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
[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. Fig. 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 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.

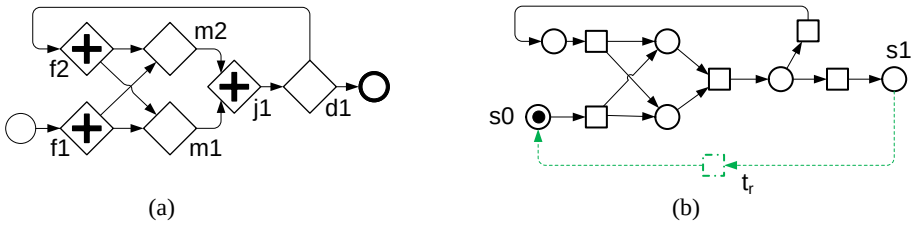

Fig. 1. (a) A workflow graph, (b) its corresponding free-choice workflow net (without dashed part) and its connected version (with dashed part)

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.

Figs. 2(a) and (d) (ignore the coloring for now) show simple examples of workflow nets with deadlocks. Figs. 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 [4–6, 2] 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 outsourced into an appendix in this version. Additional detail can also be found in an extended version of this paper in a thesis [7].

Related work. Our structural error patterns are similar to control-flow 'anti-patterns', which are sometimes given to modeling practitioners [8] 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 [9]. 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 [10] 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 [11], which requires additional techniques to trim the trace [12, 13], 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 [14] 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[3] $|{}^\bullet t_1| = |{}^\bullet t_2| = 1$. 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.

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. Fig. 1(b) with dashed part. Note that the underlying graph of $\overline{N}$ is indeed strongly connected.

[3] 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.

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*[4] 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 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, Thm.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, Thm.6.17],[9, Thm.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*

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

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). Fig. 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 Thm. 2(ii) in polynomial time. Moreover, condition (ii) of Thm. 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 (Thm. 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*:

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 Thm. 3 is deferred to Sect. 3.3. Fig. 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 Thm. 3(i), is the case already in the initial marking, cf. Fig. 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

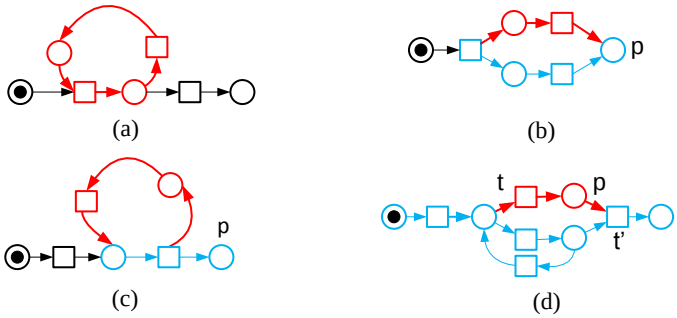

Fig. 2. Examples of the three error patterns: (a) A siphon (red) that does not contain the source, (b) A path to the sink (blue) with a (forward) T/P-handle (red), (c) A path to the sink (blue) with a (backward) T/P-handle (red), (d) A DQ-siphon (blue) with a P/T-handle (red).

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 Thm. 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 Thm. 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 Thm. 3. The execution traces can also be used to complement the main diagnostic information from Thm. 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 Appendix A. 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 Thm. 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 Appendix B. 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 Thm. 3

In Thm. 4, we have shown one direction of Thm. 3. To show the other direction of Thm. 3, suppose N is unsound. Due to Thm. 1, $\overline{N}$ is not safe or not live from the initial marking. Due to Thm. 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 Thm. 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 Appendices C, D, and E.

4 Computation of Structural Diagnostic Information

In this section, we show that structural diagnostic information as given by Thm. 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 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 $\overline{N}$. Fig. 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 ([15]). *$\overline{N}$ is SLB iff $\overline{N}$ is SMD and*

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

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

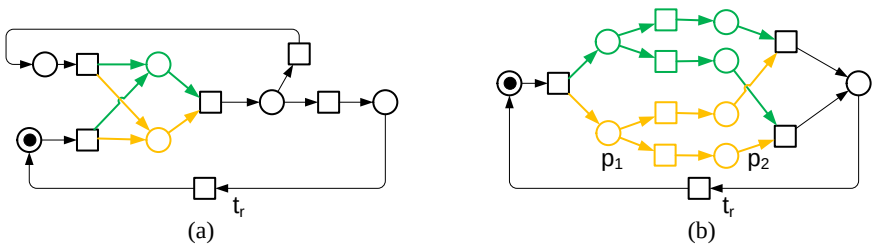

Fig. 3. Two nets, each decomposed into two overlapping P-components S_1 (green part + black part) and S_2 (orange part + black part) (a) is safe and live (sound), (b) is safe but not live (unsound)

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 equation (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 equation (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**

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 Thm. 3, which we will show below in Thm. 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, Alg.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, Alg. 17], which is denoted as *SMD-Check($\overline{N}$)*. This algorithm computes a cover of the net with minimal siphons, i.e., a set of 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, Thm.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 equation (1), denoted *Rank-Check($\overline{N}$)*. If the equation holds, then $\overline{N}$ is SLB due to Thm. 5 and safe and live due to Thm. 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. Fig. 4 shows in (a) the result of delete$(p_1, \overline{N})$ and (b) the result of 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 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.

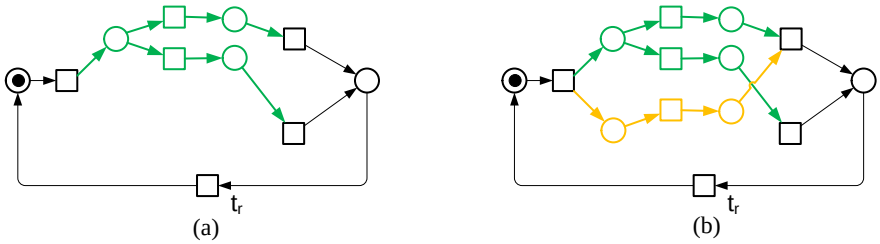

Fig. 4. The result of (a) delete$(p_1, \overline{N})$ and (b) delete$(p_2, \overline{N})$ where $\overline{N}$ is the net in Fig. 3(b)

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.

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* delete(p, N') *is nonempty and not SLB.*

We can now conclude with the correctness of Alg. 1.

Theorem 6. *Alg. 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. Alg. 1 completes in $O(|P|^2 * max(|P|, |T|)^3)$.*

Proof. We have already shown the correctness of Alg. 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 $O(|P| + |T| + |F|)$ [16]. 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. Alg. 1 performs, at most, $|P|^2$ computations of the rank. Thus, the worst case time complexity of Alg. 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. Fig. 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 Alg. 1 for $\overline{N}$ can be post-processed into diagnostic information for N. We obtain:

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 Thm. 3 in case N is unsound.*

Proof. The algorithm creates the connected version of N and applies Alg. 1 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

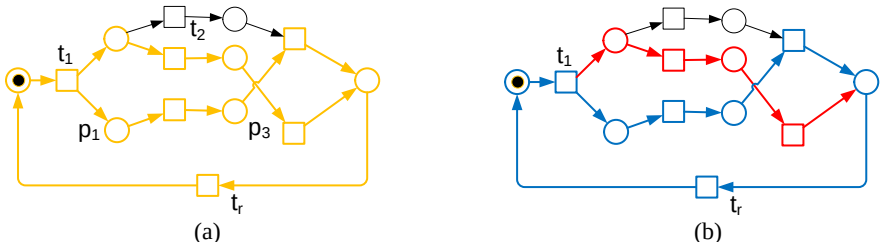

Fig. 5. (a) A minimal siphon S (in orange) that is not a state machine witnessed by t_1 (b) A circuit (in blue) with a T/P-handle (in red) computed from S and t_1.

