Data & Knowledge Engineering 68 (2009) 793–818

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

The refined process structure tree

Jussi Vanhatalo [a,b,1], Hagen Völzer [a,*], Jana Koehler [a]

[a] IBM Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
[b] Institute of Architecture of Application Systems, University of Stuttgart, Universitätsstrasse 38, D-70569 Stuttgart, Germany

ARTICLE INFO

Article history:
Available online 9 March 2009

Keywords:
Workflow management
Workflow graph parsing
Model decomposition
Subprocess detection
Graph theory

ABSTRACT

We consider a workflow graph as a model for the control flow of a business process and study the problem of workflow graph parsing, i.e., finding the structure of a workflow graph. More precisely, we want to find a decomposition of a workflow graph into a hierarchy of sub-workflows that are subgraphs with a single entry and a single exit of control. Such a decomposition is the crucial step, for example, to translate a process modeled in a graph-based language such as BPMN into a process modeled in a block-based language such as BPEL. For this and other applications, it is desirable that the decomposition be unique, *modular* and as fine as possible, where *modular* means that a local change of the workflow graph can only cause a local change of the decomposition. In this paper, we provide a decomposition that is unique, modular and finer than in previous work. We call it the *refined process structure tree*. It is based on and extends similar work for sequential programs by Tarjan and Valdes [ACM POPL '80, 1980, pp. 95–105]. We give two independent characterizations of the refined process structure tree which we prove to be equivalent: (1) a simple descriptive characterization that justifies our particular choice of the decomposition and (2) a constructive characterization that allows us to compute the decomposition in linear time. The latter is based on the tree of triconnected components (elsewhere also known as the SPQR tree) of a biconnected graph.

1. Introduction

The control flow of a business process can often be modeled as a *workflow graph* [1]. Workflow graphs capture the core of many business process languages such as UML activity diagrams, BPMN and EPCs. We study the problem of *parsing* a workflow graph, that is, decomposing the workflow graph into a hierarchy of sub-workflows that have a single entry and a single exit of control, often also called *blocks*, and labeling these blocks with a syntactical category they belong to. Such categories are *sequence*, *if*, *repeat-until*, etc., see Fig. 1a. Such a decomposition is also called a *parse* of the workflow graph. It can also be shown as a *parse tree*, see Fig. 1c.

The parsing problem occurs when we want to translate a graph-based process description (e.g. a BPMN diagram) into a block-based process description (e.g. BPEL process), but there are also other use cases for workflow graph parsing. For example, Vanhatalo, Völzer and Leymann [2] show how parsing speeds up control-flow analysis. Küster et al. [3] show how differences between two process models can be detected and resolved based on decompositions of these process models. Gschwind et al. [4] use parsing for pattern-based editing operations of process models. We believe that parsing also helps in understanding large processes and in finding reusable subprocesses.

* Corresponding author. Tel.: +41 44 724 8395; fax: +41 44 724 8953.
E-mail addresses: jussi.vanhatalo@ieee.org (J. Vanhatalo), hvo@zurich.ibm.com (H. Völzer), koe@zurich.ibm.com (J. Koehler).
[1] Tel.: +41 44 724 8111; fax: +41 44 724 8953.

0169-023X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2009.02.015

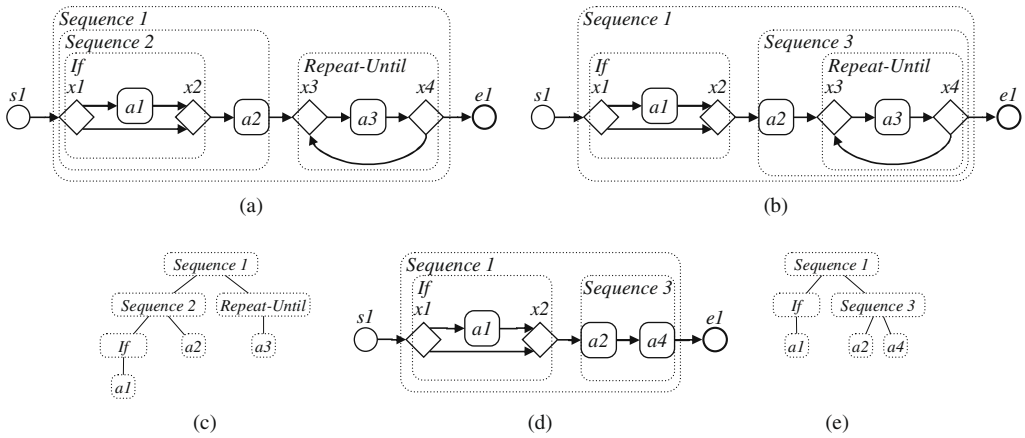

Fig. 1. (a), (b) Two parses of the same workflow graph. (c) Parse tree corresponding to (a). (d) Workflow graph obtained by a local change and its parse. (e) Parse tree corresponding to (d).

For a roundtripping between a BPMN diagram and a BPEL process, it is desirable that the decomposition be unique, i.e., the same BPMN diagram always translates to the same BPEL process. Consider, for example, the workflow graph in Fig. 1a. The translation algorithm proposed by Ouyang et al. [5] is nondeterministic. It may produce one of the two parses shown in Fig. 1a and b, depending on whether the if-block or the repeat-until-block is found first by the parsing algorithm.

One idea to resolve some of this nondeterminism is to define priorities on the syntactic categories to be found [5–7]. For example, if in each step the parsing algorithm tries to find sequences first, then if-blocks and then repeat-until-blocks, we can only obtain the parse in Fig. 1a in our example. However, this can introduce another problem. If we change a single block, say, the repeat-until block by replacing it, e.g. by a single task, we obtain the workflow graph shown in Fig. 1d. Fig. 1d also shows the parse we obtain with the particular priorities mentioned above. The corresponding parse tree is shown in Fig. 1e. It cannot be derived from the tree in Fig. 1c by just a local change, viz., by replacing the Repeat-Until subtree. For a roundtripping between a BPMN diagram and a BPEL process, it would be much more desirable that a local change in the BPMN diagram also result in only a local change in the BPEL process. Replacing a block in the BPMN diagram would therefore only require replacing the corresponding block in the BPEL process. We then call a such decomposition *modular*. The existing approach to the BPMN to BPEL translation problem [5] is not modular. Furthermore, it does not provide, because of the above problems, a specification of the translation that is independent of the actual translation algorithm.

A unique and modular decomposition is provided by the *program structure tree* defined by Johnson et al. [8,9] for sequential programs. It was applied to workflow graphs by Vanhatalo et al. [2] to find control-flow errors. The corresponding decomposition for our first example is shown in Fig. 2. It uses the same notion of a block as Ouyang et al. [5] do, that is, a block is a connected subgraph with a single entry and a single exit edge. But in contrast to the approach of Ouyang et al. [5], non-maximal sequences are disregarded in the program structure tree. For example, Sequence 2 in Fig. 1a [likewise Sequence 3 in subfigure (b)] is non-maximal: it is in sequence with another block.

Another general requirement for parsing is to find as much structure as possible, i.e., to decompose into blocks that are as fine as possible. As we will see (cf. Section 6), this allows us to map more BPMN diagrams to BPEL in a structured way. It has also been argued [5] that the BPEL process is more readable if it contains more blocks. Furthermore, debugging is easier when an error is local to a small block rather than to a large one.

In this paper, we provide a new decomposition that is finer than the program structure tree as defined by Johnson et al. [8,9]. It is based on and extends similar work for sequential programs by Tarjan and Valdes [10]. The underlying notion of a block is a connected subgraph with unique entry and exit *nodes* (as opposed to *edges* in the previous approach). Accordingly, all blocks of the previous approach are found, but more may be found, resulting in a more refined parse tree. Therefore, we call our decomposition the *refined process structure tree* (*RPST*, for short). We prove that our RPST is unique and modular.

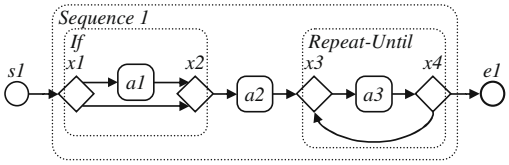

Fig. 2. Modular decomposition of the process from Fig. 1.

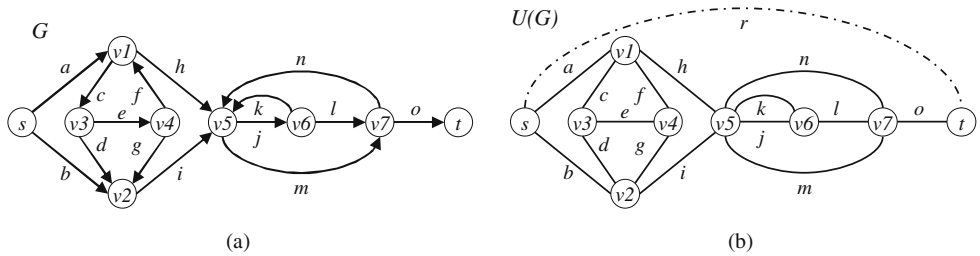

Fig. 3. (a) Two-terminal graph G, and (b) its undirected version $U(G)$, where r is the return edge.

Our parsing technique is not restricted to workflow graphs. It can be applied to any *two-terminal graph* (defined in Section 2.1). It can therefore be applied to Workflow (Petri) Nets, YAWL, flow charts, control-flow graphs of programs, and similar directed graph representations.

Some of these results have been presented earlier in a conference paper [11]. In this version, we extend the previous publication by presenting proofs of our propositions and theorems, and by providing a new, simple independent characterization of the RPST, which gives an additional justification of our particular choice of the decomposition showing that the RPST is optimal in the following sense. We cannot obtain any finer decomposition than the RPST that is based on the same definition of fragment, unless we are willing to make some choice between fragments that cannot be included in a same parse tree. Later we argue that such a choice would be somewhat arbitrary, not dependent on the structural information we would like to detect.

We prove that this new characterization is equivalent with our previous constructive characterization [11], which serves as a basis for computing the decomposition in linear time. The computation is based on the triconnected components of a biconnected graph, which can also be computed in linear time, for example by using an algorithm by Hopcroft and Tarjan [12].

The paper is structured as follows. In Section 2, we define the RPST through our simple independent characterization. In Section 3, we present our constructive characterization for the RPST. In Section 4, we prove some properties of the RPST including its modularity. In Section 5, we describe how to compute the RPST in linear time. Proofs can be found in the appendix.

2. The refined process structure tree

First we review the notion of a fragment from existing literature in Section 2.1. In Section 2.2, we define the RPST based on our particular selection of fragments that we call the *objective fragments*.

2.1. Fragments

We start by recalling some basic notions of graph theory. A *multi-graph* is a graph in which two nodes may be connected by more than one edge. This can be formalized as a triple $G = (V, E, M)$, where V is the set of nodes, E the set of edges and M a mapping that assigns an ordered or unordered pair of nodes to each edge – for a directed or undirected multi-graph, respectively. We will use multi-graphs throughout the paper, directed and undirected, but we will call them graphs for simplicity.

Let G be a graph. If e is an edge of G that connects two nodes u and v, we also say that u and v are *incident* to e, e is *incident* to u and v, and nodes u and v are *adjacent*.

Workflow graphs are based on *two-terminal graphs*.[2] A *two-terminal graph* (*TTG* for short) is a directed graph G without self-loops such that there is a unique source node s and a unique sink node $t \neq s$ and each node v is on a directed path from s to t. The *undirected version* of G, denoted $U(G)$, is the undirected graph that results from ignoring the direction of all the edges of G and adding an additional edge between the source and the sink. The additional edge is called the *return edge* of $U(G)$. Fig. 3 shows examples of (a) a two-terminal graph G, and (b) its undirected version $U(G)$, where r is the return edge.

For a subset F of edges, let V_F denote the set of nodes that are incident to some edge in F and let G_F denote the subgraph with nodes V_F and edges F. We say that G_F is *formed by* F.

Definition 1 (*Boundary node, entry, exit, fragment*). Let G be a TTG and F a subset of its edges such that G_F is a connected subgraph of G.

1. A node $v \in V_F$ is a *boundary node* of F if it is the source or sink node of G, or if G has edges $e \in F$ and $e' \notin F$ such that v is incident to e and e'.
2. A boundary node v is an *entry* of F if no incoming edge of v is in F or if all outgoing edges of v are in F.

[2] A workflow graph is a two-terminal graph in which each node is labeled with some control flow logic such as AND, OR, etc.

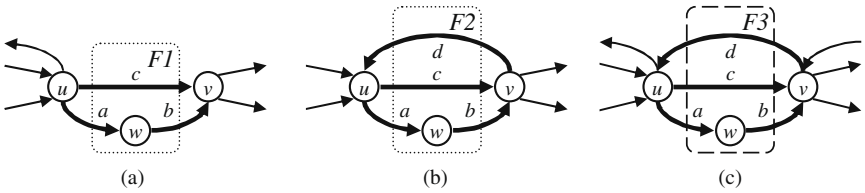

Fig. 4. (a), (b) Examples and (c) counterexamples of entry, exit and fragment.

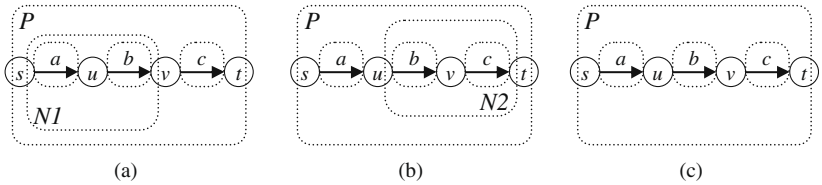

Fig. 5. (a), (b) Two maximal decompositions, (c) a non-maximal decomposition.

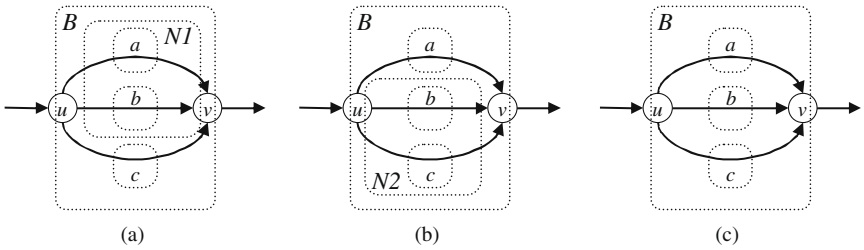

Fig. 6. (a), (b) Two maximal decompositions, (c) a non-maximal decomposition.

3. A boundary node v is an *exit* of F if all incoming edges of v are in F or if no outgoing edge of v is in F.
4. F is called a *fragment* of G if it has exactly two boundary nodes, an entry and an exit. Let $\mathscr{F}(u,v)$ denote the set of all fragments with entry u and exit v.

A fragment F is *trivial* if F contains exactly one edge e. If a fragment is not trivial, we call it *non-trivial*. As a TTG contains no self-loops, any set F of edges containing only one edge is a fragment.

Fig. 4 shows examples of fragments. A fragment is indicated as a dotted box. It contains all those edges that either are inside the box or cross the boundary of the box. Thus, the box in subfigure (a) denotes the fragment $F1 = \{a, b, c\}$. Node u is the entry and node v is the exit of $F1$. In subfigure (b), $F2 = \{a, b, c, d\}$ is a fragment with entry u and exit v. In subfigure (c), $F3 = \{a, b, c, d\}$ has two boundary nodes, u and v, neither of them is an entry or an exit of $F3$. Therefore, $F3$ is not a fragment.

Note that it can be checked locally whether a boundary node is an entry or an exit. This notion of fragment was proposed by Tarjan and Valdes [10], where a TTG modeled the control flow of a sequential program. When control flows through any of the edges of a fragment, then it must have flown through the entry before and must flow through the exit after. Their notion of fragment is, in a sense, the most general notion of fragment having this property that can still be verified locally [10].

