A-004

一般ペトリネットの可達グラフ生成途中で削除可能な状態の反復一致性に基づく推定法

Estimation Method of Removable States during Reachable Graph Generation based on Iterative Consistency in General Petri Nets

藤森 浩平 †　張江 洋次朗 ††　和崎 克己 ††
Kohei Fujimori　Yojiro Harie　Katsumi Wasaki

1　あらまし

ペトリネット (Petri net) とは, 離散事象システムの振る舞いを表す数学モデルであり [1][2], その解析を行うことでシステムの様々な性質を知ることができる. 本学で開発されたペトリネット設計ツール HiPS (Hierarchical Petri net Simulator) がある [3]. HiPS は様々なペトリネット解析ツールを備えており, その一つとして状態空間生成器がある. 大規模なネットに対する状態空間生成時には, 状態数の爆発的増加に伴ってメモリ使用量が増大する. このメモリの大半は生成した状態値 (マーキングベクトル) の保持に使用される. 生成済みの状態の中で, 以降の状態空間生成に用いない状態値をメモリ上から削除していくことでメモリ使用量を抑えられる. 本研究では, 生成中に削除可能な状態を推定・削除することでメモリ使用量を抑える手法を提案する.

2　接続行列と状態方程式

n 個のトランジションと m 個のプレースを持ったペトリネット N に対して, 接続行列 $A = [a_{ij}]$ は $n \times m$ の整数行列である. a_{ij} はトランジション i が 1 回発火した時の, プレース j のトークン数変化量を表す.

以下ではマーキング M_k は, $m \times 1$ の列ベクトルとして表す. M_k の j 番目の成分は, ある発火系列における k 番目の発火直後のプレース j 内のトークン数を表す. k 番目の発火ベクトル u_k は, $n \times 1$ 列ベクトルであり, k 番目の発火においてトランジション i が発火することを表すため i 番目の成分は 1 でその他の成分は 0 である. この時, ペトリネットの状態方程式は次のように定義される.

$$M_k = M_{k-1} + A^T u_k, k = 1, 2, ... \quad (1)$$

3　可達性と可達必要条件, 不変量

3.1　可達性

マーキング M_0 をマーキング M_n へ変換する発火系列が存在する時, マーキング M_n は M_0 から可達であるという. 発火系列は $\sigma = M_0 t_1 M_1 t_2 M_2 ... t_n M_n$ と表す. M_n が σ により M_0 から可達の時, $M_0[\sigma > M_n$ と表す. ペトリネット (N, M_0) について, 初期マーキング M_0 から発火可能なトランジションを発火させて新しい可達マーキングを順次求めていくことにより, 可達マーキングの木 (またはグラフ) 表現を得る. ペトリネットの可達グラフは, ラベルづけされた有向グラフ $G = (V, E)$ である. ノードの集合 V は, 可達木内の全てのことなったラベル (マーキング) を持つノードの集合であり, アークの集合 E は $M_i[t_k > M_j (M_i, M_j \in V)$ であるような単一トランジションの発火を表す.

3.2　無制限ネットにおける可達必要条件

目標とするマーキング M_d が, 発火系列 $\{u_1, u_2, ..., u_d\}$ を通して初期マーキング M_0 から可達であるとする. $i = 1, 2, ..., d$ に対する状態方程式 (1) を求めて和をとることで (2) を得る.

$$M_d = M_0 + A^T \sum_{k=1}^{d} u_k \quad (2)$$

これは, 次のように書き直すことができる.

$$A^T x = \Delta M$$

ここで, $\Delta M = M_d - M_0$ 及び $x = \sum_{k=1}^{d} u_k$ である. この x は非負整数の $n \times 1$ 列ベクトルであり, 発火回数ベクトルと呼ばれる. x の i 番目の成分は, M_0 から M_d へ変わるためにトランジション i が発火しなければならない回数を表す.

ΔM が $Ay = 0$ の各解 y に対して直交するという条件は, B_f 行列 [1] を用いた次の条件と同値である.

$$B_f \Delta M = 0 \quad (3)$$

ペトリネット (N, M_0) において, M_d が M_0 から可達であれば, 対応する発火回数ベクトルが存在し, 式 (3) が成り立つ.

3.3　反復一致性, T インバリアント

ペトリネット N は, あるマーキング M_0 と, 全ての (いくつかの) トランジションが少なくとも 1 回発火系列 σ 内に生起するような M_0 から始まり M_0 へ戻る発火系列 σ とが存在すれば, (準) 反復一致的であると呼ばれる. ペトリネット N は, $A^T x = 0, x \neq 0$ であるような正 (非負) 整数の n 次ベクトルが存在すれば, かつその時に限って, (準) 反復一致的である. 整数の n 次ベクトル x は, $A^T x = 0$ の時, T インバリアントと呼ばれる. n 次ベクトル $x \geqq 0$ は, マーキング M_0 と, 発火回数ベクトル $\overline{\sigma}$ が等しいような M_0 から始まり M_0 へ戻る発火系列 σ とが存在すれば, かつその時に限って, T-インバリアントである. T インバリアント $x \geqq 0$ の非ゼロ成分に対応するトランジションの集合は, インバリアントの台集合 (support of an invariant) と呼ばれ, $\|x\|$ で表される.

