3R-06

並行処理プログラムにおけるテストのための相互作用列テスト基準の提案

片山徹郎* 伊東栄典** 古川善吾† 牛島和夫‡
* 宮崎大学 工学部 情報システム工学科 ** 九州大学 情報基盤センター
† 香川大学 工学部 信頼性情報システム工学科 ‡ 九州大学大学院 システム情報科学研究院

1. はじめに

近年、並行処理プログラムが実際の場で多く書かれるようになってきた。これに伴い、並行処理プログラムの信頼性向上の方法の一つとして、テストが重要な役割を演じてきている。本稿では、並行処理プログラムのテストが満たすべきテスト基準として、新たに相互作用列テスト基準 (Interaction Sequences Testing Criteria($ISTC_k$)) を提案する。

相互作用列テスト基準は、並行処理プログラムにおける相互作用 (同期、通信、待ち) の列に基づいたテスト基準である。このテスト基準を、既に我々が提案している事象相互作用グラフ[1]上で定義する。

2. 事象相互作用グラフと協調路

この節では、今回並行処理プログラムのモデルとして用いる事象相互作用グラフと、テストケースとして定義した協調路とについて、また、並行処理プログラムが満たすべきテスト基準について簡単に述べる (詳細は[1])。

2.1 事象相互作用グラフ

プロセスの制御フローグラフの節点でプロセス間の相互作用に関連した節点、およびそれを含む分岐文からなる節点とその間の制御の移行を枝とするグラフを事象グラフと呼ぶ。

プロセス間の相互作用を、『同期』・『通信』・『待ち』の3種類で定義し、事象グラフと相互作用を用いて、プログラムをモデル化したものが「事象相互作用グラフ」*EIAG* である (図1参照)。

2.2 協調路

路は、事象グラフの開始節点を始点とし、終了節点を終点とする、隣り合う節点の対が枝集合に含まれるような節点列である。2つのプロセスから成るプログラムの場合、協調路とは、それぞれのプロセスの路の中で、相互作用の個数が等しい路の対である (図1参照)。

並行処理プログラムにおいて、プロセスが2つ以上存在する場合、すなわち事象グラフが2つ以上の場合は、任意の2つの事象グラフの間で協調路を定義する。並行処理プログラムが、k個のプロセスから成る場合、協調路は、k個の路の組である。我々は、協調路を事象相互作用グラフのテストケースと定義している。

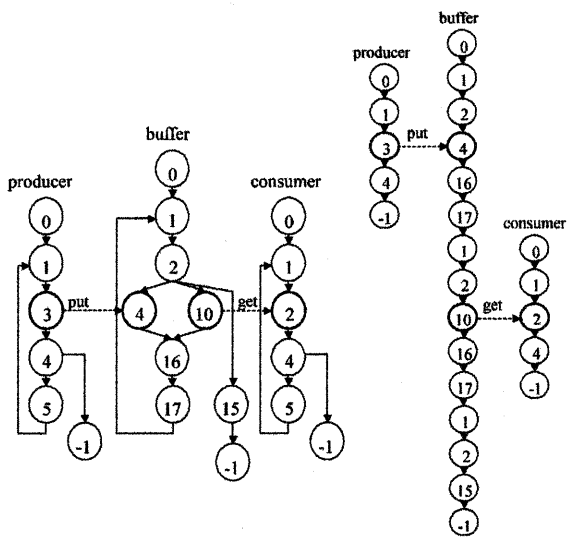


図1: 「生産者消費者問題」の事象相互作用グラフ (左) と協調路の一例 (右)

2.3 テスト基準

事象グラフ内にループが1つでも存在すれば、無限個数の路が存在する。テスト基準は、テストケース生成、および、テストの完了のための条件を定めたものである。我々が、並行処理プログラムのために採用したテスト基準を以下に示す。

i) 枝被覆テスト基準— 事象グラフのすべての枝を少なくとも1回は実行する。

ii) ループ被覆テスト基準— ループが存在する場合は、0回と1回繰り返す場合をそれぞれ考える。

iii) 相互作用被覆テスト基準— プロセス間に存在する相互作用は、少なくとも1回は実現する。

上記 i), ii) は、逐次処理プログラムにおいても用いられるテスト基準である。iii) は、実用的なテストを行なうために、相互作用に着目し定義したテスト基準である。

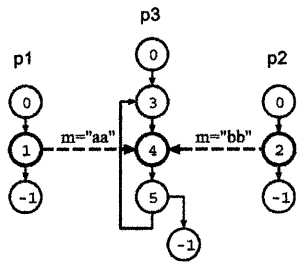
3. 相互作用列テスト基準

並行処理プログラムでは、同じテストデータを与えても、その時の計算機の状態によって、実行の振舞い、また

Proposal of Interaction Sequences Testing Criteria for Testing of Concurrent Programs
Tetsuro Katayama*, Eisuke Itoh**, Zengo Furukawa†, and Kazuo Ushijima‡
* Faculty of Eng., Miyazaki Univ.
** Conputing and Comm. Center., Kyushu Univ.
† Faculty of Eng., Kagawa Univ.
‡ Grad. Sch. of Info. Sci. and Electrical Eng., Kyushu Univ.

```
   P1                    P2
1: Send_data(aa);    2: Send_data(bb);

      P3
3: repeat
4: m:= Received_data;
5: until nodata;
```



m="aa" or "bb"

