An alternative method for refined process structure trees (RPST)

Yongsun Choi
Department of Industrial and Management Engineering, Inje University, Gimhae, Republic of Korea
N. Long Ha
Department of Information and Communication Systems, Inje University, Gimhae, Republic of Korea
Pauline Kongsuwan
Faculty of Engineering, Rajamangala University of Technology Thanyaburi, Pathum Thani, Thailand, and
Kwan Hee Han
Department of Industrial and Systems Engineering, Gyeongsang National University, Jinju, Republic of Korea

Received 10 November 2018
Revised 16 May 2019
9 July 2019
Accepted 11 July 2019

Abstract
Purpose – The refined process structure tree (RPST), the hierarchy of non-overlapping single-entry single-exit (SESE) regions of a process model, has been utilized for better comprehension and more efficient analysis of business process models. Existing RPST methods, based on the triconnected components of edges, fail to identify a certain type of SESE region. The purpose of this paper is to introduce an alternative method for generating a complete RPST utilizing rather simple techniques.
Design/methodology/approach – The proposed method first focuses on the SESE regions of bonds and rigids, from the innermost ones to the outermost ones, utilizing dominance and post-dominance relations. Then, any SESE region of a series nested in a bond or a rigid is identified with a depth-first search variation. Two-phase algorithms and their completeness proofs, a software tool incorporating visualization of stepwise outcomes, and the experimental results of the proposed method are provided.
Findings – The proposed method utilizes simple techniques that allow their straightforward implementation. Visualization of stepwise outcomes helps process analysts to understand the proposed method and the SESE regions. Experiments with 604 SAP reference models demonstrated the limitation of the existing RPST methods. The proposed method, however, completely identified all types of SESE regions, defined with nodes, in less computation time than with the old methods.
Originality/value – Each triconnected component of the undirected version of a process model is associated with a pair of boundary nodes without discriminating between the entry and the exit. Here, each non-atomic SESE region is associated with two distinct entry and exit nodes from the original model in the form of a directed graph. By specifying the properties of SESE regions in more comprehensible ways, this paper facilitates a deeper understanding of SESE regions rather than relying on the resulting RPST.

Keywords Process model, Dominance, Post-dominance, Single-entry single-exit region, Workflow graph
Paper type Research paper

1. Introduction
As a divide-and-conquer approach, single-entry single-exit (SESE) regions have been actively utilized recently for better comprehension and more efficient analysis of business process models. Those cases utilizing SESE regions include process discovery (Augusto *et al.*, 2018), behavioral relation analysis (Weidlich *et al.*, 2010, 2011), structural verification (Vanhatalo *et al.*, 2007; Fahland *et al.*, 2011), process similarity analysis (Klinkmüller and Weber, 2017), clone detection (La Rosa *et al.*, 2015), model-to-text transformation (Leopold *et al.*, 2014; Fan *et al.*, 2017; Wang *et al.*, 2017), process model restructuring (Khlif *et al.*, 2017), etc. Most of these studies utilize the so-called

This work was supported by the 2016 Inje University research grant.

refined process structure tree (RPST) (Vanhatalo *et al.*, 2009; Polyvyanyy *et al.*, 2010), which is based on the triconnected decompositions (Hopcroft and Tarjan, 1973; Tarjan and Valdes, 1980; Gutwenger and Mutzel, 2001) of a process model.

Triconnected components, defined with edges, are identified from an undirected graph, and each of them is associated with a pair of boundary nodes. However, a process model is in the form of a directed graph, and each SESE region is associated with two distinct nodes of entry and exit. Because of this definition gap, a certain type of SESE region are not found by the existing RPST methods of Vanhatalo *et al.* (2009) or Polyvyanyy *et al.* (2010). This paper introduces a simple alternative method for identifying SESE regions in an arbitrary business process model utilizing dominance and post-dominance relations (Aho *et al.*, 2006; Cormen *et al.*, 2009) and simple depth-first search variation. In the proposed method, SESE regions are defined with nodes that represent activities and gateways, which garner more attention from BPM stakeholders.

The rest of this paper is organized as follows: Section 2 briefly introduces RPST and provides an example of an SESE region that is not identified by existing RPST methods. The nature of dominance and post-dominance relations, which are well known in control-flow analysis (Aho *et al.*, 2006; Gruhn and Laue, 2007; Cormen *et al.*, 2009) are also presented. Section 3 describes the proposed method and the software tool that incorporates visualization of stepwise outcomes of the proposed method. Section 4 provides the experimental results of the proposed method compared to the method of Polyvyanyy *et al.* (2010). Section 5 gives the conclusions and suggests future research directions.

2. Backgrounds and preliminaries

2.1 Workflow graphs

Similar to existing RPST methods (Vanhatalo *et al.*, 2009; Polyvyanyy *et al.*, 2010), process models in this paper are represented as workflow graphs of $G = (N, E, Start, End)$, where N is a set of nodes, E is a set of edges, and *Start* and *End* are distinguished nodes with no incoming or outgoing edges, respectively. Nodes, including activities and gateways, are represented in the BPMN style (Figure 1). For each pair of nodes, there is one edge at most. Gateways that act as a join and a split at the same time are allowed, each to be split into a join succeeded by a split, called normalization in Polyvyanyy *et al.* (2010). For any node $n \in N$, $Adj(n)$ denotes the set of nodes adjacent to n. A workflow graph, with the control types of its gateways ignored, is a type of control-flow graph. Figure 2 shows

