Workflow Mining: Discovering Loops in Process Models

V.R. Kavitha
VCET, Madurai, India

R. Kavitha
VCET, Madurai, India

Dr. N. Suresh Kumar
VCET, Madurai, India

ABSTRACT
The growing complexity of processes in many organizations stimulates the adoption of business process management techniques. Process models typically lie at the basis of these techniques and generally, the assumption is made that the operational business processes as they are taking place in practice confirm to these models. Technologies such as workflow management, Enterprise Resource Planning (ERP) etc., typically focus on the realization of it [1], [2], [8]. The current research in process mining still has problems in mining some common constructs in workflow models. Among these constructs are loops. Because loops are the major concern for boundedness of any process model. This paper discusses about representing workflow model using Petri Nets and a method to identify loops. For identifying loops topological sorting is used. In the literature process logs are used to identify short loops of length two but the proposed algorithm identify loops of any length.

Keywords

Workflow, Petri Net, Topological sort

1. INTRODUCTION
During the last decade, workflow management concepts and technology have been applied in many enterprise information systems. Workflow management systems such as Staffware, IBM MQSeries, COSA, etc., offer generic modeling and enactment capabilities for structured business processes. By making graphical process definitions, i.e., models describing the life-cycle of a typical case in isolation, one can configure these systems to support business processes[4] , [5], [9], [10]. Besides pure workflow management systems, many other software systems have adopted workflow technology. Despite its promise, many problems are encountered when applying workflow technology. One of the problems is that these systems require a workflow design, i.e., a designer has to construct a detailed model accurately describing the routing of work. Modeling a workflow is far from trivial; it requires deep knowledge of the workflow language and lengthy discussions with the workers and management involved.

In this work Petri Net is selected for representing workflow models. The main idea behind selecting Petri Net is checking boundedness property will be easy when the workflow is represented as Petri Net. Since Petri Net is a directed graph the proposed method uses topological sort to identify loops present in a workflow. In the literature process logs are used to identify short loops of length two but using the proposed algorithm loops of any length is identified that is the major advantage of the present algorithm.

2. PRELIMINARIES

This section introduces the techniques used in the remainder of this paper. First, we introduce standard Petri Net notations, then firing rules, Boundedness and Safeness.

2.1. Petri Nets

Petri net is a graphical and mathematical modeling tool applicable to many systems. They are a promising tool for describing and studying the information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic. As a graphical tool, they can be used as visual communication aids. As a mathematical tool, it is possible to set up state equations, algebraic equations and other mathematical models governing the behavior of the system. The primary difference between Petri nets and modeling tools is the presence of tokens which are used to simulate dynamic concurrent and asynchronous activities in a system. Figure:1 shows the example Petri Net.

A Petri net [3], [11] is a 5-tuple, PN = (P,T,F,W,M_0) where:

P = { p_1,p_2,p_3,….,p_m} is a finite set of places,
T = { t_1, t_2, ……, t_n } is finite set of transitions,
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
W: F -> { 1,2,3, ….} is a weight function,
M_o : P -> { 0,1,2,……} is the initial marking,
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

A Petri net structure N = (P, T, F, W) without any specific initial marking is denoted by N. A Petri net with the given initial marking is denoted by (N, M_0).

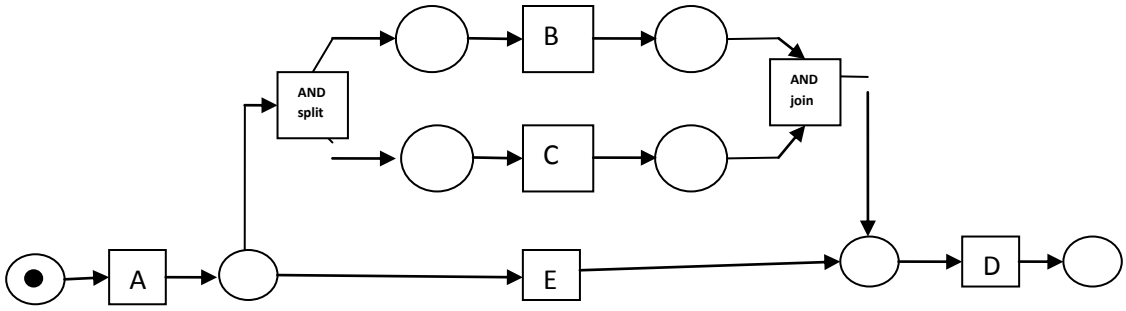

Figure 1. Example Workflow model represented using Petri net

It defines arcs and assigns to each arc a non-negative integer *arc multiplicity*; note that no arc may connect two places or two transitions. Figure 1 [2]shows a P/T-net consisting of 8 places and 7 transitions. Transition *A* has one input place and one output place, transition *AND-split* has one input place and two output places, and transition *AND-join* has two input places and one output place. The black dot in the input place of *A* represents a token. This token denotes the initial marking. The dynamic behavior of such a marked P/T-net is defined by a *firing rule*.