図 2: 動作の非決定性を持つプログラムの例

実行結果までもが変わることがある (動作の非決定性)。例として、図 2に P1, P2, P3 の 3 つのプロセスから成るプログラムを示す。

このプログラムにおいて、相互作用は下記である。ただし、$Comms$ は相互作用における通信集合を表す。

$$(1,4,m),(2,4,m) \in Comms.$$

このプログラムの実行結果における m の値は、$Comms$ の要素の実行順序に依存する。すなわち、実行結果は下記の通りである。

$$Seq1: \ <(1,4,m),(2,4,m)>; \longrightarrow \ m = \text{“}bb\text{”},$$
$$Seq2: \ <(2,4,m),(1,4,m)>; \longrightarrow \ m = \text{“}aa\text{”}.$$

このプログラムを 1 回実行すれば、$Seq1$ か $Seq2$ のどちらかだけが実行される。その際、全路 (all-paths) テスト基準 (C_∞)[2] を満足する。また同時に、我々が従来用いていた、2.3 節で述べた 3 つのテスト基準も満足する。しかしながら、上記 $Seq1$ と $Seq2$ の 2 つの列を満足しなければテストは充分ではない。

そこで、相互作用の実行順序を考慮した新たなテスト基準として、相互作用列テスト基準 ISTC (Interaction Sequences Testing Criteria) を提案し、以下のように定義する。

$$TE(\mathsf{ISTC}_k) \equiv \{\omega | \omega \in Seq(Interactions) \wedge |\omega| = k\}.$$

ここで、$TE(Cri)$ は、テスト基準 Cri におけるテストすべき項目 (テスト事象) の数を表し、$Interactions$ は相互作用集合を表す。

$k = 1$ の場合 (ISTC_1) は、我々が従来用いていた相互作用被服テスト基準と等価である。$k = 2$ の場合は、次式で表せる。

$$TE(\mathsf{ISTC}_2) \equiv \{<(a,b,X),(c,d,Y)> | \forall (a,b,X),(c,d,Y) \in Interactions\}.$$

相互作用被服テスト基準の代わりに、相互作用列テスト基準である ISTC_2 を採用すれば、図 2 における $Seq1$ と $Seq2$ の両方を、テストの際に満足させることになる。また、このように、実行しなければならない相互作用の列を提示することそのものが、テストの際に役立つと考えられる。

4. 議論および評価

相互作用列テスト基準を用いた並行処理プログラムのテストにおける信頼性について議論する。以下、ある誤りに対して「信頼できる」とは、その誤りがプログラム中に存在すれば必ず発見できることを指す。

任意のプログラムに対して信頼できるテスト法は「全数テスト」しかないので、他のあらゆるテスト基準は、ある誤りに対してのみ信頼できる。相互作用列テスト基準は、プロセス間で通信を行なえば必ず発生する誤りである「完全通信誤り」に対しては信頼できる。通信を行なっても必ず発生するとは限らない誤りである「部分通信誤り」については、誤りが発見できるとは限らないため信頼できない。また、デッドロックや飢餓状態 (ライブロック) に関しても信頼できない。ただし、協調路を用いてテストを行なうことにより、ある性質を持ったデッドロックに対しては信頼できることが分かっている[3]。

次に、テスト事象の数について議論する。ISTC_k における k の値が大きくなれば、実行しなければならない協調路はより多くなる。ISTC_k のテスト事象の数は、$|TE(\mathsf{ISTC}_k)|^k$ となる。

$k = 1$ の時、すなわち、ISTC_1 は、相互作用被服テスト基準と等価である。テスト事象の数を考慮すると、実用的なテスト基準であるが、図 2 のような動作の非決定性を持つプログラムでは、充分なテスト基準とは言えない。$k \geq 2$ の場合に、この場合は解決できる。

一般的に、テストケースの数とプログラムの振舞いの正確な記述との間には相関関係がある。正確な記述にはテストケースの数が増え、テストケースの数を減らすと、プログラムの振舞いを反映できなくなる可能性が生じる。すなわち、テストに費やすことができる時間とプログラムの振舞いや特徴とを考慮して、ISTC_k の k の値を定めなければならない。これらの詳細な関係については、今後の課題とする。

参考文献

[1] T. Katayama, Z. Furukawa and K. Ushijima: “Event Interactions Graph for Test-case Generation of Concurrent Programs,” *Proc. 1995 Asia-Pacific Softw. Eng. Conf. (APSEC'95)*, pp.29-37, 1995.

[2] W. E. Howden: “Reliability of the Path Analysis Testing Strategy,” *IEEE Trans. Softw. Eng.*, Vol.2, No.3, pp.208-215, 1976.

[3] T. Katayama, Z. Furukawa and K. Ushijima: “Design and Implementation of Test-case Generation for Concurrent Programs, ” *Proc. 1998 Asia-Pacific Softw. Eng. Conf. (APSEC'98)*, pp.262-269, 1998.