

4 R-3 並行プログラムにおける広域データフローテスト基準の拡張

有村耕治 古川善吾 牛島和夫

九州大学 工学部

1. はじめに

逐次プログラムのテストにおいてはプログラムの構造を静的に解析して得られたデータを利用する方法がある。プログラムのテスト充分性の評価に用いられるテスト基準もプログラムの解析結果に基づき定義されている。プログラムの構造解析にデータフロー解析を用い、その結果からプログラム中の特定のパス集合をテスト基準として選ぶパス選択テスト基準が提案されている。

並行プログラムにおいてもプログラムの構造の静的解析に基づくテスト基準が提案されている。Taylor は並行プログラム全体の並行状態としてタスク内の並行状態を規定する事象の組を用い、それをノードとする並行状態グラフ上で定義されるテスト基準を提案している^[1]。この並行状態グラフは、並行プログラムの並行状態をもとに定義されているために、並行に動作するタスクが増えると、並行状態が組合せ的に増加していき、並行状態グラフも非常に複雑なものとなる。並行状態グラフに基づくテスト基準も被覆すべき実行系列の数が増えて実現が困難となる。

そこで、我々は並行プログラムにおける共有変数のデータフローに基づく広域データフローテスト基準を提案した^[2]。広域データフローテスト基準は被覆すべき実行系列の数が少なく実現可能なテスト基準である。しかしながら、被覆すべき実行系列の数を少なくすることはそのテスト基準によるテスト戦略の信頼性を制限してしていることに他ならない。本論文では、この広域データフローテスト基準によるテスト戦略の適用度を増大するためにこの基準の拡張を行っている。

2. 広域データフローテスト基準

我々は、並行プログラムの共有変数のデータフロー解析に基づくテスト基準を提案した^[2]。

広域定義 プログラム中の共有変数が値を与えられる場所。

広域参照 プログラム中の共有変数が式の中に現れる場所。

広域データフロー (GDF) 同一の共有変数で異なるタスクに属する広域定義、広域参照の組 (def,ref)

広域データフローテスト基準

プログラム中の全ての広域データフロー (def,ref) を少なくとも一回は実行する。

このテスト基準の有効性を確かめるために実際のプログラムで広域データフローテスト基準の測定を行った。

Extensions of the Testing Criterion using Global Variable Data Flow for Concurrent Programs
Koji ARIMURA, Zengo FURUKAWA and Kazuo USHIJIMA
Faculty of Engineering, Kyushu University.

測定は、共有変数の定義、参照を記録するモニタを挿入して行う。測定に先立って、自動的にモニタを挿入するツールを作成した。被測定プログラムは、デッカーのアルゴリズムを実行するプログラムと共有変数を用いている 1st、2nd、3rd、4th と名付けた 4 つのプログラムである。これらの 4 つのプログラムはデッカーのアルゴリズムに至る 4 つの版である。3rd はデッドロックを起こす可能性を持つプログラム、4th はロックアウトを起こす可能性を持つプログラムである。それぞれのプログラムは約 70 ～ 90 ステップで Ada で記述されている。Ada 処理系は、VAX8800 上の Verdix Ada、Sun-3 上の TeleGen2 Ada、Sun-3 上の Alsys Ada である。測定結果を表 1. に示す。表の数字の上段は実行された広域データフローの数 / プログラム中の広域データフローの総数、下段は被覆率を表している。

表 1. 測定結果

処理系名	1st	2nd	3rd	4th	Dekker
Verdix	4/4 100%	6/6 100%	5/6 83%	4/10 40%	14/18 78%
TeleGen2	4/4 100%	5/6 83%	2/6 [†] 33%	4/10 [‡] 40%	14/18 78%
Alsys	4/4 100%	6/6 100%	3/6 [†] 50%	7/10 [‡] 70%	13/18 72%

[†]:Deadlock, [‡]:Lockout

TeleGen2 Ada 処理系、Alsys Ada 処理系の上では 3rd、4th のプログラムのデッドロック、ロックアウトを発見している。しかし、広域データフローテスト基準は 3rd のデッドロック、4th のロックアウトを発見するのに有効なテスト基準とは言えない。なぜなら、広域データフローテスト基準が共有変数の定義、参照関係のみに着眼して、場合の数を少なくしているからである。そこで、逐次プログラムのテスト基準で使われる P-use、C-use を使って広域データフローテスト基準の拡張を行う。逐次プログラムの P-use、C-use を使うテスト基準では、その基準で発見されたエラーが条件判定による誤りか、計算の誤りかというエラーの性質の判定が行える。広域データフローテスト基準においても P-use、C-use を用いてテスト基準の拡張を行いエラーの性質の判定の効果を与える。

3. 広域データフローテスト基準の拡張

広域参照を 2 つの種類に分ける。

- i) P-use : 条件文での参照
- ii) C-use : 代入文の右辺、出力文での参照

また、タスク内を制御フローグラフに表した場合の節点 n の次の節点の集合を succ(n) とする。

GDF-All-C-use テスト基準

プログラム中の全ての広域データフロー (def,ref) を少なくとも一回は実行する。ただし、ref は C-use である。

GDF-All-P-use テスト基準

広域データフロー (def,ref)、かつ、succ(ref) を実行することをプログラム中の全ての広域データフローについて少なくとも一回は行う。ただし、ref は P-use である。