(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 Alg. 1.

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 [10] and then applies a completion technique by Kiepuszewski et al. [17] (cf. also [10]) 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 [10, 11]. 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. [10].

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 [10]. 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

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. Experimental results.

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.

Tab. 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 [10].

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 Appendix I.

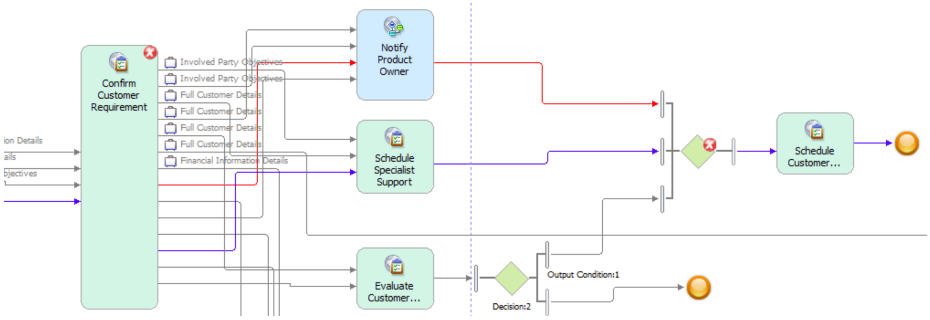

Fig. 6. Screenshot of the diagnostic information for a T/P-handle.

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. C. Favre, D. Fahland, and H. Völzer, "The relationship between workflow graphs and free-choice workflow nets," *Inf. Syst.*, vol. 47, pp. 197–219, 2015.
2. J. Desel and J. Esparza, *Free Choice Petri Nets*. Cambridge University Press, 1995.
3. W. M. P. van der Aalst, "Verification of workflow nets," in *ICATPN '97, Proceedings*, vol. 1248 of *LNCS*, pp. 407–426, Springer, 1997.
4. J. Esparza and M. Silva, "A polynomial-time algorithm to decide liveness of bounded free choice nets," *Theor. Comput. Sci.*, vol. 102, no. 1, pp. 185–205, 1992.
5. P. Kemper and F. Bause, "An efficient polynomial-time algorithm to decide liveness and boundedness of free-choice nets," in *ICATPN '92, Proceedings*, vol. 616 of *LNCS*, pp. 263–278, Springer, 1992.
6. J. Esparza, "Reduction and synthesis of live and bounded free choice Petri nets," *Inf. Comput.*, vol. 114, no. 1, pp. 50–87, 1994.
7. C. Favre, *Detecting, Understanding, and Fixing Control-Flow Errors in Business Process Models*. PhD thesis, Department of Computer Science, ETH Zurich, 2014.
8. J. Koehler and J. Vanhatalo, "Process anti-patterns: How to avoid the common traps of business process modeling," Tech. Rep. RZ 3678, IBM Research, May 2007. Also published in the WebSphere Developer Technical Journal in 2007.
9. J. Esparza and M. Silva, "Circuits, handles, bridges and nets," in *ICATPN '89, Proceedings*, vol. 483 of *LNCS*, pp. 210–242, Springer, 1989.
10. D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf, "Analysis on demand: Instantaneous soundness checking of industrial business process models," *Data Knowl. Eng.*, vol. 70, no. 5, pp. 448–466, 2011.
11. N. Lohmann and D. Fahland, "Where did I go wrong? - explaining errors in business process models," in *BPM 2014, Proceedings*, vol. 8659 of *LNCS*, pp. 283–300, Springer, 2014.

12. T. Ball, M. Naik, and S. K. Rajamani, "From symptom to cause: Localizing errors in counterexample traces," in *POPL 2003, Proceedings*, pp. 97–105, ACM, 2003.
13. A. Groce, "Error explanation with distance metrics," in *TACAS 2004, Proceedings*, vol. 2988 of *LNCS*, pp. 108–122, Springer, 2004.
14. H. M. W. E. Verbeek, T. Basten, and W. M. P. van der Aalst, "Diagnosing workflow processes using Woflan," *Comput. J.*, vol. 44, no. 4, pp. 246–279, 2001.
15. J. Esparza, "Synthesis rules for Petri nets, and how they lead to new results," in *CONCUR '90, Proceedings*, vol. 458 of *LNCS*, pp. 182–198, Springer, 1990.
16. P. Kemper, "Linear time algorithm to find a minimal deadlock in a strongly connected free-choice net," in *ICATPN '93, Proceedings*, vol. 691 of *LNCS*, pp. 319–338, Springer, 1993.
17. B. Kiepuszewski, A. H. M. ter Hofstede, and W. M. P. van der Aalst, "Fundamentals of control flow in workflows," *Acta Inf.*, vol. 39, no. 3, pp. 143–209, 2003.

Appendices: At discretion of the reviewers, not to be included in proceedings version.

A Proof of Lemma 1 and Lemma 2

We first define a few important Petri net concepts that will help us in building a trace. We start with the notion of allocation which allows us to specify an 'execution strategy'. In this appendix, we also identify a cluster with the subnet (P', T', F') generated by it, where $T' = [t], P' = {}^\bullet t$.

Definition 2 (Allocation [2]).
Let C be a subset of all the clusters of N.
An allocation of *C is a function $\alpha : C \to T$ such that $\alpha(c) \in c$ for every $c \in C$.*
A transition t is allocated by *α if $t = \alpha([t])$.*
If C contains all the clusters of N, α is called a total allocation *of N.*
An execution (or an execution sequence) σ agrees with *an allocation α if each transition $t \in \sigma$ is allocated by α.*
A path π agrees with *an allocation α if each transition $t \in \pi$ is allocated by α.*
An allocation α of C points to *a set of clusters C' if for every place $p \in P$ there exists a path π from p to a place in C' such that all the transitions of π are allocated by α.*

The notion that an allocation points to a cluster, or a set of clusters, indicates that the tokens are 'driven' to the cluster by executions following the allocation.

The following lemma establishes that an allocation pointing to a set of clusters always exists for strongly connected free-choice Petri nets. This is a known result which is proven in the literature part of the 'Pointing allocation lemma' [2].

Lemma 9 (Existence of an allocation [2]).
Let C be a non-empty set of clusters of $\overline{N}$ and $\overline{C}$ be the set of clusters of $\overline{N}$ that do not belong to C.
There exists an allocation α with domain $\overline{C}$ that points to C.

We can now show the following lemma which we use to build execution sequences following a given allocation in quadratic time.