2.2. Objective fragments and the refined process structure tree

We want to find a decomposition of a workflow graph into fragments. Formally such a *decomposition* (or a *parse*) of a TTG $G = (V, E, M)$ is a hierarchy of fragments, i.e., a set $\mathscr{E}$ of fragments such that $E \in \mathscr{E}$, and no two fragments in $\mathscr{E}$ *overlap*.[3] A trivial example of such a decomposition is the set $\{E\}$. A decomposition $\mathscr{E}$ is *maximal* if there exists no fragment F such that $\mathscr{E} \cup \{F\}$ is also a decomposition.

Fig. 5 shows a TTG and (a)–(b) its two maximal decompositions and (c) a non-maximal decomposition. Fig. 6 shows another such example. In case two fragments overlap as $N1$ and $N2$ do in Figs. 5 and 6, no decomposition contains both of them. Therefore, in order to obtain a unique decomposition, we must decide to exclude either one of them or both from the decom-

[3] Where we say that two fragments F_1 and F_2 *overlap* if $F_1 \cap F_2 \neq \emptyset$, $F_1 \setminus F_2 \neq \emptyset$ and $F_2 \setminus F_1 \neq \emptyset$. Therefore, if two fragments F_1 and F_2 do not overlap, they are either nested ($F_1 \subseteq F_2$, or $F_2 \subseteq F_1$) or disjoint ($F_1 \cap F_2 = \emptyset$).

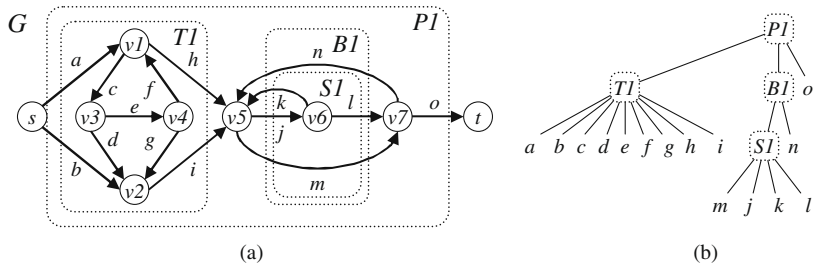

Fig. 7. (a) The objective fragments of G, and (b) the refined process structure tree of G.

position. As there is no reasonable argument to favor $N1$ over $N2$, or vice versa, that would break the symmetry between them, we exclude both fragments. Hence, we obtain the non-maximal decomposition shown in Figs. 5c and 6c, respectively. These decompositions are obtained in general by excluding all those fragments that overlap with *some* fragment. A fragment that does not overlap with any other fragment is contained in each maximal decomposition. Hence we call it *objective* fragment. The set of objective fragments is clearly a decomposition which is uniquely determined. This is how we define the *refined process structure tree*.

Definition 2 (*Refined process structure tree (RPST)*). Let $G = (V, E, M)$ be a TTG.

1. A fragment of G is *objective* if it does not overlap with any other fragment of G.
2. The set of all objective fragments of G is called the *RPST decomposition of G*.
3. The corresponding parse tree is called the *refined process structure tree (RPST)* of G, i.e., the RPST is the tree of the objective fragments such that the parent tree node of an objective fragment F is the smallest objective fragment that properly contains F.

If a fragment is not objective, we call it *non-objective*. Fragments $N1$ and $N2$ in Figs. 5 and 6 are examples of non-objective fragments. Each non-objective fragment occurs in some maximal decomposition, but not in all.

Fig. 7 shows (a) a TTG G and its objective fragments, and (b) its RPST. The root of the RPST is the largest fragment that contains all the edges of G, and has the source and the sink as the boundary nodes. If a fragment F is the parent tree node (*parent*, for short) of a fragment F' in the RPST, we say F' is a *child* of F.

Note that each trivial fragment is also an objective fragment. As there is a one-to-one correspondence between edges and trivial fragments, we represent each trivial fragment by the respective edge in the RPST. Note that each trivial fragment is a leaf node of the RPST, whereas each non-trivial fragment is a non-leaf node.

3. An equivalent characterization of the RPST

The definition of the RPST is quite simple but it does not give us a method to compute the RPST efficiently. For this purpose, we give an equivalent characterization of objective fragments (and hence the RPST) in this section, which will allow us to provide a linear time algorithm to compute the RPST. We also use this characterization in Section 4 to show how the RPST represents not only the objective fragments but also all the non-objective fragments, and to prove that the RPST is modular.

This characterization is based on the relationship of fragments and *triconnected components* that we prove in Section 3.1. In Section 3.2, we define various types of fragments including *bond fragments* on which we base our constructive characterization of objective fragments in Section 3.3.

3.1. Triconnected components

Tarjan and Valdes [10] have shown that the fragments of a TTG are closely related to the *triconnected components* of its undirected version. We have to introduce a few more notions from graph theory.

Let G be an undirected graph. The following notions are also used for directed graphs by ignoring the direction of their edges. Let W be a subset of nodes of G. A *path* from v_0 to v_k is an alternating sequence $\pi = v_0, e_1, v_1, e_2, \ldots, v_k$ of nodes and edges of G, such that e_i is incident to v_{i-1} and v_i for every $i = 1, \ldots, k$. Two nodes $u, v \notin W$ are *connected without* W if there is a path that contains u and v but no node $w \in W$. For instance, in Fig. 3a nodes s and t are connected without $v6$, but not connected without $v5$. Two edges e, f are *connected without* W if there exists a path containing e and f in which a node $w \in W$ may only occur as the first or last element. A graph without self-loops is *k-connected*, $k > 0$, if it has at least $k + 1$ nodes and for every set W of $k - 1$ nodes, any two nodes $u, v \notin W$ are connected without W. We say *connected* for 1-connected, *biconnected* for 2-connected and *triconnected* for 3-connected. A *separation point* (*separation pair*) of G is a node u (pair $\{u, v\}$ of nodes) such that there exists two nodes that are not connected without $\{u\}$ (without $\{u, v\}$). Therefore a graph is biconnected (triconnected) if and only if it has no separation point (separation pair). For instance in Fig. 3, G is not biconnected, because $v5$ is a separation point, whereas $U(G)$ is biconnected, because it has no separation points. $U(G)$ is not triconnected, because

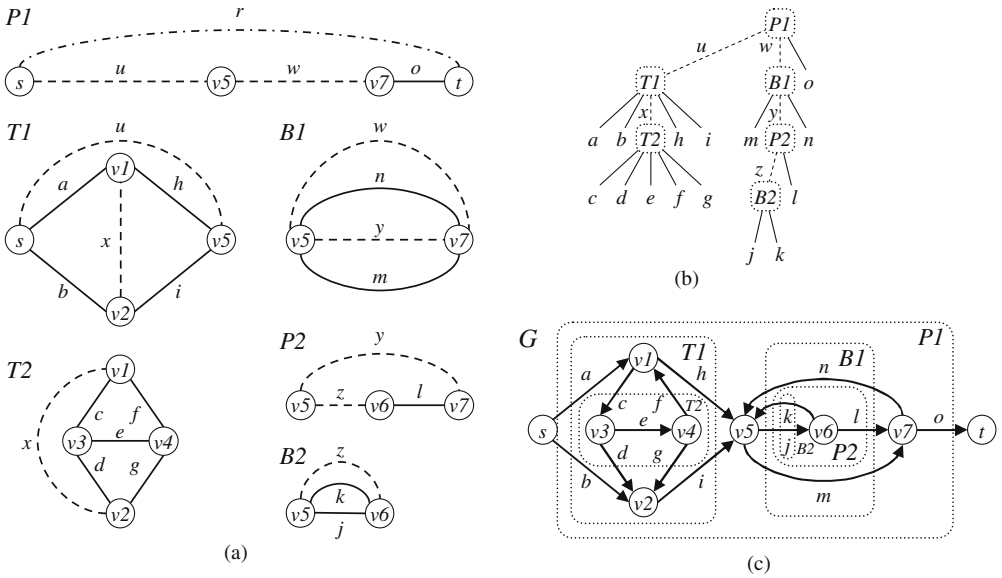

Fig. 8. (a) The triconnected components of $U(G)$ in Fig. 3, (b) the tree of the triconnected components of $U(G)$, and (c) the corresponding component subgraphs of G.

$\{v5, v7\}$ is a separation pair. In Fig. 8a, $T2$ is an example of a triconnected graph if the dashed edge x is considered as an ordinary edge.

We say that a graph is *weakly biconnected* if it is biconnected or if it contains exactly two nodes and at least two edges between these two nodes. For instance, in Fig. 8a, $B1$ is weakly biconnected, but not biconnected.

Throughout the paper, we assume that $U(G)$ is weakly biconnected. This can easily be achieved by splitting each separation point into two nodes, where the only outgoing edge of the first node is the only incoming edge of the second node. [4]

Let $\{u, v\}$ be a pair of nodes. A *separation class* with respect to (w.r.t.) $\{u, v\}$ is a maximal set S of edges such that any pair of edges in S is connected without $\{u, v\}$; S is a *proper separation class* or *branch* if it does not contain the return edge; $\{u, v\}$ is called a *boundary pair* if there are at least two separation classes w.r.t. $\{u, v\}$. Note that a pair $\{u, v\}$ of nodes is a boundary pair if and only if it is a separation pair or u and v are adjacent in G. For instance in Fig. 3b, $\{v5, v7\}$ and $\{v6, v7\}$ are boundary pairs. The pair $\{v5, v7\}$ is also a separation pair, but $\{v6, v7\}$ is not.

Each weakly biconnected graph can be uniquely decomposed into a set of graphs, called its *triconnected components* [12], where each triconnected component is either a *bond*, a *polygon* or a triconnected graph. A *bond* is a graph that contains exactly two nodes and at least two edges between them. A *polygon* is a graph that contains at least three nodes, exactly as many edges as nodes such that there is a cycle that contains all its nodes and all its edges.

Part (a) in Fig. 8 shows the six triconnected components of graph $U(G)$ from Fig. 3. $P1$ and $P2$ are polygons, $B1$ and $B2$ are bonds, and $T1$ and $T2$ are triconnected graphs. Each component contains *virtual edges* (shown as dashed lines), which are not contained in the original graph. They contain the information on how the components are related to each other: Each virtual edge occurs in exactly two components, whereas each original edge occurs in exactly one component. For example, the virtual edge x occurs in components $T1$ and $T2$. In component $T1$, x represents the component $T2$, whereas x represents $T1$ in $T2$. Therefore, we obtain the original graph by merging the triconnected components at the virtual edges (which removes them).

The triconnected components can be arranged in a tree, cf. Fig. 8b, where two components are connected if they share a virtual edge. The root of the tree is the unique component that contains the return edge. Each original edge is also shown in the tree under the unique component that contains that edge. Therefore, each component C determines a set F of edges of the original graph, namely all the leafs of the subtree that C corresponds to. For example, component $T1$ determines the set $F = \{a, b, c, d, e, f, g, h, i\}$ of edges. We call the subgraph formed by such a set F of edges the *component subgraph* of C. Fig. 8c shows the component subgraphs of G. Note that the component subgraphs $B1$, $P1$ and $T1$ are fragments, whereas $B2$, $P2$ and $T2$ are not. There are also fragments that are not component subgraphs, for instance, $\{j, k, l, m\}$.

The precise definition of the triconnected components [13,14,12] is rather lengthy and is therefore presented in Appendix A.1. Instead we present here the exact relationship between the triconnected components and fragments we are going to exploit. The proofs of the following two theorems are presented in Appendix A.2. First, we observe that triconnected components are closely related to boundary pairs.

Theorem 1. *A set $\{u, v\}$ of two nodes is a boundary pair of $U(G)$ if and only if*

[4] It is often assumed for workflow graphs that no node has both multiple incoming and multiple outgoing edges. In that case it follows that $U(G)$ is biconnected. See also Section 6.

1. *nodes u and v are adjacent in $U(G)$,*
2. *a triconnected component of $U(G)$ contains a virtual edge between u and v, or*
3. *a triconnected component of $U(G)$ is a polygon and contains u and v.*

Proof. Presented in Appendix A.2. □

We show examples based on $U(G)$ in Fig. 3b and its triconnected components in Fig. 8a. For instance, the boundary pair $\{v6, v7\}$ contains two adjacent nodes of $U(G)$, the boundary pair $\{v1, v2\}$ corresponds to a virtual edge x of $T2$, and the boundary pair $\{s, v7\}$ contains two nodes of the polygon $P1$. Boundary pairs are closely related to fragments as follows.

Theorem 2.

1. *If $F \in \mathscr{F}(u, v)$, then $\{u, v\}$ is a boundary pair of $U(G)$ and F is the union of one or more proper separation classes w.r.t. $\{u, v\}$.*
2. *Let $\{u, v\}$ be a boundary pair of $U(G)$ and F the union of one or more proper separation classes w.r.t. $\{u, v\}$. If u is an entry of F and v is an exit of F, then $F \in \mathscr{F}(u, v)$.*

Proof. Presented in Appendix A.2. □

For instance, the boundary pair $\{v5, v7\}$ has three proper separation classes $\{m\}$, $P2 = \{j, k, l\}$, and $\{n\}$. $P2$ is not a fragment, because $v5$ is neither its entry nor its exit, whereas $\{m\} \in \mathscr{F}(v5, v7)$ and $\{n\} \in \mathscr{F}(v7, v5)$ are fragments. The union of $P2$ and $\{m\}$ is a fragment, whereas $P2 \cup \{n\}$ and $\{m\} \cup \{n\}$ are not. $P2 \cup \{m\} \cup \{n\}$ is a fragment.

Theorem 1 says that the boundary pairs can be obtained from the triconnected components while Theorem 2 says that the fragments can be obtained from the boundary pairs.

3.2. Sequences and bond fragments

We define various types of fragments on which we base our constructive characterization of the objective fragments in Section 3.3.

Fig. 9 shows examples of fragments $N1$ and $N2$ that each overlap with some other fragment. As we discussed in Section 2.2, we exclude sequences $N1$ and $N2$ in Fig. 9a from the RPST, as they overlap. We can distinguish these overlapping sequences from the other sequences, such as P in Fig. 5, based on the following definition.

Definition 3 (*Maximal sequence*).

1. If $F_0 \in \mathscr{F}(v_0, v_1)$ and $F_1 \in \mathscr{F}(v_1, v_2)$ such that $F_0 \cup F_1 = F \in \mathscr{F}(v_0, v_2)$, we say that F_0 and F_1 are *in sequence* (likewise: F_1 and F_0 are in sequence) and that F is a *sequence*.
2. F is a *maximal sequence* if there is no fragment F_2 such that F and F_2 are in sequence.

If a sequence is not maximal, we call it *non-maximal*. In Fig. 5, $N1$ and $N2$ are non-maximal sequences, whereas $\{a, b, c\}$ is a maximal sequence. We observe that each non-maximal sequence overlaps with some other non-maximal sequence, and is therefore non-objective. We also note that each maximal sequence does not overlap with any other sequence. The proofs for these observations are included in the proof of Theorem 7 presented in Section 3.3.

Fig. 9b and c show examples of non-objective fragments $N1$ and $N2$, for which it is less straightforward to distinguish them from the objective fragments than for the non-maximal sequences. In order to distinguish these non-objective fragments from the objective ones, we first define various types of *bond fragments* as follows.

Definition 4 (*Bond fragments*). Let S be a proper separation class (i.e., a branch) w.r.t. $\{u, v\}$. S is *directed from u to v* if it contains neither an incoming edge of u nor an outgoing edge of v. $\mathscr{D}(u, v)$ denotes the set of directed branches from u to v. S is *undirected* if it is neither in $\mathscr{D}(u, v)$ nor in $\mathscr{D}(v, u)$. The set of undirected branches between u and v is denoted by $\mathscr{U}(u, v)$. A fragment $X \in \mathscr{F}(u, v)$ is

1. a *bond fragment* if it is the union of at least two branches from $\mathscr{D}(u, v) \cup \mathscr{U}(u, v) \cup \mathscr{D}(v, u)$.
2. a *directed bond fragment* if it is the union of at least two branches from $\mathscr{D}(u, v) \cup \mathscr{U}(u, v)$.

