Diagnostic Information for Control-Flow Analysis of Workflow Graphs (a.k.a. Free-Choice Workflow Nets)

Cédric Favre[1,2], Hagen Völzer[1(✉)], and Peter Müller[2]

[1] IBM Research, Zurich, Switzerland
hvo@zurich.ibm.com
[2] Department of Computer Science, ETH, Zurich, Switzerland

Abstract. A workflow graph is a classical flow graph extended by concurrent fork and join. Workflow graphs can be used to represent the main control-flow of e.g. business process models modeled in languages such as BPMN or UML activity diagrams. They can also be seen as compact representations of free-choice Petri nets with a unique start and a unique end. A workflow graph is said to be *sound* if it is free of deadlocks and exhibits no lack of synchronization, which correspond to liveness and safeness of a slightly modified version of the corresponding Petri net. We present a new characterization of unsoundness of workflow graphs in terms of three structural, i.e., graphical error patterns. We also present a polynomial-time algorithm that decides unsoundness and returns for each unsound workflow graph, one of the three structural error patterns as diagnostic information. An experimental evaluation on over 1350 workflow graphs derived from industrial business process models suggests that our technique performs well in practice.

1 Introduction

Workflow graphs can capture the main control flow of processes modeled in languages such as BPMN, UML Activity Diagrams, and Event Process Chains (EPC). That is, the core routing constructs of these languages can be mapped to the routing constructs of workflow graphs: alternative split and merge, as well as concurrent fork and join. Thus, a workflow graph is a classical control-flow graph or flow chart extended by concurrent fork and join. Figure 1(a) shows a simple example of a workflow graph in BPMN notation where $f1$ and $f2$ are (concurrent) forks, $j1$ is a (concurrent) join, $d1$ is a decision (i.e., alternative split), and $m1$ and $m2$ are alternative merges. Note that Fig. 1(a) shows only the pure control-flow; representations of tasks, commands, data assignments, etc. are omitted. Those could be added to the edges of the workflow graph.

A workflow graph is equivalent to a corresponding *free-choice* Petri net [1], called a *free-choice workflow net*. The corresponding net for the workflow graph in Fig. 1(a) is shown in Fig. 1(b) where the (green) dashed part is ignored. In fact, the corresponding free-choice workflow net is in some sense isomorphic

M. Chechik and J.-F. Raskin (Eds.): TACAS 2016, LNCS 9636, pp. 463–479, 2016.
DOI: 10.1007/978-3-662-49674-9_27

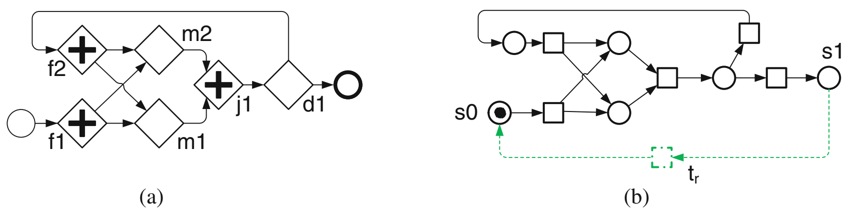


Fig. 1. (a) A workflow graph, (b) its corresponding free-choice workflow net (without dashed part) and its connected version (with dashed part)

to its workflow graph (see [1] for details) such that the workflow graph can be seen as a condensed representation of the free-choice workflow net, and all analysis information obtained on the workflow net can be easily mapped back to the workflow graph. Because of this close relationship and the rich theory available for free-choice nets [2], we will henceforth continue the technical development based on free-choice workflow nets.

A natural correctness condition for free-choice workflow nets, and the dominant one for business process modeling, called *soundness*, requires the absence of two types of control-flow errors: deadlocks and *lack of synchronization.* Lack of synchronization, called *unsafeness* in Petri net theory, occurs when there are two control-flow tokens at the same place, which gives rise to *implicit* auto-concurrency, i.e., a task executing concurrently to itself, which is usually considered a modeling error for this model class. Process languages such as BPMN have constructs for *explicit* auto-concurrency, called multiple instance tasks, where the auto-concurrency is encapsulated in a single-entry-single-exit block.

Figures 2(a) and (d) (ignore the coloring for now) show simple examples of workflow nets with deadlocks. Figures 2(b) and (c) exhibit lack of synchronization, i.e., unsafeness. In particular, Fig. 2(c) shows a special case of lack of synchronization, which causes an unbounded production of tokens, which can be a serious problem for process execution engines if undetected.

A free-choice workflow net is sound if and only if its *(strongly) connected version* is safe and live (in the Petri net sense) [3]. The connected version of the net in Fig. 1(b) is created by adding the dashed part. It can be decided in polynomial time whether a strongly connected free-choice net is safe and live using linear-algebraic techniques. However, none of the existing decision procedures [2,4–6] explicitly attempts to produce diagnostic information to support a modeler in locating, understanding and fixing the error in case the net is not live or not safe.

In this paper, we present a novel technique to detect control-flow errors and to produce diagnostic information that helps modelers to locate and fix the cause of the error. In particular, we make the following contributions:

1. We present a new characterization of unsoundness in terms of three structural error patterns, i.e., offending subgraphs (see Fig. 2) that are present in a free-choice workflow net if and only if it is unsound. This diagnostic information

is designed to be more concise and easier to consume than an error trace, which is important as many process models are created by business analysts without a strong technical background.
2. We present an algorithm that decides unsoundness in polynomial time such that one structural error pattern is returned for each unsound graph. As a byproduct, we can also generate an error trace in polynomial time to complement the main graphical diagnostic information.
3. We implemented our technique as a research prototype in the IBM WebSphere Business Modeler. An experimental evaluation on over 1350 workflow graphs derived from industrial business process models suggests that our technique is sufficiently fast to run the analysis and provide immediate concise diagnostic feedback while the process model is being developed.

