World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:3, 2009

A Technique for Reachability Graph Generation for the Petri Net Models of Parallel Processes

Farooq Ahmad, Hejiao Huang, and Xiaolong Wang

***Abstract*—**Reachability graph (RG) generation suffers from the problem of exponential space and time complexity. To alleviate the more critical problem of time complexity, this paper presents the new approach for RG generation for the Petri net (PN) models of parallel processes. Independent RGs for each parallel process in the PN structure are generated in parallel and cross-product of these RGs turns into the exhaustive state space from which the RG of given parallel system is determined. The complexity analysis of the presented algorithm illuminates significant decrease in the time complexity cost of RG generation. The proposed technique is applicable to parallel programs having multiple threads with the synchronization problem.

***Keywords*—**Parallel processes, Petri net, reachability graph, time complexity.

I. Introduction

THE enumeration of state space requires the construction of reachability graph (RG) [1, 2] and it is important and fundamental approach for computer-aided verification and behavioral analysis of the Petri net (PN) models of parallel discrete systems. Further, it provides the complete and detailed information about the dynamic behavior of the system and its global states represent the combined behavior of all parallel components in the parallel discrete systems.

Parallel events are modeled by the interleaving of transitions in the PN model and RG contains all possible partial ordering of such transitions and consequently the RG confronts with the state-explosion problem [3]. The state-explosion problem is directly related to the exponential time and space complexity involved in generating the state-space of parallel systems.

Several methods have been suggested to tackle the state-explosion problem, which includes structural analysis methods [1, 4, 5] and limited unfolding approaches [6, 7, 8]. Faced to the state-explosion problem, the reduction and refinement methods [1, 9] have been developed to reduce the complexity of the initial net. However, it is not always possible to transform a complex and larger net into simpler and smaller one. An algorithm has been proposed to obtain the finite representation of the RG by using marking abstraction process [10].To alleviate the computational complexity involved in generating the RG for parallel systems, partial order reduction [11, 12, 13, 14] has been suggested. Partial order reduction to cure the complexity issues of RG generation is the range of methods for constructing the reduced state space, which includes the stubborn set method [15, 16, 17], maximal concurrent simulation [18, 19] and symmetry method [20, 21, 22, 23]. However, partial order reduction methods have limited applicability due to their utility for specific analysis questions [3]. Another approach to manage the state-explosion problem is the compression technique using binary decision diagram (BDD) [24, 25, 26]. Gaining in memory often results in increase in the temporal complexity [19] because all known algorithms for analysis tasks in relatively small memory are extremely slow [3].

To cope with the critical problem of time complexity, this paper presents a new technique for RG generation for parallel systems. The proposed technique firstly generates independent RGs for each individual process in parallel, and then considers the cross-product of these RGs. Finally, resultant RG is the subset of exhaustive state space generated by the cross-product. The complexity analysis of the presented algorithm illuminates about a significant decrease in the time complexity for RG generation when it is compared with the time complexity of the classical method [1, 2]. In addition, the generation of independent RGs may be distributed over available processors, which increases the practical utility of the proposed technique.

The paper is organized as follows. Section II introduces the related terminology. Section III presents the algorithm for parallel RG generation. The complexity analysis of the proposed algorithm is presented in Section IV. Some concluding remarks are presented in Section V.

II. Definitions and Concepts

In this section, some basic definitions and notations of ordinary (for the sake of simplicity) PN are described. The related terminology and notations are taken from [1, 2].

Definition 1: (*Petri net*) A Petri net *PN,* is a five tuple, $PN = (P, T, I, O, M_0)$. Where, $P = \{p_1, p_2, \ldots, p_{|P|}\}$ is a finite set of places, $|P| > 0$; $T = \{t_1, t_2, \ldots, t_{|T|}\}$ is a finite set of transitions, $|T| > 0$; $I : T \to P$ is the input function, which is a mapping from transitions to the set of places and it indicates

Farooq Ahmad is with the Computer science Department, Harbin Institute of Technology Shenzhen Graduate School, China (e-mail: farooq190@gmail.com).

Hejiao Huang is with the Computer science Department, Harbin Institute of Technology Shenzhen Graduate School, China (corresponding author's e-mail: hjhuang.hitsz@hotmail.com).