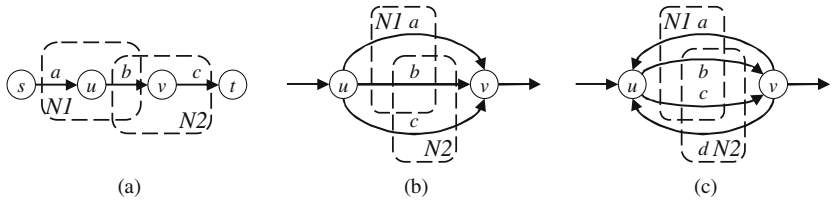

Fig. 9. Examples of overlapping fragments.

3. a *semi-pure bond fragment* if it is the union of at least two branches from $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$, and
   (a) there exists no $Y \in \mathscr{U}(u,v)$ such that $Y \subseteq X$, Y has an edge incoming to u, or
   (b) there exists no $Y \in \mathscr{U}(u,v)$ such that $Y \subseteq X$, Y has an edge outgoing from v.
4. a *pure bond fragment* if it is the union of at least two branches from $\mathscr{D}(u,v)$.

Note that the various bond-fragment types form a hierarchy, i.e., each pure bond fragment is a semi-pure bond fragment, each semi-pure bond fragment is a directed bond fragment, and each directed bond fragment is a bond fragment. Fig. 10 shows examples of various classes of bond fragments that do not belong to a lower class. We say that a bond fragment is *proper* if does not belong to a lower class. For instance, a proper semi-pure bond fragment is not a pure bond fragment.

Based on these various types of bond fragments, we define the *maximal* and the *canonical* bond fragments as follows.

Definition 5 (*Maximal bond fragment, canonical bond fragment*).

1. A bond fragment $F \in \mathscr{F}(u,v)$ is *maximal* if there is no bond fragment $F' \in \mathscr{F}(u,v)$ that properly contains F. A directed (semi-pure) [pure] bond fragment $F \in \mathscr{F}(u,v)$ is *maximal* if there is no directed (semi-pure) [pure] bond fragment $F' \in \mathscr{F}(u,v)$ that properly contains F.
2. A bond fragment $F \in \mathscr{F}(u,v)$ is *canonical* if
   (a) F is a maximal bond fragment, a maximal directed bond fragment, or a maximal semi-pure bond fragment, or
   (b) F is a maximal pure bond fragment, and F is not properly contained in any bond fragment $F' \in \mathscr{F}(v,u)$.

If a bond fragment is not canonical, we call it *non-canonical*. In Section 3.3, we prove that the canonical bond fragments are objective, whereas the non-canonical bond fragments are non-objective.

In the following, we prepare for this by studying how the bond fragments that share a common boundary pair are related to each other. Bond fragments are closed under composition.

Proposition 3. *If $X, Y \in \mathscr{F}(u,v)$ and $F = X \cup Y$, then $F \in \mathscr{F}(u,v)$. If X and Y are bond fragments, so is F. If X and Y are directed (semi-pure) [pure] bond fragments, so is F.*

Proof. Presented in Appendix A.3. □

Proposition 3 assures that a maximal bond fragment, maximal directed, maximal semi-pure, or maximal pure bond fragment is unique if it exists.

In the rest of this section, we show that any two bond fragments that share a common boundary pair do not overlap if at least one of them is canonical. In general, we can encounter two situations depending on whether the union of all branches between two boundary nodes is a fragment or not. Examples of these two cases are shown in Fig. 11.

In Fig. 11a, the union of all the branches between the boundary nodes u and v is a fragment B, because u is an entry and v is an exit. As B contains all the branches between u and v, B is the maximal bond fragment from u to v. D is the maximal directed bond fragment from u to v. S is the maximal semi-pure bond fragment from u to v. R is the maximal pure bond fragment from u to v.

Compared to part (a) of Fig. 11, part (b) has an additional edge outgoing from u that is outside of the union A of all the branches between the boundary nodes u and v. Because of this added edge, neither u nor v is an entry of A. Thus, A is not a fragment. $R1$ is the maximal pure bond fragment from u to v. S is the maximal semi-pure bond fragment from u to v. As there is no larger bond fragment from u to v, S is also the maximal directed and maximal bond fragment from u to v. $R2$ is the

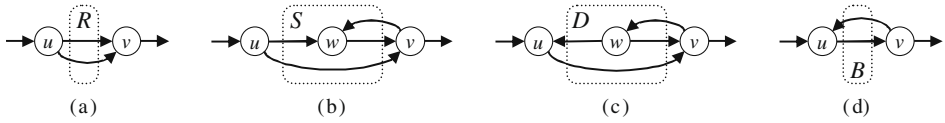

Fig. 10. Examples of (a) a pure bond fragment, (b) a semi-pure bond fragment, (c) a directed bond fragment, and (d) a bond fragment.

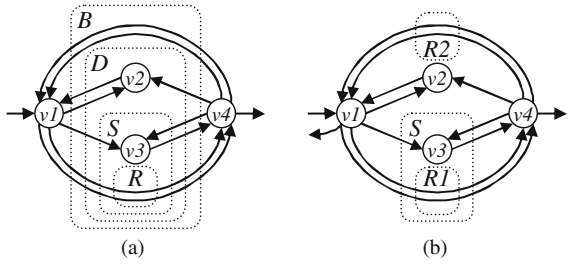

Fig. 11. Examples of various classes of maximal bond fragments.

maximal pure bond fragment from v to u. As there is no larger bond fragment from v to u, $R2$ is also the maximal semi-pure, maximal directed, and maximal bond fragment from v to u.

Next, we observe that a proper semi-pure bond fragment and a proper directed bond fragment are unique with respect to their boundary pairs if they exist.

Proposition 4. *Let G be a TTG such that its undirected version is weakly biconnected.*

1. *If $F, F' \in \mathscr{F}(u,v)$ are proper semi-pure bond fragments of G, then $F = F'$.*
2. *If $F, F' \in \mathscr{F}(u,v)$ are proper directed bond fragments of G, then $F = F'$.*

Proof. Presented in Appendix A.3. □

We can show now that a canonical bond fragment does not overlap with any other (canonical or non-canonical) bond fragment that shares the same boundary nodes.

Proposition 5. *Let G be a TTG such that its undirected version is weakly biconnected.*

1. *Let $X, Y \in \mathscr{F}(u,v)$ be bond fragments of G, and X be canonical. Then $X \subseteq Y$, or $Y \subseteq X$.*
2. *Let $X \in \mathscr{F}(u,v)$ and $Y \in \mathscr{F}(v,u)$ be bond fragments of G, and X or Y be canonical. Then $X \subseteq Y$, $Y \subseteq X$, or $X \cap Y = \emptyset$.*

Proof. Presented in Appendix A.3. □

3.3. The Constructive characterization: the canonical fragments

We give a constructive characterization of the objective fragments. For this, we define the *canonical fragments*, and prove that the objective fragments and the canonical fragments coincide.

Definition 6. (Canonical fragment). A fragment is *canonical* if it is a maximal sequence, a canonical bond fragment, or neither a sequence nor a bond fragment.

If a fragment is not canonical, we call it *non-canonical*. Note that each trivial fragment is canonical. Part (a) of Fig. 7 shows the non-trivial canonical fragments of graph G. $P1$ is a maximal sequence. $S1 \in \mathscr{F}(v5, v7)$ is a maximal semi-pure bond fragment, and $B1 \in \mathscr{F}(v5, v7)$ is a maximal bond fragment. $T1$ is neither a sequence nor a bond fragment.

In order to prove that the canonical fragments are exactly the objective fragments, we also give a constructive characterization of the non-canonical fragments.

Proposition 6. *Let G be a TTG, $F \in \mathscr{F}(u,v)$ be a fragment of G. F is a non-canonical fragment if and only if either*

1. *F is a non-maximal sequence,*
2. *F is a non-maximal pure bond fragment,*
3. *F is a non-maximal proper bond fragment, or*
4. *F is a maximal pure bond fragment and F is properly contained in a bond fragment $F' \in \mathscr{F}(v,u)$.*

Proof. Presented in Appendix A.3. □

For example, the fragments $N1$ and $N2$ in Fig. 9a are non-maximal sequences. The fragments $N1$ and $N2$ in Fig. 9b are non-maximal pure bond fragments. The fragments $N1$ and $N2$ in Fig. 9c are non-maximal proper bond fragments. The fragment $\{a,d\}$ in Fig. 9c is a maximal pure bond fragment that is properly contained in a bond fragment $\{a,b,c,d\}$.

Finally, we show that the objective fragments and canonical fragments coincide.

Theorem 7. *Let F be a fragment of a TTG. F is canonical if and only if F is objective.*

Proof. Presented in Appendix A.3. □

4. Properties of the RPST

In this section, we give two additional justifications for our definition of the RPST. First, we show in Section 4.1 how all fragments can be easily derived from the canonical fragments, for instance, as a combination of child fragments in the RPST. Secondly, we prove in Section 4.2 that the RPST is modular.

4.1. Computing all fragments from the canonical fragments

The following proposition indicates how to derive all fragments (also the non-canonical) from the canonical fragments. This is useful for example if one wants to find the smallest fragment that contains some given set of graph elements.

Proposition 8. *Let F be a set of edges in a TTG. F is a fragment if and only if F is a canonical fragment or F is*

1. *a union of consecutive child fragments of a maximal sequence,*
2. *a union of child fragments of a maximal pure bond fragment, or*
3. *a union of child fragments of a maximal bond fragment B such that B is not a maximal directed bond fragment.*

Proof. Presented in Appendix A.4. □

To define consecutive child fragments of a sequence, the child fragments of a sequence are ordered left to right from the entry to the exit of the sequence. For example, the maximal sequence $P1$ in Fig. 7 has $T1$, $B1$ and o as ordered child fragments. Besides these canonical fragments and the maximal sequence, also the union of $T1$ and $B1$ ($B1$ and o) is a fragment. However, the union of $T1$ and o is not a fragment, as these are not consecutive child fragments, i.e., they do not share a boundary node.

Part (a) of Fig. 12 shows a maximal pure bond fragment $R = \{a, b, c\}$. Its child fragments are $\{a\}$, $\{b\}$, and $\{c\}$. It follows from Proposition 8 that $\{a, b\}$, $\{b, c\}$, and $\{a, c\}$ are the non-canonical fragments in R. Part (b) of Fig. 12 shows a maximal bond fragment $B = \{a, b, c, d\} \in \mathscr{F}(u, v)$ and a maximal directed bond fragment $R = \{a, b\} \in \mathscr{F}(u, v)$ such that $B \neq R$. It follows from Proposition 8 that $\{c, d\}$, $\{a, b, c\}$ and $\{a, b, d\}$ are the non-canonical fragments in B. Note that $\{a, c, d\}$ and $\{b, c, d\}$ are not fragments, because their boundary nodes are neither entries nor exits.

4.2. Modularity

Finally, we state what we mean by saying that our decomposition is modular.

Theorem 9. *Let G be a TTG and $X \in \mathscr{F}(u, v)$ be a canonical fragment of G. Let G' be the TTG obtained by replacing the subgraph that is formed by X by some other subgraph formed by a set of (fresh) edges X' such that $X' \in \mathscr{F}(u, v)$ is again a fragment of G' (but not necessarily canonical) with the same entry-exit pair as X. Assume that A is the parent fragment of X in G and $F \neq X$ is a child fragment of A in G. Let $A' = (A \setminus X) \cup X'$ and $F' = F$. Then A' and F' are canonical fragments of G' where F' is a child fragment of A' in G'.*

Proof. Presented in Appendix A.4. □

Theorem 9 means that a local change to the TTG, i.e., changing a canonical fragment X, only affects the RPST locally. The parent and siblings of a changed canonical fragment remain in the RPST in the same place and it follows inductively that this is also true for all canonical fragments above the parent and all canonical fragments below the siblings of X.

5. Computing the RPST

In this section, we describe an algorithm that computes the RPST in linear time. We have extended the algorithm by Valdes [15] to find all the canonical fragments (his algorithm produces a coarser decomposition, cf. Section 6). The algorithm has three high-level steps, which are illustrated in Fig. 13 and described in Algorithm 1. In Step 1, the tree of the triconnected components is computed, using e.g. the linear time algorithm by Hopcroft and Tarjan [12]. Gutwenger and Mutzel [14] present some corrections to this algorithm. We illustrate the computed triconnected components through the respective component subgraphs in Fig. 13.

In Step 2, we analyze each triconnected component to determine whether the respective component subgraph is a fragment. This can be done in linear time with the following approach that takes advantage of the hierarchy of fragments. We analyze the tree of the triconnected components bottom-up – all children before a parent. For each child edge of a triconnected component c, we check whether it is incoming to or outgoing from one of the two boundary nodes of c. We count these edges to determine whether a boundary node is an entry, an exit, or neither. Based on this information, we can determine whether the respective component subgraph is a fragment. Note that when a triconnected component shares a boundary node with its parent, the same edges do not have to be counted twice, because an edge inside a child is also inside its parent. We find some canonical fragments in Step 2, but not all. For example, in Fig. 13, the fragments $P1$, $T1$, and $B1$ are found in Step 2, whereas the fragment $S1$ is not.

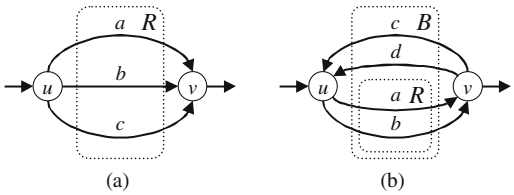

Fig. 12. Deriving non-canonical fragments from child fragments of (a) a maximal pure bond fragment and (b) a maximal bond fragment.

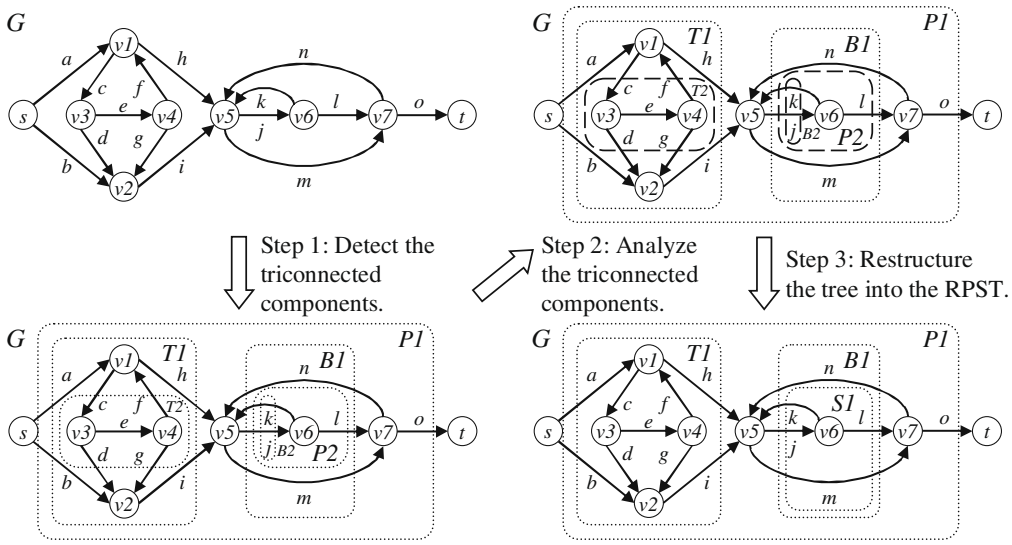

Fig. 13. The high-level steps of Algorithm 1. Step 1: Detect the triconnected components. Step 2: Analyze each triconnected component to determine whether the respective component subgraph is a fragment. Step 3: Create the missing canonical fragments and merge the triconnected components that are not fragments.