Some proofs are omitted in this version. They and additional detail can be found in an extended version of this paper [7] and in a thesis [8].

Related Work. Our structural error patterns are similar to control-flow 'anti-patterns', which are sometimes given to modeling practitioners [9] in terms of erroneous instructive examples. However, in contrast to those anti-patterns, our structural error patterns are formally characterized as graph structures and proved to capture all situations where deadlock or lack of synchronization may occur.

Our patterns are also strongly related to a graph-theoretic characterization of live and bounded free-choice nets given by Esparza and Silva [10]. However, we are not aware of any polynomial-time decision procedure for their characterization.

An alternative approach to detect control flow errors and to provide diagnostic information is to compute an error trace through state-space exploration. An experimental study [11] has shown that whereas naive state-space exploration is not sufficient to analyze the state spaces of business process models, with appropriate reduction techniques, state space exploration can check soundness of an industrial process model in less than a second. However, the traces obtained can be large and contain many transitions that do not contribute to the actual control-flow error [12], which requires additional techniques to trim the trace [13,14], whereas our structural error patterns represent control-flow errors concisely. Moreover, many traces are difficult to visualize in the context of a process model, for instance, because they include several iterations through a cycle, whereas our structural error patterns can be displayed and understood within the process model.

The tool Woflan [15] implements a complex set of (partially exponential-time) Petri net analysis techniques. Some diagnostic hints, e.g., so-called mismatches, can be helpful in many cases to understand an error, but they do not imply unsoundness in general. For other diagnosis results, it remains unclear how they can pinpoint the cause of an error.

2 Preliminaries

2.1 Free-Choice Workflow Nets and Soundness

A *Petri net* $N = (P, T, F)$ consists of disjoint, finite and non-empty sets P of places and T of transitions and a relation $F \subseteq (P \times T) \cup (T \times P)$. An element of $P \cup T$ is also called an *element* of N. Note that $(P \cup T, F)$ is a directed (bipartite) graph and we apply well-known graph-theoretic concepts such as path and strong connectedness to it. For an element x of N, we define ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$ and for a set X of elements, we set ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$. N is *free-choice* if for all transitions t_1, t_2, ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ implies[1] $|{}^\bullet t_1| = |{}^\bullet t_2| = 1$. A *subnet* of N is a Petri net $N' = (P', T', F')$ such that $P' \subseteq P$, $T' \subseteq T$ and $F' = F \cap ((P' \times T') \cup (T' \times P'))$. The *incidence matrix* of a Petri net N is given by the integers $c_{t,p} = \chi_F(t, p) - \chi_F(p, t)$ for $t \in T, p \in P$ where χ_F denotes the characteristic function of F.

A *marking* of N is a mapping $m : P \to \mathbb{N}$, i.e., a bag over P. If $m(p) = k$, we say that p has k *tokens* in m. If $m(p) > 0$, we say that p is *marked* in m. We will sometimes treat a set of places $X \subseteq P$ as a marking by identifying it with its characteristic function. Addition and containment of markings is defined pointwise: $(m_1 + m_2)(p) = m_1(p) + m_2(p)$ and $m_1 \leq m_2$ if there is a marking m such that $m_1 + m = m_2$. A transition t is *enabled* in m if ${}^\bullet t \leq m$. For two markings m_1, m_2 and a transition t, the relationship $m_1 \xrightarrow{t} m_2$ holds whenever t is enabled in m_1 and $m_1 + t^\bullet = {}^\bullet t + m_2$. We write $m \to m'$ if $m \xrightarrow{t} m'$ for any t and $\xrightarrow{*}$ for the reflexive and transitive closure of $\to$. We say m' is *reachable* from m if $m \xrightarrow{*} m'$. A transition t (place p) is *dead* in m if no marking reachable from m enables t (marks p). A transition or place x is *live* in m if x is not dead in each marking reachable from m. A *local deadlock* is a marking in which a transition t is dead and a place $p \in {}^\bullet t$ is marked. A *global deadlock* is a marking in which no transition is enabled. N is *live* in a marking m_0 if every transition is live in m_0.

We say N is *bounded* from a marking m_0 if there is a marking m^* such that for every marking m reachable from m_0, we have $m \leq m^*$. A marking is *safe* if each place has at most one token. N is *safe* from a marking m_0 if every marking reachable from m_0 is safe.

A *workflow net* is a Petri net N with a unique source p_s and a unique sink $p_\mathrm{t} \neq p_\mathrm{s}$ such that $p_\mathrm{s}, p_\mathrm{t} \in P$ and every element of N is on a path from p_s to p_t. The marking m_s (resp. m_t) of N which has a single token on the source (sink) and no token elsewhere is called the *initial* (*final*) marking of N. An *execution sequence* of N is a sequence $\sigma = m_0, m_1, m_2, \ldots$ of markings of N such that $m_i \to m_{i+1}$ for each $i \geq 0$. If $\sigma = m_0, \ldots, m_n$ is finite, we also write $m_0 \xrightarrow{\sigma} m_n$. An *execution trace* of N is an execution sequence $\sigma = m_0, m_1, m_2, \ldots$ of N such that $m_0 = m_\mathrm{s}$. A marking of N is said to be a *reachable marking* of N if it is reachable from the initial marking of N.

[1] Often, a more liberal definition is given for free-choice, which is sometimes also called *extended free-choice*. However an extended free-choice net can be converted into an equivalent free-choice net by a simple and well-known construction.