GDF-All-DU-paths テスト基準

広域データフロー (def,ref)、かつ、succ(ref) を実行することをプログラム中の全ての広域データフローについて少なくとも一回は行う。

4. 広域データフローテスト基準の包含関係

テスト基準 Cr_1 が満足されるといつもテスト基準 Cr_2 も満足する時、テスト基準 Cr_1 はテスト基準 Cr_2 を包含するという^[3]。2 節、3 節で挙げたテスト基準の包含関係を解析すると、図 1. のようになる。図では、矢印の元のテスト基準が矢印の先のテスト基準を包含していること表している。

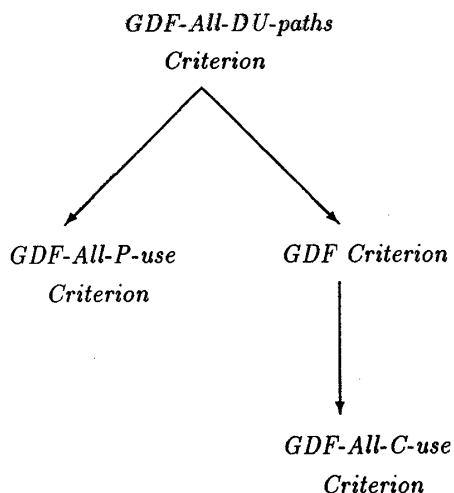


図 1. 広域データフローテスト基準の包含関係

広域データフローテスト基準は GDF-All-DU-paths テスト基準に包含されているが、GDF-All-C-use テスト基準を包含している。また、GDF-All-DU-paths テスト基準は GDF-All-P-use テスト基準を包含している。ただし、広域データフローテスト基準は GDF-All-P-use テスト基準とは、包含関係はない。したがって、広域データフローテスト基準を使用しても、時間的、コスト的余裕があり、もっと有効なテストが必要なときは、GDF-All-DU-paths テスト基準を使用し、余裕がなくもっと実現が簡単なテストが必要な時は、GDF-All-C-use- テスト基準を使用する。

5. 考察

テスト基準が有効であるとは、誤りを発見する可能性の高いパスを選択することである。広域データフローテスト基準はタスク間の共有変数の定義参照関係を基にしたテ

スト基準であり、必要条件にしているパスの数が少ない。従って、並行状態グラフを用いたテスト基準よりは有効性は低い。しかし、必要条件としているパスの数が少ないので実現の可能性は並行状態グラフを用いたテスト基準よりも高いと考えられる。

広域データフローテスト基準で発見できる誤りは、タスク間での共有変数の解釈の違い、つまり、用途の違いによるものである。このような誤りが実際のプログラムの中でどの程度存在するかはこのテスト基準を本論文で広域データフローを測定したプログラム以外にも適用して見なければならぬ。

広域データフローテスト基準はタスク間の共有変数に着目している。このテスト基準において被覆率を 100% とするには並行プログラムの実行の制御が必要となる。実行制御はテスト再現においても問題とされる。なぜなら、並行プログラムの実行には非決定性があるために、ある入力データをプログラムに与えてエラーが見つかったとしても、次にその入力データでエラーが見つかるとは限らないからである。Tai はエラーの再現性を克服するために SYN-sequence を用いた並行プログラムのテスト再現の手法を提案した^[4]。広域データフローテスト基準を達成するには、何らかの制御機構を実現しなければならない。

広域データフローテスト基準の有効性の検証がもっと必要である。広域データフローテスト基準を適用したプログラムは現在のところ数 10 ステップの Ada プログラムだけである。テスト基準の有効性を言うにはもっと多くのもっと大規模なプログラムに適用してみなければならない。本論文で提案した広域データフローテスト基準の拡張版でも、それぞれの有効性の違いを実際のプログラムで確認する必要がある。すると、広域データフローテスト基準と GDF-All-P-use テスト基準が実際にはどのような関係にあるかも明らかにできる。

また、広域データフローの測定はモニタを挿入して行っている。モニタ挿入による被測定プログラムへの影響を考察するために、被測定プログラムとモニタを挿入したプログラムとのスケジューリングの違いを調べてみなければならない。

参考文献

- [1] Taylor R.N., Kelly C.D.: "Structural testing of Concurrent Programs," Proc. Workshop on Software Testing, Banff, Canada, pp.164-169, July 1986.
- [2] 有村耕治, 古川善吾, 牛島和夫: "並行プログラムにおける広域データフローを用いたテスト基準の提案", 情報処理学会第 40 回全国大会, 1990 年 3 月.
- [3] Clarke L.A.: "A Formal Evaluation of Data Flow Path Selection Criteria," IEEE Trans. Software Eng., vol.15, no.11, pp.1318-1332, November 1989.
- [4] Carver R.H., Tai k.: "Reproducible Testing of Concurrent Programs based on Shared Variable," Proc. the 6th International Conference of Distributed Computing Systems, pp.428-433, May 1986.