Algorithm 1. Computes the RPST for a two-terminal graph

buildRPST(Graph *G*)
Require: *G* is a weakly biconnected TTG.
{Step 1. Compute the tree of the triconnected components of the TTG}.
`Tree` := Compute the tree of the triconnected components of *G*.
{Step 2. Analyze each component to determine whether it is a fragment.}
for each `Component` *c* in `Tree` in a post-order of a depth-first traversal **do**
Count the number of edges (that are children of *c*) incoming to/outgoing from each boundary node. (4 counts)
Add these edge counts to the respective edge counts of the parent component for each shared boundary node.
Compare these edge counts to the total number of incoming/outgoing edges to determine whether each boundary node is entry, exit, or neither.
Based on these boundary node types, determine whether *c* is a fragment.
if *c* is a polygon **then**
Count the number of entry and exit nodes of the child components.
If a child component is a fragment, order the child components from entry to exit.
{Step 3. Restructure the tree of the triconnected components into the tree of the canonical fragments (the RPST).}
for each `Component` *c* in `Tree` in a post-order of a depth-first traversal **do**
if *c* is a polygon **then**
Merge consecutive child components (that are not fragments if any exist) if those form a minimal child fragment.
if *c* is not a fragment and *c* has at least two child fragments **then**
Create a maximal sequence (that contains a proper subset of children of *c*).
if *c* is a bond **then**
Classify each branch of *c* based on the edge counts of the boundary nodes of the respective child components of *c*.
if *c* is a fragment **then**
Based on the classifications of the branches, create the maximal pure, the maximal semi-pure, and the maximal directed bond fragment, if any exists.
else
Based on the classifications of the branches, create the maximal pure bond fragments, the maximal semi-pure bond fragment, if any exists.
for each `Component` *d* that has been created in this iteration **do**
Merge the child fragments of each component in the to-be-merged list of *d* to *d*.
if *c* is not a fragment **then**
Add *c* to the to-be-merged list (a linked list) of its parent component.
Concatenate the to-be-merged list of *c* to the to-be-merged list of its parent.
else
Merge the child fragments of each component in the to-be-merged list of *c* to *c*.
return `Tree`

In Step 3, we create the missing canonical fragments, and merge each component subgraph that is not a fragment to the smallest canonical fragment that contains it. This restructuring is based on the information computed in Step 2. New fragments are created only in those cases where a bond or a polygon contains canonical fragments that are not component subgraphs. Such a fragment is created as a union of at least two (but not all) children of this bond or polygon. We show examples in the following.

We process the tree of the triconnected components bottom-up as in Step 2. Thus, in Fig. 13, we can begin with $T2$. It contains no new canonical fragments, because it is neither a sequence nor a bond. $T2$ is not a fragment, because $v1$ is neither its entry nor its exit. Thus, it will be merged into its parent fragment $T1$, that is, the children of $T2$ become children of $T1$.

The bond $B2$ is not a fragment, so it will be merged. $B2$ contains no new canonical fragments, because it has only two children. The same applies to $P2$. More interestingly, $B1$ is a fragment and has three children. Each child of a bond is a branch, and we classify them to find out whether they form new canonical bond fragments. $\{m\}$ is a directed branch from $v5$ to $v7$, $P2$ is an undirected branch that has no outgoing edges from $v7$, and $\{n\}$ is a directed branch from $v7$ to $v5$. Note that the branches can be classified based on the counts of edges incident to each boundary node of a branch computed in Step 2. There is a new semi-pure bond fragment $S1 = \{m\} \cup P2$. $B2$ and $P2$ are merged to $S1$. $S1$ and $\{n\}$ become the children of the restructured $B1$. Finally, $P1$ and all its children are fragments, thus there is no need to restructure $P1$.

In the following examples we show polygons and bonds in which more restructuring is required. In Fig. 14a, $B1$ and $P1$ are not fragments. However, polygon $P1$ has two consecutive child fragments $\{d\}$ and $\{e\}$ that form a maximal sequence $P = \{d\} \cup \{e\}$. To determine whether a polygon contains such a new maximal sequence, we compute the number of entries and exits of its children already at the end of Step 2. A polygon that is not a fragment contains a maximal sequence as a union of its children if and only if its children together have at least three entries or at least three exits in total.

In Fig. 14b, $B1$, $P1$, $B2$, and $P2$ are not fragments and will be merged. Bond B is a fragment from $v1$ to $v4$ and has six branches: two edges as directed branches from $v1$ to $v4$, and an undirected branch $P2$ that has no edge incoming to the entry of B, an undirected branch $P1$ that has both an edge incoming to the entry of B and an edge outgoing from the exit of B, and another two edges as directed branches from $v4$ to $v1$. The directed branches from the entry to the exit of B form a new maximal pure bond fragment R. The union of $P2$ and R is a new maximal semi-pure bond fragment S. The union of $P1$ and S is a new maximal directed bond fragment. D and the remaining two directed branches are the children of B. $B1$ and $P1$ are merged to D, and $B2$ and $P2$ to S. P is a maximal sequence.

Fig. 14c shows an example of a bond B that is not a fragment, but its children form new canonical fragments. As there are at least two directed branches to each direction, these branches form two new pure bond fragments, $R1$ and $R2$. The union of $R1$ and branch $P2$ is a semi-pure bond fragment S. Thus, $B2$ and $P2$ are merged to S. The polygon P has four children $\{a\}$, $B3$, B, and $\{b\}$. $B3$ and B not fragments, but the union of these consecutive siblings is a fragment. Thus, B is merged to $B3$ to form a new fragment M. $B1$ and $P1$ are also merged to M. The fragment P has only three children.

Each step of the algorithm can be performed in linear time. Thus, also the entire algorithm has linear time complexity. We conclude:

Theorem 10. *The RPST of a TTG G can be computed in time linear in the number of edges of G.*

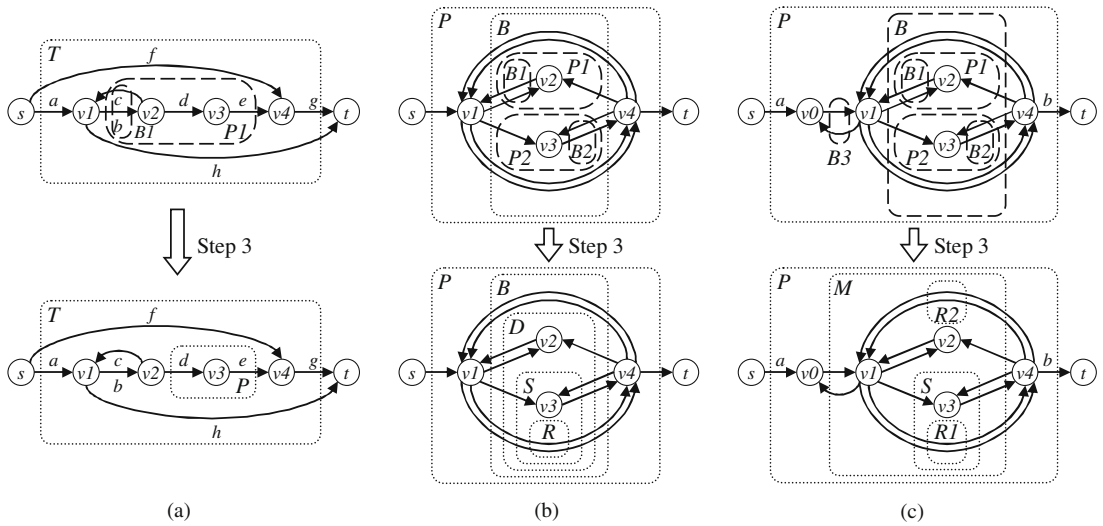

Fig. 14. Step 3: From the tree of the component subgraphs to the tree of the canonical fragments.

6. Conclusion

We have presented a modular technique of workflow graph parsing to obtain fine-grained fragments with a single entry and single exit node. The result of the parsing process, the refined process structure tree, is obtained in linear time. We have mentioned a couple of use cases in Section 1. Coarser-grained decompositions may be desirable for some use cases. Those can easily be derived from the refined process structure tree by flattening. One such coarser decomposition, which can be derived and which is also modular, is the decomposition into fragments with a single entry edge and a single exit edge presented by Vanhatalo, Völzer and Leymann [2]. The new, refined decomposition presented here allows us to translate more BPMN diagrams to BPEL in a structured way. As an example, consider the workflow graph in Fig. 15 and (a) its decomposition with the existing techniques [5,2] and (b) with our new technique. In Fig. 15a, X cannot be represented as a single BPEL block, whereas in Fig. 15b each fragment can be represented as a single BPEL block.

The main idea of our technique to use the tree of triconnected components (elsewhere also known as the SPQR tree) is taken from Tarjan and Valdes [10,15]. They describe an algorithm that produces a unique parse tree. However, they do not provide a specification of the parse tree, i.e., a definition of canonical fragments or claim or prove modularity. Moreover, our RPST is more refined than their parse tree. Fig. 14 shows examples of workflow graphs where this is the case. The fragments that are not identified by them are P in (a), D, S and R in (b), and S, $R1$ and $R2$ in (c).

We have made some simplifying assumptions about workflow graphs. The assumption that we have unique source and sink nodes can be lifted. One way to achieve this is to complete such a workflow graph into a TTG with an automatic technique presented by Vanhatalo et al. [16]. Also the assumptions that the undirected version of the workflow graph is weakly biconnected and does not contain self-loops can be lifted. This can be achieved by splitting some nodes into two nodes as we discuss in Section 3.1. The above techniques to lift these assumptions require some changes to the original graph. The necessary constructions to deal with these cases without changing the original graph will be presented in [13]. Thus, the only remaining assumption on workflow graphs is that each node is on a path from some source to some sink.

Note that we cannot obtain any finer decomposition than the RPST that is based on the same definition of fragment, unless we are willing to make some choice between two non-objective fragments.

The RPST is implemented as a part of the IBM WebSphere Business Modeler version 6.2 and IBM WebSphere Process Server version 6.2. It is used in a BPMN as well as in a BPEL context. To do so, workflow graphs are extracted from BPMN diagrams and BPEL process graphs by dedicated transformations. This intermediate representation of workflow graphs allows us to easily apply our technique to business process models represented in different languages.

Our RPST data structure represents the canonical fragments explicitly, and the non-canonical fragment implicitly as the latter can be easily derived from the former based on Proposition 8. Each canonical fragment is also associated with its parent fragment and its child fragments. The child fragments of sequences are ordered from its entry to its exit. Each edge is associated with its unique parent fragment, and each node with its parent fragments.

The RPST is used to speed up control-flow analysis of workflow graphs using the divide-and-conquer approach presented by Vanhatalo et al. [2]. Our implementation can also refactor a workflow graph based on its RPST as described by Vanhatalo et al. [16].

However, we did not implemented the RPST specifically for one of these use cases. Instead, we see the RPST as a fundamental parsing technique for workflow graphs, as it can be used as a basis of many other use cases, such as subprocess detection and extraction, abstraction, visualization, and process merging [3] techniques. Many other use cases that are based on parse trees and known from compiler theory, such as data-flow analysis and code optimization, seem to be relevant in the context of business process management.

Acknowledgement

The work published in this article was partially supported by the SUPER project (http://www.ip-super.org/) under the EU 6th Framework Programme Information Society Technologies Objective (Contract No. FP6-026850).

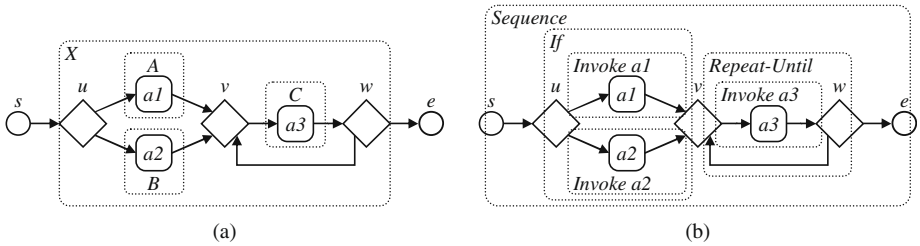

Fig. 15. A workflow graph and (a) decomposition presented in [5,2] and (b) our decomposition.

Appendix A. Appendix – Proofs

This appendix includes the definitions and the proofs that we exclude from the previous sections in order to enhance readability of this paper. In Section A.1, we define the triconnected components. In Section A.2, we prove the relationship between the fragments and the triconnected components. In Section A.3, we prove that the objective and the canonical fragments coincide. In Section A.4, we prove the properties of the RPST.

A.1. The triconnected components

In this section, we give a precise definition of *triconnected components* based on which we can prove the relationship between the fragments and the triconnected components. We define triconnected components through *split graphs*, *split components*, and *merge graphs*, and follow the presentation of Hopcroft and Tarjan [12] and Gutwenger and Mutzel [14].

Assume G is a TTG and its undirected version $U(G)$ is weakly biconnected. We split $U(G)$ into its triconnected components through multiple steps. In the first step, splitting $U(G)$ is based on *split pairs*. A boundary pair $\{u, v\}$ of a weakly biconnected graph G is a *split pair* of G unless (i) there are exactly two separation classes of G w.r.t $\{u, v\}$, and one has exactly one edge, or (ii) there are exactly three separation classes of G w.r.t $\{u, v\}$, and each has exactly one edge.

Fig. 16 shows (a) a TTG G, and (b) its undirected version $U(G)$, which has three split pairs $\{s, v\}$, $\{u, v\}$, and $\{u, t\}$. The boundary pairs $\{s, u\}$, $\{s, t\}$, and $\{v, t\}$ are not split pairs.

A weakly biconnected graph $G = (V, E, M)$ can be split into two *split graphs* based on a split pair $\{u, v\}$ as follows. Assume E_1 is the union of some separation classes in $\mathscr{S}(u, v)$ and $E_2 = E \setminus E_1$ such that E_1 and E_2 have at least two edges each. Two graphs $G_{E_1} = (V_{E_1}, E_1 \cup \{e\}, M_{E_1})$ and $G_{E_2} = (V_{E_2}, E_2 \cup \{e\}, M_{E_2})$ are formed by $E_1 \cup \{e\}$ and $E_2 \cup \{e\}$, where $e \notin E$ and e is incident to the nodes u and v. G_{E_1} and G_{E_2} are called *split graphs* of G with respect to $\{u, v\}$. The new edge e is called a *virtual edge*.

For example, Fig. 17a shows the split graphs G_{E_1} and G_{E_2} of $U(G)$ w.r.t. the split pair $\{s, v\}$. The edge e is a virtual edge, whereas the other edges of these split graphs are also edges of $U(G)$. Replacing a graph G with its split graphs G_{E_1} and G_{E_2} is called *splitting* G. The virtual edge identifies this split, for instance, it contains the information needed to merge two split graphs in order to obtain the original graph. Each split graph is a weakly biconnected multi-graph [12].

Some split graphs can be further split into their split graphs. Suppose a weakly biconnected graph G is split, and the obtained split graphs are split until no more splitting is possible. The resulting graphs are called *split components* of G. For example, in Fig. 17a G_{E_1} cannot be further split, its split graphs $G_{E_{21}}$, and $G_{E_{22}}$ shown in Fig. 17b. The graphs G_{E_1}, $G_{E_{21}}$, and $G_{E_{22}}$ are the split components of $U(G)$. Hopcroft and Tarjan [12] prove that each split component of a weakly biconnected graph G is either a *double bond*, *triple bond*, *triangle*, or triconnected graph. These special kinds of graphs are defined as follows.

A graph $G = (V, E, M)$ is a *bond* if V contains exactly two nodes u and v, E contains at least two edges and each edge in E is incident to both u and v. A *double (triple) bond* is a bond that contains exactly two (three) edges. In Fig. 17c, $G_{E_{22}}$ is a triple bond. Note that a split component G' of G is a double bond if and only if $G = G'$.

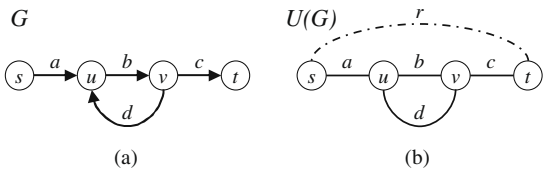

Fig. 16. (a) A TTG G, and (b) its undirected version $U(G)$.