A workflow net N is said to be *sound* [3] if the following three conditions are satisfied: (i) the sink is live in the initial marking, (ii) the final marking is the only reachable marking of N that marks the sink, and (iii) no transition of N is dead in the initial marking. Condition (ii) says that a token on the sink signals 'proper termination', i.e., there is no token left in the interior of the net. The example in Fig. 1(b) is sound. Soundness appears to be an especially natural correctness condition for free-choice workflow nets as the following theorem suggests. We define for a workflow net N the *(strongly) connected* version of N, denoted by $\overline{N}$. $\overline{N}$ is obtained from N by adding a fresh transition t_r, called the *return* transition and connect it such that ${}^\bullet t_\mathrm{r} = p_\mathrm{t}$ and $t_\mathrm{r}^\bullet = p_\mathrm{s}$, i.e., the return transition moves a token from the sink to the source of N, cf. Figure 1(b) with dashed part. Note that the underlying graph of $\overline{N}$ is indeed strongly connected.

Theorem 1 ([3]). *Let N be a free-choice workflow net. The following five statements are equivalent: (i) N is sound, (ii) N is safe from the initial marking and no reachable marking is a local deadlock (iii) N is safe from the initial marking and no reachable marking is a global deadlock, (iv) $\overline{N}$ is bounded and live from the initial marking, and (v) $\overline{N}$ is safe and live from the initial marking.*

It is not necessarily obvious to a modeler whether a reachable marking is a local deadlock or not. However it is fairly obvious whether a marking is a global deadlock. We could therefore call a global deadlock an *explicit* error marking and a local deadlock an *implicit* error marking. So far, we have three explicit error markings that imply unsoundness: a global deadlock, an unsafe marking, and an *improper termination*, i.e., a reachable marking that has a token on the sink and some token elsewhere. An execution trace that ends in an explicit error marking can be considered as diagnostic information for unsoundness.

2.2 Structural Characterizations for Safeness and Liveness in Free-Choice Nets

We will use the following concepts from Petri net and graph theory to build graphical diagnostic information: *siphons, circuits, handles and bridges.* A *siphon*[2] is a non-empty set S of places such that ${}^\bullet S \subseteq S^\bullet$; S is *minimal* if it does not contain another siphon. The central property of a siphon, which makes it suitable as diagnostic information, and which is an immediate consequence of its definition, is that if S is not marked in a marking m, then each transition $t \in S^\bullet$ is dead in m. We will also identify a siphon S with the subnet *generated* by it, which is the subnet (P', T', F') such that $P' = S$ and $T' = {}^\bullet S$.

A path is said to be *trivial* if it contains only one element. A *circuit* of N is a non-trivial path from an element to itself such that all other elements are pairwise distinct. A *handle* on a subnet N' is a non-trivial path H in N from some element x of N' to some element y of N' such that H is disjoint from N' apart from x and y. H is a *P/T-handle* if $x \in P$ and $y \in T$ and a *T/P-handle* if $x \in T$ and $y \in P$. A *bridge* between two subnets N' and N'' is a non-trivial

[2] Unfortunately, a siphon is also often called a *deadlock*.

path B from an element x of N' to an element y of N'' that is disjoint from N' apart from x and disjoint from N'' apart from y. B is a *T/P-bridge* if $x \in T$ and $y \in P$.

N is *structurally live* if there exists a marking from which it is live; it is *structurally bounded* if N is bounded from each marking of N. N is *SLB* if it is structurally live and structurally bounded. Note that SLB is equivalent with the notion of *well-formedness* [2] of a free-choice net, where N is *well-formed* if there exists a marking from which N is live and bounded. This is because a well-formed free-choice net is structurally bounded [2, Theorem 5.8].

The following propositions are directly derived from the literature. Let for the rest of this paper, N denote a free-choice workflow net and $\overline{N}$ its connected version.

Theorem 2 ([2, Theorem 6.17],[10, Theorem 4.2]). *We have: (i) $\overline{N}$ is safe and live from the initial marking iff it is SLB and every siphon contains the source and (ii) $\overline{N}$ is SLB iff it contains no circuit with a T/P-handle and for every circuit C with a P/T-handle H, there is a T/P-bridge between C and H.*

Imagine the connected versions for the examples in Fig. 2. Then the red part in Fig. 2(a) shows a siphon that does not contain the source. In Figs. 2(b) and (c), the red part shows a T/P-handle on a circuit (blue part + imagined return edge). Figure 2(d) shows a P/T-handle without bridges (red part) on a circuit (part of the blue plus imagined return edge). We discuss these examples in more depth in Sect. 3.1.

It can be computed in polynomial time whether every siphon is initially marked [5]. However, we do not know any way to compute Theorem 2(ii) in polynomial time. Moreover, condition (ii) of Theorem 2 has another drawback to be used directly as diagnostic information. Although a circuit with a handle is an explicit error condition as it is easily verified by a user, a circuit with a handle without bridges is less suited because the absence of bridges is not obvious to a user in a large graph.

3 New Diagnostic Information for Unsoundness

In this section, we present a new characterization of unsoundness in terms of the presence of three types of graph structures, which are suitable as diagnostic information. The new characterization is derived from the Esparza-Silva characterization (Theorem 2(ii)) with an essential change and some additional adaptations based on the structure of workflow nets. We present the new characterization in Sect. 3.1, and we show in Sect. 3.2 that each of the error patterns indeed indicates unsoundness.

3.1 A New Structural Characterization of Unsoundness

The new characterization is based on the new notion of a *DQ-siphon*:

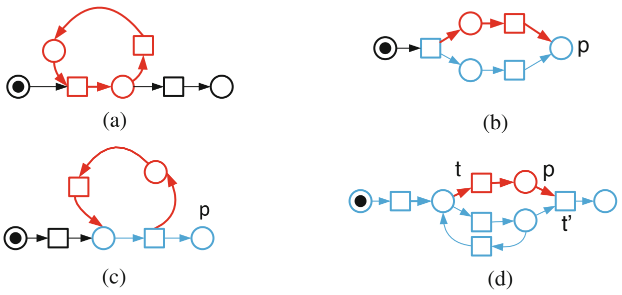


Fig. 2. Examples of the three error patterns: (a) A siphon (red) that does not contain the source, (b) A path to the sink (blue) with a (forward) T/P-handle (red), (c) A path to the sink (blue) with a (backward) T/P-handle (red), (d) A DQ-siphon (blue) with a P/T-handle (red).

