3D-4

Ada@タスキングにおけるデッドロックの検出方法

程 京徳　　牛島 和夫
(九州大学)

1. まえがき

Ada[1]並列プログラムのタスク間でランデブーにより通信する際に待ち合せによって起こるデッドロック(以下、Adaタスキング・デッドロックという)を検出することは、Adaプログラミング支援環境が提供すべき機能の一つである。しかし、これについて、今迄提案されていた検出方法は、Adaタスキングの幾つかの重要機能(例えば、terminate選択肢によるタスクの終了、abort文によるタスクの強制終了など)に対処することができなかった[2][3]。その原因は、Ada並列プログラムのタスク間の動的な依存関係を厳密に考慮していないことによると考える。

本論文では、Adaタスキング・デッドロックの検出について、タスク間の動的依存関係を考慮した新しい方法を提案する。この方法は、Adaタスキングの上記の機能にも対処することができる。

2. 基本概念

Ada並列プログラムの実行過程でタスクが経過する状態を以下の18通りに分類する:「起動」、「起動待機」、「起動成功」、「起動失敗」、「実行」、「遅延による待機」、「呼び出しによる待機」、「受け付けによる待機」、「選択待機」、「ブロック起動待機」、「ブロック終了待機」、「異常」、「完了した」、「終了待機」、「終了した」、「ランデブー」、「ランデブーによる待機」、「再開」。

「起動」、「起動成功」、「実行」、「遅延による待機」、「ランデブー」、「再開」をタスクの続行状態といい、残りのそれ以外をタスクのロッキング状態という。

Ada並列プログラムPの実行中に、一部分のタスク間のエントリ呼び出し関係が循環になると、Pが循環エントリ呼び出しになったという。

Ada並列プログラムPの実行中に、Pの終了していないタスクの状態が全てロッキング状態になると、Pがグロバール・デッドロックになったという。

[定義1] Ada並列プログラムPの実行中にタスクAとタスクBとに対して次の何れかが成立すれば、AをBの親タスクといい、BをAの子タスクという:

1) Bが直接に依存しているマスタ[1]はAである。
2) Bが直接に依存しているマスタはAの本体中のブロック文又はAの本体中で宣言されている副プログラムである。
3) Bが直接に依存しているマスタはライブラリ・パッケージであり、Aは主タスクである。

[定義2] タスクTの時刻tにおけるタスキング動的従属木DT(T, t)は、次の条件1)~3)を満足する3項組(task, state, entries)(以下、節点taskという)の有限集合である。但し、taskは、T或いは時刻tでTに依存しているタスクの識別子であり、stateは、時刻tにおけるタスクtaskの状態であり、entriesは、タスクtaskが持つエントリの集合である:

1) 特殊な節点(T, T_s, T_e)が存在し根節点と呼ぶ。
2) 根節点以外の節点をn≧0個の交わらない集合$DT(T_1, t), ..., DT(T_n, t)$に分けることができる。$DT(T_1, t), ..., DT(T_n, t)$は、それぞれタスク$T_1, T_2, ..., T_n$の時刻tにおけるタスキング動的従属木である。節点Tを節点$T_1, ..., T_n$の親節点といい、節点$T_1, ..., T_n$を節点Tの子節点という。もしタスクAがタスクBの親タスクであれば、DT(T, t)の中で、節点Aは節点Bの親節点であり、節点Bは節点Aの子節点である。
3) 時刻tでタスクTに依存しているタスクT'に対して、DT(T, t)は、T'に対応する節点を一つかつ一つに限り持つ。

Ada並列プログラムPの主タスクの時刻tにおけるタスキング動的従属木をDT(P, t)と書く。これは、Ada並列プログラムPの実行過程の時刻tにおける各タスクの状態とタスク間の動的依存関係とを反映する。

タスク間のエントリ呼び出し関係を反映するために、DT(P, t)の節点T_cとT_aのエントリe(以下$T_{a.e}$で表す)との間をT_cから$T_{a.e}$に向かう有向枝で繋ぐことができる。以下、このような有向枝をエントリ呼び出し枝といい、$(T_c, T_{a.e})$で表す。

DT(P, t)に対して、次の操作を許す:

1) 新しい節点を生成する。
2) 指定した節点及びその子孫節点を探索する。
3) 指定した節点の状態を更新する。
4) 指定した節点を除去する。
5) 新しい呼び出し枝を繋ぐ。
6) 指定した呼び出し枝を除去する。

3. Adaタスキング・デッドロックの検出

被テストAda並列プログラムPの各タスクの状態変化に応じてDT(P, t)に対する操作を行って、Pの実

Detecting Deadlocks in Ada Tasking Programs
Jingde CHENG and Kazuo USHIJIMA
Kyushu University