Xiaolong Wang is with the Computer science Department, Harbin Institute of Technology Shenzhen Graduate School, China (e-mail: wangxl@insun.hit.edu.cn).

the input places of transitions; $O: T \to P$ is the output function, which is a mapping from transitions to the set of places and it indicates the output places of transitions, $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$.

Let $I(t_j)$ represents the set of *input places* of transition $t_j \in T$ and $p_i \in P$ is an input place of a transition t_j if $p_i \in I(t_j)$; $O(t_j)$ represents the set of *output places*, then p_i is an output place of t_j if $p_i \in O(t_j)$. For example, a PN structure given in Fig. 1 has $P = \{p_1, p_2, p_3, p_4\}$ and $T = \{t_1, t_2, t_3, t_4, t_5\}$ with $I(t_1) = \{p_1\}$, $I(t_2) = \{p_1\}$, $I(t_3) = \{p_2\}$, $I(t_4) = \{p_3\}$, $I(t_5) = \{p_4\}$ and $O(t_1) = \{p_2\}$, $O(t_2) = \{p_3\}$, $O(t_3) = \{p_4\}$, $O(t_4) = \{p_4\}$, $O(t_5) = \{p_1\}$.

The input and output functions can be extended to map the set of places P into the set of transitions T such as $I: P \to T$ and $O: P \to T$. Then, set $I(p_i)$ represents the set of *input transitions* of place $p_i \in P$ and set $O(p_i)$ represents the set of *output transitions* of place $p_i \in P$. Therefore, $I(p_1) = \{t_5\}$, $I(p_2) = \{t_1\}$, $I(p_3) = \{t_2\}$, $I(p_4) = \{t_3, t_4\}$ are sets of input transitions and $O(p_1) = \{t_1, t_2\}$, $O(p_2) = \{t_3\}$, $O(p_3) = \{t_4\}$, $O(p_4) = \{t_5\}$ are sets of output transitions of all the places of a structure given in Fig. 1.

The incoming arc from p_i to t_j is represented by $(p_i, I(t_j))$ and outgoing arc from t_j to p_i be $(p_i, O(t_j))$. Similarly, $(t_j, I(p_i))$ represents the incoming arc from t_j to p_i as $t_j \in I(p_i)$ and arc $(t_j, O(p_i))$ represents outgoing arc from p_i to t_j as $t_j \in O(p_i)$, when the set of places maps into the set of transitions; $\forall t_j \in T, \forall p_i \in P$.

The structure of a PN is defined by the set of places, set of transitions, input function and output function. A PN structure without M_0 is denoted by $N = (P, T, I, O)$. A PN structure N is said to be strongly connected if and only if every node $x_i \in P \cup T$ is reachable from every other node $x_j \in P \cup T$ by a directed path. A PN structure N is said to be *self-loop-free* or *pure* if and only if $\forall t_j \in T$, $I(t_j) \cap O(t_j) = \varnothing$ i.e. no place can be both an input and an output of the same transition.

A marking is a function $M: P \to \mathbb{N}$ (non-negative integers) and initial marking is denoted by M_0. A PN with given initial marking is denoted by (N, M_0). The set of all reachable markings from M_0 is denoted by $R(M_0)$ which is a definite set of markings of PN such that, if $M_k \in R(M_0)$ and $M_k \xrightarrow{t_j} M_k'$ for some $t_j \in T$, then $M_k' \in R(M_0)$.

Definition 2: (*Firing rule*) The firing rule identifies the transition enabling and the change of marking. Let $M(p_i)$ be the number of tokens in place p_i, then for $\forall t_j \in T$; t_j is enabled under marking M if and only if $\forall p_i \in I(t_j): M(p_i) \geq 1$. The change of marking M to M' by firing the enabled transition t_j is denoted by $M \xrightarrow{t_j} M'$ and defined for each place $p_i \in P$ by $M'(p_i) = \begin{cases} M(p_i) - 1 \text{ for every } p_i \in I(t_j) \\ M(p_i) + 1 \text{ for every } p_i \in O(t_j) \\ M(p_i) \text{ otherwise.} \end{cases}$