Lemma 10 (An execution sequence to mark a place). *Let p_1 and p_2 be two places of $\overline{N}$, and α be a total allocation of $\overline{N}$ pointing to $[p_2]$.*
If a marking m of N marks p_1, we can obtain, in quadratic time, an execution sequence σ such that $m \xrightarrow{\sigma} m'$ and at least one of the following propositions is true:

1. *m' marks p_2,*
2. *m' marks the sink p_t of N, or*
3. *m' is an error marking.*

Proof. Let $C = \{[p_2]\}$ and $\overline{C}$ be the set of clusters in $\overline{N}$ that are not contained in C. By Lemma 9, there exists an allocation α of $\overline{C}$ pointing to C.
We will use α to 'drive' the construction of σ in quadratic time. To do so we start with some necessary definition and the proof of some properties:
Let $R = (P', T', F')$ be a subnet of $\overline{N}$ such that:

– $p \in P'$ *iff there exists a path in* $\overline{N}$ *from a place marked by* m *to* p *that contains only transitions allocated by* α,
– $t \in T'$ *iff* t *is allocated by* α, *and*
– $F' = F \cap ((P' \times T') \cup (T' \times P'))$.

It can be easily proven that any execution sequence σ_α *agreeing with* α *can mark only places in* P' *by induction on the length of* σ_α.

Let S *be a set of places and* m^* *be a marking. We call* marked border of m^* with respect to S, *the set of places* B *such that for each place* $p \in B$, *there exists a path* π^* *from* p *to any place* $p' \in S$ *such that* π^* *contains no place, other than* p *and possibly* p', *that is marked by* m^*.

Let $S = \{p_2\}$ *and* B *be the marked border of* m *with respect to* S. *Figure 7 illustrates schematically* R, S, *and* B. *The highlighted path represents* π. *The dashed lines represent portions of* $\overline{N}$, *which do not belong to* R. *The transitions and their connection to* $\overline{N}$ *are omitted from the illustration. When a place contains a dot (*token*), it means that the place is marked.*

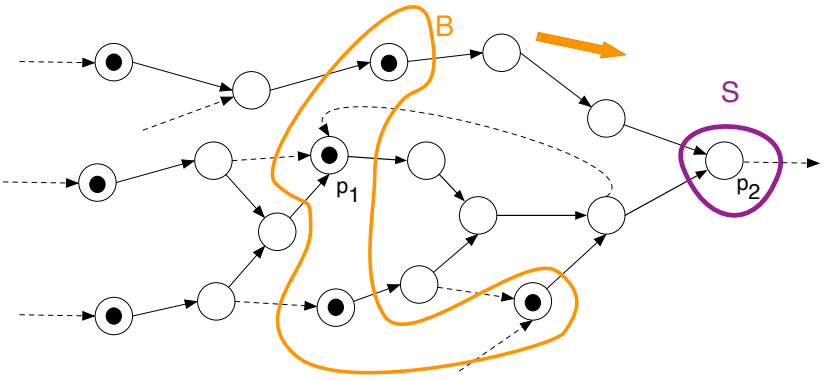

Fig. 7. Illustration of S and its marked border B.

Let σ_1 *be an execution sequence obtained by executing any allocated and enabled transitions in* $B^\bullet$ *until no allocated transition in* $B^\bullet$ *is enabled or either* p_2 *or* p_t *is marked. Note that by the free-choice property, if any transition in a cluster* c *is enabled, then* $\alpha(c)$ *is also enabled. Also note that* B *and thus* $B^\bullet$ *are updated at every transition execution. Figure 8 illustrates the result of executing* σ_1.

We define the function $\Delta(m^*, S, R) = k$ *such that* k *is the number of places that are on a path in* R *from a place in the marked border of* m^* *with respect to* S, *to a place in* S.

We show that for each transition $t \in \sigma_1$ *such that* t *is enabled by* m, *when* t *is executed, the marking* m' *resulting from the execution of* t *has the property* $\Delta(m', S, R) < \Delta(m, S, R)$: *By definition of* σ_1, $\alpha([t]) = t$. *By definition of* R, *for each place* $p \in {}^\bullet t$, R *contains only one transition in* $p^\bullet$: *the transition* t. *Therefore, we have* $\Delta(m', S, R) = \Delta(m, S, R) - |{}^\bullet t|$.

As $\Delta(m, S, R) \leq |P|$ *and* Δ *is strictly decreasing with any transition of* σ_1, *we know that the number of transitions in* σ_1 *is bounded by* $|P|$.

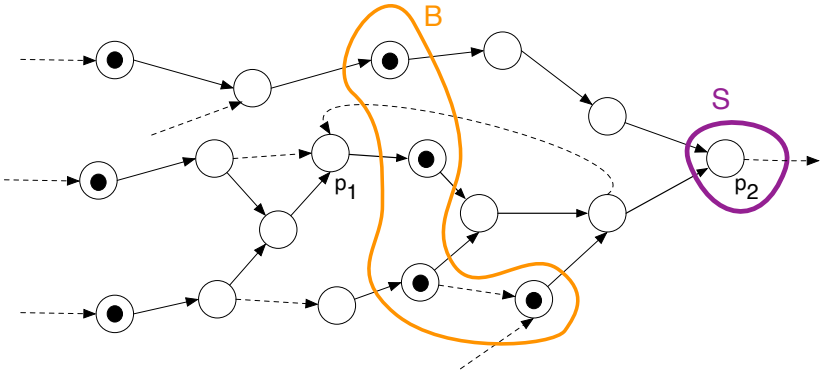

Fig. 8. Illustration of S and its marked border B after σ_1.

Let m_1 be the marking such that $m \xrightarrow{\sigma_1} m_1$. If p_2 or p_t is marked by m_1, we are done: we constructed $\sigma = \sigma_1$ in quadratic time. In the following, we only deal with the case where m_1 does not mark p_2 or p_t and there is no transition in $B^\bullet$ enabled by m_1.

We now show that for any place $p \in B$ and each marking m_i reachable from m_1, $m_i(p) \geq 1$. One place $p' \in {}^\bullet\alpha([p])$ is not marked by m_1. We show that p' never gets marked and thus $\alpha([p])$ is never enabled and $m_i(p) \geq 1$. We proceed by contradiction:

Suppose that there exists a marking m^* and an execution sequence σ^* such that $m_1 \xrightarrow{\sigma^*} m^*$ and $m^*(p') > 0$. We can assume without loss of generality that p' is the place in ${}^\bullet(B^\bullet)$ which is not marked by m_1 and is first marked during σ^*. As $m_1 \xrightarrow{\sigma^*} m^*$ and $m^*(p') > 0$, there exists a path π in R from a place p^* such that $m_1(p^*) > 0$ to p' such that no place on π, other than p^*, is marked by m_1. Thus, p^* belongs to B and therefore no transition in $p^{*\bullet}$ is enabled by m_1. To enable $\alpha([p^*])$, another place than p' in ${}^\bullet\alpha([p])$ which is not marked by m_1 would have needed to be marked by σ^* which contradicts our supposition that p' is the first.

