性違反の危険性
ある時点の時間幅ではあるタプル群は変化しない
　→ その時点の時間幅で移動のあったタプルはごみ
長い時間待たれていたタプル
　→ プログラム上重要度が高い
読まれるのがほとんど
　→ プログラムの定まった状態を示す

④ 状態分布解析

これまでの手法はあまりにもミクロ的であり効果的でない場合がある．そのため並列プログラムの協調作業全体を包括的に解析する手法が必要になる．そこでタプル空間の状態分布解析を導入する．タプルを分類するためのタプルパターンの組み合わせよりタプル空間パターンを定義することができる．これによりタプル空間を分類することができる．また，タプル空間パターンを複数組み合わせてタプル空間パターンとすることもできる．ここでユーザは上記の経験則を用いた解析により発見されたプログラム上重要な（バグに関係する）幾つかのタプルパターンよりタプル空間パターンを組み立てる．そしてタプル空間パターンにマッチするようなタプル空間が，タプル空間の時間的状態遷移中のどこに分布するか，そのマッチしたタプル空間にどのようなタプルが存在するかを解析する．また，複数のタプル空間パターンを用いて解析を行ない，そのマッチするタプル空間の間にある遷移をマッピングとして取り出すことができる．

この処理は複雑な並列プログラミングを感覚的に捕えられる形に変換する作業で，解析作業の複雑さを軽減する．また，知識情報とそれに対する並列的な推論処理を施した場合などに適用できる手法と考えている．

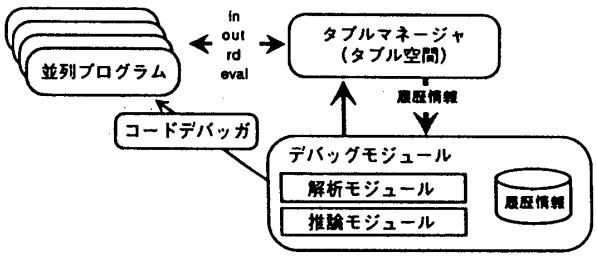

図2：デバッグシステム構成

5．コードデバッガを用いたプロセスの再現

プログラムの異常がありそうなプロセス及びその時間的な位置を指定する．再現するプロセスの生成時のタプル空間の状態を履歴から再現する．そのプロセスを以前記録したタプル空間へのオペレーションの時間的変移通りに動かす．他のプロセスの操作はシュミレートする．時間的変移通りにどうしてもならない時は再現中のプロセス内に非決定性プロセスが存在するものとし，再現を中断する．これにより，プロセス間の非決定性を決定的にすることができ，並列プログラムであるために起こる再現性のないバグを修正することができる．

6．おわりに

ここで示した解析法を組み合わせることで，有効なデバッグ情報を得ることができる．個々のプロセスの動きに注目するのではなく，システム全体として包括的に並列プログラムの動きをとらえることができる．それと同時に今まで困難であった並列プログラムの再現も実現できる．また，ここで用いたLindaモデルは一般的にどんな既存言語にも付加することができる．またその上で今までの並列プログラムのモデルをLinda モデルを用いて実現することが可能である．よってここで示したデバッグ法は，今までのどんな並列プログラムのモデルにも適用できるものである．

7．参考文献

[1] Carriero,N. and Gelernter,D.：Linda in Context, Communications ACM, Vol 32 No.4（Apr.1989）pp.444-458.

[2] Roman,G.-C. and Cox,K.C.：Declarative Visualization in the Shared Dataspace Paradigm, 11th I CSE Proc.,（May 1989）pp.34-43.