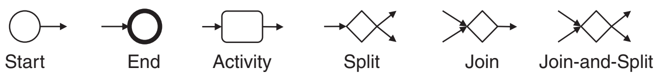


Figure 1.
Types of nodes and their representations

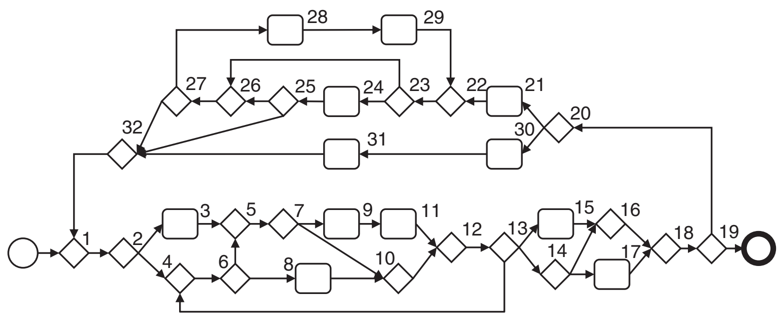


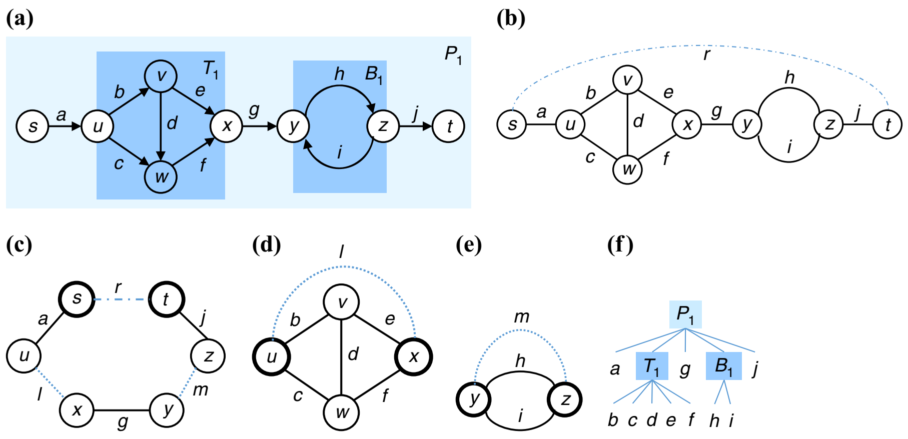
Figure 2.
Example workflow graph in normalized form

an example of a workflow graph in normalized from, slightly modified from the one introduced in Choi *et al.* (2015).

2.2 Triconnected components, canonical fragments and RPST

Tarjan and Valdes (1980) introduced a flow graph parsing method based on the triconnected decomposition of a biconnected graph, where multiple edges are allowed for a pair of nodes. Triconnected components are defined with edges, each associated with a pair of boundary nodes, and are uniquely configured with undirected graphs (Hopcroft and Tarjan, 1973). Di Battista and Tamassia (1990, 1996) introduced the data structure of the SPQR-tree to represent the hierarchy of triconnected components. After decades, linear time implementation to identify an SPQR tree was presented by Gutwenger and Mutzel (2001). Triconnected components are classified into four types (Polyvyanyy *et al.*, 2012): a trivial component of a single edge; a polygon that is a maximal sequence of multiple components where the exit node of the preceding component is the entry node of the succeeding component; a bond which is a maximal set of multiple components with the same pair of boundary nodes; and a rigid which is not a trivial, a polygon or a bond component.

Vanhatalo *et al.* (2009) proposed a workflow graph parsing technique introducing the RPST, which requires a fairly complex post-processing of the triconnected components (Polyvyanyy *et al.*, 2010). The RPST of a workflow graph, G, is the hierarchy of all non-overlapping canonical fragments of G, where a fragment is a connected subgraph associated with two distinct entry and exit nodes. For a fragment F, its entry has no incoming edge contained in F, or all its outgoing edges belong to F; and its exit has no outgoing edge belonging to F, or all its incoming edges belong to F. All other nodes, called interior to F, are connected only to nodes in F. Polyvyanyy *et al.* (2010) proposed a simpler alternative to computing the RPST. Their step, called normalization, split nodes first to force every node to have at most one incoming edge or at most one outgoing edge. Then, the RPST of the original model is derived from the SPQR-tree (Gutwenger and Mutzel 2001) of the resulting extended model, after "omitting" additionally introduced components incurred by normalization. Figure 3 shows how their method works for an example process model (Figure 3(a)), which is simplified from the one introduced in Polyvyanyy *et al.* (2010). Figure 3(b) is the undirected version of Figure 3(a). Figure 3(c) shows the polygon, $P1$, with virtual



Notes: (a) Example process model; (b) Undirected version of Figure 3(a); (c) Polygon P1; (d) Rigid T1; (e) Bond B1; (f) RPST of (a)

Figure 3. An example summarizing the RPST method in Polyvyanyy *et al.* (2010)

edges l and m, respectively representing the rigid $T1$ in Figure 3(d) and the bond $B1$ in Figure 3(e). Nodes in bold represent the pair of boundary nodes for each triconnected component (Figures 3(c)-(e)). The RPST of this example is shown in Figure 3(f). Normalization step is not required for this example.

2.3 A counter example: an SESE region not found by existing RPST methods

The RPST of a workflow graph G is the hierarchy of all its canonical fragments, each associated with two distinct entry and exit nodes. However, triconnected components are identified from the undirected version of G, each associated with a pair of boundary nodes, without discriminating the entry and the exit. Due to this definition gap, some canonical fragments, i.e., SESE regions, are not identified by the existing RPST methods of Vanhatalo *et al.* (2009) or Polyvyanyy *et al.* (2010). For example, according to their methods, two bonds, $B1$ and $B2$, are found for the structure shown in Figure 4(a) utilizing its normalized structure (Figure 4(b)) with trivial components of edges $s_1 \rightarrow s_2$ and $t_1 \rightarrow t_2$ omitted. However, only one bond, $B1$, is found for the structure shown in Figure 4(c), which is already in normalized form. By definition, the structure, $B2$, in Figure 4(c) is also a canonical fragment, with t as the entry and s as the exit.

2.4 The dominator tree and the post-dominator tree of a control-flow graph

In a control-flow graph, node d dominates node n if and only if all paths from *Start* to node n pass through node d; and node p post-dominates node n if and only if all paths from node n to *End* pass through node p. Every node dominates and post-dominates itself. The dominance and the post-dominance relations among nodes in a control-flow graph are represented by a dominator tree and by a post-dominator tree, with *Start* and *End* as the root, respectively. (Sreedhar *et al.*, 1996; Cooper *et al.*, 2001; Aho *et al.*, 2006):

Definition 1. (a) In the dominator tree, node n plus the set of its descendants (if any) is called the inverse dominators of n, denoted by $Dom^{-1}(n)$; (b) in the post-dominator tree, node n plus the set of its descendants (if any) is called the inverse post-dominators of n, denoted by $Pdom^{-1}(n)$.

Figure 5 shows the dominator tree and the post-dominator tree of the workflow graph in Figure 2 as an example. For instance, $Dom^{-1}(20) = \{20, \ldots, 32\}$ in Figure 5(a), and $Pdom^{-1}(32) = \{20, \ldots, 32\}$ in Figure 5(b).

