7M－5

テストによって同期・通信誤りを発見するための並行処理プログラムのモデルに関する一考察

片山徹郎＊　古川善吾＊＊　牛島和夫＊

＊ 九州大学 工学部 情報工学科　＊＊ 九州大学 情報処理教育センター

1. はじめに

近年、並行処理プログラムが実際の場で多く使わ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。逐次処理プログラムに比べると、並行処理プログラムは同期や通信などの機構を備えているため動作が複雑である。

並行処理プログラムで発見される誤りは、以下のように分類できる[1]。

i) プロセス内での誤り — 逐次処理プログラムにおいて考えられる誤り[2]。

ii) 通信に関する誤り — 並行に実行されるプロセス間でデータの受渡し時に発生する誤り。

iii) 同期に関する誤り — 並行に実行されるプロセス間で同期の取り方の誤り。これについては、さらに以下のように分類できる。

(1) 安全性の破壊 — 相互排除の失敗によるデータの一貫性の喪失。

(2) 生存性の破壊 — 並行処理プログラムが意味のある動作を全く行わない。デッドロックとも呼ばれる。

(3) 公平性の破壊 — ある特定のプロセスだけが実行を不当に待たされる。ライブロックとも呼ばれる。

本論文では、並行処理プログラム特有の同期・通信に関する誤りを、テストによって発見するための、並行処理プログラムのモデルを提案する。

2. 事象相互作用グラフ

2.1 事象グラフ

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（これについての詳細は次節）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 EG と呼ぶ。

$$EG = (N, E, s, f).$$

ただし、N は節点の集合、$E = \{e = (a, b) | a, b \in N\}$ は枝の集合である。節点 $s, f \in N$ は以下の条件を満足し、それぞれ開始節点、終了節点と呼ぶ。

$$s \in N \wedge \forall a[a \in N \rightarrow < a, s > \notin E],$$
$$f \in N \wedge \forall b[b \in N \rightarrow < f, b > \notin E].$$

プログラム $\mathbf{P}$ を構成するプロセス $P_i, 1 \leq i \leq numProc(\mathbf{P})$（ここで、$numProc(\mathbf{P})$ はプログラム $\mathbf{P}$ を構成するプロセス数である）の事象グラフ EG_i の集合が事象グラフ集合 EGs である。

$$EGs(\mathbf{P}) = \{EG_i = (N_i, E_i, s_i, f_i) | 1 \leq i \leq numProc(\mathbf{P})\}.$$

2.2 相互作用

プロセス間の相互作用を、以下の3種類で定義する。

i) 『同期』
識別子 α で関係付けられた節点 a と節点 b とが同時に実行されることを表す『同期』 $sync$ の集合が『同期集合』 $Sync$ である。

$$Sync = \{sync = (a, b, \alpha)\}.$$

ii) 『通信』
識別子 β を通して節点 a から節点 b にデータが渡されることを表す『通信』 $comm$ の集合が『通信集合』 $Comm$ である。

$$Comm = \{comm = (a, b, \beta)\}.$$

iii) 『待ち』
識別子 γ によって節点 a が節点 b を待つ可能性があることを表す『待ち』 $wait$ の集合が『待ち集合』 $Wait$ である。

$$Wait = \{wait = (a, b, \gamma)\}.$$

2つの文が『同期』をとるというのは、それぞれの文がお互いに「ある文の実行開始を必ず待つ」とも言える。節点 a は、必要に応じて実行開始節点 a_s と実行終了節点 a_f とに分割することができる。分割した2つの節点の間には、枝 $e_{sf} = (a_s, a_f)$ が存在するものとする。

事象グラフと相互作用を用いて、プログラム $\mathbf{P}$ をモデル化したものが「事象相互作用グラフ」 $EIAG$ である。

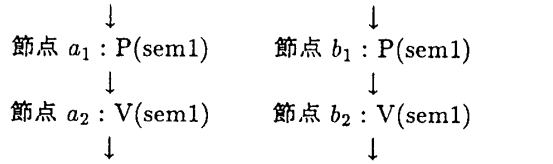
$$EIAG(\mathbf{P}) = \{EGs, Sync, Comm, Wait\}.$$

3. 事例研究

この章では、様々な並行処理機構が事象相互作用グラフで表現可能なことを示す。

A Model for Concurrent Programs to Detect Communication Errors and Synchronization Errors in Testing.
Tetsuro KATAYAMA*, Zengo FURUKAWA** and Kazuo USHIJIMA*
*Department of Computer Science and Communication Engineering, Kyushu University.
**Educational Center for Information Processing, Kyushu University.

i) セマフォ



相互作用を用いると以下のように表せる。

$$Wait = \{(b_1, a_2, sem1), (a_1, b_2, sem1)\}.$$

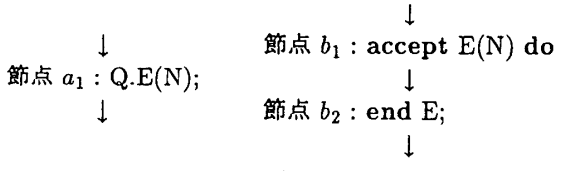
ただし、sem1 がバイナリセマフォでなく、多重値セマフォである場合は、以下のようになる。

$$Wait = \{(m, n, sem1)\},$$

ここで $m \in \{a_1, b_1, c_1, \ldots\}, n \in \{a_2, b_2, c_2, \ldots\}$ である.