We have established that there is, and there will be, no enabled transition in any cluster of B. If no transition at all is enabled by m_1, we are done: we constructed $\sigma = \sigma_1$ in quadratic time and m_1 is a global deadlock. In the following, we only deal with the case where at least one transition is enabled by m_1 and hence, by construction of R, one place preceding B in R is marked.

Now that we have proven the main properties of σ_1, we show how we can iteratively assign S and continue to build an execution until finding an error. Let B become S_2 and a B^* be a new set of places initially equal to B.

***Move iteration:** Let B_2 be the marked border of m_1 with respect to S_2. Figure 9 illustrates S_2 and its marked border B_2. Let σ_2 be the execution sequence obtained by executing any allocated and enabled transitions in $B_2^\bullet$ until no transition in $B_2^\bullet$ is enabled, and let m_2 be the marking such that $m_1 \xrightarrow{\sigma_2} m_2$. As argued earlier for σ_1, the number of steps in σ_2 is bounded by $|P|$. If $B_2 \cap S \neq \emptyset$, then a place in B_2 carries two tokens and m_2 is a lack of synchronization. If not, then either m_2 is a global deadlock, or there exists a transition preceding B_2 which is enabled in m_2. In the latter case, we repeat the move iteration with $S_3 := B_2$ and $B^* := B^* \cup B_2$.*

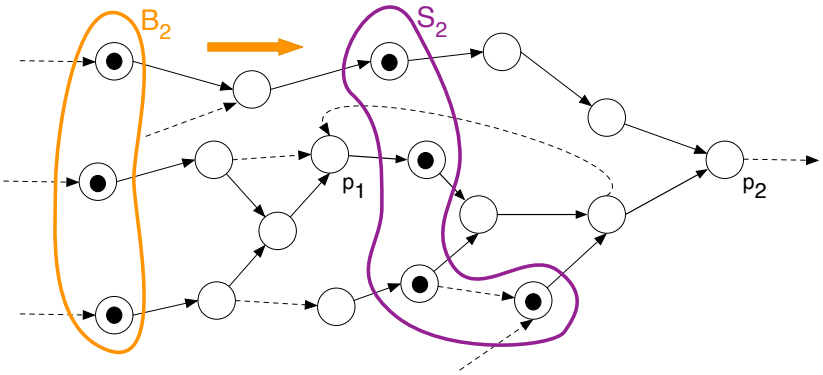

Fig. 9. Illustration of S_2 and its marked border B_2.

As the number of places is finite and the number of places on a path to a place in B^* is strictly decreasing at each move iteration, the number of times the step is repeated is bounded by $|P|$. As each step has a maximum of $|P|$ transitions and there is a maximum of $|P|$ steps, the length of σ is bounded by $|P|^2$.

The following lemma allows us to obtain, for a given path, a total allocation such that the path agrees with the allocation and the allocation points to the last element of the path.

Lemma 11 (Path allocation).
For any simple path π in N, there exists a total allocation α of N such that α points to the last element of π and for every transition in π we have $\alpha[t] = t$.

Proof. Let C be the set of clusters containing the elements of $\pi \setminus \{p_t\}$. Let $\overline{C}$ be the set of clusters in $\overline{N}$ that are not contained in C. By Lemma 9, there exists an allocation α_1 of $\overline{C}$ pointing to C.

We first show a that, for each cluster c in C, there is one and only one transition in c which is in π. This will help us in building an allocation α_2 of the clusters in C:

For each cluster c in C, we have: $|T \cap c \cap \pi| = 1$ because

1. *There exists at least one transition $t \in c$ such that $t \in \pi$ because, by definition, c contains at least one element x of c if x is a transition, then $t = x$. If x is a place, by definition of a path $x^\bullet \cap \pi \neq \emptyset$ (remember that we do not consider $[p_t]$) and by definition of cluster $x^\bullet \subset [x]$.*
2. *There can be only one transition $t \in c$ such that $t \in \pi$: Suppose that two distinct transitions t_1 and t_2 both belong to c and π. By cluster definition, ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$. By free-choice property, they have ${}^\bullet t_1 = {}^\bullet t_2$. Both sets ${}^\bullet t_1$ and ${}^\bullet t_2$ are equal and contain more than one place because t_1 and t_2 are on a simple path and thus they have a distinct place preceding them on π. Thus t_1, t_2 and ${}^\bullet t_1$ form an extended free-choice pattern which we ruled out in our definition of free-choice Petri net.*

As we have $|T \cap c \cap \pi| = 1$ for each cluster c in C, we can define the allocation α_2 of the clusters in C, for any transition t in a cluster of C we have $\alpha_2[t] = ({}^\bullet t)^\bullet \cap \pi$ and for every

place $p \in C$ we have $\alpha_2([p]) = p^\bullet \cap \pi$. By definition of π and of pointing allocation, α_2 points to p_1.

Lemma 10 has the two following corollaries for workflow nets. These corollaries will allow us to simplify some proofs in the rest of this appendix.

Lemma 2 (Alternative error marking). *Let m be a reachable marking of N such that m marks two places on a simple path π to the sink of N.*

Then, an execution sequence σ, such that $m \xrightarrow{\sigma} m'$ and m' is an error marking, can be obtained in $O(|P|^2)$.

Proof.

Let $\overline{N}$ be the connected version of N and p_t be the sink of $\overline{N}$.

We first show that there exists a total allocation α of $\overline{N}$ pointing to p_t. This allocation will 'drive' the construction of σ.

Let C be the set of clusters containing the elements of $\pi \setminus \{p_t\}$. Let $\overline{C}$ be the set of clusters in $\overline{N}$ that are not contained in C. By Lemma 9, there exists an allocation α_1 of $\overline{C}$ pointing to C.

We first show a property which will help us in building an allocationα_2 of the clusters in C, for each cluster c in C, we have: $|T \cap c \cap \pi| = 1$ because

1. *There exists at least one transition $t \in c$ such that $t \in \pi$ because, by definition, c contains at least one element x of c if x is a transition, then $t = x$. If x is a place, by definition of a path $x^\bullet \cap \pi \neq \emptyset$ (remember that we do not consider $[p_t]$) and by definition of cluster $x^\bullet \subset [x]$.*
2. *There can be only one transition $t \in c$ such that $t \in \pi$: Suppose that two distinct transitions t_1 and t_2 both belong to c and π. By cluster definition, ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$. By free-choice property, they have ${}^\bullet t_1 = {}^\bullet t_2$. Both sets ${}^\bullet t_1$ and ${}^\bullet t_2$ are equal and contain more than one place because t_1 and t_2 are on a simple path and thus they have a distinct place preceding them on π. Thus t_1, t_2 and ${}^\bullet t_1$ form an extended free-choice pattern which we ruled out in our definition of free-choice Petri net.*

Thanks to this property, we can define the allocation α_2 of the clusters in C, for any transition $t \in C$ we have $\alpha_2[t] = ({}^\bullet t)^\bullet \cap \pi$ and for every place $p \in C$ we have $\alpha_2([p]) = p^\bullet \cap \pi$. By definition of π and of pointing allocation, α_2 points to p_1.