3. SESE regions by dominance and post-dominance relations

3.1 SESE regions defined with nodes

Definition 2. A SESE region SESE(s, t) in a normalized workflow graph G is a connected subgraph spanned by boundary nodes of the entry s and the exit t plus its interior nodes, which are dominated by s, post-dominated by t, and

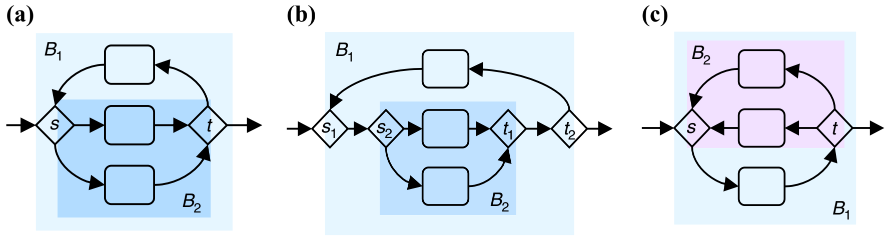
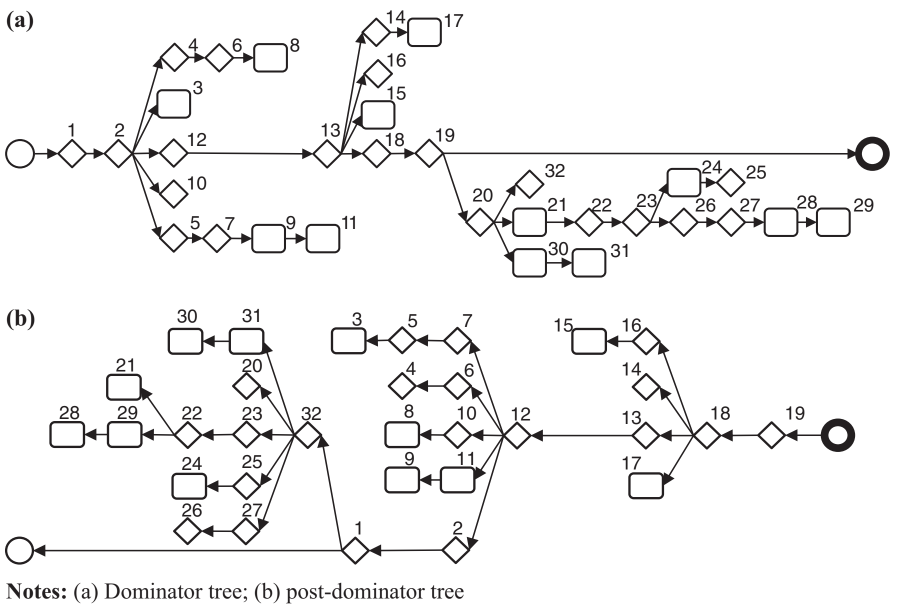


Figure 4.
Comparative examples illustrating an SESE region not found by existing RPST methods



Notes: (a) Dominator tree; (b) post-dominator tree

Figure 5. Dominance and post-dominance relations for the workflow graph of Figure 2

connected only to nodes in SESE(s, t). An SESE(s, t) is canonical if it shares any of its interior nodes with other canonical SESE regions only when they are in a nesting relation.

Definition 3. A canonical SESE region in a normalized workflow graph G is one of the following:

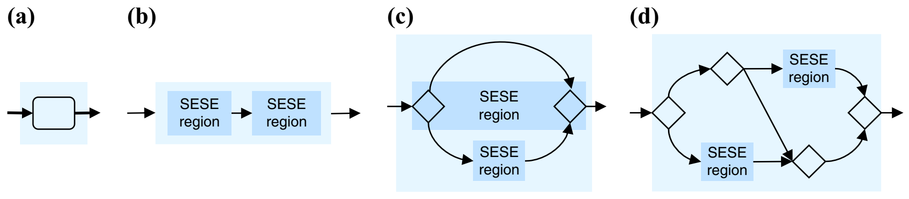
(1) a singleton of an activity node;

(2) a series is a maximal sequence of multiple SESE regions, each connected with a unique other region via a single edge or a shared boundary node;

(3) a bond is a maximal set of multiple SESE regions or paths, each connected with others only at the entry and at the exit, where each path contains zero or one SESE region; and

(4) a rigid is none of the above.

Figure 6 illustrates the four types of canonical SESE regions, defined with nodes in a normalized workflow graph G. Each singleton is an activity node bounded by itself and is introduced to define SESE regions in self-referenced form. A series is comparable to but



Notes : (a) A singleton; (b) a series; (c) a bond; (d) a rigid

Figure 6. Four types of SESE regions defined with nodes

different from a polygon of triconnected components, defined with edges. A bond or a rigid is bounded by gateways, and contains at least one interior node. An SESE region in this paper is a canonical SESE region, if not specified otherwise.

Considering that the main purposes of identifying SESE regions are for better comprehension and more efficient further analysis of process models, the proposed method first focuses on bonds and rigids.

3.2 Bonds and rigids defined with nodes

Definition 4. For a pair of distinct gateways, s and t, in a normalized workflow graph G, $Rg(s, t)$, is defined as the connected subgraph spanned by s and t plus the set of interior nodes n satisfying the following conditions:

(1) $n \in Dom^{-1}(s) \cap Pdom^{-1}(t)$;

(2) n is connected only to any other node in $\mathrm{Rg}(s, t)$;

(3) $|Adj(s) \cap Rg(s, t)| \geqslant 2$ and $|Adj(t) \cap Rg(s, t)| \geqslant 2$; and

(4) $|Adj(s) \cap Rg(s, t)| > |Adj(s) \cap Rg(s, w)|$ or $|Adj(t) \cap Rg(s, t)| > |Adj(t) \cap Rg(u, t)|$ for any $Rg(s, w)$ and $Rg(u, t)$ nested in $Rg(s, t)$.

Lemma 1. $Rg(s, t)$ of Definition 4 is a bond or a rigid with *entry s* and *exit t.*

Proof. Condition (1) assures that s and t, respectively, act as the entry and the exit of $Rg(s, t)$ of Definition 4. Condition (2) assures that each node n is interior to $Rg(s, t)$ of Definition 4. Condition (3) is required for $Rg(s, t)$ to be a bond or a rigid. Condition (4) assures that $Rg(s, t)$ of Definition 4 is not a series. If $Rg(s, t)$ shares any of its interior nodes with $Rg(q, r)$, with $s \neq q$ and $t \neq r$, then that interior node is dominated by both s and q, and post-dominated by both t and r, by definition. This implies that there exists a dominance relation between s and q and a post-dominance relation between t and r. It is further implied that $Rg(s, t)$ and $Rg(q, r)$ are in a nesting relation so as not to violate Condition (2). In a similar manner, $Rg(s, t)$ is in a nesting relation with any $Rg(s, r)$ or $Rg(q, t)$ if any of its interior nodes is shared with any of those, respectively. Thus, $Rg(s, t)$ of Definition 4 is a SESE region of a bond or a rigid, by Definition 2 and Definition 3. ■

Refinement of Rg(s, t) of Definition 4. Carriage return for any $Rg(s, t)$ of Definition 4, whether it is a bond or a rigid needs to be clarified. Additionally, as with triconnected components, the entry-exit pair may be associated with more than one bond or rigid (Hopcroft and Tarjan, 1973; Gutwenger and Mutzel, 2001). The following notions are used for a bond, a rigid, or a series, associated with its entry and its exit:

Definition 5. An SESE(s, t) in a normalized workflow graph G, with $s \neq t$, is denoted more specifically as follows according to its type: $B(s, t)$ for a bond, $R(s, t)$ for a rigid and $S(s, t)$ for a series. For rigids or series nested in $B(s, t)$, with the same entry s and the same exit t, distinct subscript indices are used for each type, e.g., $R_i(s, t)$ or $S_i(s, t)$.