2.2. Firing rule

Let (N = (P; T; F); s) be a marked P/T-net. Transition $t \in T$ is enabled, denoted (N, s)[t>, iff $\bullet t \leq s$.

The firing rule –[->-$\subseteq$ N X T X N is the smallest relation satisfying for any (N = (P,T,F),s) $\in$ N and any $t \in$ T, $(N, s)[T > \Rightarrow (N, s[t > (N, s - \bullet t + t\bullet)$.

2.3. Boundedness

A given Petri net with initial marking M0 is said to be bounded if for any reachable marking, the number of tokens in each place does not exceed a finite value [6] , [7].

A marked net ($N = (P, T, F)$, s) is *bounded* iff the set of reachable markings [N, s is finite. It is *safe* iff, for any $s' \in [N, s >$ and any p $\in$ P, $s'(p) \leq 1$. Note that safeness implies boundedness. The marked P/T-net shown in Figure 1 is safe (and therefore also bounded) because none of the 8 reachable states puts more than one token in a place.

3.1. TOPOLOGICAL SORT

Topological sort is an ordering of the vertices in a directed acyclic graph, such that: If there is a path from u to v, then v appears after u in the ordering. The graphs should be directed, otherwise for any edge (u,v) there would be a path from u to v and also from v to u, and hence they cannot be ordered. The graphs should be acyclic, otherwise for any two vertices u and v on a cycle u would precede v and v would precede u. The ordering may not be unique.

Topological sorting can be used to schedule tasks under precedence constraints. Suppose we have a set of tasks to do, but certain tasks have to be performed before other tasks. These precedence constraints form a directed acyclic graph, and any topological sort defines an order to do these tasks such that each is performed only after all of its constraints are satisfied.

3.2 Algorithm for Topological Sort

After the initial scanning to find a vertex of degree 0, we need to scan only those vertices whose updated in-degrees have become equal to zero.

1. Store all vertices with in-degree 0 in a queue
2. **get** a vertex U and place it in the sorted sequence (another queue).
3. For all edges (U,V) update the in-degree of V, and **put** V in the queue if the updated in-degree is 0.
4. Perform steps 2 and 3 while the queue is not empty.

Complexity of topological Sort

The number of operations is O(|E| + |V|), where |V| - number of vertices, |E| - number of edges. The number of operations needed to compute the in-degree depends on the representation:

Adjacency lists: O(|E|)
Matrix: O($|V|^2$)
For complete graph |E|=O($|V|^2$)

For example consider Figure 2 to find out the topological sort. Compute the indegrees for the nodes present in Figure2:

V1: 0
V2: 1
V3: 2
V4: 2
V5: 2

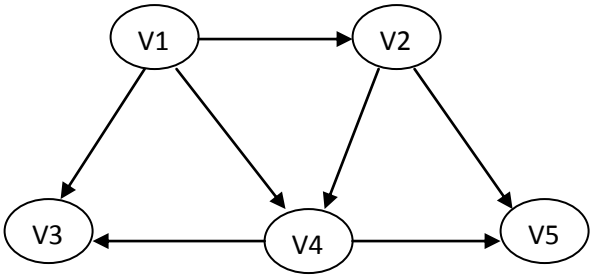

Figure 2 : Example diagram for topological sort.

- Find a vertex with indegree 0: V1
- Output V1, remove V1 and update the indegrees:

  Sorted: V1
  Remove edges: (V1,V2) , (V1, V3) and (V1,V4)
  Updated indegrees:

  V2: 0
  V3: 1
  V4: 1
  V5: 2

- Find a vertex with indegree 0: V2
- Output V2 , remove V2 and update the indegrees:

  Sorted: V2
  Remove edges: (V2, V4) and (V2, V5)

Repeat the same process and remove the vertices V4, V3 and V5.

The resultant topological sorting order is given by
: V1, V2, V4, V3, V5

4. PROPOSED ALGORITHM TO CHECK FOR BOUNDEDNESS

In this section the details of new algorithm that can handle loop identification is presented. The Petri net given in Figure 3 representing the workflow processes is taken as an input since Petri net is also directed graph this property helps in finding the loops present in the net by using the topological sorting algorithm. The Petri net given in Figure 3 is converted into a digraph which is shown in Figure 4. Topological sorting is applied on the graph given in Figure 4.