By Lemma 10 we can build, in quadratic time, an execution sequence σ (agreeing α) of N that $m \xrightarrow{\sigma} m'$ and m' is an error marking (in which case we are done) or m' marks p_t. In the following, we will consider the latter case.

It can be proven by an induction on the length of σ that, for any marking m^* reachable from m by any execution sequence σ^* agreeing with α, $m^*(\pi) \geq 2$. The main argument for the induction step is that, because π is a simple path and $\overline{N}$ is free-choice, executing any transition agreeing with α cannot consume both tokens.

We know that a place p^*, preceding p_t on π, is marked by m' because m' is not an error marking and $m'(\pi) \geq 2$. The sink p_t remains marked by any execution sequence agreeing with alpha because no transition of $[p_t]$ is allocated by α. By Lemma 10, we can continue to build σ in quadratic until reaching a marking m'' such that either is an error marking or $m''(p_t) > 2$ which is a lack of synchronization, i.e., an error marking.

The second corollary is well known in the literature: it is a direct violation of one of the clauses of the classical definition of soundness (See Section 2).

Lemma 12 (Improper termination). *Let m be a reachable marking of N such that m marks the sink of N and another place.*

Then, an execution sequence σ, such that $m \xrightarrow{\sigma} m'$ and m' is an error marking, can be obtained in quadratic time.

Proof. This is a special case of Lemma 2 where the 'second' token on π is already marking the sink.

Finally, we prove Lemma 1:

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' is an explicit error marking or m' marks p_2.*

Proof. Let $\overline{N}$ be the connected version of N. By Lemma 11, there exists a total allocation α of $\overline{N}$ allocating the transitions of π and pointing to p_2.

By Lemma 10, we can obtain in quadratic time an execution sequence σ agreeing with α such that $m \xrightarrow{\sigma} m'$ and m' marks p_2, m' is an error marking, or m' marks p_t. If m' marks p_2 or m' is an error marking, we are done. We consider the case where none of these two conditions is satisfied and thus m' marks p_t. As σ agrees with α and α allocates the transitions of π, p_2 would have needed to be marked during for m' to not mark a place of π, therefore m' must mark a place of π because. By Lemma 12, we can compute in quadratic time an execution sequence σ' such that $m' \xrightarrow{\sigma'} m''$ and m'' is an error marking.

B Proof of Theorem 4

We prove that, for the each of the three error patterns described in Theorem 3, we are able to compute an error trace in quadratic time.

B.1 Error trace corresponding to a siphon that does not contain the source place

A siphon that does not contain the source place is not marked by any execution. In fact, any execution where we try to mark a place in the siphon will end in an error marking:

Lemma 13 (A non-marked siphon can be used to build an error trace in quadratic time).

Let S be a siphon of N and m be a reachable of N.

An execution σ, such that $m \xrightarrow{\sigma} m'$ and m is an error marking, can be computed in quadratic time.

Proof. Let $p \in S$. By the workflow net definition there exists a simple path from the source place of N. By Lemma 1 we can compute in quadratic time an execution trace σ such that $m \xrightarrow{\sigma} m'$ and m' either marks p or is an error marking. By the unmarked siphon property, m' cannot mark p. Therefore m' is an error marking.

B.2 Error trace corresponding to DQ siphon with a P/T handle

A siphon with a P/T handle also allows us to compute an error trace in quadratic time:

Lemma 14 (A siphon with a P/T handle can be used to build an error trace in quadratic time).

Let S be a siphon of N with a P/T-handle H.

An execution σ, such that $m_0 \xrightarrow{\sigma} m$ and m is an error marking, can be computed in quadratic time.

Proof. *Let p_1 be the first place of the handle. By definition of workflow net, there is a simple path from the source place of N to p_1. By Lemma 1, we can obtain an execution sequence σ_1 such that $m_0 \xrightarrow{\sigma_1} m_1$ and either m_1 is an error marking in which case $\sigma = \sigma_1$ or m_1 marks p_1. By DQ siphon definition (cf. Definition 1), the number of tokens in S cannot increase. The number of tokens is originally 1. As the last transition of σ_1 removes one token from S, $m_1(S) = 0$. By Lemma 13, we can compute an error trace in quadratic time.*

B.3 Error trace corresponding to a simple path to the sink with a T/P handle

We now show how we can build an error trace when there exists a simple path from an element of N to the sink of N with a T/P handle:

Lemma 15 (A simple path to the sink with a T/P handle can be used to build an error trace in quadratic time).

Let π be a path in N to the sink of N and H be a T/P-handle of π.

An execution σ, such that $m_0 \xrightarrow{\sigma} m$ and m is an error marking, can be computed in quadratic time.

Proof. *Let α be an allocation pointing to a place p_1 preceding the first transition t of H. By Lemma 1, we can obtain an execution sequence σ_1 such that $m_0 \xrightarrow{\sigma_1} m_1$ and either m_1 is an error marking in which case $\sigma = \sigma_1$ or m_1 marks p_1. Let p be the last node of H. As H is a T/P-handle of π, t and p are on π, t is a transition, and p is a place.*

Let β be a total allocation pointing to p, which allocates the transitions in H and π. We show that such allocation exists by showing that for each cluster c containing an element of H or π we have $|H \cap c| = 1$ or $|\pi \cap c| = 1$, and there exist no pair of transitions $t_1 \in \pi$ and $t_2 \in H$ such that t_1 and t_2 belong to the same cluster.

1. *There exists at least one transition $t \in c$ such that $t \in \pi$ ($t \in H$) because by definition c contains at least an element x of c if x is a transition, then $t = x$. If x is a place, by definition of path (handle), $x^\bullet \cap \pi \neq \emptyset$ ($x^\bullet \cap H \neq \emptyset$) and by definition of cluster $x^\bullet \subset [x]$.*
2. *There can be only one transition $t \in c$ such that $t \in \pi \cup H$: Suppose that two distinct transitions t_1 and t_2 that both belong to c and $\pi \cup H$. By cluster definition ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$. By free-choice property they have ${}^\bullet t_1 = {}^\bullet t_2$. Both sets ${}^\bullet t_1$ and ${}^\bullet t_2$ are equal and contain more than one place because t_1 and t_2 are on (disjoints) simple paths and thus they have a distinct place preceding them on π and H. Thus t_1, t_2 and ${}^\bullet t_1$ form an extended free-choice pattern which we ruled out in our definition of free-choice Petri net.*

It is clear that during any execution sequence σ_2 following β such that $m_1 \xrightarrow{\sigma_2} m_2$ and no marking before m_2 marks p, there is always a token on H and a token on π. By Lemma 10, we can compute in quadratic time execution sequence σ_2 such that $m_1 \xrightarrow{\sigma_2} m_2$ and m_2 is either an error marking, m_2 marks the sink p_t, or $m_2(p) > 1$. In case m_2 is an error marking, we are done and $\sigma = \sigma_1\sigma_2$. We now consider the two other cases. As the marking m'_2 preceding m_2 in σ_2 marks a place on π and a place on H, m_2 still marks a place on H or a place on π because the last transition on π is distinct from the last transition on H. Therefore m_2 marks two places and there exists a simple path from one of the marked place to the sink which contains the other place. By Lemma 2, we can obtain, in quadratic time, an execution sequence σ_3 such that $m_2 \xrightarrow{\sigma_3} m_3$ and m_3 is error marking. We have $\sigma = \sigma_1\sigma_2\sigma_3$.