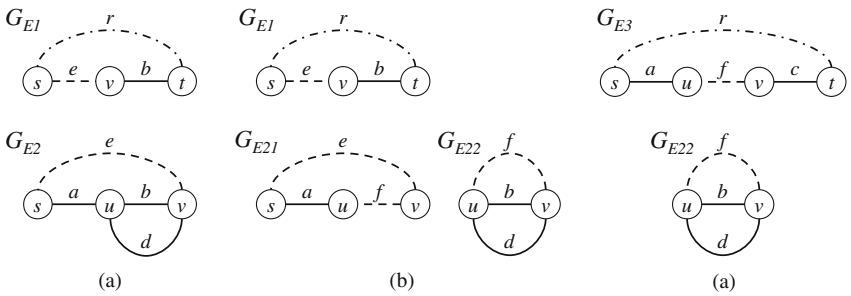

Fig. 17. (a) Two split graphs of G from Fig. 16, (b) the split components of G, and (c) the triconnected components of G.

A graph $G = (V, E, M)$ is a *polygon* if V contains at least three nodes, exactly as many edges as nodes and there exists a cycle of G that contains all nodes and edges of G. A *triangle* is a polygon that contains exactly three nodes. For example, in Fig. 17, G_{E_3} is a polygon, but not a triangle, whereas G_{E_1} is a triangle. G_{E_2} is neither a polygon or a bond.

Hopcroft and Tarjan [12] prove the following observations on split components. Each edge of G is contained in exactly one split component of G, and each virtual edge of a split component of G is contained in a exactly two split components of G. The split components of a graph G are not necessarily unique. However, unique triconnected components can be obtained from split graphs by merging some of these split graphs as follows.

Assume $G_{E_1} = (V_{E_1}, E_1, M_{E_1})$ and $G_{E_2} = (V_{E_2}, E_2, M_{E_2})$ are distinct split components of a weakly biconnected graph G containing the same virtual edge e. The graph $G' = (V', E', M')$ is called *merge graph* of G_{E_1} and G_{E_2}, where $V' = V_{E_1} \cup V_{E_2}$, $E' = (E' \cup E'') \setminus \{e\}$, and M' maps each edge in E' to the same pair of nodes as M_{E_1} or M_{E_2}. For example, in Fig. 17, G_{E_3} is the merge graph of the split components G_{E_1}, $G_{E_{21}}$. Replacing distinct split components G_{E_1} and G_{E_2} with a merge graph G' is called merging G_{E_1} and G_{E_2}.

Triconnected components of G are obtained from split components of G by merging any two bonds (polygons) that contain the same virtual edge, and replacing these bonds (polygons) with the obtained bond (polygon) until no more merging is possible. For example, G_{E_3} and $G_{E_{22}}$ in Fig. 17c are the triconnected components of $U(G)$ in Fig. 16b.

Hopcroft and Tarjan [12] prove the following observations on the triconnected components of a weakly biconnected graph. Each triconnected component of G is either a bond, polygon, or triconnected graph. Each edge of G is contained in exactly one triconnected component of G, and each virtual edge of a triconnected component of G is contained in exactly two triconnected components of G. The triconnected components of G are unique.

A.2. Proofs – The relationship between the fragments and the triconnected components

In this section, we prove the relationship between the fragments and the triconnected components through Theorems 1 and 2. First, we define the following notations. Recall that a directed edge e is mapped to an ordered pair of nodes (u, v). We denote this mapping $e \mapsto (u, v)$. Whereas, an undirected edge e is mapped to a multi-set of nodes $\{u, v\}$, and we denote this mapping $e \mapsto \{u, v\}$

Theorem 1. *A set $\{u, v\}$ of two nodes is a boundary pair of $U(G)$ if and only if*

1. *nodes u and v are adjacent in $U(G)$,*
2. *a triconnected component of $U(G)$ contains a virtual edge $e \mapsto \{u, v\}$, or*
3. *a triconnected component of $U(G)$ is a polygon and contains u and v.*

Proof. ($\Rightarrow$) Assume $\{u, v\}$ is a boundary pair of $U(G)$. A boundary pair is a split pair or a pair of adjacent nodes. Thus, we distinguish two cases:

1. Assume nodes u and v are adjacent in $U(G)$. Then, there is nothing to prove.
2. Assume $\{u, v\}$ is a split pair of $U(G)$. We construct the triconnected component of $U(G)$ as follows. First, we split $U(G)$ w.r.t $\{u, v\}$ into split graphs G_1 and G_2. The resulting split graphs contain a virtual edge $e \mapsto \{u, v\}$. We split G_1 and G_2 into their split components, which are also split components of $U(G)$. Let G_{e1} and G_{e2} be two of these split components, such that each G_{e1} and G_{e2} contains the virtual edge e. We build the triconnected components of $U(G)$ from these split components distinguishing the following subcases:
   (a) Assume G_{e1} and G_{e2} are bonds. Since G_{e1} and G_{e2} contain the same virtual edge, G_{e1} and G_{e2} and possibly other split graphs are merged to obtain a maximal bond G_B. It follows that G_B contains exactly two nodes u and v, which are connected by an edge of $U(G)$ or a virtual edge. Thus, nodes u and v are adjacent in $U(G)$, or triconnected component G_B of $U(G)$ contains a virtual edge $f \mapsto \{u, v\}$.
   (b) Assume G_{e1} and G_{e2} are polygons. Since G_{e1} and G_{e2} contain the same virtual edge, G_{e1} and G_{e2} and possibly other split graphs are merged to obtain a maximal polygon G_P. It follows, polygon G_P contains two nodes u and v.
   (c) Assume G_{e1} and G_{e2} are not both bonds, nor both polygons. Thus, G_{e1} and G_{e2} cannot merged along e, but each can be merged with some other split components to obtain triconnected components G_{e1m} and G_{e2m}. Since G_{e1} and G_{e2} are not both polygons and not both bonds, polygons are merged only with polygons, and bonds are merged bonds, then their merge graphs are not merged, and the resulting triconnected components are distinct. Each G_{e1m} and G_{e2m} contains a virtual edge $e \mapsto \{u, v\}$.

($\Leftarrow$) We distinguish the following cases based on the cases in this theorem:

1. Assume nodes u and v are adjacent. It follows that there exists an edge $e \mapsto \{u, v\}$ that is a separation class w.r.t $\{u, v\}$. It follows from the definitions of a TTG and the undirected version that the $U(G)$ of any TTG G has at least two edges. Thus, there exists another separation class w.r.t $\{u, v\}$. Thus, $\{u, v\}$ is a boundary pair.
2. Assume a triconnected component of $U(G)$ contains a virtual edge $e \mapsto \{u, v\}$. It follows from the definition of a split graph and a virtual edge that $\{u, v\}$ is a split pair. Since each split pair is a boundary pair, then $\{u, v\}$ is a boundary pair.

3. Assume a triconnected component P of $U(G)$ is a polygon and contains nodes u and v. Each pair of nodes in a polygon is a split pair of the polygon, or a pair of adjacent nodes. We already showed that a pair of adjacent node is a boundary pair. Each split pair of P is a split pair of $U(G)$. Since each split pair is a boundary pair, then $\{u, v\}$ is a boundary pair. □

For the next proof, we define the following notations. We denote the set of all separation classes of a graph G w.r.t. a set of nodes $\{u, v\}$ by $\mathscr{S}(u, v)$ of G. We denote the set of all proper separation classes of a graph G w.r.t a set of nodes $\{u, v\}$ by $\mathscr{P}(u, v)$ of G.

Theorem 2. *Let G be a TTG such that $U(G)$ is weakly biconnected.*

1. *If $F \in \mathscr{F}(u, v)$, then $\{u, v\}$ is a boundary pair of G and F is the union of one or more proper separation classes in $\mathscr{P}(u, v)$.*
2. *Let $\{u, v\}$ be a boundary pair of G and F the union of one or more proper separation classes in $\mathscr{P}(u, v)$. If u is an entry of F and v is an exit of F, then $F \in \mathscr{F}(u, v)$.*

Proof. Two parts of the theorem are proved separately.

1. Assume $F \in \mathscr{F}(u, v)$ and G_F is formed by F. It follows from the definition of a fragment that the subgraph formed by F is connected. Since u and v are the boundary nodes of F, G_F has no other boundary nodes, F contains at least one edge e, and at least one edge r (the return edge) is outside of the fragment then e and r are not connected without $\{u, v\}$. Thus, $\{u, v\}$ is a boundary pair of $G = (V, E, M)$. Since the return edge r is not an edge of G, and F contains only edges of G, then F does not contain r.
   If $r \mapsto \{u, v\}$ or $e \mapsto \{u, v\}$ in $U(G)$, then u and v are adjacent nodes and $\{u, v\}$ is a boundary pair. Otherwise, there exists a node x in G_F and a node y outside G_F such that $x \neq u$, $x \neq v$, $y \neq u$, and $y \neq v$. Assume x and y are connected without $\{u, v\}$, then there exists a node w such that $w \neq u$, $w \neq v$, an edge incident to w is in F, and an edge incident to w is outside F. Thus, w is a boundary node of F. This is a contradiction as the fragment F has exactly two boundary nodes u and v. Thus, $\{u, v\}$ is a separation pair of G, and also a boundary pair of G.
   Assume there is a separation class $S \in \mathscr{S}(u, v)$ that has an edge f in F and an edge g outside F. Then, f and g are connected without $\{u, v\}$. Assume there exists a node z such that $z \neq u$, $z \neq v$, and an edge incident to z is in F, and an edge incident to z is outside F. Thus, z is a boundary node of F. This is a contradiction as the fragment F has exactly two boundary nodes u and v. It follows that all edges in a separation class are either inside or outside F. Since $e \in F$, F contains at least one separation class. Since $r \notin F$, the separation class containing r is outside F. Thus, F is the union of one or more proper separation classes in $\mathscr{P}(u, v)$.
2. Assume $\{u, v\}$ is a boundary pair of $G = (V, E, M)$ and F is the union of one or more proper separation classes in $\mathscr{P}(u, v)$. Let G_F be formed by F. Since each subgraph formed by separation class is a connected subgraph of G, and contains nodes u and v, then G_F is a connected subgraph of G. Since u is the entry and v is the exit, then F has two boundary nodes. Assume there exists another boundary node w of F. Then there exists two edges e and e' incident to w such that $e \in F$ and $e \notin F$. Then F a proper separation class $S \in \mathscr{P}(u, v)$ that contains e, but not e'. However, e and e' are connected without $\{u, v\}$, which is a contradiction. Thus, F has exactly two boundary nodes. Since F has one entry and one exit, $F \in \mathscr{F}(u, v)$. □

A.3. Proofs – The canonical and the objective fragments coincide

In this section, we prove that the canonical and the objective fragments coincide. First, we prepare for this by proving Propositions 3–6 and Lemmas 10–12. Then we can prove this equivalence in Theorem 7.

Proposition 3. *If $X, Y \in \mathscr{F}(u, v)$ and $F = X \cup Y$, then $F \in \mathscr{F}(u, v)$.*

1. *If X and Y are bond fragments, F is a bond fragment.*
2. *If X and Y are directed bond fragments, F a directed bond fragment.*
3. *If X and Y are semi-pure bond fragments, F a semi-pure bond fragment.*
4. *If X and Y are pure bond fragments, F a pure bond fragment.*

Proof. Since u is the entry of X, all edges incoming to u are outside of X or all edges outgoing from u are inside of X. The same is true for Y. We distinguish two subcases:

1. Assume all edges incoming to u are outside X and Y. Then, all edges incoming to u are outside $X \cup Y$ and u is hence an entry of F.
2. Assume all outgoing edges of u are inside X or all outgoing edges of u are inside Y. Then, all outgoing edges of u are inside $X \cup Y$ and u is hence an entry of F.

The proof that v is an exit of F is symmetric. It is clear that F does not have any other boundary nodes. Thus, $F \in \mathscr{F}(u, v)$. Claims 1,2 and 4 are now trivial.

We now prove claim 3. Assume X and Y are semi-pure bond fragments. It follows that each X and Y is a union of at least two branches from $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$. It follows from the definitions of semi-pure bond fragments, undirected and directed branches that each branch of X and Y has both an edge outgoing from u and an edge incoming to v.

We distinguish two cases in the following, (1) X or Y has at least one undirected branch, and (2) both X and Y have no undirected branches.

1. Without loss of generality, assume that X has at least one undirected branch. Since X is a semi-pure bond fragment, either (a) X has no branch that has an edge incoming to u, or (b) X has no branch that is an edge outgoing from v. We distinguish these two cases in the following.
   (a) Assume X has no branch that has an edge incoming to u. Assume S is an undirected branch of X. Since S is an undirected branch and S has no edge incoming to u, then S has an edge outgoing from v. Since v is the exit of X, then X contains all branches that have an edge incoming to v. Since each branch of Y, contains an edge incoming to v, then $Y \subseteq X$. It follows that $X \cup Y = X$. Thus, $F = X$. It follows that F is a semi-pure bond fragment.
   (b) Assume X has no branch that has an edge outgoing from v. Assume S is an undirected branch of X. Since S is an undirected branch and S has no edge outgoing from v, then S has an edge incoming to u. Since u is the entry of X, then X contains all branches that have an edge outgoing from u. Since each branch of Y, contains an edge outgoing from v, then $Y \subseteq X$. It follows that $X \cup Y = X$. Thus, $F = X$. It follows that F is a semi-pure bond fragment.
2. Assume both X and Y have no undirected branches. It follows that both X and Y are pure bond fragments. Then it follows from the claim 4 of this proposition that F is a pure bond fragment. It follows from the hierarchy of bond fragment types that F is also a semi-pure fragment. □

Proposition 4. *Let G be a TTG such that its undirected version is weakly biconnected.*

1. *If $F, F' \in \mathscr{F}(u,v)$ are proper semi-pure bond fragments of G, then $F = F'$.*
2. *If $F, F' \in \mathscr{F}(u,v)$ are proper directed bond fragments of G, then $F = F'$.*

Proof

1. Suppose $F, F' \in \mathscr{F}(u,v)$ and $F \neq F'$. As F (F') is a semi-pure bond fragment and not a pure bond fragment, then F (F') is a union of at least two branches each, contains at least one undirected branch S (S') and each undirected branch in F (F') has either (i) no edge incoming to u, or (ii) no edge outgoing from v. Thus, such an undirected branch S of F (S' of F') has either (i) at least one edge outgoing from u, at least one edge incoming to v, and at least one edge outgoing from v, or (ii) at least one edge outgoing from u, at least one edge incoming to v, and at least one edge outgoing from v.
   As F is a proper semi-pure bond fragment, the undirected branch $S \in F$, F has an edge $e \in S$ incoming to u, or an edge $e' \in S$ outgoing from v. As u is the entry of F, and $e \in F$, all outgoing edges of u are in F. As v is the exit of F, and $e' \in F$, all incoming edges of v are in F. It follows from the definition of branch types that every directed branch from u to v has an edge outgoing from u, or an edge incoming to v. Therefore, F contains all directed branches from u to v. With the same arguments F' contains all directed branches from u to v.
   As F and F' contain the same directed branches from u to v, and $F \neq F'$, one of them contains an undirected branch that the other does not. Without loss of generality, assume S is not in F'. Undirected branches S and S' can either both have no incoming edge to u, both have no edge outgoing edge from v, or one has no incoming edge to u, and the other has no outgoing edge. The first two cases are symmetric. Without loss of generality, we distinguish the following two cases (a) both S and S' have both no edge incoming to u, (b) S has no edge incoming to u, and S' has no edge outgoing from v.
   (a) Assume S and S' have both no edge incoming to u. Since S' has both incoming and outgoing edges of v, and S has an edge incoming to v, then v is not an exit of F'. This contradicts F' being a fragment.
   (b) Assume S has no edge incoming to u, and S' has no edge outgoing from v. Since S' has both incoming and outgoing edges of v, and S has an edge incoming to v, then v is not an exit of F'. This contradicts F' being a fragment.
2. Assume $F, F' \in \mathscr{F}(u,v)$ are directed bond fragments of G and not semi-pure bond fragments. As F is a directed bond fragment and not a semi-pure bond fragment, F has an undirected branch that has an edge incoming to u and an undirected branch that has an edge outgoing from v. As u is the entry of F and F has an edge incoming to u, F contains all edges outgoing from u. Analogously, as v is the exit of F and F has an edge outgoing from v, F contains all edges incoming to v. As each directed branch in $\mathscr{D}(u,v)$ contains an edge outgoing from u, F contains all branches in $\mathscr{D}(u,v)$. As each undirected branch in $\mathscr{U}(u,v)$ contains an edge outgoing from u or an edge incoming to v, F contains all branches in $\mathscr{U}(u,v)$. As F is a directed bond fragment, F contains no branch in $\mathscr{D}(v,u)$. Thus, F is the union of all branches in $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$. With the same arguments, F' is the union of all branches in $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$. Therefore, $F = F'$. □

We prepare to prove the subsequent Proposition 5 by showing in Lemma 10 that two canonical bond fragments that share an entry and an exit do not overlap.

