1B-1

プログラム合成における入出力条件のチェックでのペトリネットの利用

恐神正博† 山西輝也† 魚崎勝司†
†福井工業大学経営情報学科

1 はじめに

1968 年に NATO の国際会議でソフトウェア危機が提唱されて以来，各方面においてソフトウェアの生産性を向上させるための技術が，研究されている．

プログラムの生産性を向上させる一つの手法であるプログラムの自動合成において，我々はその実験システム MAPP の開発を行ってきたが [1]，ここでは，プログラムの自動合成時に行っているモジュールの入出力条件のチェックに対し，ペトリネットを利用する方法について検討を行う．ペトリネットは数学的に解析が可能なモデル化ツールであり，離散事象システムの解析などに用いられている．

モジュールの入出力条件のチェックを，ペトリネットの可達性問題として考え，その状態方程式を解析することで，合成されたプログラムの動作可能性を数学的に証明することを検討する．

2 実験システム MAPP における入出力条件のチェックおよび自動補填

我々は，従来より Prolog を用いた自動合成システムの一つの構成法について，その実験システム MAPP(Module Aided Programming system by Prolog) を通し報告を行なってきている [1]．MAPP では Prolog の述語定義により，日本語等の自然言語から目的のプログラムへ，変換辞書（Prolog の述語定義による）を用いてプログラムを合成する．この中で，機能単位に準備してあるモジュールごとに入出力条件を持たせ，プログラムの合成時に条件が満たされているかどうかを適用条件としている．具体的には，図１のようなイメージとなる．

A method of using Petri nets for Input − Output conditions check during program generation
Masahiro OSOGAMI†, Teruya YAMANISHI†and Katsuji UOSAKI†
†Department of Management Information Science, Fukui University of Technology
3-6-1 Gakuen, Fukui, 910-8505, Japan
{osogami, yamanisi, uosaki}@fukui-ut.ac.jp

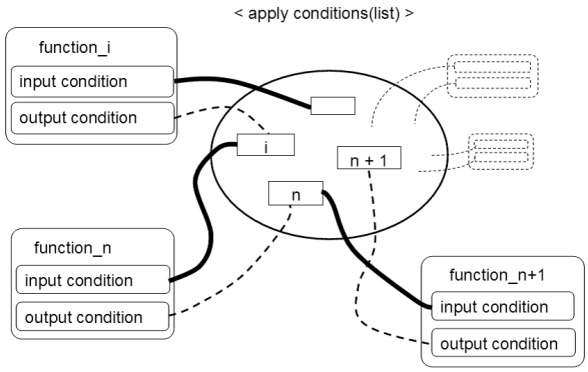

図１．入出力条件のチェック概念図

図１では，まず function_n がリンクされる際に入力条件を蓄えているリスト (apply conditions) に自身の入力条件（input condition）が存在しているかどうかを調べ，存在した場合自身の出力条件 (output condition) をリスト上に追加しリンクされる．次に function_n+1 がリンクされる際，同様に自身の入力条件が既に存在しているかどうかを調べる．例えば，この入力条件が function_n の出力条件であった場合，すでに apply conditions のリストに存在しているので function_n+1 がリンクできることになる．すなわち，finction_n+1 がリンクされるための前提条件が function_n であったことになる．

このように，処理仕様に記述されているすべてのモジュールに対しそれらの入力条件をチェックしていくが，１つでも条件を満たしていない場合，このリンクは失敗に終わってしまう．そこで，もし自分の入力条件が存在しなかった場合，MAPP ではモジュールを蓄えている辞書の中から必要な前提条件を持つモジュールを探索し，自動的に自分の前にリンクしていく自動補填の考えを新たに導入し行っている．

3 入出力条件のチェックにおけるペトリネットの利用

前節で述べた入出力条件のチェックに，ペトリネットを利用できれば，その可達性問題としてその状態方程式を解析し，合成されたプログラムの動作可能性を

数学的に証明することができるはずである．

3.1 ペトリネットの可達判定

ペトリネットにおける可達判定としては，接続行列を A，発火回数ベクトルを x，初期マーキング M_0 から目標マーキング M_d までの差を b とした場合，その状態方程式 $Ax = b$ の解 x が存在することが言えればよい．これには，接続行列 A が，$n \times m$ の行列で，$rank = r$ である場合，

$$A = \begin{matrix} \overset{m-r}{\leftrightarrow} & \overset{r}{\leftrightarrow} \\ \left[\begin{matrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{matrix}\right] & \begin{matrix} \updownarrow \ r \\ \updownarrow \ n-r \end{matrix} \end{matrix} \tag{1}$$