Given $Rg(s, t)$ of Definition 4, any bond and/or rigid(s) sharing both the entry s and the exit t is found using Lemma 2 below utilizing the subgraphs of $Rg(s, t)$ defined as follows:

Definition 6. Each connected subgraph of $Rg(s, t)$, denoted by $sub_i(Rg(s, t))$, is connected with others only at the *entry s* and at the *exit t.*

Lemma 2. (Refinement of $Rg(s, t)$):

(1) When $Rg(s, t)$ of Definition 4 has only a single subgraph of Definition 6, then $Rg(s, t)$ is a rigid, $R(s, t)$.

(2) Otherwise, $Rg(s, t)$ is a bond, $B(s, t)$, and each of its subgraph $\text{sub}_i(Rg(s, t))$ of Definition 6 is a rigid if it is not a series.

Proof: Each connected subgraph $sub_i(Rg(s, t))$ is of two types: A path containing zero or one SESE region; or another SESE(s, t) nested in $Rg(s, t)$. The former ones are not contained in a rigid but in a bond. Any SESE(s, t) nested in $Rg(s, t)$ is not a bond by Definition 3: it is a series if it is a sequence of multiple SESE regions each connected with unique other one via a single edge or a shared boundary node; otherwise, it is a *rigid* which contains an interior gateway not contained in any bond or rigid nested in $Rg(s, t)$. ■

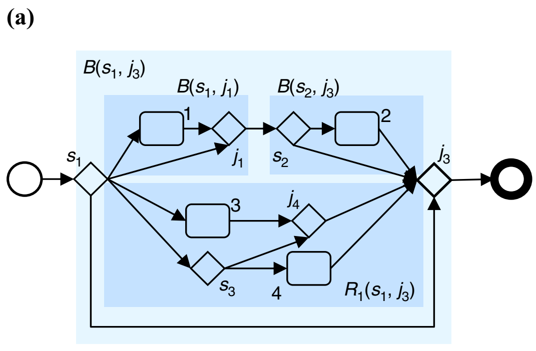
Figure 7 illustrates Lemma 2. $Rg(s_1, j_3)$ in Figure 7(a), found by Definition 4, has three subgraphs connected only at the entry s_1 and at the exit j_3. The edge $s_1 \rightarrow j_3$ is one of those. The sequence of two nested bonds, $B(s_1, j_1)$ and $B(s_2, j_3)$, assumed to have been previously identified, is a series. Thus, $Rg(s_1, j_3)$ is a bond, $B(s_1, j_3)$ nesting a rigid of $R_1(s_1, j_3)$, with the entry s_1 and the exit j_3 shared. Note that $R_1(s_1, j_3)$, not nesting any bond or rigid, contains interior gateways s_3 and j_4. Figure 7(b) shows a cyclic structure $Rg(h, e)$, dominated by the single loop-entry h and post-dominated by the single loop-exit e. It has two subgraphs of Definition 6, each connected only at the entry h and at the exit e. Thus, it is a bond, $B(h, e)$, and the subgraph of $Rg(h, e)\backslash\{3\}$ is a rigid of $R_1(h, e)$:

Theorem 1. For a normalized workflow graph G, each subgraph corresponding to a canonical fragment of a bond or a rigid is identified by Lemma 1 and Lemma 2.

Proof: A canonical fragment of a bond or a rigid is bounded by a pair of gateways. For a normalized workflow graph G, any bond or rigid with distinct gateways of entry s and exit t, is found by Lemma 1. Any rigid $R_i(s, t)$ nested in a bond $B(s, t)$ is found by Lemma 2. ■

3.3 The algorithm and its illustration

Algorithm 1 summarizes the proposed method that identifies the SESE regions defined with nodes. The proposed method first identifies bonds and rigids, bounded by gateways, from the innermost ones to the outermost ones. For that purpose, the proposed method utilizes two ordered sets of gateways, each respectively sorted in the bottom-up order of the dominator tree and in the bottom-up order of the post-dominator tree, for a normalized workflow graph G. Any gateway that dominates or post-dominates only itself is deleted from each of those two sets, respectively. After all bonds and rigids are identified, each series nested in a bond or a rigid is identified with a simple depth-first search variation, as



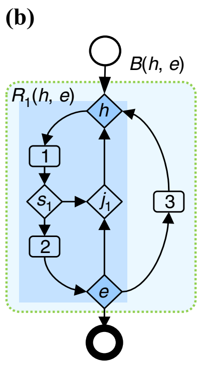


Figure 7. Examples of nested rigids sharing the entry and exit with the nesting bond

described in Appendix. The workflow graph *G*, with distinguished *Start* and *End*, is a series as itself.

Algorithm 1. Compute the RPST defined with nodes

```
Input: Normalized workflow graph G
Output: RPST defined with nodes for G
{Step 1. Preprocessing of G}
cndEntry ← A set of gateway s of G, where | Dom^-1(s) | > 1, in bottom-up order of
            Dominator Tree of G
cndExit ← A set of gateway t of G, where | Pdom^-1(t)) | > 1, in bottom-up order of
            Post-dominator Tree of G
{Step 2. Identifying the SESE regions}
for each s in cndEntry do
    for each t in cndExit do
        if Rg (s, t) of Definition 4 is found then
            Extend RPST with R (s, t); or with B (s, t) and any nested R_i (s, t), identified by
            Lemmas 1 and 2
        end if
    end for
end for
for any bond or rigid in the RPST do
    Extend RPST with any series nested in the current bond or rigid, identified by
    Appendix 1
end for
return RPST
```

As shown in Figure 4, a bond *B*(*t*, *s*) nested in another bond *B*(*s*, *t*), with their entry and exit interchanged, is not identified by the existing RPST methods of Vanhatalo *et al.* (2009) or Polyvyanyy *et al.* (2010). Such a phenomenon happens for a cyclic structure when: all its nodes are dominated by a single loop-entry *s* and post-dominated by a single loop-exit *t*; and it has multiple paths from *t* to *s*, with neither *t* nor *s* revisited, which are connected only at *t* and at *s*. The proposed method identifies such nested bond structures, each prior to its nesting bond, simply by considering all the entry-exit pairs of gateways in the order as described in Algorithm 1.