Lemma 10. *Let G be a weakly biconnected TTG. Let $X, Y \in \mathscr{F}(u,v)$ be canonical bond fragments of G. Then $X \subseteq Y$, $Y \subseteq X$ or $X \cap Y = \emptyset$.*

Proof. Follows from the fact that the maximal elements of the bond fragment classes are unique if they exist, the fact that the bond classes form a hierarchy and from Proposition 3.

For example, assume that X is a maximal directed bond fragment and Y is a maximal semi-pure bond fragment. It follows from the definitions of bond fragment types that Y is also a directed bond fragment. It follows from Proposition 3 that $X \cup Y$ is also a directed bond fragment. $X \cup Y = X$, as X is the maximal directed bond fragment. Thus, $Y \subseteq X$. □

Proposition 5. *Let G be a TTG such that its undirected version is weakly biconnected.*

1. *Let $X, Y \in \mathscr{F}(u,v)$ be bond fragments of G, and X be canonical. Then $X \subseteq Y$, or $Y \subseteq X$.*
2. *Let $X \in \mathscr{F}(u,v)$ and $Y \in \mathscr{F}(v,u)$ be bond fragments of G, and X or Y be canonical. Then $X \subseteq Y$, $Y \subseteq X$, or $X \cap Y = \emptyset$.*

Proof

1. As X is a canonical bond fragment, X is a maximal pure bond fragment X_P, a maximal semi-pure bond fragment X_S, a maximal directed bond fragment X_D, or a maximal bond fragment X_B. It follows from the definition of the bond fragment types that $X_P \subseteq X_S \subseteq X_D \subseteq X_B$ if they exist. We distinguish the following cases Y is (a) a pure bond fragment, (b) a proper semi-pure bond fragment, (c) a proper directed bond fragment, and (d) a proper bond fragment.
   (a) Assume Y is a pure bond fragment. Thus, $Y \subseteq X_P$. It follows that $Y \subseteq X_P \subseteq X_S \subseteq X_D \subseteq X_B$. Thus, $Y \subseteq X$.
   (b) Assume Y is a proper semi-pure bond fragment. It follows from Proposition 4 that $Y = X_S$. As Y is canonical, it follows from Lemma 10 that $X \subseteq Y$, or $Y \subseteq X$.
   (c) Assume Y is a proper directed bond fragment. It follows from Proposition 4 that $Y = X_D$. As Y is canonical, it follows from Lemma 10 that $X \subseteq Y$, or $Y \subseteq X$.
   (d) Assume Y is a proper bond fragment. It follows Proposition. 3 that a maximal bond fragment $Y \in \mathscr{F}(u,v)$ contains every bond fragment $F \in \mathscr{F}(u,v)$. Thus, $Y \subseteq X_B$. As Y is a proper bond fragment, Y has a directed branch S from v to u. It follows that Y has an edge $e \in S$ incoming to u, and an edge $e' \in S$ outgoing from v. As u is the entry of Y, and $e \in Y$, all outgoing edges of u are in Y. As v is the exit of Y, and $e' \in Y$, all incoming edges of v are in Y. It follows from the definition of branch types that every directed branch from u to v and every undirected branch between u and v has an edge outgoing from u, or an edge incoming to v. Therefore, Y contains all directed branches from u to v and all undirected branches between u and v. If X_D exists, X_D is the union of these branches. Thus, $X_D \subseteq Y$. Note that if X_D does not exist, neither does X_S or X_P.If $X = X_B$, $Y \subseteq X$. Otherwise $X = X_D$, $X = X_S$, or $X = X_P$, and as $X_P \subseteq X_S \subseteq X_D \subseteq Y$, it follows that $X \subseteq Y$.
2. Assume that $X \in \mathscr{F}(u,v)$ and $Y \in \mathscr{F}(v,u)$ are bond fragments of G, and X or Y is canonical. Let $X_B \in \mathscr{F}(u,v)$ and $Y_B \in \mathscr{F}(v,u)$ be the maximal bond fragments, $X_D \in \mathscr{F}(u,v)$ and $Y_D \in \mathscr{F}(v,u)$ be the maximal directed bond fragments, $X_S \in \mathscr{F}(u,v)$ and $Y_S \in \mathscr{F}(v,u)$ be the maximal semi-pure bond fragments and $X_P \in \mathscr{F}(u,v)$ and $Y_P \in \mathscr{F}(v,u)$ be the maximal pure bond fragments if they exist. It follows from the definition of the bond fragment types that $X_P \subseteq X_S \subseteq X_D \subseteq X_B$ and $Y_P \subseteq Y_S \subseteq Y_D \subseteq Y_B$. If either X_B or Y_B does not exist, there is nothing to show. Otherwise, it follows that both u and v have both incoming and outgoing edges. Therefore, neither u nor v is the sink or the source of G. Since u and v are nodes of a TTG, then u and v are in a path from the source to the sink. We can assume without loss of generality that u has an incoming edge outside $X_B \cup Y_B$, and v has an outgoing edge outside $X_B \cup Y_B$. Let A be the union of all the branches between u and v. We distinguish two cases (1) u has no outgoing edge outside A, and v has no incoming edge outside A, and (2) u has an outgoing edge outside A, or v has an incoming edge outside A.
   (a) Assume that u has no outgoing edge outside A, and v has no incoming edge outside A. It follows that A is a bond fragment. As A has all branches between u and v, then it is a maximal bond fragment and $X_B = A$. Thus, X_B contains all directed branches from u to v, from v to u and all undirected branches between u and v. It follows that $Y_B \subseteq X_B$.Next, we show that Y_B does not contain any directed branches in $\mathscr{D}(u,v)$ or undirected branches in $\mathscr{U}(u,v)$.
      (i) Suppose $S_1 \in \mathscr{D}(u,v)$ such that $S_1 \subseteq Y_B$. Since there exists an edge outgoing from v outside of Y_B, and v is the entry of Y_B, then Y_B and hence S_1 contain no edges incoming to v. This contradicts S_1 being a directed branch from u to v.
      (ii) Suppose $S_2 \in \mathscr{U}(u,v)$ such that $S_2 \subseteq Y_B$. Since there exists an edge outgoing from v outside of Y_B, and v is the entry of Y_B, then Y_B and hence S_2 contain no edges incoming to v. Because S_2 is an undirected branch, it must contain an edge incoming to u and an edge outgoing from u. This contradicts u being the exit of Y_B and u having an incoming edge outside Y_B.
   It follows that $Y_P = Y_S = Y_D = Y_B$, which is the union of all directed branches from v to u. Furthermore, $Y \subseteq Y_B$. As $X_B \in \mathscr{F}(u,v)$, then X_B has at least one branch in $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$. As Y and Y_B are properly contained in X_B, Y and Y_B are not canonical. Thus, X is canonical. We distinguish the following cases (a) $X = X_B$, and (b) $X = X_D$, $X = X_S$, or $X = X_P$.
      (i) Assume $X = X_B$. As $Y \subseteq X_B$, $Y \subseteq X$.
      (ii) Assume $X = X_D$, $X = X_S$, or $X = X_P$. As X_D contains only branches $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$ and Y contains only branches in $\mathscr{D}(u,v)$, $X_D \cap Y = \emptyset$. As $X_P \subseteq X_S \subseteq X_D$, $X \cap Y = \emptyset$.
   (b) Let A be the union of all the branches between u and v. Assume that u has an outgoing edge outside A, or v has an incoming edge outside A. As $X \in \mathscr{F}(u,v)$ and $Y \in \mathscr{F}(v,u)$, A contains an incoming and an outgoing edge of u and an incoming and an outgoing edge of v. It follows that A is not a fragment. Without loss of generality, we can assume v has an incoming edge outside of A.Next, we show that Y_B does not contain any directed branches in $\mathscr{D}(u,v)$ or undi-

rected branches in $\mathscr{U}(u,v)$. As there is an edge outgoing from v outside Y_B, an edge incoming to u outside Y_B and $Y_B \in \mathscr{F}(v,u)$, all edges incoming to v and all edges outgoing from u are outside Y_B. Thus, no edge incoming to v is in Y_B. It follows that Y_B does not contain any branch in $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$. Thus, $Y_P = Y_S = Y_D = Y_B$, which is the union of all directed branches from v to u.Since X_B is a fragment and v is the exit of X_B, and there exists incoming and outgoing edges of v outside X_B, X_B contains no fragment that has edges outgoing from v. It follows that X_B does not contain a directed branch from v to u. Therefore, $X_B \cap Y_B = \emptyset$. Since X_B and Y_B are disjoint, $X \subseteq X_B$, and $Y \subseteq Y_B$, then X and Y are disjoint. □

Proposition 6. *Let G be a TTG, $F \in \mathscr{F}(u,v)$ be a fragment of G. F is a non-canonical fragment if and only if either*

1. *F is a non-maximal sequence,*
2. *F is a non-maximal pure bond fragment,*
3. *F is a non-maximal proper bond fragment, or*
4. *F is a maximal pure bond fragment and F is properly contained in a bond fragment $F' \in \mathscr{F}(v,u)$.*

Proof. We prove the theorem for each direction separately:

1. ($\Rightarrow$) Assume $F \in \mathscr{F}(u,v)$ is a non-canonical fragment. It follows from the definition of a canonical fragment that F is (a) a non-maximal sequence, or (b) a non-canonical bond fragment. In case (a), the claim directly follows. Assume that F is a non-canonical bond fragment. It follows from the definition of a canonical bond fragment and the hierarchy of different bond fragment types that F is (i) a non-maximal proper pure bond fragment, (ii) a non-maximal proper semi-pure bond fragment, (iii) a non-maximal proper directed bond fragment, (iv) a non-maximal proper bond fragment, or (v) a maximal pure bond fragment such that F is not properly contained in any bond fragment $F' \in \mathscr{F}(v,u)$. It follows from Proposition 4 that a proper semi-pure bond fragment is maximal if it exists, and that a proper directed bond fragment is maximal if it exists. Therefore, if F is a non-canonical bond fragment, F is either (i) a non-maximal pure bond fragment, (iv) a non-maximal proper bond fragment, or (v) a maximal pure bond fragment such that F is not properly contained in any bond fragment $F' \in \mathscr{F}(v,u)$. The claim directly follows.
2. ($\Leftarrow$) The claim follows directly from the definition of a non-canonical bond fragment and a non-canonical fragment. □

We prepare for Theorem 7 by showing in Lemma 11 that two canonical fragments do not overlap.

Lemma 11. *Let G be a weakly biconnected TTG. Let X and Y be two fragments of G, and X or Y be canonical. Then $X \subseteq Y$, $Y \subseteq X$ or $X \cap Y = \emptyset$.*

Proof. Suppose X and Y are overlapping, that is $Z = X \cap Y$, $Z \neq \emptyset$, $X \setminus Z \neq \emptyset$ and $Y \setminus Z \neq \emptyset$. Thus, there exists edges a, b, c such that $a \in X \setminus Z$, $b \in Z$ and $c \in Y \setminus Z$. Let G_X be the subgraph formed by X, and G_Y be the subgraph formed by Y. Since $a, b \in X$, and G_X is a connected subgraph, then there exists a path from a to b in G_X.[5] Thus, there exists a node $u \in G_X$ and edges $d, e \in X$ such that d and e are incident to u, $d \in X \setminus Z$ and $e \in Z$. It follows that $d \notin Y$ and $e \in Y$, thus u is a boundary node of Y. With similar arguments, there exists a node $v \in G_Y$ such that v is a boundary node of X.

Let $x1, x2$ be the distinct boundary nodes of X and $y1, y2$ be the distinct boundary nodes of Y. We distinguish the following three cases, where X and Y have (1) two common boundary nodes ($x1 = y1$ and $x2 = y2$), (2) exactly one common boundary node (without loss of generality, we assume $x1 = y1$ and $x2 \neq y2$), and (3) no common boundary nodes ($x1, x2, y1, y2$ are pairwise distinct nodes).

1. Assume $x1 = y1$ and $x2 = y2$. Fig 18 illustrates this case. Since any fragments have only two boundary nodes, $x1$ and $x2$ are the only nodes crossing the boundary of X and Y.
   It follows from Theorem 2 that each X and Y is a union of one or more branches between $x1$ and $x2$ (that are also called as proper separation classes in $\mathscr{P}(x1, x2)$. Since $x1$ and $x2$ are the only boundary nodes of X and Y, $x1$ and $x2$ are also the only boundary nodes of $X \setminus Z$, Z, and $Y \setminus Z$. It follows that each $X \setminus Z$, Z, and $Y \setminus Z$ is a union of one or more branches between $x1$ and $x2$. Furthermore, there exists pairwise distinct branches S_a, S_b, and S_c between $x1$ and $x2$ such that $a \in S_a$, $b \in S_b$, $c \in S_c$, $S_a \subseteq Z \setminus X$, $S_b \subseteq Z$, and $S_c \subseteq Z \setminus Y$.
   Without loss of generality, assume $X \in \mathscr{F}(x1, x2)$. Since a is on a path from s to t and any path from s to a goes via the entry node of X, and any path from a to t goes via the exit node of X, then a is a part of a branch S_a between $x1$ and $x2$, and $S_a \subseteq (X \setminus Z)$. With similar arguments, b is a part of a branch S_b between $x1$ and $x2$, and $S_b \subseteq Z$; and c is a part of a branch S_c between $x1$ and $x2$, and $S_c \subseteq (Y \setminus Z)$. It follows that S_a, S_b, S_c are distinct branches. Since $S_a, S_b \in X$, and $S_b, S_c \in Y$, then X, Y must be bond fragments. We distinguish two subcases:
   (a) Assume $Y \in \mathscr{F}(x1, x2)$. It follows from Proposition 5 that X, Y are nested or disjoint, which is a contradiction.
   (b) Assume $Y \in \mathscr{F}(x2, x1)$. It follows from Proposition 5 that X, Y are nested or disjoint, which is a contradiction.

[5] Note that this path does not have to be a directed path. A path $\pi = v_0, e_1, v_1, e_2, \ldots, v_k$ of a directed graph G' is *directed* if $e_i \mapsto (v_{i-1}, v_i)$ for every $i = 1, \ldots, k$.

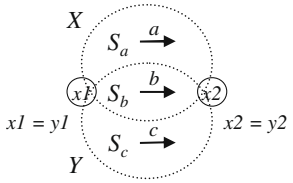

Fig. 18. Case 1: The fragments X and Y have two common boundary nodes $x1$ and $x2$.