Definition 1. *A* decreasing quasi-component siphon, DQ-siphon for short, *is a siphon S such that for each transition t, $|t^\bullet \cap S| \leq 1$.*

A DQ-siphon has initially at most one token in N and since the number of tokens in it cannot increase, it has never more than one token. Recall that a path in a graph is said to be *simple* if it does not visit any node twice. The new characterization is:

Theorem 3. *N is unsound iff at least one of the following statements holds:*

1. N has a siphon that does not contain the source.
2. N has a simple path from some element to the sink that has a T/P-handle.
3. N has a DQ-siphon S with a P/T-handle (more precisely, the subnet generated by S has a P/T-handle).

The proof of Theorem 3 is deferred to Sect. 3.3. Figure 2 shows examples for the error patterns. They indicate unsoundness as follows. As stated already above, all transitions $t \in S^\bullet$ are dead once the siphon S is unmarked, which for Theorem 3(i), is the case already in the initial marking, cf. Figure 2(a). For a simple path with a T/P-handle, we can distinguish two cases, a *forward handle*, cf. Fig. 2(b) and a *backward handle*, cf. Fig. 2(c). The intuition of the forward handle is that we can execute, unless there is an obstruction by a deadlock, the path and the handle independently, which generates two tokens at the merging place p, i.e., an unsafe marking. Likewise, the intuition for the backward handle is that, if not obstructed by a deadlock, the handle and the path can be executed concurrently to produce an unbounded number of tokens at p. Finally, for a DQ-siphon S, we can assume it contains the source (otherwise we resort to Theorem 3(i)). Since S has always at most one token, it becomes unmarked when the first transition of the handle occurs, which marks the handle and all transitions $t \in S^\bullet$ become dead. The token on the handle can be brought, unless obstructed by a deadlock, to the last place $p \in {}^\bullet t'$ of the handle, where t' is the last transition of the handle, hence $t' \in S^\bullet$ and t' is therefore dead, which is a local deadlock. These intuitions are substantiated in Sect. 3.2 below.

3.2 Error Patterns Indicate Unsoundness

In this section, we prove that given that the workflow net exhibits one of the error patterns in Theorem 3, we can compute an execution trace to an explicit error marking in polynomial time. The existence of the traces forms the underpinning for the intuition for the error patterns given above. In particular, this proves the 'if' direction of Theorem 3. The execution traces can also be used to complement the main diagnostic information from Theorem 3. We need the following lemma, which generalizes the central property of a sound free-choice net that every place can be marked and every transition can be enabled from the initial marking.

Lemma 1. *Let π be a path in N from a place p_1 to a place p_2, and m a marking of N such that p_1 is marked in m. Then, we can compute in $O(|P|^2)$ time an execution sequence from m to a marking m' such that m' marks p_2 or m' is an explicit error marking, i.e., a global deadlock, an improper termination, or unsafe.*

The proof of Lemma 1 is provided in the extended version of this paper [7]. The following observation will help us later: An unsafe marking or an improper termination marking are two special cases of a more general error marking:

Lemma 2. *Let p_1, p_2 be two distinct places of N such that there exists a simple path from p_1 via p_2 to the sink of N. If there exists a reachable marking m that marks both places p_1 and p_2, then N is unsound. An execution sequence from m to an explicit error marking can be computed in $O(|P|^2)$ time.*

By help of Lemmas 1 and 2, we can show the following:

Theorem 4. *If N exhibits any of the three error patterns in Theorem 3, then a trace of N to an explicit error marking can be computed in $O(|P|^2)$ time from the error pattern.*

A complete proof of Theorem 4 is provided in the extended version of this paper [7]. We now give a brief account of the proof for each error pattern. For any siphon that does not contain the source, there is a path from the source to a place of the siphon. As no place of an unmarked siphon can be marked, it follows directly from Lemma 1 that we can compute, in quadratic time, a trace from the initial marking to an error marking.

Consider now the case where N has a simple path π from some element e_1 to the sink that has a T/P-handle H. Applying Lemma 1 to a path from the source of N to the place p_1 that precedes the first transition of H, we obtain either an error marking or a marking that marks p_1. In the latter case, we apply Lemma 1 to the path H. This will result in an error marking or, without going into the details, a marking m where two places of π are marked. By Lemma 2, we can obtain from m, an error trace in quadratic time.

Finally consider the case where N has a DQ-siphon S with a P/T-handle H. By the DQ-siphon property, S has always at most one token. Applying Lemma 1 to a path from the source of N to the first place of H, we obtain, in quadratic time, an execution leading to an error marking or a marking that marks a single place in S, viz. the first place of H. In the latter case, we apply Lemma 1 to the path H and we obtain an execution that leads to either an error marking or a marking that marks the last place p of H. Because S became unmarked through the execution of the handle, the last transition t' of H is dead because $t' \in S^\bullet$, cf. Fig. 2(d). Hence we obtain an explicit error marking by once again applying Lemma 1 to a path from p to the sink of N.

3.3 Proof of Theorem 3

In Theorem 4, we have shown one direction of Theorem 3. To show the other direction of Theorem 3, suppose N is unsound. Due to Theorem 1, $\overline{N}$ is not safe or not live from the initial marking. Due to Theorem 2, we have either (i) some siphon of $\overline{N}$ does not contain the source, (ii) $\overline{N}$ contains a circuit with a T/P-handle, or (iii) $\overline{N}$ contains a circuit with a P/T-handle without T/P-bridges. In case (i), we conclude that N has a siphon that does not contain the source because each siphon in $\overline{N}$ is also a siphon in N. For the cases (ii) and (iii), we use the following lemmas.