Table I summarizes the steps identifying the bonds and rigids, and then series nested in them, for the workflow graph in Figure 2. Two bonds and three rigids are found represented

Table I. Summary of steps identifying SESE regions for the workflow graph in Figure 2

Entry *s*	Exit *t*	Type of Rg(*s*, *t*)	Elements in Rg(*s*, *t*)	Series in Rg(*s*, *t*)
22	32	*R*(22, 32)	{19, …, 29, 32}	*S*(28, 29)
20	32	*B*(20, 32)	{20, 21, 30, 31, *R*(22, 32)}	*S*(30, 31); *S*(21, 32)
13	18	*R*(13, 18)	{13, …, 18}	–
2	13	*R*(2, 13)	{2, …, 13}	*S*(9, 11)
1	19	*B*(1, 19)	{1, 19, *R*(2, 13), *R*(13, 18), *B*(20, 32)}	*S*(2, 18)
Start	End	–	–	*S*(*Start, end*)

in the third column in Table I. Neither of those two bonds nests other rigids sharing each corresponding entry-exit gateways. The fourth column of Table I shows the elements of activities, bonds, or rigids, contained in each bond or rigid. The last column shows the series nested in each bond or rigid. For example, $R(22, 32) = \{22, \ldots, 29, 32\}$ and it nests $S(28, 29) = \{28, 29\}$. Elements contained in other series are as follows: $S(30, 31) = \{30, 31\}$, $S(21, 32) = \{21, R(22, 32)\}$, $S(9, 11) = \{9, 11\}$, and $S(2, 18) = \{R(2, 13), R(13, 18)\}$ in Table I. The workflow graph G itself is a series of $S(Start, End) = \{Start, B(1, 19), End\}$.

Figure 8 shows the resulting RPST defined with nodes for the workflow graph in Figure 2. Bonds and rigids are represented in solid boxes shaded in blue, and series are represented by dashed boxes. Leaf nodes of the RPST in Figure 8 are singletons of activities or gateways. In the RPST defined with nodes, boundary gateways shared by multiple SESE regions are redundantly contained in each of those, only when they are not in a nesting relation. For instance, gateway 13 is contained in both $R(2, 13)$ and $R(13, 18)$, whereas gateways 18 and 32 are contained only in each bottommost SESE region, $R(13, 18)$ and $R(22, 32)$, respectively.

3.4 The software tool incorporating visualization of stepwise outcomes

The proposed method was implemented in Microsoft C# as the SESE region identification module in the graph-based process analysis tool *gProAnalyzer* (Choi *et al.*, 2015). The execution file of this tool, its user guide, and sample input data files of 604 SAP reference models are available for download at GitHub[1]. Figure 9 shows the selected screenshots of the visualizations provided by the tool for the example workflow graph in Figure 2. The tool provides six types of visualizations corresponding to each stepwise outcome of the proposed method: (1) input model in its original representation (e.g. an EPC model); (2) input model transformed into a normalized workflow graph; (3) the dominator tree (Figures 9(a)); (4) the post-dominator tree (Figures 9(b)); (5) the resulting RPST (Figure 9(c)); and (6) the extended workflow graph model with SESE regions marked (Figure 9(d)). For better visualization of types (3) to (6), they are represented with the extended workflow graph where each entry (in yellow-green) and each exit (in pink) of a SESE region is split to have a single external edge to that SESE region.

4. Experimental results

The proposed method, implemented as a module in the *gProAnalyzer* (Choi *et al.*, 2015) was tested with SAP reference models (Curran *et al.*, 1997). For comparison, the method in

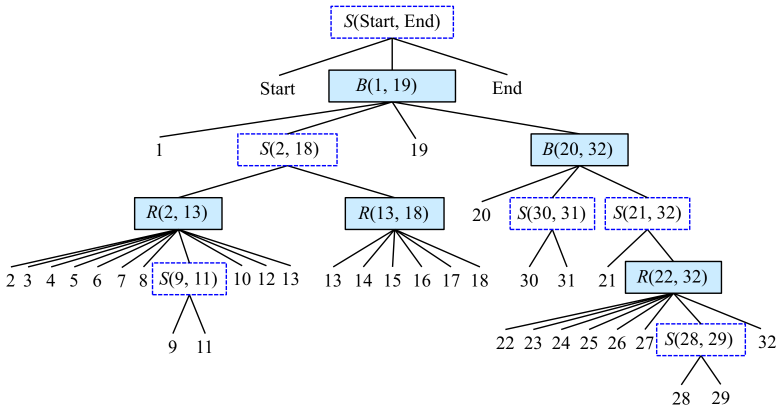


Figure 8. The RPST defined with nodes for the workflow graph in Figure 2

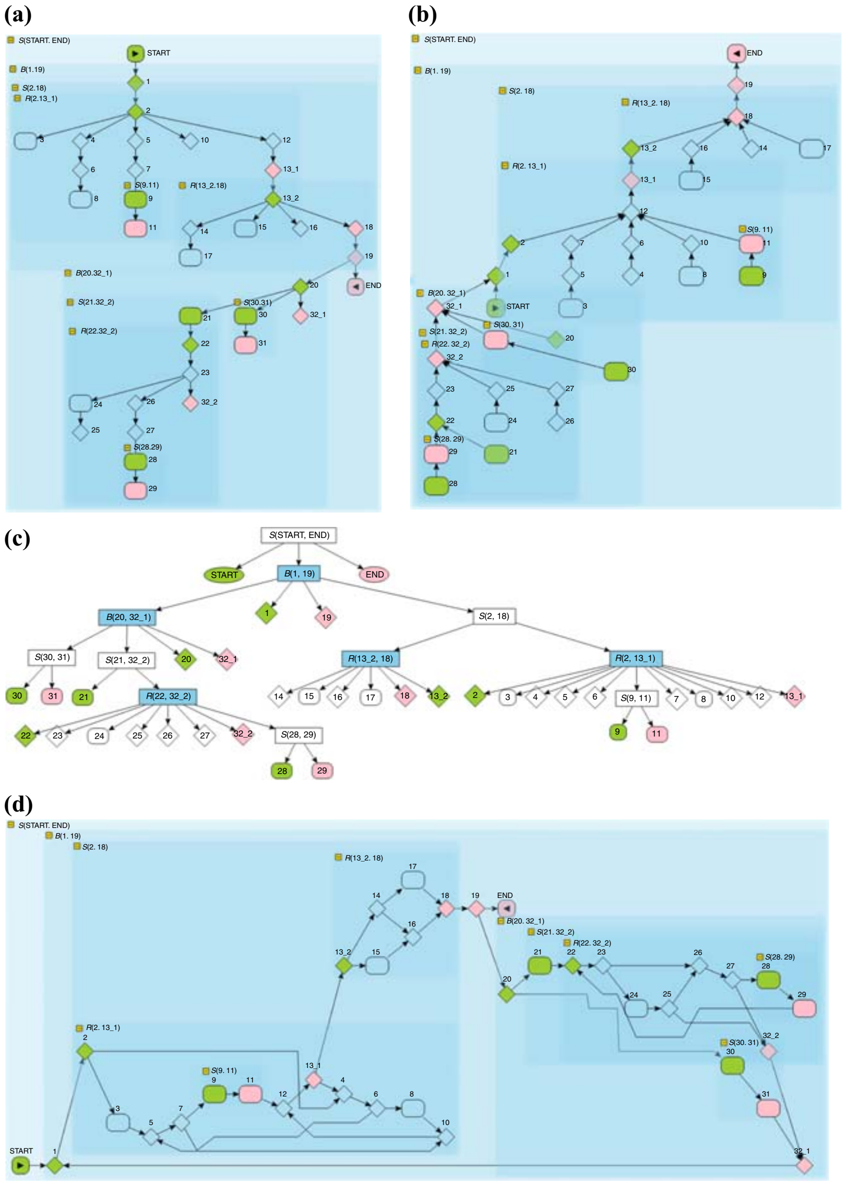


Figure 9.
Selected screenshots of the SESE region module of the tool *gProAnalyzer*