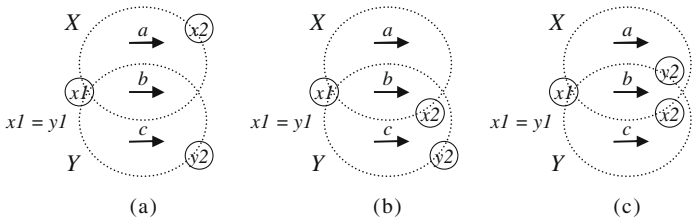

Fig. 19. Three distinguished subcases of case 2: (a) $x2 \notin Y$ and $y2 \notin X$, (b) $x2 \in Y$ and $y2 \notin X$, and (c) $x2 \in Y$ and $y2 \in X$.

2. Assume $x1 = y1$ and $x2 \neq y2$. (This also covers the symmetric case, where $x1 \neq y1$ and $x2 = y2$.) We distinguish the following subcases illustrated in Fig. 19:
   (a) Assume $x2 \notin Y$ and $y2 \notin X$. Then $x1$ is the only node which has two incident edges such that one is inside Z and the other is outside Z. Thus, $x1$ is the only boundary node of the subgraph formed by Z. Thus, either b is a self-loop, or $x1$ is a separation point, which contradict G being weakly biconnected.
   (b) Assume $x2 \in Y$ and $y2 \notin X$. (This also covers the symmetric case, where $x2 \notin Y$ and $y2 \in X$.) Then $x1$ is the only node which has two incident edges such that one is inside $X \setminus Z$ and the other is outside $X \setminus Z$. Thus, $x1$ is the only boundary node of the subgraph formed by $X \setminus Z$. Thus, either a is a self-loop, or $x1$ is a separation point, which contradict G being weakly biconnected.
   (c) Assume $x2 \in Y$ and $y2 \in X$. The source (sink) is a boundary node of any fragment that includes the source (sink). Since X, Y share only one boundary node and boundary nodes of X (Y) are inside Y (X), then both source and sink cannot be inside $X \cup Y$. Thus, there exists an edge e outside of $X \cup Y$. Since $x2 \in Y$ and $y2 \in X$, then $x1$ is the only boundary node that can have an edge outside $X \cup Y$. It follows that $x1$ has at least one edge outside $X \cup Y$. Since $x1$ is the only node that has at least one incident edge inside $X \cup Y$ and at least one incident edge outside of $X \cup Y$, then all paths from e to a go through $x1$. Thus, $x1$ is a separation point, which contradicts G being weakly biconnected.
3. Assume $x1, x2, y1, y2$ are pairwise distinct nodes. We distinguish the following subcases illustrated in Fig. 20:
   (a) Assume $x1, x2 \in Y$ and $y1, y2 \in X$. Since all those boundary nodes are inside both G_X and G_Y and they are not shared boundary nodes, then there is no path from an edge in $X \cup Y$ to an edge outside of $X \cup Y$. Since all boundary nodes are in $G_X \cup G_Y$ and X and Y share no boundary nodes, then the source and sink are outside of $G_X \cup G_Y$. It follows that the edge a are not on a directed path from the source to the sink, which is a contradiction.
   (b) Assume $x1, x2 \in Y$ and $y1 \in X$ and $y2 \notin X$. (This also covers three other symmetric cases.) Let $G_{X \setminus Z}$ be the subgraph formed by $X \setminus Z$. It follows that $y1$ is the only node in $G_{X \setminus Z}$ such that there exists two edges incident to $y1$ and one is inside $X \setminus Z$ and the other is outside $X \setminus Z$. Since both boundary nodes of X are in G_Y and X and Y have no common boundary nodes, then the start node and the end node are outside of $G_{X \setminus Z}$. Thus, either a is a self-loop, or $y1$ is a separation point, which are contradictions as G is weakly biconnected.
   (c) Assume $x1 \notin Y$ and $x2 \in Y$ and $y1 \in X$ and $y2 \notin X$. (This also covers three other symmetric cases.) Let G_Z be the subgraph formed by Z. Since X, Y have no common boundary node, then there is no start node s or end node t in G_Z. Without loss of generality, assume $x2$ is the exit of X. Since $x2$ is the exit of X and $t \notin Z$, then there is no simple directed path from s to b via $x2$. Since there is a directed path from s to b, then there must be a directed path from s to b via $y1$. It

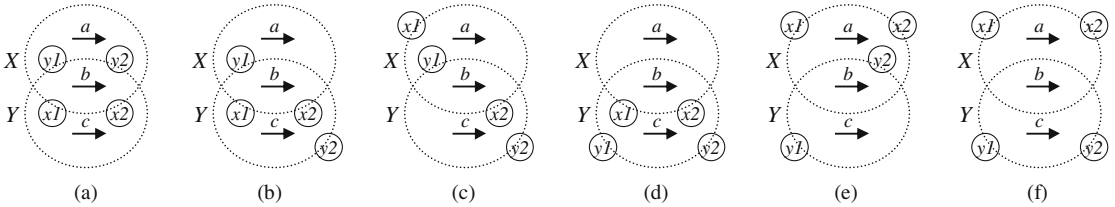

Fig. 20. The distinguished subcases, where $x1, x2, y1, y2$ are pairwise distinct nodes. All omitted subcases are symmetric to these.

follows $y1$ must be an entry of Y.It follows that $X \setminus Z$ has two boundary nodes $x1$ as the entry and $y1$ as the exit; Z has two boundary nodes $y1$ as the entry and $x2$ as the exit; and $Y \setminus Z$ has two boundary nodes $x2$ as the entry and $y2$ as the exit. Thus, $Y \setminus Z, Z, Y \setminus Z$ are fragments. It follows that X and Y are sequences, but not maximal. This contradicts X or Y being a canonical fragment.
(d) Assume $x1, x2 \in Y$ and $y1, y2 \notin X$. (This also covers the symmetric case, where $x1, x2 \notin Y$ and $y1, y2 \in X$.) No boundary node of Y is in G_X which is a contradiction. Recall that in the beginning of this proof, we showed that a boundary node of Y must be in G_X.
(e) Assume $x1, x2 \notin Y$ and $y1 \notin X$ and $y2 \in X$. (This also covers three other symmetric cases.) No boundary node of X is in G_Y, which is a contradiction.
(f) Assume $x1, x2 \notin Y$ and $y1, y2 \notin X$. No boundary node of X is in G_Y, which is a contradiction. □

We prepare for Theorem 7 by showing in Lemma 12 that every non-canonical fragment overlaps with some other non-canonical fragment.

Lemma 12. *Let G be a TTG.*

1. *If F is a non-maximal sequence of G, then there exists a non-maximal sequence F' of G, such that F and F' are overlapping.*
2. *If F is a non-maximal pure bond fragment of G, then there exists a non-maximal pure bond fragment F' of G, such that F and F' are overlapping.*
3. *If F is a non-maximal proper bond fragment of G, then there exists a non-maximal proper bond fragment F' of G, such that F and F' are overlapping.*
4. *If $F \in \mathscr{F}(u, v)$ is a maximal pure bond fragment of G and F is properly contained in a bond fragment $F'' \in \mathscr{F}(v, u)$, then there exists a non-maximal proper bond fragment $F' \in \mathscr{F}(v, u)$ of G, such that F and F' are overlapping.*
5. *If F is a non-canonical fragment of G, then there exists a non-canonical fragment F' of G, such that F and F' are overlapping.*

Proof. We prove each claim separately in the following.

1. Assume F is a non-maximal sequence. Since F is a sequence it is a union of two fragments. Assume A and B are these fragments. Since F is a non-maximal sequence, there exists a fragment C such that $C \neq A$, $C \neq B$, and F and C are in sequence. It follows that C has a common boundary node with A or B. Without loss of generality, assume B and C have a common boundary node. It follows that the union of B and C is fragment F'. Moreover, F' is non-maximal. Clearly, F and F' are overlapping.
2. Assume F is a non-maximal pure bond fragment. Since F is a pure bond fragment it is a union of at least two directed branches from u to v. Assume A and B are two of these directed branches. Since F is a non-maximal pure bond fragment, there exists a directed C such that $C \neq A$, $C \neq B$, and the union of F and C is a pure bond fragment. It follows that the union F' of A and C is also a pure bond fragment. Moreover, F' is non-maximal. Clearly, F and F' are overlapping.
3. Assume F is a non-maximal bond fragment, and not a directed bond fragment. As u is the entry and v is the exit of F and the bond fragment F is not a directed bond fragment, F contains all branches in $\mathscr{D}(u, v) \cup \mathscr{U}(u, v)$ and at least one directed branch in $\mathscr{D}(v, u)$. Assume $S \in \mathscr{D}(v, u)$ and $S \subseteq F$. As F is not a maximal bond fragment, there exists a directed branch S' such that $S' \in \mathscr{D}(v, u)$ and $S' \cup F = \emptyset$. Let F' be the union of S' and the branches in $\mathscr{D}(u, v) \cup \mathscr{U}(u, v)$. It follows that F' is also a bond fragment, and not a directed bond fragment. Moreover, F' is non-maximal. Clearly, F and F' are overlapping.
4. Assume F a maximal pure bond fragment and F is properly contained in a bond fragment $F'' \in \mathscr{F}(v, u)$. By definition, F is a union of at least two directed branches in $\mathscr{D}(u, v)$. As F' is a bond fragment in $\mathscr{F}(v, u)$, F' has at least one outgoing edge of v and at least one incoming edge of u. Thus, F' has at least one branch in $\mathscr{D}(v, u) \cup \mathscr{U}(v, u)$. It follows from the definition of a fragment that F' contains all the branches in $\mathscr{D}(v, u) \cup \mathscr{U}(v, u)$. Assume$S$ is a directed branch in $\mathscr{D}(v, u)$ and $S \subseteq F$. It follows from the definition of a bond fragment that the union of all branches in $\mathscr{D}(v, u) \cup \mathscr{U}(v, u)$ and S is a non-maximal proper bond fragment F'. It follows that F and F' are overlapping.
5. Assume $F \in \mathscr{F}(u, v)$ is a non-canonical fragment. It follows from Proposition 6 that F is either (1) a non-maximal sequence, (2) a non-maximal pure bond fragment, (3) a non-maximal proper bond fragment, or (4) a maximal pure bond fragment and F is properly contained in a bond fragment $F'' \in \mathscr{F}(v, u)$. It follows from the claims 1–4 of this lemma and Lemma 11 that in each of these four cases, there exists a non-canonical fragment F' of G, such that F and F' are overlapping. □

Finally, we are ready to prove Theorem 7.

Theorem 7. *Let F be a fragment of a TTG. F is a canonical fragment if and only if F is an objective fragment.*

Proof

(⇒) Assume F is a canonical fragment. It follows from Lemma 11 that F is an objective fragment.
(⇐) Suppose an objective fragment F is not a canonical fragment. As F is non-canonical, it follows from Lemma 12 that F overlaps with some fragment, which contradicts F being an objective fragment. □

A.4. Proofs – The properties of the RPST

In this section, we prove the properties of the RPST. Proposition 8 describes how all fragments can be derived from the RPST. Finally, Theorem 9 is used to show the modularity of the RPST.

Proposition 8. *Let F be a set of edges in a TTG. F is a fragment if and only if F is a canonical fragment or F is*

1. *a union of consecutive child fragments of a maximal sequence,*
2. *a union of child fragments of a maximal pure bond fragment, or*
3. *a union of child fragments of a maximal bond fragment B such that B is not a maximal directed bond fragment.*

Proof

1. ($\Rightarrow$) Assume F is a fragment. We distinguish two cases: F is either (1) a canonical or (2) a non-canonical fragment. In case (1) the claim directly follows. In the following, we deal with the remaining case (2).
   Assume $F \in \mathscr{F}(u,v)$ is a non-canonical fragment. It follows from Proposition 6 that F is either (a) a non-maximal sequence, (b) a non-maximal pure bond fragment, (c) a non-maximal proper bond fragment, or (d) a maximal pure bond fragment and F is properly contained in a bond fragment $F'' \in \mathscr{F}(v,u)$. We distinguish these cases in the following:
   (a) Assume F is a non-maximal sequence. It follows from Lemma 12 that F overlaps with another non-maximal sequence.
   Assume A and B are two fragments that are not sequences, but that are in sequence. It follows that their union is a sequence that is union of two fragments that are not sequence. It can be shown through induction that each sequence is a union of two or more fragments that are not sequences. As a maximal sequence does not overlap with any non-canonical fragment, it follows that a maximal sequence is a union of child fragments, where two consecutive child fragments share a boundary node that is the entry of the former and the exit of the latter and are pairwise in sequence. Moreover, each non-maximal sequence is a union of consecutive child fragments of a maximal sequence.
   (b) Assume F is a non-maximal pure bond fragment. By definition, a pure bond fragment is a union of directed branches in $\mathscr{D}(u,v)$. As each such a branch S has no incoming edges of u and no outgoing edges of v, S is a fragment. Moreover, the union of two or more branches in $\mathscr{D}(u,v)$ is a pure bond fragment. The union of all branches in $\mathscr{D}(u,v)$ is a maximal pure bond fragment R. Each branch in $\mathscr{D}(u,v)$ is a child fragment of R, as the other unions of these branches are non-maximal pure bond fragments. It follows that a union of a child fragments of a maximal pure bond fragment is a fragment.
   (c) Assume F is a non-maximal proper bond fragment. By definition, a proper bond fragment is a union of all branches in $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$ and at least one directed branch in $\mathscr{D}(v,u)$. Since F is a fragment, F has at least one edge incoming to its entry u and at least one edge outgoing from its exit v, then F has all edges outgoing from u and all edges incoming to v. It follows that the union A of all branches in $\mathscr{D}(u,v) \cup \mathscr{U}(u,v)$ is a fragment. Moreover, D is canonical. Each directed branch in $\mathscr{D}(v,u)$ is also a canonical fragment, and a child fragment of a maximal bond fragment B that is the union of all branches between u and v. Also D is a child fragment of B. It follows that F is a union of child fragments of a maximal bond fragment B such that B is not a maximal directed bond fragment.
   (d) Assume F is a maximal pure bond fragment and F is properly contained in a bond fragment $F' \in \mathscr{F}(v,u)$. It follows that F is properly contained in a maximal bond fragment $B \in \mathscr{F}(v,u)$. With similar arguments than in the previous case (c), F is a union of child fragments of a maximal bond fragment B such that B is not a maximal directed bond fragment.
2. ($\Leftarrow$) If F is a canonical fragment, it is clear that F is also a fragment. Assume F' is the parent fragment of F. Therefore, we distinguish the following cases based on the structure of the proposition. In each case, it follows directly from the definition of the RPST that a union of exactly one child fragment of F' is a (canonical) fragment.
   (a) Assume F is a union of two or more consecutive child fragments of a maximal sequence, and F'. It can be shown based on the definitions of the RPST and the maximal sequence that each child fragment of a maximal sequence F' is canonical, but not a sequence. Furthermore, on the same basis a union of two or more consecutive child fragments of F' is a sequence. Thus, F is a fragment.
   (b) Assume F is a union of child fragments of a maximal pure bond fragment. It can be shown based on the definition of the RPST and the canonical bond fragment that each child fragment of a maximal pure bond fragment is a canonical fragment and a directed branch $F \in \mathscr{D}(u,v)$, but not a bond fragment. As the union F' of directed branches in $\mathscr{D}(u,v)$ has no incoming edges to u and no outgoing edge from v, F' is a (pure bond) fragment.
   (c) Assume F is a union of child fragments of a maximal bond fragment B such that B is not a maximal directed bond fragment. It can be shown based on the definition of the RPST and the canonical bond fragment that each child fragment of F' is (i) either a directed branch in $\mathscr{D}(u,v)$, an undirected branch in $\mathscr{U}(u,v)$, or a maximal directed bond fragment in $\mathscr{F}(u,v)$, whereas (ii) each of the other child fragments of F' is directed branch in $\mathscr{D}(v,u)$ and not bond fragment. If F is a union of two or more directed branches in $\mathscr{D}(v,u)$, then based on the same arguments than in the previous case (b), F is a (pure bond) fragment. Whereas if F is a union of a child fragment in $\mathscr{D}(u,v) \cup \mathscr{U}(u,v) \cup \mathscr{F}(u,v)$, and some child fragments in $\mathscr{D}(v,u)$, then F is a (proper) bond fragment. □

Theorem 9. *Let G be a TTG and $X \in \mathscr{F}(u,v)$ be a canonical fragment of G. Let G' be the TTG that is obtained by replacing the subgraph that is formed by X by some other subgraph formed by a set of edges X' such that $X' \in \mathscr{F}(u,v)$ is again a fragment of G' (but not necessarily canonical) with the same entry-exit pair as X. Assume A is the parent fragment of X in G and $F \neq X$ is a child fragment of A in G. Let $A' = (A \setminus X) \cup X'$ and $F' = F$. Then A' and F' are canonical fragments of G' where F' is a child fragment of A' in G'.*

Proof. First we show that (1) F' and (2) A' are fragments.