Lemma 3. *If $\overline{N}$ has a circuit with a T/P-handle, then N has a path to the sink with a T/P-handle, which can be computed in $O(|F|)$ time.*

Lemma 4. *If $\overline{N}$ has a circuit with a P/T-handle without T/P-bridge, then N has a minimal siphon S with a P/T-handle or a siphon that does not contain the source.*

When the minimal siphon S with P/T-handle that Lemma 4 returns is not a DQ-siphon, then there is a transition t such that $|t^\bullet \cap S| > 1$. Then, we apply Lemma 5 below and obtain a path to the sink of N with a T/P-handle, which concludes the proof of Theorem 3. Lemma 5 will be re-used in Sect. 4.

Lemma 5. *If S is a minimal siphon of $\overline{N}$ such that there is a transition t such that $|t^\bullet \cap S| > 1$, then $\overline{N}$ has a circuit with a T/P-handle, which can be computed in $O(F)$ time.*

The proofs of the lemmas are provided in the extended version of this paper [7].

4 Computation of Structural Diagnostic Information

In this section, we show that structural diagnostic information as given by Theorem 3 can be computed in polynomial time. We employ and extend an algorithm by Kemper and Bause [5], which in turn is based on the *rank equation* for free-choice nets. We need the following definitions. A *state machine (also called S-graph)* is a Petri net such that for each transition t, $|^\bullet t| = |t^\bullet| = 1$, i.e., it has no concurrency. A *P-component (also called S-component)* of $\overline{N}$ is a

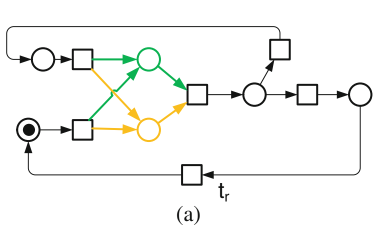


Fig. 3. Two nets, each decomposed into two overlapping P-components S_1 (green part + black part) and S_2 (orange part + black part) (a) is safe and live (sound), (b) is safe but not live (unsound)

subnet (P', T', F') of $\overline{N}$ that is a strongly connected state machine such that $T' = {}^\bullet P' \cup P'^\bullet$. $\overline{N}$ is *state-machine decomposable (SMD)* if each element of $\overline{N}$ belongs to some P-component of $\overline{N}$. Figure 3 shows two nets that are SMD, each with a decomposition. SMD is necessary for $\overline{N}$ to be SLB but not sufficient. The net in Fig. 3(a) is SLB (sound), the net in Fig. 3(b) is not SLB (unsound). More precisely, SMD is sufficient for $\overline{N}$ being safe, but not for being live. Note that the net in Fig. 3(b) has deadlocks.

The difference between $\overline{N}$ being SMD and SLB can be captured using the *rank equation*. To this end, we need the notion of a *cluster*. For a transition t of $\overline{N}$, let $[t] = \{t' \in T \mid {}^\bullet t \cap {}^\bullet t' \neq \emptyset\}$ be the *cluster* of t. In a free-choice net, each cluster is an equivalence class, hence clusters provide a partition of T. We have the following:

Theorem 5. ([16]) $\overline{N}$ is SLB iff $\overline{N}$ is SMD and

$$\text{rank}(\overline{N}) = |\{[t] \mid t \in T\}| - 1 \tag{1}$$

where rank$(\overline{N})$ is the rank of the incidence matrix of $\overline{N}$.

Note that the rank of the incidence matrix can be computed in time cubic in $\max(|P|, |T|)$. For the example net (connected version) in Fig. 3(b), the rank of the incidence matrix is 6 and the number of clusters is also 6.

We now present an algorithm that decides whether $\overline{N}$ is safe and live, which is, as stated earlier, equivalent with N being sound. This algorithm, shown as Algorithm 1, returns corresponding diagnostic information for the connected net, viz. either a siphon that does not contain the source of N (line 1), a circuit with a T/P-handle or a DQ-siphon with a P/T-handle (lines 2 and 10). This can then be post-processed into the desired diagnostic information for N as stated in Theorem 3, which we will show below in Theorem 7.

Algorithm 1 proceeds as follows. It first checks whether $\overline{N}$ has a siphon that does not contain the source of N using an algorithm by Esparza and Silva [4, Algorithm 6.6], cf. also [5]. This is denoted as *SiphonCheck*$(\overline{N})$.

Next the algorithm tries to compute a state-machine decomposition of $\overline{N}$ using an algorithm by Kemper and Bause [5, Algorithm 17], which is denoted

Algorithm 1. Decides whether $\overline{N}$ is safe and live and returns diagnostic information.

```
CheckSafeAndLive($\overline{N}$)
1: SiphonCheck($\overline{N}$): if $\overline{N}$ has a siphon $S$ that does not include the source of $N$ then return (false, $S$) end if
2: SMD-Check($\overline{N}$): if a minimal siphon $S$ of $\overline{N}$ is found that is not a state machine then return (false, $D$) where $D$ is either a circuit with a T/P-handle or a DQ-siphon with a P/T-handle computed from $S$ end if
3: Rank-Check($\overline{N}$): if the rank Eq. (1) holds for $\overline{N}$ then return true end if
4: loop
5:   Unprocessed := the set of all places of $\overline{N}$
6:   loop
7:     pick $p \in$ Unprocessed
8:     $N'$ := delete($p, \overline{N}$)
9:     if $N'$ is not empty then
10:      SMD-Check($N'$): if a minimal siphon $S$ of $N'$ is found that is not a state machine then return (false, $D$) where $D$ is either a circuit with a T/P-handle or a DQ-siphon with a P/T-handle computed from $S$ end if
11:      Rank-Check($N'$): if the rank Eq. (1) does not hold for $N'$ then $\overline{N}$ := $N'$; break end if
12:    end if
13:    Unprocessed := Unprocessed $\setminus \{p\}$
14:  end loop
15: end loop
```