† 信州大学大学院総合理工学研究科, Graduate School of Science and Technology, Shinshu University
†† 信州大学大学院総合工学系研究科, Interdisciplinaty Graduate School of Science and Technology, Shinshu University

4　削除可能状態判定を用いた提案手法

ペトリネットにおける状態空間は有界な可達グラフとして与えられる．有界な可達グラフ生成アルゴリズムが提案されていて，HiPS の状態空間生成器に実装されている [4]．

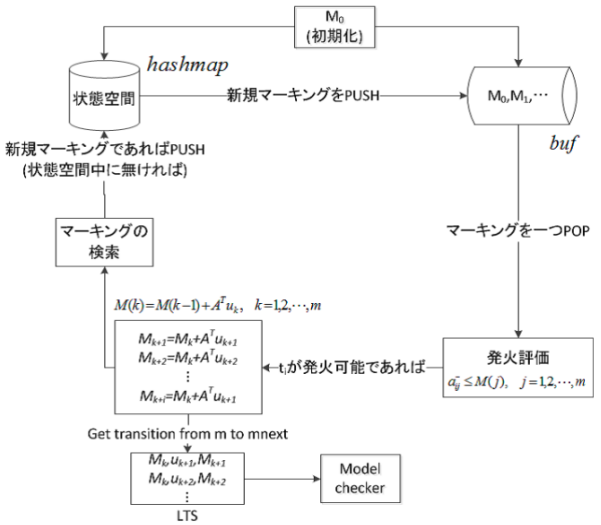

図 1　可達グラフ生成アルゴリズム動作フロー

従来法の状態空間生成器では，生成した全ての状態値 (マーキングベクトル) をハッシュマップに登録して保持している (図 1)．大規模なペトリネットに対して状態空間生成を実行すると，状態数の爆発的増加に伴ってメモリ使用量が増大する．生成済みの状態値の中で，以降の状態空間プロセスで使用されない状態値をメモリ上から削除していくことで状態空間生成時のメモリ使用量を抑える手法を提案する．

削除可能な状態の判定・削除プロセスを追加した可達グラフ生成アルゴリズムを Algorithm 1 に示す．

可達グラフにおいて，状態 M の可達な次段マーキングが全て生成済みであり，かつ状態 M へ戻る可達な経路が全て生成済みであるとき，状態 M は以降の状態空間生成で使用されないため削除可能であるとする．

可達グラフ生成プロセス中で次段マーキングの生成を終えた状態 M に対して，M へ戻る可達な経路が全て生成済みであるか判定することで，状態 M は削除可能かどうか判定できる．$M_p[t > M$ を満たす可達な状態 M_p とトランジション t が存在する時，状態 M へ戻る可達な経路が存在する．

提案する手法では，待ち行列 buf から発火評価待ちの状態 M を一つずつ取り出して発火評価・次段マーキング生成を行う．次段マーキングの生成を終えた状態 M でトランジション t を逆発火させて得られた状態 M_p に対して，式 (3) を用いて可達判定を行うことで状態 M へ戻る経路を列挙する．列挙した可達な経路の中で未生成の経路を記録しておく．これらの未生成経路が以降の状態空間生成で全て生成された時点で状態 M は削除可能となるので，メモリ上から削除する．この方法で生成プロセス中に削除可能になった状態から削除していくことで実行時のメモリ使用量を抑えられる．

5　まとめと今後の課題

提案手法により，可達グラフ生成中に削除可能な状態をメモリ上から削除してグラフ生成中のメモリ使用量を抑えられた．しかし，状態 M へ戻る可達な経路を列挙するとき，全てのトランジションについて逆発火させて式 (3) による判定を行うため実行時間が増加する．実行時間を短縮するためには，可達な経路の候補を絞り込む必要がある．経路の候補を絞り込むためには，対象ネットが反復一致性を有する時，T インバリアント発火系列が存在することを用いる方法が考えられる．T インバリアント x で，トランジション $t \in \|x\|$ が状態 M で発火可能な時，$\|x\|$ による発火系列で M へ戻るものが存在すると推定できる．ネットの反復一致性を用いて経路の候補を絞り込み，式 (3) を適用する対象を減らして削除可能判定にかかる時間を短縮できる．

Algorithm 1 削除可能状態判定つき可達グラフ生成アルゴリズム

```
1:  buf ← M_0
2:  map ← (M_0, stateNumber ← stateNumber + 1)
3:  while buf is not empty do
4:      m ← buf.pop()
5:      for i in 0..row do
6:          if t_i is not fireble then
7:              continue
8:          end if
9:          m_next ← m + A^T u_i
10:         if m_next is new marking then
11:             map ← (m_next, stateNumber)
12:             stateNumber ← stateNumber + 1
13:             buf ← m_next
14:         else
15:             m_next.count ← m_next.count − 1
16:             if m_next.count = 0 then
17:                 map.erase(m_next)
18:             end if
19:         end if
20:     end for
21:     for i in 0..row do
22:         m_prev ← m − A^T u_i
23:         ΔM ← m_prev − M_0
24:         if B_f ΔM = 0 then
25:             m.count ← m.count + 1
26:         end if
27:     end for
28:     if m.count = 0 then
29:         map.erase(m)
30:     end if
31: end while
```

参考文献

[1] T. Murata : “Petri Nets: Properties, Analysis and Applications”, Proc. of the IEEE, 77(4), 1989.

[2] J.L. ピーターソン: “ペトリネット入門 情報システムのモデル化”, 共立出版, 1984.

[3] HiPS : Hierarchical Petri net Simulator, https://sourceforge.net/projects/hips-tools/

[4] 太田淳也, 和崎克己: “ペトリネット援用ツールを用いたモデル設計とポスト検証ツール向け状態空間生成アルゴリズム”, 第 12 回情報科学技術フォーラム, FIT2013, pp. 171-174, 2013.