Definition 3: (*reachability graph*) the RG of the PN model is a directed graph $G = (V, L, E, v_0)$, V is the set of vertices and each vertex $v_k \in V$ represents the reachable marking $M_k \in R(M_0)$; L is a set of labels where each $l_i \in L$ directly corresponds to fireable transitions in any reachable marking of PN; E is the set of edges such that $E = \{(v_k, l_i, v_{k+1}) \in E \setminus v_k, v_{k+1} \in V, l_i \in L\}$ and for each $e \in E$, v_k is a initial vertex of e, v_{k+1} is a terminal vertex of e and l_i is a label of the edge. Basically, set of labeled directed edges E is a firing relation such as $E \subset V \times L \times V$. Therefore, each $e \in E$ represents directed edge from given marking M_k to other reachable marking M_{k+1} and labeled by the fired transition t_i at given marking; $v_0 = M_0$ is the initial vertex.

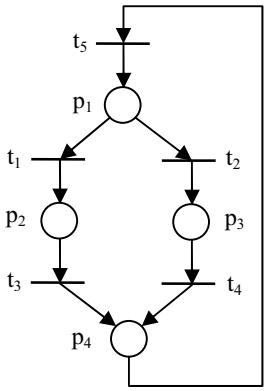

Fig. 1 An example of PN model

III. Parallel Generation of Reachability Graph

The PN modeling of parallel processes depicts the interleaving of transitions (events), which can be observed from Fig. 2(a), and state-space explosion is directly related to the exploration of all possible interleavings of parallel events in RG. For instance, the execution of k parallel events (independent transitions) is investigated by exploring all $k!$ interleavings of these transitions (events) and states in RG.

The proposed technique initiates by extracting the independent parallel processes from the PN model of a concurrent system. Every individual parallel process is the sequential representation of events as shown in Fig. 2(c) and consequently the RG is the sequence of states. The technique permits the concurrent execution of independent parallel processes to aim at time complexity involved in RG generation.

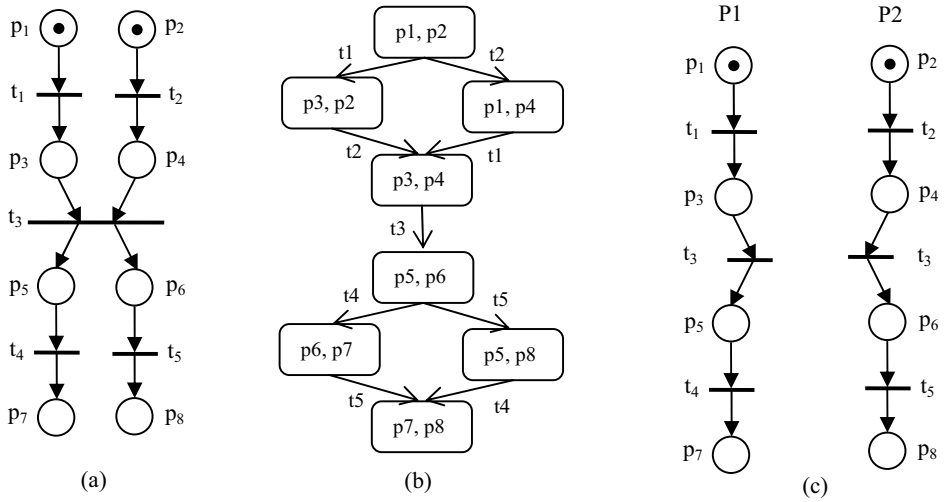

Fig. 2 (a) PN model with parallel processes, (b) its classical RG generation and (c) independent parallel processes obtained from Fig. 2(a)

The cross-product of the individual RGs of parallel processes generates the exhaustive state-space of the whole PN model of the system. The RG is a sub-graph of this cross product and it is obtained by discarding the invalid states from the exhaustive state-space.

The interaction of the parallel processes often incorporated with the synchronization structure in its PN model. Fig. 2(a) shows the PN model with interacting parallel processes, where transition t_3 is the synchronic point of the PN structure. The synchronizing transition in PN model becomes sequential transition in each independent parallel process.

