ペトリネットによる並列プログラムの動作解析に関する一考察

2R－3

前田 賢一*　中島 俊介**　長谷川 晴朗*
*沖電気工業(株)　**沖通信システム(株)

1.はじめに

最近、並列計算機の普及に従い、並列プログラムの開発支援の必要性が論じられ、また研究が盛んに行われている。筆者らは、その開発支援環境構築の一環として、並列プログラムの動作評価である動的解析及び静的解析の研究を行っている[1]。

並列プログラム(ここではプログラミング言語としてGHCをとりあげる)はその特性として、計算が並列に進行し、それらが互いに干渉し合い、また実行順序に再現性がないので、デバッグが非常に困難であると言われている[2]。そこで、実行前に、できるだけ多くのバグを取り除き、かつプログラムの性質を抽出しておくことは、実行時のデバッグに対する負荷を軽減することになる。

今回は、非同期並行系システムのモデル化に適していると言われているペトリネット[3]を利用したGHCプログラムの静的解析について報告する。

2.ペトリネットによるGHCのモデル化

現在、GHCプログラムをペトリネットで表現する方法として2種類のモデルについて検討を進めている。それぞれについて、ペトリネットとGHCの対応を表1に示す。

表1 ペトリネットとGHCの対応

PN / モデル	プレース	トランジション	アーク	トークン
(a)	ヘッド ボディゴール	ガードゴール	リダクション	ゴールの 起動
(b)	出力変数		出力変数の 値の代入 ガード部での 値の参照	出力変数の 具体化

表に示した通り、モデル(b)はモデル(a)を包含した形になっている。モデル(a)は、変換が単純である。一方モデル(b)は、述語の変数をプレースで表現しているため、プログラムの同期関係がうまく表現できる。

これらの対応に基づいて下記のGHCプログラムをペトリネットへ変換した例を示す(図1)。

```
p(X) :- true | q(S),r(S,X).   %(1)
q(S) :- true | S=m.           %(2)
r(S,X) :- S=m | X=a.          %(3)
r(S,X) :- S=m | X=b.          %(4)
```

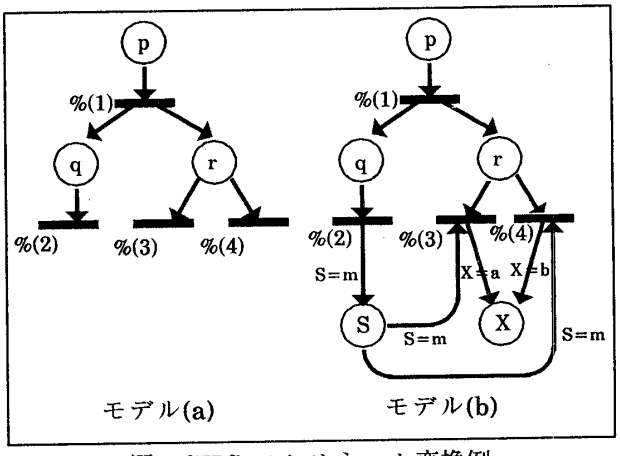

図1 GHC-ペトリネット変換例

3.ペトリネットを利用した解析

プログラムに含まれているバグを検出するためにペトリネットの解析能力を利用する。

モデル(b)はモデル(a)を包含しているので、ここでは先程のプログラムをモデル(b)に基づいて変換したペトリネットについての解析例を示す。ただしここで、初期マーキングとは、実行時に設定されているマーキング、すなわちプログラムの起動の述語に対応する。目的マーキングとは、ペトリネットがある発火系列で発火したときに最終的に残るマーキング、すなわちプログラムの終了時に残っている情報に対応する。

プログラムをペトリネットで表した場合の接続行列Aと初期マーキングM、目的マーキングM'が与えられた時、MからM'に可達であると仮定する。その時のトランジションの発火系列σ(空の場合を含める)において、発火ベクトル$f(\sigma)$が次の行列方程式(1)における非負整数解xであることが知られている。

$$M' = M + xA \quad \cdots (1)$$

また、先程のプログラムは図2の様な関係になる。すなわち、

初期マーキングM:(1,0,0,0,0)
目的マーキングM':(0,0,0,0,1)

接続行列 A:

	P1	P2	P3	P4	P5
T1	-1	1	1	0	0
T2	0	-1	0	1	0
T3	0	0	-1	-1	1
T4	0	0	-1	-1	1

A Study on a Static Analysis of Parallel Program by using Petri Nets,
Ken'ichi MAEDA*, Shunsuke NAKAJIMA**, Haruo HASEGAWA*
*Oki Electric Industry Co., Ltd., ** Oki Telecommunication Systems Co., Ltd.