Once the algorithm finishes its iteration it finds no node is left out without process this implies that there is no loop present in the net. This is because topological sort is possible only if the given net is directed and acyclic. So, if all the nodes are present in the output list then it is concluded that no loop present in the net and the net is bounded.

Take another net given in Figure 6 and apply topological sort. Nodes are selected with zero indegree. S1 is selected because it is having in-degree zero the same way all other nodes are added to the output list. Since the nodes P2, P4, P6 and P8 are in a loop it will never get indegree zero so it will not be added in the output list when the algorithm terminates after specified number of iterations. This implies that those nodes which are not included in the output list are part of a loop. This leads to the conclusion that this workflow may lead to Unboundedness.

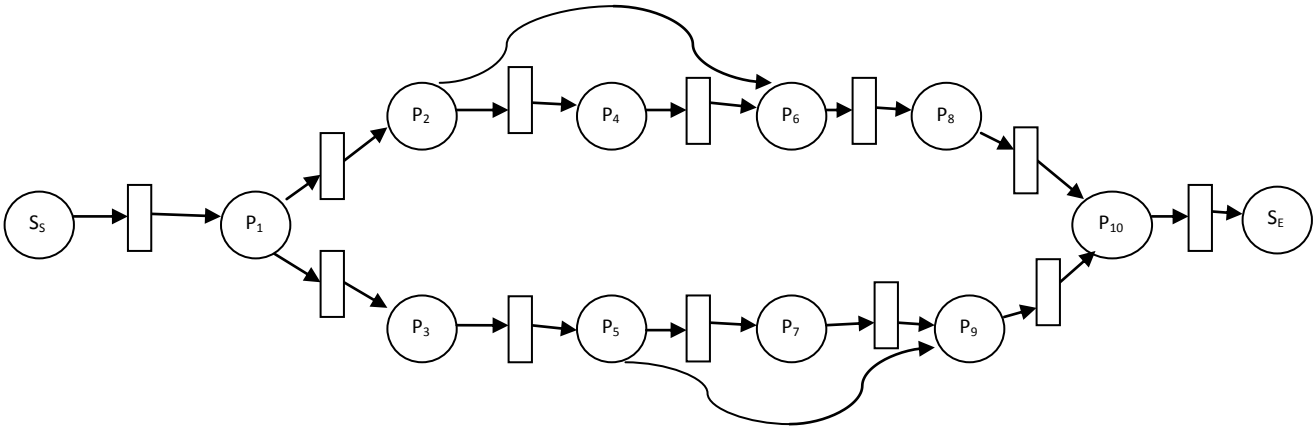

Figure 3: Workflow represented as Petri net

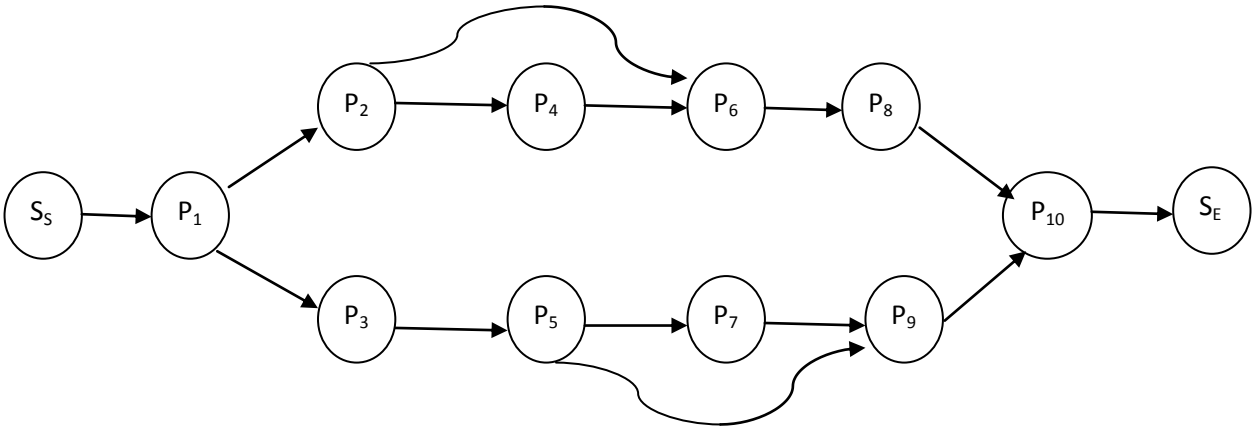

Figure 4: Graph representation of the above Petri net for topological sort

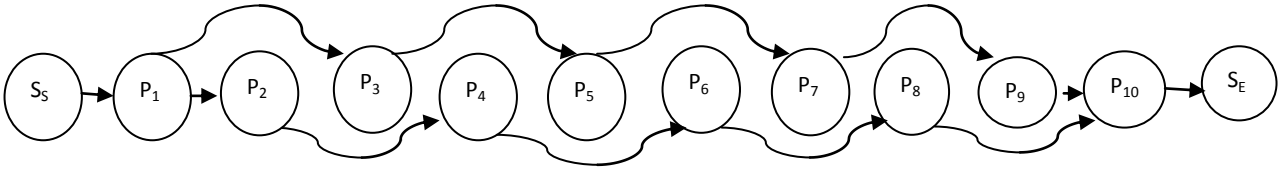

Figure 5: Resultant graph after applying topological sort

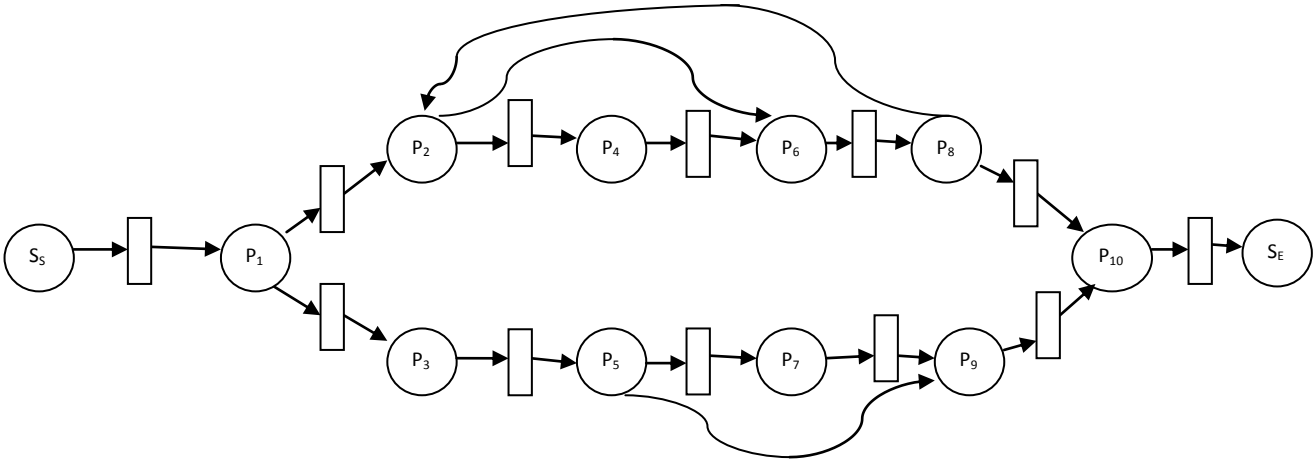

Figure 6: Petri net with loop

4. CONCLUSION

This paper introduced the topic of how topological sort is applied to find the loops present in a Workflow. Using a number of simple examples the work is illustrated. The main advantage of applying topological sort is it is easy to implement and practically possible to apply if the workflow grows in size. But there is a need to further address scientific challenges. Problems like length of the loop, persons involved in the loop duplicate tasks with noise etc., needs more attention.