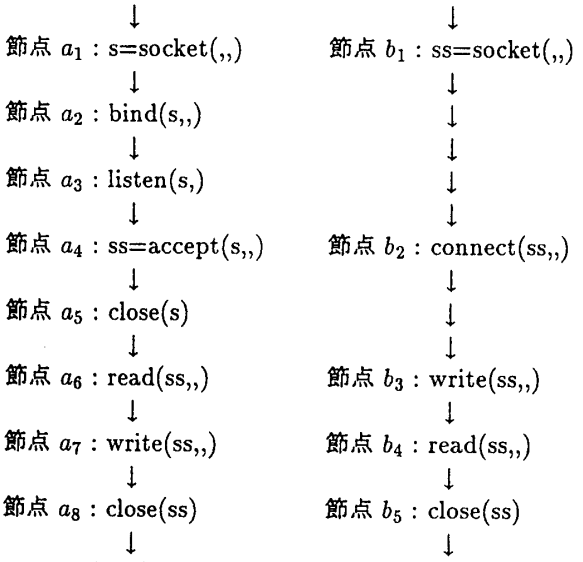
ii) ランデブー

Ada のランデブーは、以下のように表せる。



$$Comm = \{(a_{1s}, b_1, N)\},$$
$$Sync = \{(a_{1s}, b_1, E), (a_{1f}, b_2, E)\}.$$

iii) ソケット



このソケットは以下のように表せる。

$$Sync = \{(a_4, b_2, ss)\},$$
$$Comm = \{(b_3, a_6, ss), (a_7, b_4, ss)\},$$
$$Wait = \{(a_6, b_3, ss), (b_4, a_7, ss)\}.$$

4. テスト基準の設定と発見できる誤り

テスト基準として、まずこのグラフ上の「『同期』・『通信』・『待ち』を少なくとも 1 回は実行する」というものが考えられる。『待ち』に関しては、待つ場合と待たない場合との 2 種類について調べるものとする。この場合、プログラム中に存在する『同期』の数を m、『待ち』の数を n、『通信』の数を l とすると、測定事象数は $m + 2n + l$ となる。このテスト基準は「完全通信誤り」[1] に関しては信頼できる。なぜなら、『通信』を少なくとも 1 回は実行することが義務づけられるからである。また、プログラム内の変数や同期・通信命令の実行系列にかかわらず発生するクラス 0 のデッドロック[3] についても発見することができる。

このグラフ上で、『待ち』の待つ場合と『同期』との組合せに着目してテストすれば、プロセス間の通信・同期命令がある特定の実行系列にしたがった場合にのみ発生するクラス 1 のデッドロックを発見することができる。『同期』の数を m とし、『待ち』の数を n とすると、測定事象数は ${}_{m+n}C_1 + \cdots + {}_{m+n}C_{m+n} = 2^{m+n} - 1$ となる。

安全性の破壊を発見するためには、『待ち』の待たない場合の組合せに着目してテストすれば良い。『待ち』の数を n とすると、測定事象数は ${}_nC_1 + \cdots + {}_nC_n = 2^n - 1$ である。ここでバイナリセマフォの場合、待たない場合の 2 個の組合せについて考えれば良いので 測定事象数は、${}_nC_2 = n(n+1)/2$ となる。

公平性の破壊を発見するためには、『待ち』の待つ場合と『同期』とに着目して、その動作がどのような順番で行なわれるかをテストする必要がある。この場合、どこか 1 つでもループが存在すると、測定事象数は無限となる。仮にループが存在しない場合でも、『同期』の数を m とし、『待ち』の数を n とすると、測定事象数は $(m+n)!$ となり、公平性の破壊の発見については期待できない。

5. おわりに

本論文では、同期・通信誤りを発見するための並行処理プログラムのモデルとして事象相互作用グラフを提案した。この事象相互作用グラフ上で、様々なテスト基準を設定することにより、公平性の破壊を除く同期・通信誤りを発見することが可能である。今後の課題として、実際の並行処理プログラムからグラフの自動生成と、グラフ上からこれらのテスト基準を満足するようなテストケースの自動作成とが挙げられる。具体的には、文献[4] で用いた手法の適用を考えている。

参考文献

[1] 古川善吾, 牛島和夫: “並行処理プログラムのテスト法に関する一考察,” 日本ソフトウェア科学会第 6 回大会論文集, pp.185-188, 1989.

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

[3] 伊東栄典, 川口豊, 古川善吾, 牛島和夫: “同期誤りに対する順序列テスト基準の信頼性について,” 平成 5 年度電気関係学会九州支部連合大会論文集, p.808, 1993.

[4] 片山徹郎, 菰田敏行, 古川善吾, 牛島和夫: “並行処理プログラムにおけるテストケースの定義と生成ツールの試作,” 情報処理学会論文誌, 第 34 巻, 11 号, pp.2223-2232, 1993.