Notes: (a) The dominator tree; (b) the post-dominator tree; (c) the RPST; (d) the extended workflow graph with SESE regions marked

Polyvyanyy *et al.* (2010) was also tested utilizing its open-source implementation in Java, as a module in the jBPT (business process technologies for Java) version 0.2.429 (Polyvyanyy and Weidlich, 2013). Both methods were tested with a desktop PC with a 3.6 GHz Intel Core i7 and 16 GB RAM.

First, 604 EPC models were transformed into normalized workflow graphs as follows correcting some syntactic errors:

- Models with multiple start events and/or multiple end events were transformed to have a single start and a single end, as in Polyvyanyy *et al.* (2009). This also fixed 74 models where each of them was composed of multiple disconnected subgraphs.
- Regardless of node types in EPC models, nodes with single incoming and single outgoing edges were converted into activities; and nodes with multiple incoming (or outgoing) edges were converted into gateways.
- Each gateway that acts as a join and a split at the same time is split into a join succeeded by a split.

Figure 10 shows the number of bonds and the number of rigids identified by the proposed method, plotted by the number of nodes for each model. The following are observed:

- The number of rigids was rather restricted. More specifically, zero for 406 models, one for 178 models, and a maximum of two for 20 models, and its relation to the number of nodes is not apparent.
- The number of bonds tended to increase when the number of nodes increases.
- All bonds and rigids found by the method in Polyvyanyy *et al.* (2010) were identified by the proposed method.
- Ten cyclic structures each dominated by its single loop-entry and post-dominated by its single loop-exit were found, all as bonds. From one of those ten bonds, another bond structure nested in that bond, with their entry and exit interchanged, was identified by the proposed method but not by the method of Polyvyanyy *et al.* (2010).

Figure 11 shows the total computation time of both methods, averaged after 100 runs for each model, plotted by the number of nodes for each model. The following are observed:

- The computation times did not significantly increase with the number of nodes in the models, but those fit better with a quadratic increase rather than a linear increase for both methods.
- Computation time was not our main concern for comparison, which would require a more elaborate experiment; however, the proposed method required less computation time for each of the 604 models (Table II).

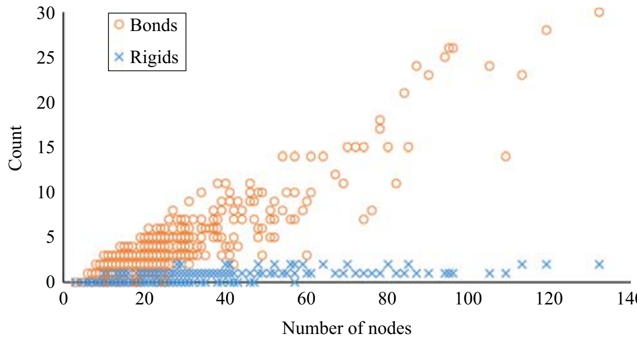


Figure 10.
Number of bonds and rigids identified by the proposed method

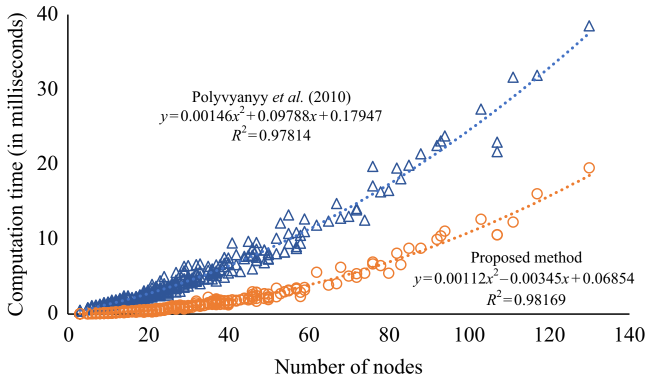


Figure 11.
Computation times of both methods

Table II.
Summary of the empirical analysis results of the proposed method

	Minimum	Average	Maximum
No. of bonds	0	3.60	30
No. of rigids	0	0.36	2
No. of bonds and rigids	0	3.96	32
Depth of RPST (bonds and rigids only)	0	2.07	9
Computation time (in milliseconds)	0.016	0.870	19.511

5. Conclusions

This paper introduced an alternative method for generating a complete refined process structure tree (RPST) for arbitrary business process models. It was also shown that an RPST computed by the existing methods was not complete. The method proposed here utilizes relatively simple techniques of dominance and post-dominance relations among nodes, whereas the existing methods are based on the triconnected components of edges from the undirected version of a process model.

The proposed method first identifies all SESE regions of bonds and rigids, from the innermost ones to the outermost ones, utilizing dominance and post-dominance relations. Those structures are crucial for better comprehension and more efficient analysis of a process model. Then, any SESE region of a series nested in each bond or rigid is identified utilizing a depth-first search variation. The specific properties for each type of SESE regions, the two-phase algorithms aligned with those properties, their completeness proofs, a software tool incorporating visualization of the stepwise outcomes of the process, and the experimental results of the proposed method were introduced. Experiments with 604 SAP reference models demonstrated that the existing RPST methods failed to identify a bond structure nested in another bond with their entry and exit interchanged. However, the method proposed here completely identified all types of SESE regions, defined with nodes, in less computation time than with the old method.

In this paper, the properties of SESE regions are specified in more comprehensible ways utilizing simple techniques. Thus, this paper facilitates a deeper understanding of SESE regions, especially for business analysts, than relying on the resulting RPST. The implementation of each of two-phase algorithms is straightforward, and either of those can be further integrated with additional modules for any specific purpose. In sum, the findings here should allow further process analysis studies utilizing SESE regions to be more effectively pursued.

Note

1. https://github.com/InjeBPM/Single-Entry-Single-Exit-Identification

References

Aho, A.V., Lam, M.S., Sethi, R. and Ullman, J.D. (2006), *Compilers: Principles, Techniques, and Tools*, 2nd ed., Addison-Wesley Longman Publishing, Boston, MA.