B.4 Error trace corresponding to an error pattern

Finally, the proof of Theorem 4 summarizes the previous results of this appendix:

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

Proof. We consider the three error patterns separately:

1. *If N contains a siphon that does not contain the source place, by Lemma 13, we can compute an error trace in quadratic time.*
2. *If N contains a DQ siphon with a P/T handle, by Lemma 14, we can compute an error trace in quadratic time.*
3. *If N contains a simple path to the sink with a T/P handle, by Lemma 15, we can compute an error trace in quadratic time.*

C Proof of Lemma 3

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.*

Proof. Let Ω be a circuit with a T/P handle H in $\overline{N}$ as illustrated by Figure 10 on page 24. Let t be the transition in $\Omega \cap H$ and p be the place in $\Omega \cup H$

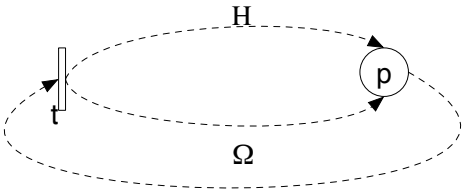

Fig. 10. A Circuit with a T/P handle.

Note the symmetry in the pattern: One can build a circuit Ω' by concatenating H and the portion of Ω from p to t. The portion of Ω from t to p is a T/P handle of Ω'.

We will show the existence of a simple path in N to the sink f of N with a T/P handle.

We distinguish two cases:

1. *$f \in \Omega \cup H$: We can assume without loss of generality that $f \in \Omega$ because of the symmetry in the pattern. In $\overline{N}$, we have, by definition of connected workflow net, that $(f^\bullet)^\bullet$ contains only the source i of N. The portion of Ω from i to f is a simple path to f and H is a T/P handle of that path (See Figure 11 on page 25). Note that, while the figure illustrates a case where i is on the portion of Ω between t and p, this same reasoning applies in the case where i is on the portion of Ω between p and t.*

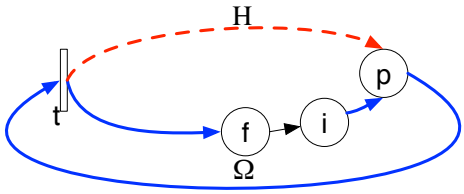

Fig. 11. Path with T/P handle case 1.

2. *$f \notin \Omega \cup H$: Consider a simple path π' in N from p to f. Such a path exists by definition of a workflow net.*

   Let x be the last element in $(\Omega \cup H) \cap \pi'$. We can assume without loss of generality that $x \in \Omega$ by the symmetry of the pattern.

   Let π^* be the path obtained by the concatenation of the portion of Ω from the element following x on Ω to x and the suffix of π' from x. π' is a simple path to f and H is a T/P handle of π^* (See Figure 12 on page 25). Again, the figure illustrates the most complex case where x is on the portion of Ω between t and p, the same reasoning applies in case x is on the portion of Ω between p and t.

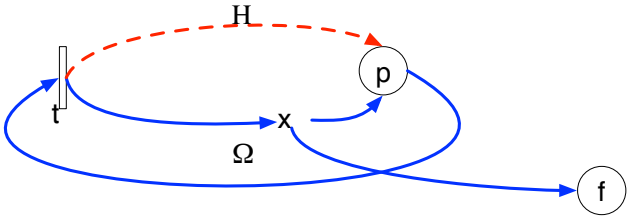

Fig. 12. Path with T/P handle case 2.

It is easy to see that, in both cases the path to the sink with the T/P handle can be computed in $O(|F|)$ time.

D Proof of Lemma 4

We start by enunciating a theorem from Esparza and Silva [4] which characterizes minimal siphons:

Theorem 8 (Minimal siphon property [4]). *Let $N = (P, T, F)$ be a free-choice Petri net, $S \subset P$ be a siphon of N and $N' = (P', T', F')$ the subnet generated by S.*

S is minimal if and only if S is strongly connected and for every transition $t \in T'$, we have $|pre_{N'}(t)| \leq 1$.

We can now prove Lemma 4:

Lemma 4. *If $\overline{N}$ has a circuit with a P/T handle without a T/P bridge, then N has a minimal siphon with a P/T handle or a siphon that is not initially marked.*

Proof. Let Ω be circuit of $\overline{N}$ and let H be a P/T handle of Ω without a T/P bridge. Let p be the place in $H \cap \Omega$ and t be the transition in $H \cap \Omega$.

Consider a minimal siphon S containing Ω. (We can use an algorithm presented by Esparza and Silva [4] to compute a minimal siphon containing a circuit. Therefore S exists.) There is a suffix π of H such that the first and last elements of π belong to S and all the other elements of π are disjoint from S: Note that t and p belong to the subnet generated by S. By Theorem 8, we have that $|{}^\bullet t \cap S| = 1$ and because $|pre_\Omega(t) \cap S| = 1$ and $pre_\Omega(t) \cap pre_H(t) = \emptyset$, we have $pre_H(t) \not\subset S$. Therefore, π exists and is non-trivial because it contains at least $pre_H(t)$ and t.

We now show that π is a P/T handle of S. By construction, π is disjoint from S aside from its first and last elements. The last element of π is a transition. We are left to show that the first element x of π is a place. We proceed by contradiction: Suppose that x is a transition. By definition of siphon, S is strongly connected, therefore there is a simple path π' from x to an element p' of Ω. Without loss of generality, we assume that p' is the only element of π' that belongs to Ω. As $|{}^\bullet p' \cup S| > 1$, p' must be place by definition of siphon. Thus π' is a T/P bridge from H to Ω which contradicts the initial assumption that H does not have a T/P bridge.

If S is not initially marked in $\overline{N}$, then S is also a non-initially marked siphon of N. If S is initially marked in $\overline{N}$, S contains the source place of $\overline{N}$ and therefore, by definition of siphon, must contain the sink of $\overline{N}$. By definition of handle, H is disjoint from S aside from t and p and therefore H does not contain the return transition. Thus, S is also a siphon in N with a P/T handle.

E Proof of Lemma 5

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 a circuit with a T/P handle in $\overline{N}$ can be computed in $O(|F|)$.*

Proof. Consider two places $p_1, p_2 \in t^\bullet \cap P'$. Because N' is strongly connected, there exists two circuits $\Omega =< t, p_1, \ldots, t >$ and $\Omega' =< t, p_2, \ldots, t >$ in N'. Consider the prefix H of Ω' such that H only intersects with Ω with its first node t and its last node n. By Theorem 8, no transition in N' has more than one incoming arc in N'. Thus n is a place because it has at least two incoming arcs in N'. We have shown that H is a T/P-handle of Ω. The two circuits and their intersections can be computed in $O(|F|)$.