1. We have $X \cap F = \emptyset$. Let G_X be formed by X and G_F be formed by F. Since F is connected, F has two boundary nodes, and $F = F'$, then F' is connected and has two boundary nodes. Since X and F are disjoint, then G_X and G_F are either disjoint, or share the entry or the exit of F. We distinguish these cases in the following:
   (a) Assume that the boundary nodes of F are not in G_F. Since $F = F'$, $X \cap F = \emptyset$ and only G_X has been replaced by $G_{X'}$, then Y and F' have the same boundary nodes, and the edges incident to these boundary nodes are the same. It follows that the entry of F is also the entry of F'. With analogous arguments, the exit of F is also the exit of F'.
   (b) Assume the entry of F is a boundary node of X. We distinguish two cases, the entry of F is either u, or v.
      (i) Assume u is the entry of F. Since u is the entry of F and X, then each has an outgoing edge of u. Since $X \cap F = \emptyset$, these outgoing edges are distinct. Since an outgoing edge of u is outside F, then all outgoing edges of u are not in F. Since u is an entry of F, it follows that F has no incoming edges of u. Then F' has also no incoming edge of u and u is an entry of F'.
      (ii) Assume v is the entry of F. Since v is the entry of F, then F has no incoming edges of v, or all outgoing edges of v. We distinguish these two cases in the following:
         (A) Assume F has no incoming edges of v. It follows that F' has no incoming of v and v is an entry of F'.
         (B) Assume F has all outgoing edges of v, and at least one incoming edge of v. Since X' and F are disjoint, there is both an incoming edge and an outgoing edge of v outside X', and v is the exit of X', then X' has no outgoing edges of v. Thus, F has all outgoing edges of v, and v is an entry of F'.
   (c) With analogous arguments, the exit of F, is also the exit of F'.

   Since F' is a connected subgraph of G', and F' has exactly two boundary nodes, an entry and an exit, then F' is a fragment in G'.
2. We have $X \subseteq A$. Let G_X be formed by X and G_A be formed by A. Since G_X and G_A are connected subgraphs, $X \subseteq A$, G_X is connected to $A \setminus X$ only via u and v, $A' = (A \setminus X) \cup X'$ and X' is connected to $A' \setminus X'$ via u and v, then A' is a connected subgraph. Since $A' = (A \setminus X) \cup X'$, it follows that the boundary nodes of A are also the boundary nodes of A'. Since X and A are disjoint, then G_X and G_A are either disjoint, or share the entry or the exit of A. We distinguish these cases in the following:
   (a) Assume u and v are not shared by G_X and G_A. Since $A = A'$, $X \subseteq A$ and only G_X has been replaced by $G_{X'}$, then A and A' have the same boundary nodes, and the edges incident to these boundary nodes are the same. It follows that the entry of A is also the entry of A'.
   (b) Assume the entry of A is a boundary node of X. We distinguish two cases, the entry of A is either u, or v.
      (i) Assume u is the entry of A. We distinguish two cases, either there is an edge outgoing from u outside A, or there is no edge outgoing from u outside A.
         (A) Assume that there is an edge outgoing from u outside A. Since u is entry of A' and A', then there is no edge incoming to u in A' and A'. Thus, u is an entry of A'.
         (B) Assume that there is no edge outgoing from u outside A. Thus, all outgoing edges are in A and A'. Thus, u is an entry of A'.
      (ii) Assume v is the entry of A. Since v is the exit of X, then X has an edge outgoing from v. Since $X \subseteq A$ and v is the entry of A, then A has both incoming and outgoing edge of v. It follows that there is no edge outgoing from v outside A. It follows that there is no edge outgoing from v outside of A'. Thus, v is an entry of A'.
   (c) Assume the exit of A is a boundary node of X. With analogous arguments to the previous case, the exit of Y is the exit of A'.

   Since A' is a connected subgraph of G', and A' has exactly two boundary nodes, an entry and an exit, then A' is a fragment in G'.

Next, we show that A' and F' are the canonical fragments of G'. We distinguish the following cases, A is (1) a maximal sequence, (2) neither a sequence nor a bond fragment, (3) a canonical bond fragment.

1. Assume A is a maximal sequence. Since F is a child of A and a canonical fragment, then F is not a sequence. It can be shown that each child of a maximal sequence is a union of all branches between the entry and the exit of this child. Thus, F is either (a) a maximal bond fragment, or (b) neither sequence nor a bond fragment. We distinguish these cases in the following:
   (a) Assume F is a maximal bond fragment. Since F' is a fragment and $F = F'$, then F' is also a bond fragment. Since F' is a fragment and F' contains all branches between its boundary nodes, then F' is a maximal bond fragment. Thus, F' is canonical.
   (b) Assume F is neither sequence nor a bond fragment. Since F' is a fragment and $F = F'$, then F' is a fragment that is neither sequence nor a bond fragment. Thus, F' is canonical.

Since X' is a fragment, and X' replaces a child X of a maximal sequence, then X' is either (1) a sequence, (2) a maximal bond fragment, or (3) neither sequence nor bond fragment. Since X is a child of the maximal sequence A, then X is in sequence with a child of A. In each three cases above, it follows that X' is also in sequence with this child fragment of A that is also a child fragment of A'. In case X' is a sequence, then X' is not a canonical fragment. However, the other children of A are canonical fragments and children of A'. It follows that A' is also a maximal sequence. Thus, A' is a canonical.

2. Assume A is neither a sequence nor a bond fragment. Since A is not a bond, then A contains only one branch between its entry and exit. Since $A \neq X$, then A has at least two children. Replacing X with X' changes only a proper subset of this branch, and thus does not introduce any new branches between the entry and the exit of A. Thus, A' is not a bond. Any two children of A are not in sequence. It follows that X' is also not in a sequence with any child of A, because it replaces X that is not in sequence with any child of A. It follows that any union of X' and children of A is not a sequence. It follows that A' is not a sequence, because its children are not in sequence.Thus, A' is neither a sequence nor a bond fragment. Therefore, A' is a canonical fragment. With similar arguments, F' is a canonical fragment, and F' is a child of A'.
3. Assume A is a canonical bond fragment. Then A is either (a) a maximal pure bond fragment, (b) a maximal semi-pure bond fragment, but not a pure bond fragment, (c) a maximal directed bond fragment, but not a semi-pure bond fragment, or (d) a maximal bond fragment, but not a directed bond fragment.

   (a) Assume A is a maximal pure bond fragment. Then A is a union of at least two directed branches from its entry to its exit. Each such a branch (e.g. X, F) is a canonical fragment, and a child of A. Since X' is a fragment, then X' is also a union of at least one directed branch from the entry of A to the exit of A'. It follows that the other children of A are canonical fragments and children of A'. It follows that A' is a pure bond fragment. Thus, A' is a canonical.

   (b) Assume A is a maximal semi-pure bond fragment, but not a pure bond fragment. A maximal semi-pure bond fragment has at most one child fragment that has a common entry and a common exit. Such a child is a union of all directed bonds from the entry to the exit of A, because A is not a semi-pure but not a pure bond fragment. The other children are proper subsets of an undirected branch of A. Based on this, we distinguish the following two cases.

      (i) Assume X is a child that is the union of all directed branches from the entry to the exit of A. Since X' is a fragment, then X' is also a union of all directed branches from the entry to the exit of A. Since the other branches are the same in A and A', then A' is also a semi-pure bond fragment, but not a pure bond fragment.

      (ii) Assume X is a child that is a proper subset of an undirected branch of A. Replacing X with X' changes only this part of this undirected branch, and the other branches are the same in A' and A. It follows that A' is also a semi-pure bond fragment, but not a pure bond fragment.

   (c) Assume A is a maximal directed bond fragment, but not a semi-pure bond fragment. A has at most one child that is either a maximal semi-pure bond, an undirected branch between the entry and the exit of A, or a directed branch from the entry to the exit of A. The other children of A are proper subsets of an undirected branch between the entry and the exit of A. We distinguish these two cases in the following.

      (i) Assume X is either a maximal semi-pure bond, an undirected branch between the entry and the exit of A, a directed branch from the entry to the exit of A. Since X' is a fragment, then X' has no directed branches from the exit to the entry of A. It follows that A' contains at least two branches between the entry and the exit of A', and no directed branches from the entry to the exit of A'. Thus, A' is a maximal directed bond fragment, but not a semi-pure bond fragment. It can also be shown that F' is canonical and a child of A'.

      (ii) With similar arguments than in the respective case of the maximal semi-pure bond fragment that is not a pure bond fragment, it follows that A' is a maximal directed bond fragment, but not a semi-pure bond fragment. Furthermore, F' is canonical and a child of A'.

   (d) Assume A is a maximal bond fragment, but not a directed bond fragment. A bond fragment that is not a directed bond fragment, has at most one child that has a common entry and a common exit. This child is either a maximal directed bond fragment, a undirected branch, or a directed branch between the entry and exit of the parent. Each other child is a directed branch from the exit to the entry of the parent. Based on this, we distinguish the following two cases.

      (i) Assume X is a directed branch from the exit to the entry of A. Since X' is a fragment, then X' is also a union of at least one directed branch from the exit to the entry of A. Since A' is a fragment and has at least two branches, then A' is a bond fragment. Since there are no branches between the entry and the exit of A outside A, then there are no branches between the entry and the exit of A' outside A'. Since A' has a directed branch from its exit to its entry, then A' is a maximal bond fragment, but not a directed bond fragment.

      F is either a directed branch from the exit to the entry of A, or the only fragment that has a common entry and a common exit with A. We distinguish these cases in the following:

         (A) Assume F is a directed branch from the exit to the entry of A. Then, F' is a directed branch from the exit to the entry of A'. It follows that F' is child of A' and a canonical fragment.

(B) Assume F is the only fragment that has a common entry and a common exit with A. It follows that F is a union of all branches between the entry and the exit of A, except all the directed branches from the exit to the entry of A. Since $F' = F$, and X' has only directed branches from the exit to the entry of A, then F' also include all branches except these directed branches. Thus, F' is either a maximal directed bond fragment, or not a bond fragment. It follows that F' is a child of A', and a canonical fragment.

(ii) Assume X is a child that is the only child from the entry to the exit of A. Then X' is a union of branches between the entry and the exit of A. Since F is a child of A, then F is a directed branch from the exit of A to the entry of A'. Since $F = F'$, and A' is a bond, then F' is a child of A'. It follows that F' is a canonical fragment. □

References

[1] W. Sadiq, M.E. Orlowska, Analyzing process models using graph reduction techniques, Inf. Syst. 25 (2) (2000) 117–134.
[2] J. Vanhatalo, H. Völzer, F. Leymann, Faster and more focused control-flow analysis for business process models though SESE decomposition, in: B.J. Krämer, K.-J. Lin, P. Narasimhan (Eds.), ICSOC 2007, LNCS, vol. 4749, Springer, 2007, pp. 43–55.
[3] J. Küster, C. Gerth, A. Förster, G. Engels, Detecting and resolving process model differences in the absence of a change log, in: M. Dumas, M. Reichert, M.-C. Shan (Eds.), BPM 2008, LNCS, vol. 5240, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 244–260.
[4] T. Gschwind, J. Koehler, J. Wong, Applying patterns during business process modeling, in: M. Dumas, M. Reichert, M.-C. Shan (Eds.), BPM 2008, LNCS, vol. 5240, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 4–19.
[5] C. Ouyang, M. Dumas, A.H.M. ter Hofstede, W.M.P. van der Aalst, From BPMN process models to BPEL web services, in: ICWS, IEEE Computer Society, 2006, pp. 285–292.
[6] K.B. Lassen, W.M.P. van der Aalst, WorkflowNet2BPEL4WS: A tool for translating unstructured workflow processes to readable BPEL, in: OTM Conferences (1), vol. 4275 of LNCS, Springer, 2006, pp. 127–144.
[7] N. Lohmann, J. Kleine, Fully-automatic translation of open workflow net models into human-readable abstract BPEL processes, in: Modellierung 2008, vol. P-127 of Lecture Notes in Informatics (LNI), GI, 2008, pp. 57–72.
[8] R. Johnson, D. Pearson, K. Pingali, The program structure tree: Computing control regions in linear time, in: Proceedings of the ACM SIGPLAN'94 Conference on Programming Language Design and Implementation (PLDI), 1994, pp. 171–185.
[9] R.C. Johnson, Efficient program analysis using dependence flow graphs, Ph.D. thesis, Cornell University, Ithaca, NY, USA, 1995.
[10] R.E. Tarjan, J. Valdes, Prime subprogram parsing of a program, in: POPL '80: Proceedings of the Seventh ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM, New York, NY, USA, 1980, pp. 95–105.
[11] J. Vanhatalo, H. Völzer, J. Koehler, The refined process structure tree, in: M. Dumas, M. Reichert, M.-C. Shan (Eds.), BPM 2008, LNCS, vol. 5240, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 100–115.
[12] J. Hopcroft, R.E. Tarjan, Dividing a graph into triconnected components, SIAM J. Comput. 2 (1973) 135–158.
[13] J. Vanhatalo, Structural analysis of business process models using the process structure trees, to be submitted as a dissertation.
[14] C. Gutwenger, P. Mutzel, A linear time implementation of SPQR-trees, in: J. Marks (Ed.), Graph Drawing, LNCS, vol. 1984, Springer, 2000, pp. 77–90.
[15] J. Valdes Ayesta, Parsing flowcharts and series-parallel graphs., Ph.D. thesis, Stanford University, CA, USA, 1978.
[16] J. Vanhatalo, H. Völzer, F. Leymann, S. Moser, Automatic workflow graph refactoring and completion, in: A. Bouguettaya, I.H. Krueger, T. Margaria (Eds.), ICSOC 2008, LNCS, vol. 5364, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 100–115.

Jussi Vanhatalo received the M.Sc. (Tech.) degree (2004) from the Helsinki University of Technology, in Finland, and another Master's degree in research (DEA) from the University of Nice Sophia Antipolis, in France. There, he also studied at the Eurecom Institute as an exchange student. During 2000–2004, he joined a company called Nice-business Solutions Finland, where he worked on software projects for Nokia. Since 2004, he has worked in research projects at the IBM Zurich Research Laboratory, in Switzerland. He is a doctoral student at the University of Stuttgart, in Germany. His current research interests include business process management and service-oriented architectures.

Hagen Völzer received his master (1995) and doctoral (2000) degrees in computer science from Humboldt-University Berlin, Germany. He has been a research fellow at the Software Verification Research Centre at the University of Queensland, Australia (2001–2003) and a senior research and teaching associate at the University of Lübeck, Germany (2003–2007). He joined IBM Research in March 2007. His current research interests include modeling, semantics and verification of business process models.

Jana Koehler holds a Ph.D. from Saarland University Saarbrücken, Germany, and leads the Business Integration Technologies group at the IBM Zurich Research Laboratory. She joined IBM in Spring 2001 after having worked at the German Research Center for Artificial Intelligence, the University of Freiburg, and Schindler Elevators R&D. Her current work focuses on software engineering and compiler-based methods for process-oriented distributed systems. Major milestones of her previous work are novel algorithmic techniques that enabled AI planning systems to scale to complex realistic problems and a software architecture and algorithms that permitted the commercial breakthrough of destination control systems in the elevator industry.