Augusto, A., Conforti, R., Dumas, M., La Rosa, M. and Bruno, G. (2018), "Automated discovery of structured process models from event logs: the discover-and-structure approach", *Data & Knowledge Engineering*, Vol. 117, pp. 373-392, doi: 10.1016/j.datak.2018.04.007.

Choi, Y., Kongsuwan, P., Min, C. and Zhao, J.L. (2015), "Stepwise structural verification of cyclic workflow models with acyclic decomposition and reduction of loops", *Data & Knowledge Engineering*, Vol. 95, pp. 39-65, doi: 10.1016/j.datak.2014.11.003.

Cooper, K., Harvey, T. and Kennedy, K. (2001), "A simple, fast dominance algorithm", *Software: Practice and Experience*, Vol. 4 Nos 1-10, pp. 1-8.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2009), *Introduction to Algorithms*, 3rd ed., MIT Press, Cambridge, MA.

Curran, T., Keller, G. and Ladd, A. (1997), *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*, Prentice Hall, Upper Saddle River.

Di Battista, G. and Tamassia, R. (1990), "On-line graph algorithms with SPQR-trees", in Paterson, M.S. (Ed.), *Automata, Languages and Programming. ICALP 1990. Lecture Notes in Computer Science*, Vol. 443, Springer, Berlin, Heidelberg.

Di Battista, G. and Tamassia, R. (1996), "On-line maintenance of triconnected components with SPQR-trees", *Algorithmica*, Vol. 15 No. 4, pp. 302-318, doi: 10.1007/BF01961541.

Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H. and Wolf, K. (2011), "Analysis on demand: instantaneous soundness checking of industrial business process models", *Data & Knowledge Engineering*, Vol. 70, pp. 448-466, doi: 10.1016/j.datak.2011.01.004.

Fan, J., Wang, J., An, W., Cao, B. and Dong, T. (2017), "Detecting difference between process models based on the refined process structure tree", *Mobile Information Systems*, Vol. 2017, pp. 1-17, doi: 10.1155/2017/6389567.

Gruhn, V. and Laue, R. (2007), "What business process modelers can learn from programmers", *Science of Computer Programming*, Vol. 65 No. 1, pp. 4-13, doi: 10.1016/j.scico.2006.08.003.

Gutwenger, C. and Mutzel, P. (2001), "A linear time implementation of SPQR-trees", in Marks, J. (Ed), *Graph Drawing: 8th International Symposium, GD 2000 Colonial Williamsburg, VA, USA, September 20-23*, Springer, Berlin and Heidelberg, pp. 77-90.

Hopcroft, J.E. and Tarjan, R.E. (1973), "Dividing a graph into triconnected components", *SIAM Journal on Computing*, Vol. 2 No. 3, pp. 135-158, doi: 10.1137/0202012.

Khlif, W., Ben-Abdallah, H. and Ben Ayed, N.E. (2017), "A methodology for the semantic and structural restructuring of BPMN models", *Business Process Management Journal*, Vol. 23 No. 1, pp. 16-46, doi: 10.1108/BPMJ-12-2015-0186.

Klinkmüller, C. and Weber, I. (2017), "Analyzing control flow information to improve the effectiveness of process model matching techniques", *Decision Support Systems*, Vol. 100, pp. 6-14, doi: 10.1016/j.dss.2017.06.002.

La Rosa, M., Dumas, M., Ekanayake, C.C., García-Bañuelos, L., Recker, J. and Ter Hofstede, A.H.M. (2015), "Detecting approximate clones in business process model repositories", *Information Systems*, Vol. 49, pp. 102-125, doi: 10.1016/j.is.2014.11.010.

Leopold, H., Mendling, J. and Polyvyanyy, A. (2014), "Supporting process model validation through natural language generation", *IEEE Transactions on Software Engineering*, Vol. 40 No. 8, pp. 818-840, doi: 10.1109/TSE.2014.2327044.

Polyvyanyy, A., García-Bañuelos, L. and Dumas, M. (2012), "Structuring acyclic process models", *Information Systems*, Vol. 37 No. 6, pp. 518-538, doi: 10.1016/j.is.2011.10.005.

Polyvyanyy, A., Smirnov, S. and Weske, M. (2009), "On application of structural decomposition for process model abstraction", *2nd International Conference on Business Process and Services Computing*, pp. 110-122.

Polyvyanyy, A., Vanhatalo, J. and Völzer, H. (2010), "Simplified computation and generalization of the refined process structure tree", Web Serv Form Methods 7th Int Work WS-FM 2010, Hoboken, NJ, Revis Sel Pap 6551 LNCS:25–41, September 16-17, doi:10.1007/978-3-642-19589-1_2.

Polyvyanyy, A. and Weidlich, M. (2013), "Towards a compendium of process technologies the jBPT library for process model analysis", *Proceedings of the Forum of the 25th International Conference on Advanced Information Systems Engineering (CAiSE Forum)*, pp. 106-113.

Sreedhar, V.C., Gao, G.R. and Lee, Y.-F. (1996), "Identifying loops using DJ graphs", *ACM Transactions on Programming Languages and Systems*, Vol. 18 No. 6, pp. 649-658, doi: 10.1145/236114.236115.

Tarjan, R. and Valdes, J. (1980), "Prime subprogram parsing of a program", *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Language*, pp. 95-105, doi: 10.1145/567446.567456.

Vanhatalo, J., Völzer, H. and Koehler, J. (2009), "The refined process structure tree", *Data Knowl Eng*, Vol. 68 No. 9, pp. 793-818, doi: 10.1016/j.datak.2009.02.015.

Vanhatalo, J., Völzer, H. and Leymann, F. (2007), "Faster and more focused control-flow analysis for business process models through SESE decomposition", *Service-Oriented Computing – ICSOC 2007: Fifth International Conference, Proceedings*, *Vienna*, *September 17-20*, pp. 43-55.

Wang, J., Cao, B., Fan, J. and Dong, T. (2017), "FB-Diff: a feature based difference detection algorithm for process models", *2017 International Conference on Web Services*, pp. 604-611, doi: 10.1109/ICWS.2017.71.

Weidlich, M., Polyvyanyy, A., Mendling, J. and Weske, M. (2010), "Efficient computation of causal behavioural profiles using structural decomposition", *International Conference on Application and Theory of Petri Nets and Concurrency*, pp. 63-83.

Weidlich, M., Polyvyanyy, A., Mendling, J. and Weske, M. (2011), "Causal behavioural profiles – efficient computation, applications, and evaluation", *Fundamenta Informaticae*, Vol. 113 Nos 3-4, pp. 399-435.