F Proof of Lemma 6

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

Proof. We show that N' has a P/T-handle:

Because $\overline{N}$ is strongly connected there exists a circuit $\Omega' =< p, t, \ldots, p >$ in $\overline{N}$. Let H be the prefix of Ω' from p to the first node n on Ω' after p such that $n \in P' \cup T'$. (i.e., H is a path in N disjoint from N' aside from p and n). Thus H is an handle of N'. The element n is a transition because, if n was a place, all the transitions in ${}^\bullet n$ would belong to N' by definition of subnet and therefore n would not be the first node of H after p in N'. Thus H is P/T-handle of N'. The handle can be computed in $O(|F|)$.

G Proof of Lemma 7

In this section, given that x is an element of a net N and N' is a subnet of N, we use $\text{pre}_{N'}(x)$ to denote ${}^\bullet x \cap N'$ and $\text{post}_{N'}(x)$ to denote $x^\bullet \cap N'$.

We start by showing that a siphon of the reduced net is also a siphon of the original net.

Lemma 16. *Let $\overline{N}$ be an unsound connected free-choice workflow net, p be a place in $\overline{N}$ and N' be a non-empty strongly connected free-choice Petri net obtained by using delete$(p, \overline{N})$. If S is a siphon of N', we have:*

1. *$pre_{\overline{N}}(S) = pre_{N'}(S)$.*
2. *and $post_{N'}(S) \subseteq post_{\overline{N}}(S)$.*
3. *S is also a siphon of $\overline{N}$.*

Proof. By construction, S is also a subset of the places of $\overline{N}$.

1. *$pre_{\overline{N}}(S) = pre_{N'}(S)$ because the delete function never removes a transition without removing its target place(s). Therefore any place remaining in N' has the same set of incoming transitions as in $\overline{N}$.*
2. *$post_{N'}(S) \subseteq post_{\overline{N}}(S)$ because the delete function only removes elements of the workflow net.*
3. *By definition of siphon, $pre_{N'}(S) \subseteq post_{N'}(S)$. By 1 and 2, $pre_{\overline{N}}(S) \subseteq post_{\overline{N}'}(S) \subseteq post_{\overline{N}}(S)$ and therefore S is a siphon of $\overline{N}$.*

We can now prove that a siphon that does not generate a P-component in the reduced net, i.e., a siphon that points to a violation of the structural characterization, does not generate a P-component in the original net:

Lemma 7. *Let N' be a non-empty subnet of $\overline{N}$ obtained from delete$(p, \overline{N})$. (1) If S is a minimal siphon of N' such that there is a transition t such that $|post_{N'}(t) \cap S| > 1$ then S is also a minmal siphon of $\overline{N}$ such that there is a transition t such that $|post_{\overline{N}}(t) \cap S| > 1$. (2) If S is a minimal siphon of N' such $post_{N'}(S) \setminus pre_{N'}(S) \neq \emptyset$, then S is also a minmal siphon of $\overline{N}$ such that $post_{\overline{N}}(S) \setminus pre_{\overline{N}}(S) \neq \emptyset$.*

Proof.

1. *There exists a transition $t \in pre_{N'}(S)$, such that $|post_{N'}(t) \cap S| \geq 2$. By Lemma 16.1, $pre_{N'}(S) = pre_{\overline{N}}(S)$ and therefore $t \in pre_{\overline{N}}(S)$. The transition $t \in pre_{\overline{N}}(S)$ is such that $|post_{\overline{N}}(t) \cap S| \geq 2$.*
2. *There exists a transition $t \in post_{N'}(S)$, such that $t \notin pre_{N'}(S)$. By Lemma 16.2, $post_{N'}(S) \subseteq post_{\overline{N}}(S)$ and therefore $t \in post_{\overline{N}}(S)$. By Lemma 16.1, $pre_{N'}(S) = pre_{\overline{N}}(S)$ and therefore $t \notin pre_{\overline{N}}(S)$.*

H Proof of Lemma 8

We now show that whenever there is a SMD connected workflow net that is unsound, it can always be reduced by Alg. 1 . We proceed using two lemmas showing that the delete function can remove at least one place of a strongly connected free-choice Petri net N that is not structurally sound, i.e., if there is a circuit with a T/P handle in N or there is a circuit with a P/T handle without a T/P bridge in N, respectively:

In this section, given that x is an element of a net N and N' is a subnet of N, we use $\text{pre}_{N'}(x)$ to denote ${}^\bullet x \cap N'$ and $\text{post}_{N'}(x)$ to denote $x^\bullet \cap N'$.

Lemma 17. *Let $\mathcal{S}$ be a set of P-components covering N.*

If there is a circuit with a T/P handle in N, then there exists a place p such that $\text{delete}(p, N)$ *contains a circuit with a T/P handle.*

Proof. Let Ω be a circuit with a T/P handle H in N (cf. Figure 13 on page 29). Let t be the first transition of H and p the last place of H. Let $p_1 \in t^\bullet \cap H$ and $p_2 \in t^\bullet \cap \Omega$. As t is a transition and $p_1, p_2 \in t^\bullet$, by definition of P-component p_1 and p_2 must belong to a different P-component of $\mathcal{S}$. Let $S_1 \in \mathcal{S}$ be a P-component which covers p_1. Let t' be the first element following Ω backward from t such that $pre_\Omega(t') \cap pre_{S_1}(t') = \emptyset$. The element t' exists because $p_2 \notin S_1$.

Let $S_2 \in \mathcal{S}$ be a P-component covering the element in $pre_\Omega(t')$. As $pre_{S_1}(t') \neq pre_{S_2}(t')$ and $t' \in S_1 \cap S_2$, by definition of P-component, the element t' must be a transition. As P-components are strongly connected, the set ${}^\bullet t'$ contains a place $p' \in S_1$ and a place $p'' \in S_2$ such that $p' \neq p''$.

We now show that the place p' can be deleted by showing that $p' \notin H \cup \Omega$ which implies that removing p' would preserve Ω and H. We proceed by contradiction: Suppose that $p' \in H \cup \Omega$. By construction, the arc from p' to t' cannot be in Ω. Moreover, the arc from p' to t' cannot be in H because the only transition of H which is on Ω is t and it is clear that $t \neq t'$. Therefore, there must be another arc succeeding p' on H resp. Ω. Hence, there should be a transition $t^* \in post_{H \cup \Omega}(p')$ such that $t^* \neq t'$ and as $|{}^\bullet t'| > 1$ this would contradict the assumption that N is free-choice.

Lemma 18. *Let $\mathcal{S}$ be a set of P-components covering N.*

If there is a circuit with a P/T handle without a T/P bridge in N, then there exists a place p such that $\text{delete}(p, N)$ *is nonempty and not SLB.*

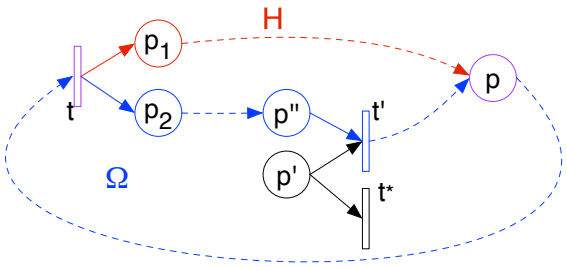

Fig. 13. Illustration for the proof of Lemma 17.