However, synchronizing transition does not permit the all possible partial ordering of transition executions in RG. The transitions before and after the synchronizing transition have the causal relationship. Further, the transition after the synchronizing transition can not be fired before the execution of synchronizing transition. Synchronizing transition would be in every independent process, by tracking the synchronizing transition, invalid states can be eliminated from exhaustive state space and remaining states are global states of RG.

From the above discussion, an algorithm can be developed to reduce the time complexity of RG generation with concurrent executions of parallel processes.

Algorithm for parallel generation of RG:

1. Input: N Parallel processes; P_i, $i = 1, 2, \ldots, N$.
2. $\forall P_i$: generate R_i, where R_i is the RG of P_i.
3. Tag each state by the fired transition obtaining that state in R_i, $\forall R_i$.
4. Do, $R_1 \times R_2 \ldots \times R_N$ such that $(S_{1j}, S_{2j}, \ldots S_{ij} \ldots, S_{Nj}) \in R_1 \times R_2 \ldots \times R_N$. Where, S_{ij} is the *jth* state in R_i and $RG \subseteq R_1 \times R_2 \ldots \times R_N$.
   Now, each S_{ij} is a local state of the global state $(S_{1j}, S_{2j}, \ldots S_{ij} \ldots, S_{Nj})$.
5. For each global state $(S_{1j}, S_{2j}, \ldots S_{ij} \ldots, S_{Nj})$, do; if $\exists t_j$ as synchronizing transition, discard the global states if any pair of local states S_{ij}, S_{jk} has the tag of transition t_j, before or after the transition t_j in parallel process.
6. Output: RG of PN model.

To formalize the RG for the PN model of a system having parallel processes with the implementation of the algorithm given above, the algorithm proceeds as follow:

- For each parallel process P_i, RG R_i is generated, which corresponds to sequence of places as states and transitions as edges in that individual process. For example, for P_1 which is shown in Fig. 2(c), $p_1 \xrightarrow{t_1} p_3 \xrightarrow{t_3} p_5 \xrightarrow{t_4} p_7$ is R_1. The RG generation for all P_is is carried out in parallel.
- Each state in R_i and $\forall R_i$ is tagged by the transition, through which it is reached, e.g. the state p_3 has a tag t_1 in the example given above.
- The cross product of R_is (i.e. $R_1 \times R_2 \ldots \times R_N$) is performed to construct the exhaustive state space containing the global states of the form $(S_{1j}, S_{2j}, \ldots S_{ij} \ldots, S_{Nj})$. For example, Table I shows the exhaustive state space for PN given in Fig. 2(a). Moreover, each element of the global state is the state for any individual process as well as local state for that global state.
- RG is obtained by discarding the invalid states in the exhaustive state space. Therefore, the set of reachable states in the resultant RG is the subset of exhaustive state space. Invalid states in the presence of synchronizing transition in PN structure are identified with the assistance of the tags of local states. Every global state is discarded from exhaustive state space, if any pair of local states has the tag of synchronizing transition, before or after the synchronizing transition in PN structure.

TABLE I
EXHAUSTIVE STATE SPACE DUE TO $R_1 \times R_2$

			Tags	m_{02}	t_2	t_3	t_5
			P2				
			States	p_2	p_4	p_6	p_8
	Tags	States					
	m_{01}	p_1		(p_1, p_2)	(p_1, p_4)	~~(p_1, p_6)~~	~~(p_1, p_8)~~
P1	t_1	p_3		(p_3, p_2)	(p_3, p_4)	~~(p_3, p_6)~~	~~(p_3, p_8)~~
	t_3	p_5		~~(p_5, p_2)~~	~~(p_5, p_4)~~	(p_5, p_6)	(p_5, p_8)
	t_4	p_7		~~(p_7, p_2)~~	~~(p_7, p_4)~~	(p_7, p_6)	(p_7, p_8)

TABLE II
RG OF FIG. 2(a) OBTAINED BY DISCARDING THE INVALID GLOBAL STATES FROM TABLE I

(p_1, p_2)	(p_1, p_4)		
(p_3, p_2)	(p_3, p_4)		
		(p_5, p_6)	(p_5, p_8)
		(p_7, p_6)	(p_7, p_8)