as *SMD-Check($\overline{N}$)*. This algorithm computes a cover of the net with minimal siphons, i.e., a set of minimal siphons such that each place is in some minimal siphon. It then checks whether each of the computed minimal siphons is a P-component. If this is not the case, $\overline{N}$ is not SLB [5, Theorem 6]. A minimal siphon S of $\overline{N}$ is not a P-component iff one of the following two conditions holds: (i) there is a transition t such that $|t^\bullet \cap S| > 1$. In this case, we can compute a circuit with a T/P-handle as proved in Lemma 5. In the other case (ii), we have $S^\bullet \setminus {}^\bullet S \neq \emptyset$. In this case, we can compute a P/T-handle attached to S as proved in Lemma 6. Note that condition (i) is checked first, so when condition (ii) is checked, we know that there is no transition t such that $|t^\bullet \cap S| > 1$ and hence, if (ii) holds for S, S must be a DQ-siphon.

Lemma 6. *If S is a minimal siphon of $\overline{N}$ such that $S^\bullet \setminus {}^\bullet S \neq \emptyset$, then S has a P/T-handle in $\overline{N}$, which can be computed in $O(|F|)$ time.*

In case the SMD-Check in line 2 passes, we know that each of the computed minimal siphons is a P-component and therefore, $\overline{N}$ is SMD and hence safe. The algorithm proceeds with computing the rank of the incidence matrix of $\overline{N}$ by standard techniques and checks the rank Eq. (1), denoted *Rank-Check($\overline{N}$)*. If the equation holds, then $\overline{N}$ is SLB due to Theorem 5 and safe and live due to Theorem 2. If the rank equation does not hold, we know that $\overline{N}$ is not live.

In this case, we iteratively reduce the net as described below until the SMD check for the reduced net returns diagnostic information.

For the reduction, we define $N' = \text{delete}(p, \overline{N})$ as the largest strongly connected subnet of $\overline{N}$ that does not contain p. Figure 4 shows in (a) the result of $\text{delete}(p_1, \overline{N})$ and (b) the result of $\text{delete}(p_2, \overline{N})$ where $\overline{N}$ is the net shown in Fig. 3(b). $N' = \text{delete}(p, \overline{N})$ can be computed in linear time. N' may be empty for a particular p, but if $\overline{N}$ is SMD and not SLB, a place p can be found such that $\text{delete}(p, \overline{N})$ is not empty (see Lemma 8). If N' is not empty, we perform an SMD check on it. If the SMD check on the reduced net N' (line 10) generates diagnostic information, this information is returned as diagnostic information of the original net $\overline{N}$. We argue in Lemma 7 below that this is correct.

Lemma 7. *Let $N' = \text{delete}(p, \overline{N})$ for some p such that N' is nonempty.*

(i) If S is a minimal siphon of N' such that there is a transition t such that $|t^\bullet \cap S| > 1$, then S is also a minimal siphon of $\overline{N}$ such that there is a transition t such that $|t^\bullet \cap S| > 1$.
(ii) If S is a minimal siphon of N' such that $S^\bullet \setminus {}^\bullet S \neq \emptyset$, then S is also a minimal siphon of $\overline{N}$ such that $S^\bullet \setminus {}^\bullet S \neq \emptyset$.

Otherwise, if the SMD check passes for N', we check the rank equation for N'. If the rank equation for N' does not hold, we know that N' is not SLB. Hence we found a smaller net that contains an error, and, in this sense, the reduction was successful. In this case, we proceed with N' as the subject of further analysis, we break from the inner loop and go into a new iteration of the outer loop to reduce the net further.

If the rank equation holds for N', then N' is SLB, i.e., removal of p has removed the deadlock (i.e., made the net live). In this case, we try to reduce with another place p.

Lemma 8 below proves that, as long as a strongly connected free-choice workflow net is SMD but not SLB, we find a place p such that the reduced net is not empty and not SLB, so either the SMD check or the rank check fails on the reduced net. In the former case, we are done and in the latter case, we proceed with a smaller net that is SMD but not SLB. Since this reduction can be performed at most $|P|$ many times, the SMD check must fail eventually and return diagnostic information.

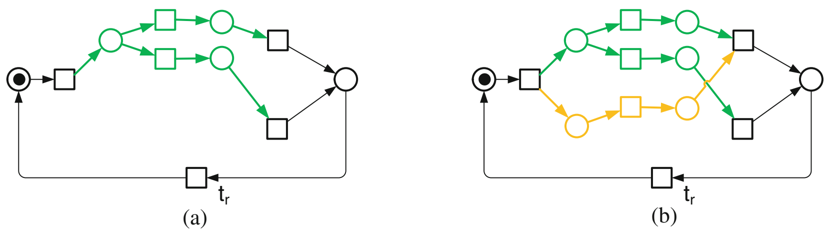


Fig. 4. The result of (a) $\text{delete}(p_1, \overline{N})$ and (b) $\text{delete}(p_2, \overline{N})$ where $\overline{N}$ is the net in Fig. 3(b)

Lemma 8. *Let N' be a strongly connected subnet of $\overline{N}$ such that N' is SMD. If N' is not SLB then there exists a place p such that* $\text{delete}(p, N')$ *is nonempty and not SLB.*

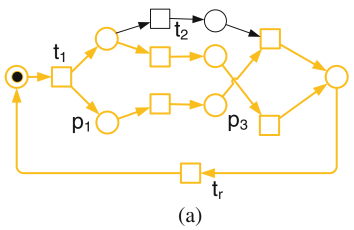
We can now conclude with the correctness of Algorithm 1.