@ Adaはアメリカ政府Ada Joint Program Officeの登録商標である。

行中に起こる循環エントリ呼び出しとグロバール・デッドロックとを動的に検出することができる。

循環エントリ呼び出しの検出　Pの各タスクの状態変化に応じてDT(P, t)に対して次の操作を行う：

1) タスクT_cが エントリ$T_{a.e}$を呼び出して「呼び出しによる待機」状態になると、エントリ呼び出し枝$(T_c, T_{a.e})$を繋ぐ。
2) タスクT_cが エントリ$T_{a.e}$を呼び出して「ランデブー」状態になり、もしエントリ呼び出し枝$(T_c, T_{a.e})$がまだ繋がっていなければそれを繋ぐ。
3) タスクT_cが エントリ$T_{a.e}$を呼び出しランデブーを終えて「再開」状態になると、エントリ呼び出し枝$(T_c, T_{a.e})$を除去する。
4) タスクT_aが「完了した」状態になり、もし$(T_c, T_{a.e})$のような形式のエントリ呼び出し枝がまだ繋がっているならそれらを全て除去する。

DT(P, t)に対して以上のような操作を行いながら、新しいエントリ呼び出し枝が既に存在しているエントリ呼び出し枝と共に有向閉路になるかどうかをチェックする。もしある時点tで新しいエントリ呼び出し枝の加入によって有向閉路が生成されれば、Pは時点tで循環エントリ呼び出しになったに違いない。

グロバール・デッドロックの検出　被テストAda並列プログラムPに対して、EXECUTABLEとWAITINGという二つのカウンタを設定して、DT(P, t)に対する操作に応じてEXECUTABLEとWAITINGとを次のように更新する：

1) 新しい節点を生成すると、
   EXECUTABLE := EXECUTABLE + 1
2) ある節点を除去すると、
   EXECUTABLE := EXECUTABLE − 1
3) ある節点の状態が続行状態からロッキング状態に変化すると、
   EXECUTABLE := EXECUTABLE − 1
   WAITING :=WAITING + 1
4) ある節点の状態がロッキング状態から続行状態に変化すると、
   EXECUTABLE := EXECUTABLE + 1
   WAITING :=WAITING − 1

1)~4)の更新の直後に、それらの値をチェックする。もしある時点tでこの二つのカウンタの値が等しければ、Pは時点tでグロバール・デッドロックになったに違いない。

4. Adaプログラム変換に基づく実現方法

上で述べたAdaタスキング・デッドロックの検出方法を実現するために、被テストAda並列プログラムのタスキング状態に関する情報が必要である。それは被テストAda並列プログラムの各タスクの状態をモニタすることによって得ることができる。モニタリングはAdaプログラム変換に基づいて行う。

タスキング動的従属木DTは、抽象データ型としてAdaパッケージで実現することができる。実現に当たって、各タスク間の動的な依存関係を反映するために、被テストAda並列プログラムの各タスクに一意的識別子を付けて各タスク間の動的な依存関係と共にタスク名前表に登録し更新しなければならない[4]。

被テストAda並列プログラムPを次のように変換すればよい。Pのタスキング状態をモニタするもの(以下、情報収集部という)をタスクとしてPの主タスクの宣言部で宣言する。情報収集部は、若干のエントリを持つ。Pのソース・テキストの適当な場所に情報収集部のエントリを呼び出す文を挿入する。また、Pの各タスクごとにその終了をテストするタスクをそのマスタの宣言部で宣言する。このように変換したプログラムを実行すると、それと情報収集部との通信によって、Pのタスキング状態に関する情報をエントリのパラメータとして情報収集部に渡すことができる。

情報収集部は、収集した情報に応じてDT(P, t)で次の操作を行う：

1) タスクTが「起動成功」状態になったら、Tに対応する節点を生成する。
2) タスクTが「起動成功」、「異常」、「終了した」以外の状態になったら、Tに対応する節点を探索しその状態を更新する。
3) タスクTが「異常」状態になったら、Tに対応する節点及びその子孫節点を探索しこれらの節点の状態が「完了した」でない限り「完了した」に更新する。
4) タスクTが「終了した」状態になったら、Tに対応する節点を除去する。

情報収集部は、以上の操作の上で、Pの各タスクの状態変化に応じてDT(P, t)に対してエントリ呼び出し枝を繋がいだり除去したり、カウンタEXECUTABLEとWAITINGとを更新したりして、Pの実行中に起こる循環エントリ呼び出しとグロバール・デッドロックとを動的に検出することができる。

5. むすび

本論文で提案したAdaタスキング・デッドロックの検出方法をData General社が提供しているAda処理系ADEの上で実現中である。

参考文献

1) Reference Manual for the Ada Programming Language (ANSI/MIL-STD-1815A), United States Department of Defense, Jan. 1983.
2) German,S.M. : Monitoring for Deadlock and Blocking in Ada Tasking, IEEE Trans. Softw. Eng., Vol.SE-10, No.6, pp.764-777, 1984.
3) Helmbold,D. and Luckham,D. : Debugging Ada Tasking Programs, IEEE Software, Vol.2, No.2, pp.47-57, 1985.
4) 程京徳、牛島和夫：Adaタスクに一意的識別子を付ける方法、電気関係学会九州支部第39回連合大会講演論文集、1986.