For example, local states p_5, p_2 of (p_5, p_2) has the respective tags t_3, m_{02} in Table I, where m_{01} and m_{02} are initial states of individual parallel processes P1 and P2 respectively, in Fig. 2(c). However, synchronizing transition t_3 can not be fired at initial state of any process. Therefore, global state (p_5, p_2) is invalid and discarded. In the same manner, Table II shows the RG for PN in Fig. 2(a) by discarding all the invalid states. Global states in Table 2 are compared to the reachable states in RG obtained by classical method [1, 2] of Fig. 2(b) and found exactly same.

IV. Complexity Analysis of Algorithm

The exact analytical evaluation of computational complexity of the algorithm for generating the RG is not a trivial task even for a restricted structure of the PN model of parallel discrete systems. The effectiveness of algorithm can be proved by comparing the reduction in complexity of the new algorithm with respect to the time complexity involved in the classical method of RG generation. The asymptotic complexity (the worst case complexity) is given below, which allows estimating the maximal duration of the computation of RG.

Time complexity for classical method: Classical method for generating the RG considers the space complexity of $O(|P|^N)$ for PN structure having N parallel branches, where each branch has $|P|$ places. Now in the worst case, each reachable state in RG may be connected to every other state. Therefore, the temporal complexity, which is proportional to the number of edges is bounded by $O(|P|^N)^2$. Hence the temporal complexity, in worst case, involved in generating the RG is quadratic with respect the number of states in RG.

Time complexity for proposed algorithm: For calculating the time complexity of RG with the implementation of proposed algorithm, the cost of each step of the algorithm is calculated as follows:

- First step of the algorithm, i.e. construction of R_i $i = 1, \ldots, N$, takes time of $O(|P|.N)$ and tagging takes the time of $O(|T|.N)$, where $|T|$ represents the number of transitions in each individual process.
- In second step of the algorithm, the cross-product takes the time of $O(|P|.N)$.
- The third step considers the elimination of invalid states from exhaustive state space of size $O(|P|^N)$. Therefore, in the worst case, the total cost for discarding the invalid states is $O(|T|.N.|P|^N)$.
- Finally, the asymptotic total cost of algorithm is $O(|P|.N + |T|.N + |P|.N + |T|.N.|P|^N) \sim O(|T|.N.|P|^N)$. Now, each step is independent, by considering the parallel execution of each step on K available processors, the time complexity is reduced to $O((|T|.N.|P|^N)/K)$.

Hence the asymptotic computational complexity of the proposed algorithm with single available processor is $O(|T|.N.|P|^N)$. It is linear with respect to the number of states $|P|^N$ and total number of transitions, which is equal to $|T|.N$, in the PN model of parallel processes.

V. Conclusions

A new technique for generating the RG is proposed, which leads to the significant reduction in time complexity. In

addition, sequential execution of the algorithm itself shows significant reduction in time complexity, where total cost of the algorithm is $O(|T|.N.|P|^N)$, which is linear with respect to the number of states and the number of transitions. The proposed technique has practical application to the behavioral and reachability analysis of parallel programs having multiple parallel threads with the synchronization problem.

ACKNOWLEDGMENT

This work was supported in part by National High-Tech R&D Program (863 Program) under grant No. 2007AA01Z194 and National Natural Science Foundation of China with Grant No. 10701030.