Proof. Assume that there exists a circuit Ω with a P/T handle H without a T/P bridge in N. Let p be a place and t be a transition in $\Omega \cap H$.

We consider the P-component $S \in \mathcal{S}$ that covers the place $p_1 \in pre_\Omega(t)$. Note that the definition of P-component implies that S must cover t and the place $p_2 \in pre_H(t)$ is not covered by S. We first show that either S covers Ω or a place can be deleted and the resulting net contains $\Omega \cup H$. We prove this indirectly:

- *Suppose that there is an element $x \in \Omega$ that is not covered by S. Let t' be the first element after x on Ω that is covered by S. We know that such an element exists because p_1 is covered by S. Because $pre_\Omega(t') \not\subset S$ and $t' \in S$, and by definition of a P-component, t' is a transition. Let p' be the place such that $p' \in pre_\Omega(t')$. Because S is strongly connected, there exists a place $p^* \in pre_S(t')$. By construction $p^* \neq p'$. We show that $p^* \notin H \cup \Omega$ and therefore removing p^* would result in a net which contains $\Omega \cup H$. We show this indirectly:*
  - *Suppose that $p^* \in \Omega \cup H$. We distinguish two cases:*
    1. *If $p^* \in \Omega$, the arc between p^* and t' cannot belong to Ω by definition of a circuit. Because Ω is a circuit, there is a transition t^* in $post_\Omega(p^*)$. Because the arc between p^* and t' does not belong to Ω, $t^* \neq t'$.*
    2. *If $p^* \in H$, the arc between p^* and t' cannot belong to H because H does not intersect with Ω aside from p and t and we have $t' \neq t$ by construction. Because H ends with a place, there must be a transition t^* in $post_H(p^*)$. Because the arc between p^* and t' does not belong to Ω, $t^* \neq t'$.*

    In both cases, the subnet containing p', t', p^*, t^*, and their connecting arcs contradict the assumption that N is free-choice.

  Therefore p^* does not belong to $H \cup \Omega$ and delete(p^*, N) *contains $H \cup \Omega$.*

We are left with the case where Ω is covered by S. Let t' be the last element of $H \setminus \{t\}$ covered by S. The element t' exists because $p \in S$ and $p_2 \notin S$. The same argument as above can be used to show that either all elements of H preceding t' belong to S or there exists a place that can be deleted without removing an element of $H \cup \Omega$. Therefore, we are left with the case where Ω and all the elements of H until t' are covered by S. Let π be a simple path in S from t' to p. Such a path exists because S is strongly connected. Let p^* be the first element of π such that $p^* \in H \cup \Omega$. We distinguish two cases:

1. *$p^* \in H$. By construction, p^* must precede t' on H because there is no element in $H \setminus \{p\}$ that is covered by S, by definition of t'. By definition of a P-component, the element p^* is a place because $|pre_S(p^*)| > 1$. Figure 14 illustrates the main elements of this portion of the proof. Let Ω^* be the circuit obtained by the concatenation of H and the portion of Ω from p to t. Let H^* be the prefix of π until p^*. H^* is a T/P handle of Ω^*. By Lemma 17, it contains a circuit with a T/P handle.*

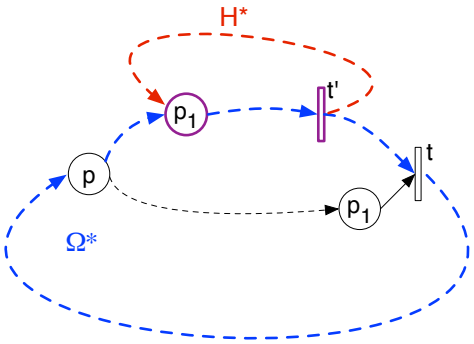

Fig. 14. The path H^* is a T/P handle of Ω^*.

2. *$p^* \in \Omega$. We rule out this case by contradiction: By definition of a P-component, the element p^* is a place because $|pre_S(p^*)| > 1$. The prefix of π until p^* is a T/P bridge from H to Ω which is ruled out by assumption.*

We can now prove that the delete function can reduce any unsound SMD strongly connected free-choice Petri net:

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* delete(p, N') *is nonempty and not SLB.*

Proof. By Theorem 2, N' contains a circuit with a T/P handle or a circuit with a P/T handle without a T/P bridge.

If N' contains a circuit with a T/P handle, then there exists a place p such that delete(p, N') *contains a circuit with a T/P handle by Lemma 17.*

If N' contains a circuit with a circuit with a P/T handle without a T/P bridge, then there exists a place p such that delete(p, N') *is not SLB by Lemma 18.*

I Additional Examples of Diagnostic Information

Our technique uses three structural error patterns. We gave an example for T/P handles in Fig. 6. In the following, we show examples for the other two patterns. Both can be displayed and understood locally, in the context of the process model and do not contain any irrelevant information.

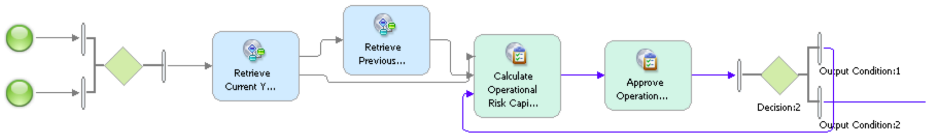

Fig. 15. Screenshot of the diagnostic information for a siphon that is not initially marked.

Fig. 15 shows the error report for a siphon that is not initially marked. The siphon (in blue) does not contain a start event. Consequently, the siphon is never marked and the portion of the model colored in blue cannot be executed.

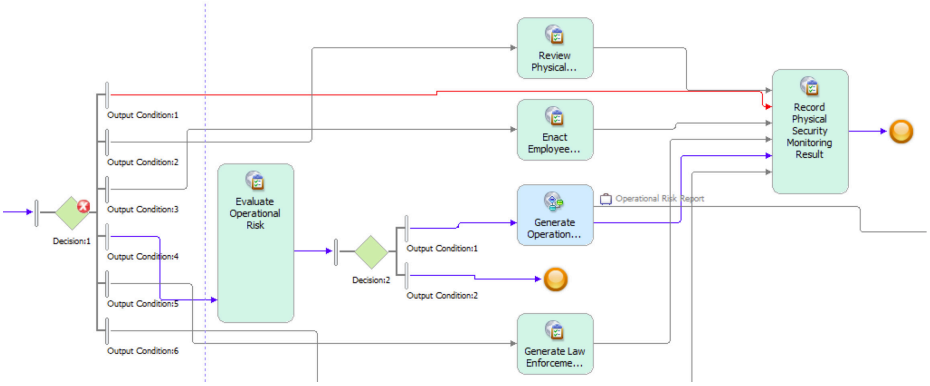

Fig. 16. Screenshot of the diagnostic information for a DQ-siphon with a P/T handle.

Fig. 16 shows the error report for a DQ-siphon with a P/T handle. Once the first outcome of 'Decision:1' is chosen, i.e., the first transition of the P/T handle (in red) is executed, the DQ-siphon (in blue) loses its only token. The token on the handle is stuck because the activity 'Record Physical Security Monitoring Result', which is part of the unmarked DQ-siphon, cannot be executed.