Appendix. Identification of series nested in a bond or a rigid

Each boundary gateway of any bond or rigid could be split to have a single external edge to that bond or rigid. This helps better visualization of models and further simplifies identifying series structures. Figure A1 shows the workflow graph extended from the one in Figure 2, where double-bordered gateways are those split to let each bond or rigid have a single external edge at its entry and at its exit. Each of those gateways is labeled the same as the original gateway but with a distinct subscript. Each bond or rigid, as shown in Table I is represented in shaded rectangles. If reduced, each of them is represented in the form of an activity, i.e., with a single incoming edge and a single outgoing edge, in the reduced model. Any series nested in a bond or a rigid is found by a simple depth-first search, with the traversal of any directly nested bond or rigid is limited only to its entry, as a maximal sequence of multiple activities or directly nested bonds or rigids.

Algorithm 2 demonstrates the steps for identifying any series nested in a bond or a rigid. Figure A2, as an example, shows the depth-first search tree for each bond or rigid in Figure A1. Shaded gateways represent the entry of any directly nested bond or rigid which is traversed only for that entry. Each series found by Algorithm 2 is represented as a dashed box in blue, which returns the whole structure of the directly nested bond or rigid corresponding to each shaded entry.

Note that existing RPST methods may conclude the incoming edge and the outgoing edge of an activity node which succeeds and precedes two distinct gateways (e.g. nodes 3, 8, 15, 17 or 24 in Figures 2 and A1) also as a polygon. Neither visualization nor analysis is enhanced by such polygons; they just make the RPST more complex. Series structures comparable to such polygons are excluded in the proposed method.

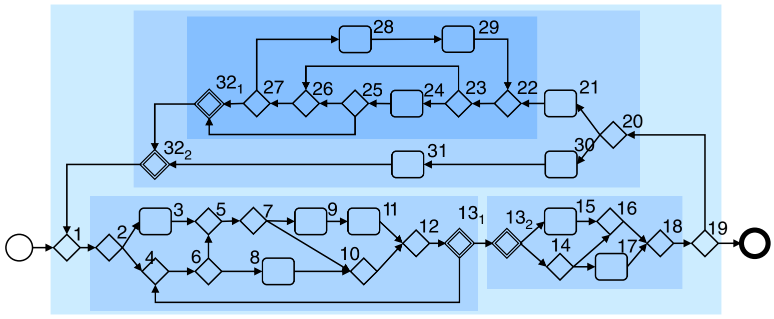
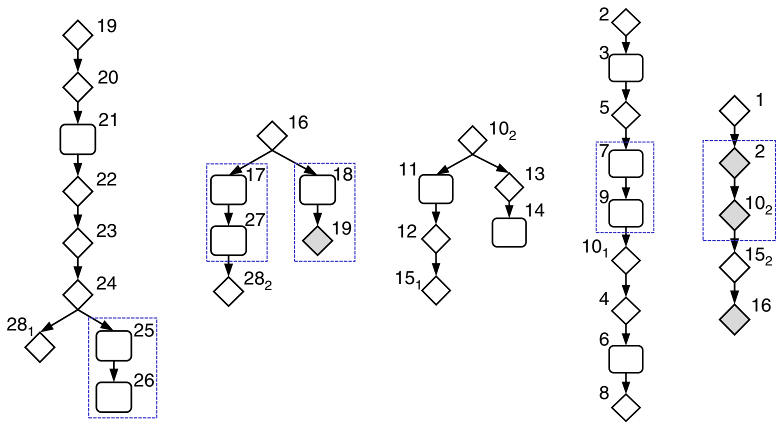


Figure A1. Workflow graph extended from Figure 2 marked with bonds and rigids, each having single external edge at the entry and at the exit



Notes: (a) $R(19, 28_1)$; (b) $B(16, 28_2)$; (c) $R(10_2, 15_1)$; (d) $R(2, 10_1)$; (e) $B(1, 15_2)$

Figure A2. Series structures found from the depth-first search tree of each bond or rigid in Figure A1

Algorithm 2: Identify any series nested in a bond or rigid

```
Input: $SESE(s, t)$ of a bond or a rigid, any bond or rigid $SESE(q, r)$ directly nested in
        $SESE(s, t)$
Output: Any series nested in $SESE(s, t)$
$S$: A candidate series containing activities or $SESE(q, r)$ in $SESE(s, t)$; $SoS$: A set of
series; $V$: a stack of nodes
$V$.push($s$)
while $V$ is not empty do
  $n \leftarrow V$.pop()
  if $n$ is not visited then
    Mark $n$ as visited
    select case $n$
      case An activity node: $S \leftarrow S \cup \{n\}$;
      case The entry $q$ of $SESE(q, r)$:
        $S \leftarrow S \cup SESE(q, r)$; $n \leftarrow r$
        if ($r = s$) or ($r = t$) then CheckCandidateSeries
        if ($r = s$) then continue;
      otherwise:
        CheckCandidateSeries
    end case
    for all nodes $m$ succeeding $n$ do
      $V$.push($m$)
  end if
end while
return $SoS$

Procedure CheckCandidateSeries
  if $S$ contains more than one activity node or $SESE(q, r)$ then $SoS \leftarrow SoS \cup S$
  $S \leftarrow$ Null
Return
```

About the authors

Dr Yongsun Choi is full Professor in the Department of Industrial and Management Engineering at Inje University, Korea. He holds PhD and MS Degrees in Industrial Engineering from Korea Advanced Institute of Science and Technology; and BS Degree in Industrial Engineering from Seoul National University, Korea. His research interests include process modeling and analysis, process model generation, decision support systems, ontologies, and information extraction. He has publications in *Data & Knowledge Engineering*, *International Journal of Technology Management*, *Information Systems Frontier*, *International Journal of Web Services Research*, *Computers and Operations Research*, *International Journal of Advanced Manufacturing Technology*, etc. Dr Yongsun Choi is the corresponding author and can be contacted at: yschoi@inje.edu

N. Long Ha is PhD Candidate in the Department of Information and Communication Systems at Inje University, Korea. He holds MS Degree in Industrial and Management Engineering from Inje University, Korea; and BS Degree in Economics Information Systems from Hue University, Vietnam. His research interests include business process management, process modeling and analysis, and information systems.

Dr Pauline Kongsuwan is Professor in the Department of Computer Engineering at Rajamangala University of Technology Thanyaburi, Thailand. She holds PhD and MS Degrees in Systems and Management Engineering from Inje University, Korea; and BS Degree in Computer Engineering from Kasetsart University, Thailand. Her research interests include process modeling and analysis, information system, software engineering, quality engineering and network security. She has publications in the *European Journal of Operational Research*, *Data & Knowledge Engineering* and *Mathematical Problems in Engineering*.

Dr Kwan Hee Han is full Professor in the Department of Industrial and Systems Engineering at the Gyeongsang National University, Korea. He received PhD Degree in Industrial Engineering from KAIST, Korea and has authored or co-authored numerous papers published in international journals and conference proceedings. His current research interests include process mining, simulation modeling and smart factory.