REFERENCES

[1] T. Murata, "Petri nets: properties, analysis and application," *In Proceedings of IEEE,* vol. 77, No. 4, pp. 541-580, 1989.
[2] J. L. Peterson, "Petri Net Theory and the Modeling of Systems", Prentice-Hall: Englewood Cliffs, NJ, 1981.
[3] A. Valmari, "The state explosion problem", In: W. Reisig, G. Rozenberg (Eds.), Lectures on Petri nets I: Basic Models, LNCS 1491, Springer-Verlag, pp. 429-528, 1998.
[4] L. Recalde, "Structural methods for the design and analysis of concurrent systems modeled with Place/Transition nets", PhD Thesis, DIIS, University of Zaragoza, 1998.
[5] H. Huang, "Enhancing the property preserving Petri net process algebra for component-based system design (with application to designing multi-agent systems and manufacturing systems)", PhD thesis of City University of Hong Kong, 2004.
[6] K. L. McMillan, "Using unfolding to avoid the state explosion problem in the verification of asynchronous circuits", LNCS 663, Springer-Verlag, pp. 164-177, 1992.
[7] M. Heiner, "Petri net based system analysis without state explosion", In Proceedings of High Performance Computing, Boston, pp. 394-403, 1998.
[8] C. Girault and R. Valk, "Petri Net for System Engineering: A Guide to Modeling, Verification, and Application", Springer-Verlag, Berlin Heidelberg, 2003.
[9] H. Huang, T. Y. Cheung and W. M. Mak, "Structure and behavior preservation by Petri-net-based refinements in system design", *Theoretical Computer Science,* vol. 328, pp. 245-269, 2004.
[10] X. Ye, J. Zhou, and X. Song, "On reachability graphs of Petri nets", *Computers and Electrical Engineering,* vol. 29, pp. 263-272, 2003.
[11] P. Godefroid, "Partial-Order Method for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem", LNCS 1032, Springer-Verlag, New York, USA, 1996.
[12] M. Ceska, L. Hasa, and T. Vojnar, "Partial-order reduction in model checking object-oriented Petri nets", Proc. EUROCAST 2003, LNCS 2809, Springer, p.265-278, 2003.
[13] C. Flanagan, and P. Godefroid, "Dynamic partial-order reduction for model checking software", Proc. 32nd ACM symposium on POPL'05, pp. 110-121, 2005.
[14] X. Wang, and M. Kwiatkowska, "Compositional state space reduction using untangled actions", Electronic *Notes in Theoretical Computer Science,* vol. 175**,** pp. 27-46, 2007.
[15] A. Valmari, "State of the art report: Stubborn sets", *Petri net Newsletter,* vol. 46, pp. 6-14, 1994.
[16] K. Schmidt, "Stubborn set for model checking the EF/AG fragment of CTL", *Fundamenta Informaticae,* vol. 43(1-4), pp. 331-341, 2000.
[17] L. M. Kristensen, K. Schmidt and A. Valmari, "Question-guided stubborn set methods for state properties", *Formal Methods in System Design*, vol. 29, No. 3, pp. 215-251, 2006.
[18] R. Janicki, and M. Koutny, "Optimal simulations, nets and reachability graphs", In: Rozenberg, G. (Ed.), Advances in Petri Nets: LNCS 524, Springer-Verlag, Berlin, pp. 205-226, 1991.
[19] A. Karatkevich, "Dynamic Analysis of Petri Net-based Discrete Systems", LNCIS, vol. 358, Springer-Verlag, Berlin, 2007.
[20] E. M. Clarke, O. Grumberg, M. Minea and D. A. Peled, "State space reduction using partial order techniques", *STTT,* vol. 2, No. 3, pp. 279-287, 1999.
[21] K. Schmidt, "How to calculate symmetries of Petri nets", *Acta Informatica,* vol. 36, No. 7, pp. 545-590, 2000.
[22] T. Junttila, "New canonical representative marking algorithms for place/transition nets", In: J. Cortadella, W. Reisig (Eds.), ICATPN 2004, LNCS 3099, Springer, Heidelberg, pp. 258-277, 2004.
[23] L. Capra, "Colored Petri nets state-space reduction via symbolic execution", Proc. IEEE International Symposium SYNASC'05, page 231, 2005.
[24] K. Bilinski, "Application of Petri Nets in Parallel Controllers Design", PhD thesis, University of Bristol, 1996.
[25] G. Labiak, "Symbolic state exploration of UML state charts for hardware description", In: A. Adamski, A. Karatkevich, M. Wegrzyn (Eds.), Design of Embedded Control Systems, Springer, NY, pp. 73-83, 2005.
[26] P. Miczulski, "Calculating state space of hierarchical Petri nets using BDD", In: A. Adamski, A. Karatkevich, M. Wegrzyn (Eds.), Design of Embedded Control Systems, Springer, NY, pp. 85-94, 2005.