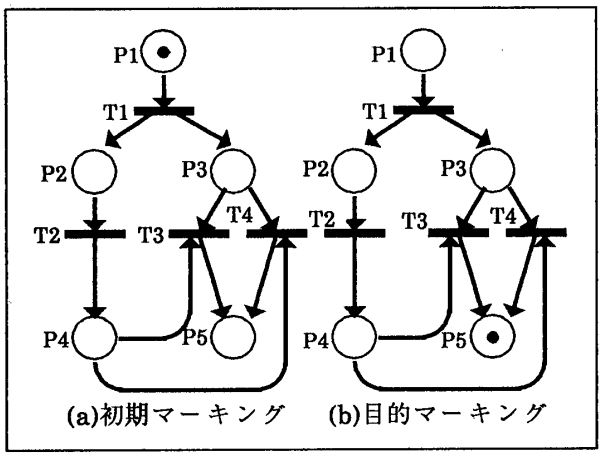

図2 ペトリネットのマーキング

となり、式(1)より発火ベクトルxを算出すると

$x=(1,1,1,0)$ または $x=(1,1,0,1)$

となる。

この結果から目的マーキングになるための発火ベクトルは2通り存在していてそれらは、T1, T2, T3が1回ずつ発火する場合とT1, T2, T4が1回ずつ発火する場合であることがわかる。これは、GHCプログラムに置き換えると目的通りに終了する場合は2通りあり節%(1), %(2), %(3)と節%(1), %(2), %(4)が実行される場合があることを示している。

次に節%(4)を誤って節%(5)の様に記述してしまった場合を考える。ただし、初期マーキングと目的マーキングは同じだが接続行列が異なってくる。

```
r(S,X) :- S=m | true.      %(5)
```

このプログラムをペトリネットに変換すると図3のようになる。

接続行列 A:

	P1	P2	P3	P4	P5
T1	-1	1	1	0	0
T2	0	-1	0	1	0
T3	0	0	-1	-1	1
T4	0	0	-1	-1	0

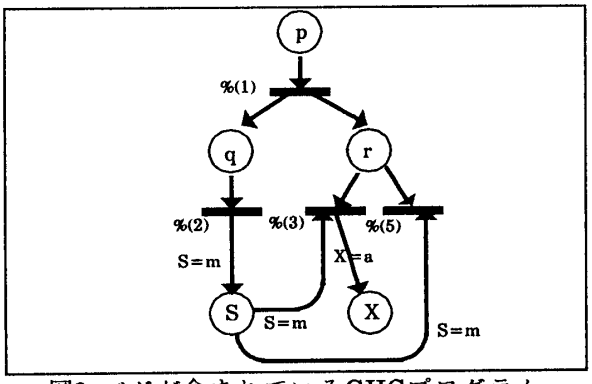

図3 バグが含まれているGHCプログラム

このような場合、先程と同様にして式(1)により発火ベクトルxを求めると

$x=(1,1,1,0)$

となる。

すなわち目的マーキングになるための発火ベクトルは1つあるが、発火しないトランジションT4が存在するので、T4のトランジションは冗長であるか、又はこのトランジションが発火することがないのはおかしいことがわかる。このことは、GHCプログラムに置き換えると目的通りに終了する動作系列が1つあるが、目的通りに終了するためには、節%(5)は実行することがないのでこれは冗長であるか又はバグであることがわかる。

このようにペトリネットの解析能力を利用したバグ検出の例を示したが、これらの解析を行うためには、ユーザにペトリネットを意識させないことは当然であるが、目的マーキング、すなわち最後に情報が残る節や変数を設計者に問い合わせる必要がある。プログラムだけでは、そのプログラムが設計者の意図通りのものであるかどうかはわからないからである。このような静的解析では、設計者とのインタラクションを行いながらバグを取り除く方法が必要である。またそうすることにより、設計者はプログラムの動作を確認していくことができる。

4.おわりに

GHCプログラムをペトリネットで表現することによりバグの検出を行えることを示した。さらにプログラムをペトリネットにモデル化することによりペトリネットの各種技術を利用することが考えられる。ペトリネットの研究は最近特に盛んに行われるようになってきているので、今後おおいに期待できる分野であろう。

ただし、すべてのGHCプログラムがモデル(b)によってペトリネットに変換できていないので(例えば、GHCにおけるパイプライン処理のように生成されたプロセス自体を区別する必要のあるものプログラム)、このようなプログラムの変換については検討を続けていく。

なお、本研究は第5世代コンピュータ・プロジェクトの一環として行っているものである。日頃ご指導を頂くICOT第1研究室・長谷川室長に深謝する。

[参考文献]

[1]安藤他:並列プログラムの動作評価に関する一検討, 情報処理学会第40回全国大会

[2]古川・溝口共編:並列論理型言語GHCとその応用, 共立出版(1987)

[3]T. Murata:Petri Nets Properties, Analysis and Applications, Proceedings of the IEEE, Vol.77, No.4, April 1989