Theorem 6. *Algorithm 1 decides whether $\overline{N}$ is safe and live from the initial marking. If not, it outputs either a siphon that does not include the source, a DQ-siphon with a P/T-handle, or a circuit with a T/P-handle. Algorithm 1 completes in $O(|P|^2 * max(|P|, |T|)^3)$.*

Proof. We have already shown the correctness of Algorithm 1. For the complexity, note that the time complexity of the Siphon Check is $O(|P|^2 * |T|)$ [5]. The complexity of the SMD Check is $O(|P| * (|P| + |T| + |F|))$ because the worst case is bounded by the need to identify $|P|$ minimal siphons and finding a minimal siphon containing a place can be done in time $O(|P| + |T| + |F|)$ [17]. Finally, checking the rank of the incidence matrix can be done in $O(max(|P|, |T|)^3)$. Thus, the complexity is dominated by the computation of the rank of the matrix. Algorithm 1 performs, at most, $|P|^2$ computations of the rank. Thus, the worst case time complexity of Algorithm 1 is $O(|P|^2 * max(|P|, |T|)^3)$.

As an example, we consider again the net from Fig. 3(b). This net passes the siphon check but might or might not pass the first SMD check, depending on the nondeterministic choice of minimal siphons in the SMD check. Suppose the algorithm finds the state machine decomposition shown in Fig. 3(b). The subsequent rank check for this net fails. If in the reduction, place p_1 is picked, we obtain the net shown in Fig. 4(a), which is SLB and hence passes the subsequent SMD and rank check. If however place p_2 is picked for reduction, we obtain the net shown in Fig. 4(b), which is not SMD and hence the SMD check must fail. Figure 5(a) shows a minimal siphon S (in orange) that covers p_3. S is not a state machine for two reasons: $|t_1^\bullet \cap S| > 1$ and $t_2 \in S^\bullet \setminus {}^\bullet S$. The first violation is picked up first, and from S and t_1, a circuit with a T/P-handle is computed (Lemma 5), which is shown in Fig. 5(b).

The diagnostic information obtained from Algorithm 1 for $\overline{N}$ can be post-processed into diagnostic information for N. We obtain:



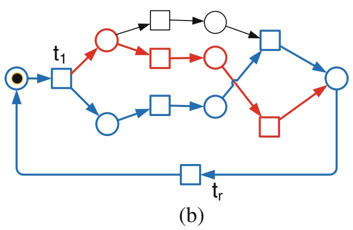


Fig. 5. (a) A minimal siphon S (in orange) that is not a state machine witnessed by t_1 (b) A circuit (in blue) with a T/P-handle (in red) computed from S and t_1.

Theorem 7. *Soundness of N can be decided in time $O(|P|^2 * max(|P|, |T|)^3)$ such that the algorithm returns one of the structural error patterns in Theorem 3 in case N is unsound.*

Proof. The algorithm creates the connected version of N and applies Algorithm1 to it. A siphon that does not contain the source of N is returned as it is. Note that each siphon in $\overline{N}$ is a siphon in N. A DQ-siphon S with a P/T-handle in $\overline{N}$ is a DQ-siphon with a P/T-handle in N and is also returned as it is. Note that in this case, S contains the source (otherwise we would return the first error pattern) and therefore the sink, because of the siphon property. Therefore, the return edge belongs to the siphon and cannot belong to the handle, so removal of the return edge retains the error pattern. A circuit with a T/P-handle in $\overline{N}$ is transformed into a path to the sink with a T/P-handle using Lemma 3. The complexity of this algorithm is dominated by the complexity of Algorithm1.

5 Experimental Evaluation

We implemented our technique as a research prototype in the IBM WebSphere Business Modeler and used this implementation to evaluate the performance of our technique and demonstrate that our technique provides useful diagnostic information.

Our implementation translates business process models described in the IBM WebSphere Business Modeler language into workflow graphs [11] and then applies a completion technique by Kiepuszewski et al. [18] (cf. also [11]) to obtain a workflow graph with a unique sink, as required by our technique.

Data Set. We ran our control flow analysis on 1353 business process models from industrial projects in the insurance and banking domain, which were also used as a benchmark in other work [11,12]. The 1353 models are organized into four libraries A, B1, B2, and B3. The libraries B1 and B2 are older releases of the library B3, where some process models were refined or changed, possibly removing or adding errors. Counting only libraries A and B3, we have 703 unique original models. Over the four libraries, the average number of nodes per derived Petri net ranges between 89 and 107. There are several large nets with up to 627 nodes. For example, 47 nets from library B3 have 200 or more nodes. Some models have state spaces with more than 1 million states, cf. [11].

We validated the correctness of the results, i.e., the detected soundness or unsoundness of the processes, by comparing them with the results obtained during previous experiments applying different analysis techniques to the same data [11]. The techniques agree on all process models.

Performance Evaluation. We ran our experiments on a notebook with a 2 GHz processor and 2 GB RAM. The analysis times are computed as an average over 10 runs. They also include the time spent by the tool to generate the error report for the user. The overhead for loading the process models from disk into memory during the first run is excluded.

Table 1. Experimental results.

Library	**A**	**B1**	**B2**	**B3**	**Total**
Processes	282	287	363	421	1,353
Unsound processes	130	180	202	214	726
Siphon without source	0	17	15	13	45
Path with a T/P-handle	47	138	158	174	517
DQ-siphon with a P/T-handle	83	25	29	27	164
Average library analysis time [ms]	752	492	627	928	2799

Table 1 summarizes the results of our experiments for the four libraries. All three structural error patterns occurred in the data set. The average time to analyze a process model is 2ms, which is sufficiently fast to run our analysis and provide immediate feedback while the process model is being developed. In particular, our control-flow analyzer is roughly 2 to 6 times faster than existing tools that produce diagnostic information, that is, tools based on state space exploration [11].