とした際の，A の部分行列 A_{11}, A_{12} と，$m-r$ 次の単位行列 I_μ を用いた，行列 B_f を求め，

$$B_f = [\ I_\mu \ :\ -A_{11}^T\ (A_{12}^T)^{-1}] \tag{2}$$

さらに，初期マーキング M_0 と目標マーキング M_d との差を, 式 (3) のように ΔM としたときの，

$$\Delta M = M_d - M_0. \tag{3}$$

式 (4) が成り立つことが示せればよい [2].

$$B_f \Delta M = 0 \tag{4}$$

3.2 入出力条件チェックの実例 [3]

今，図2の generated program について考えたとき，プログラムが実行可能であるには，modeled by Petri net のペトリネットが，○の Initial marking から●の Destination marking まで可達であることが言えればよい．これはすなわち，図2のペトリネットの状態方程式について解が存在することを証明できればよい．

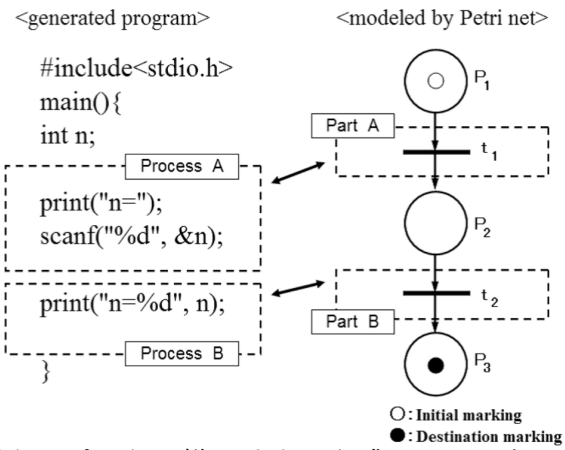

図2．プログラム例とそれをモデル化したペトリネット

そのためには式 (4) が言えればよい．ここで，接続行列 A は，

$$A = \left[\begin{matrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{matrix}\right] \tag{5}$$

となり，式 (1) より式 (5) の部分行列 A_{11}, A_{12} は，

$$A_{11} = \left[\begin{matrix} 1 & 0 \end{matrix}\right], \tag{6}$$

$$A_{12} = \left[\begin{matrix} -1 & 1 \\ -1 & 1 \end{matrix}\right] \tag{7}$$

であるため，式 (2) より，

$$B_f = \left[\begin{matrix} 1 & 1 & 1 \end{matrix}\right] \tag{8}$$

となる．また，初期マーキング (Initial marking) $M_0 = [\,1\ \ 0\ \ 0\,]^T$, と，目標マーキング (Destination marking) $M_d = [\,0\ \ 0\ \ 1\,]^T$, のマーキング差 ΔM は

$$\begin{aligned} \Delta M &= M_d - M_0 \\ &= \left[\begin{matrix} -1 & 0 & 1 \end{matrix}\right]^T \end{aligned} \tag{9}$$

であるので，

$$B_f \Delta M = 0$$

となり，式 (4) が言えたことから，M_d は M_0 から可達であることが証明できる．

4 まとめ

プログラム合成におけるモジュールの入出力条件のチェックにペトリネットを利用する方法について検討を行った．プログラムを正しくペトリネットに置き換える方法，また，規模の大きなプログラムにおける適用など課題は多いものの，正しさの証明を数学的に行える意義は大きい．今回は順次処理についてのみ検討を行ったが，今後については分岐・反復などさらに複雑な構造に対しても扱っていけるよう検討して行く必要がある．

参考文献

[1] 恐神, 西田:“プロログによるモジュールの検索とプログラムの合成”, 福井工業大学研究紀要第 23 号, pp.313-320, 1993.

[2] 村田: ペトリネットの解析と応用, 近代科学社, 1992.

[3] Osogami, Yamanishi and Uosaki:“Input-Output Conditions for Automatic Program Generation Using Petri Nets”, Proc. of The KES2011, in Germany, Part I, LNAI 6881, pp. 296-305, 2011.