5. REFERENCES

[1] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, 16(9):1128{1142, 2004.

[2] A.J.M.M. Weijters and W.M.P. van der Aalst. Process Mining: Discovering Workflow Models from Event-Based Data. In B. Kr¨ose, M. de Rijke, G. Schreiber, and M. van Someren, editors, *Proceedings of the 13th*

Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001), pages 283–290, 2001.

[3] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[4] Ana Karla Alves de Medeiros, Antonella Guzzo, Gianluigi Greco, Wil M. P. Van der Aalst, A.J.M.M. Weijiters, Boudewijn F. van Dongen, and Domenico Sacca. Process Mining Based on Clustering : A Quest for Precision. BPM Workshops, LNCS 4928, Springer-Verlag Berlin Heidelberg 2008

[5] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.

[6] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[7] W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business Process Mining: An Industrial Application. Information Systems, 32(5):713{732, 2007.

[8] S. Kumanan and K. Raja. Modeling and Simulation of Projects with Petri Nets. American Journal of Applied Sciences 5 , 2008.

[9] Yu Ru and Christoforos N. Hadjicostis. Reachability Analysis for a Class of Petri Nets. Joint IEEE Conference on Decision and Control and 28th Chinese Control Conference P.R. China, 2009.

[10] Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, And Domenico Sacca. Mining and Reasoning on Workflows. IEEE Transaction on Knowledge and Data Engineering, Vol 17, No.4 April 2005.

[11] Hemant Kr. Meena, Indradeep Saha, Koushik Kr. Mondal, T. V. Prabhakar. An Approach to Workflow Modeling and Analysis, OOPSLA, Oct 16-17, 2005 Saniego, CA, USA.