Our implementation highlights structural error patterns inside the process model. For instance, Fig. 6 shows the error report for a path with a T/P-handle. Note that the activity 'Confirm Customer Requirement' produces a token on each of its outgoing edges according to the semantics of the used high-level language. Therefore, this activity creates a concurrent fork (as $f1$ in Fig. 1(a)) in the corresponding workflow graph. In Fig. 6, the activity 'Confirm Customer Requirement' is on a simple path (blue) to the final place and starts multiple concurrent paths including a T/P-handle (in red). Both concurrent paths join on the alternative merge (empty diamond with error flag) without being properly synchronized, which would need a concurrent join instead (as $j1$ in Fig. 1(a)). We provide examples for the other two structural patterns in the extended version of this paper [7].

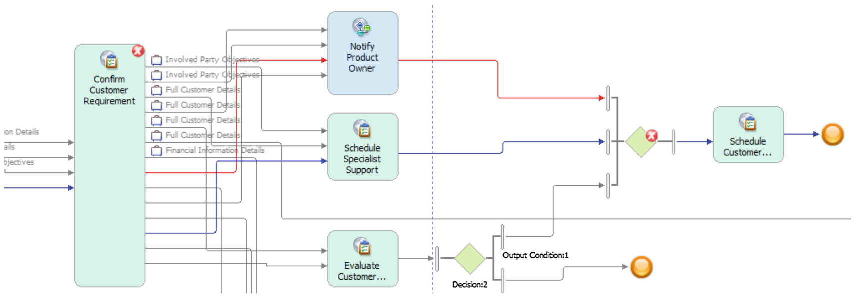


Fig. 6. Screenshot of the diagnostic information for a T/P-handle.

Diagnostic Information. As this example illustrates, our technique provides concise diagnostic information. In particular, the structural patterns have advantages over the traces computed by state space exploration. They can be displayed and understood directly in the context of the process model (which is difficult for traces that may be long and may contain iterations). Moreover, they are concise and do not contain any information that is not immediately relevant for understanding and fixing the error (in contrast to traces, which typically describe complete executions including aspects irrelevant for the error).

6 Conclusion

We presented a new characterization of control-flow errors in workflow graphs in terms of three structural error patterns, as well as an algorithm that decides whether one of the error patterns is present. To our knowledge, this is the first algorithm that runs in polynomial time and produces diagnostic information. It is applicable to a wide range of business process models modeled in languages such as BPMN or UML activity diagrams; features of these languages that do not translate to workflow graphs are either used rarely or orthogonal to soundness checking. Our experiments show that our technique is sufficiently fast to give instant feedback while the process model is being developed; the experiments also provide anecdotal evidence that our technique generates useful diagnostic information. Evaluating the benefit of this information in comparison, for instance, to error traces, requires a user study and involves various aspects beyond the scope of this paper, such as error visualizations and user interactions. We leave such a study for future work.

References

1. Favre, C., Fahland, D., Völzer, H.: The relationship between workflow graphs and free-choice workflow nets. Inf. Syst. **47**, 197–219 (2015)
2. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press, Cambridge (1995)
3. van der Aalst, W.M.P.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) Application and Theory of Petri Nets 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
4. Esparza, J., Silva, M.: A polynomial-time algorithm to decide liveness of bounded free choice nets. Theor. Comput. Sci. **102**(1), 185–205 (1992)
5. Kemper, P., Bause, F.: An effcient polynomial-time algorithm to decide liveness and boundedness of free-choice nets. In: Jensen, K. (ed.) ICATPN 1992. LNCS, vol. 616, pp. 263–278. Springer, Heidelberg (1992)
6. Esparza, J.: Reduction and synthesis of live and bounded free choice Petri nets. Inf. Comput. **114**(1), 50–87 (1994)
7. Favre, C., Völzer, H., Müller, P.: Diagnostic information for control-flow analysis of workflow graphs (a.k.a. free-choice workflow nets). Technical Report, ETH Zurich, revised, 2015, revised (2016). http://e-citations.ethbib.ethz.ch

8. Favre, C.: Detecting, Understanding, and Fixing Control-Flow Errors in Business Process Models. Ph.D. thesis, Department of Computer Science, ETH Zurich (2014)
9. Koehler, J., Vanhatalo, J.: Process anti-patterns: How to avoid the common traps of business process modeling. Technical Report RZ 3678, IBM Research, Also published in the WebSphere Developer Technical Journal in 2007, May 2007
10. Esparza, J., Silva, M.: Circuits, handles, bridges and nets. In: Rozenberg, G. (ed.) Advances in Petri Nets 1990. LNCS, vol. 483, pp. 210–242. Springer, Heidelberg (1989)
11. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on demand: Instantaneous soundness checking of industrial business process models. Data Knowl. Eng. **70**(5), 448–466 (2011)
12. Lohmann, N., Fahland, D.: Where did I go wrong? In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 283–300. Springer, Heidelberg (2014)
13. Ball, T., Naik, M., Rajamani, S.K.: From symptom to cause: Localizing errors in counterexample traces. In: POPL, Proceedings, pp. 97–105, ACM (2003)
14. Groce, A.: Error explanation with distance metrics. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 108–122. Springer, Heidelberg (2004)
15. Verbeek, H.M.W.E., Basten, T., van der Aalst, W.M.P.: Diagnosing workflow processes using Woflan. Comput. J. **44**(4), 246–279 (2001)
16. Esparza, J.: Synthesis rules for Petri nets, and how they lead to new results. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR '90 Theories of Concurrency: Unification and Extension. LNCS, vol. 458, pp. 182–198. Springer, Heidelberg (1990)
17. Kemper, P.: Linear time algorithm to find a minimal deadlock in a strongly connected free-choice net. In: Ajmone Marsan, Marco (ed.) ICATPN 1993. LNCS, vol. 691, pp. 319–338. Springer, Heidelberg (1993)
18. Kiepuszewski, B., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of control flow in workflows. Acta Inf. **39**(3), 143–209 (2003)