Information leak detection in business process models: Theory, application, and tool support[☆]

Rafael Accorsi

University of Freiburg, Department of Telematics, 79098 Freiburg, Germany

Andreas Lehmann

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany

Niels Lohmann

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany

Abstract

Despite the correct deployment of access control mechanisms, information leaks can persist and threaten the reliability of business process execution. This paper presents an automated and effective approach for the verification of information flow control for business process models. Building on the concept of place-based non-interference and declassification, the core contribution of this paper is the application of Petri net reachability to detect places in which information leaks occur. Such a feature allows for the use of state-of-the-art tool support to model-check business process models and detect leaks. We show that the approach is sound and complete, and present the Anica tool to identify leaks. An extensive evaluation comprising over 550 industrial process models is carried out and shows that information flow analysis of process models can be done in milliseconds. This motivates a tight integration of business process modeling and non-interference checking.

1. Introduction

Business processes (BPs) specify how activities are executed to provide a service; for instance steps in a supply-chain or profile updates in customer relationship management. In doing so, they handle sensitive data that must remain confidential. However, design flaws in BPs design may lead to a violation of such confidentiality requirements [2]; that is, they can cause leaks.

[☆]Extended and thoroughly revised version of [1].

Email addresses: `rafael.accorsi@iig.uni-freiburg.de` (Rafael Accorsi), `andreas.lehmann@uni-rostock.de` (Andreas Lehmann), `niels.lohmann@uni-rostock.de` (Niels Lohmann)

Leaks happen when data or information flows from a secret domain to a public domain. The former is called *data leak*; the latter is referred to as *information leak*. A data leak is a direct but illegal access to a data object. An information leak concerns the fact that unauthorized subjects can infer secret information. Although certification standards (e.g., ISO/IEC 27001 [3] and TCSEC [4]) demand the identification of both kinds of leaks, current state of the art mainly provides mechanisms to detect data leaks (e.g., [5–10]).

In this paper, we focus on the use of formal methods to identify information leaks in BP models. BPs are the main asset of companies as they describe their value chain and fundamentally define "the way businesses are done". In this scenario, information leaks can have catastrophic consequences for the business owner, be it for legal or financial reasons. *Information flow control* provides a powerful abstraction to reason about information leaks [11]. Assuming that the BP model under analysis is split into two logical security domains (*high* for secret, *low* for public), a BP model is assumed "secure" with regard to information leaks if it enforces *non-interference*; that is, the actions in the high domain do not produce an observable effect (*interference*) in the low domain. Details on the security domains follow in Sect. 2. By showing non-interference for a BP model one thus ensures that subjects in the low domain cannot deduce information about the high domain, thereby guaranteeing its confidentiality and isolation.

We consider the Petri net representation of BP models as a basis for the analysis. For this, mappings from common modeling languages, such as WS-BPEL, BPMN and EPC, exist [12]. Subsequently, the activities — denoted as Petri net transitions — are separated into high and low domains. The analysis of these models is carried out with *place-based non-interference* (PBNI) [13]. PBNI is an approach to encode and reason about *structural non-interference* (and hence information flow control) in Petri nets. The rationale is that specific types of places in the net encode different non-interference properties; that is, they denote information leaks. In showing the absence of such places in a Petri net, one rules out structural interferences.

The current approach [14] and tool support [15, 16] for information flow analysis of BP models exhibits the following drawbacks: *Firstly*, the decision procedures to detect these places are based on the generation and analysis of the *complete* state space of the model, which due the state space explosion renders a very inefficient approach for complex BP models. *Secondly*, formal proofs of the decision procedure are missing — in particular soundness and completeness properties. The lack of these guarantees weakens the security guarantees provided by tool-support, because results may contain false-negatives, for instance that a BP model is considered secure even though information leaks exist.

Contributions. This paper reports on the following contributions:

- We introduce an approach for the information flow analysis of BP models based on the reachability problem for Petri nets [17]. By reducing the analysis to reachability, we obtain a decision procedure based on standard Petri net reasoning, hence inheriting a broad spectrum of verification tools and techniques.

- We prove that the approach based on reachability is sound and complete.
- We present the design and implementation of *Anica*, a novel tool for information flow analysis of BP models. Anica employs Petri net analysis and state space reduction methods to generate and analyze only relevant parts of the state space. Its realization employs the model-checking tool LoLA [18].
- We evaluate Anica on a BP repository comprising over 550 industrial models. The information flow analysis of each BP takes only fractions of a second. Anica analyzes the entire repository, whereas other tools relying on complete state space exploration may fail for complex BPs.

This paper substantiates that the detection of information leaks in BP models can be carried out in an effective, push-button manner. Process modelers can employ these techniques to enforce non-leak security guarantees "by design" before process deployment. We show that the well-founded information flow analysis of industrial BPs can in fact be done on-the-fly, as we demonstrate in [19]. This opens the possibility of security analysis *during* BP editing, or at run-time upon the event of ad-hoc process changes. In doing so, we contribute to the reliably secure modeling of flexible BP models.

Practical relevance. In the security jargon, the aforementioned information leaks are a consequence of *covert channels* [20] induced by the faulty structure of BP models. The detection of covert channels (and, hence, potential leaks) is required in various certification standards. For example, the TCSEC's "Part B: Mandatory Protection" demands the "analysis of covert channels and their bandwidth" [4]. Similarly, ISO/IEC 27001 Section 12.5 "Security in development and support processes" prescribes that "checks should be made for information leakage for example via covert channels" [3]. Concrete audit and certification procedures for SAS 70 and SAS 117 equally demand security guarantees [21, Chap. 24]. Several papers expose the risks of covert channels in fields related to BP management and deployment, e.g. Cloud Computing environments [22], virtualization [23] and Web Services [24]. The threat of leaks is considered imminent [25] and real [26, Chap. 8], thereby motivating our work and attesting its practical relevance.

In contrast to information leak detection, the *enhancement* of processes, (e.g. with process rewriting [27]) to deter information leaks is usually not required [3]: the correction of one detected covert channel can in turn create new covert channels. The introduction of explicit declassification transitions appears to be an effective way of repairing business processes [19].

Approach overview. The proposed approach and tool support Anica take (the Petri net representation of a) BP model produces by a process modeler as input and produces a certificate containing the set of places where information leaks may occur. This process comprises two main steps: *Firstly*, the activities in the model are labeled with the corresponding security domain (namely: "high" for secret and "low" for public). This step can be carried out manually, or

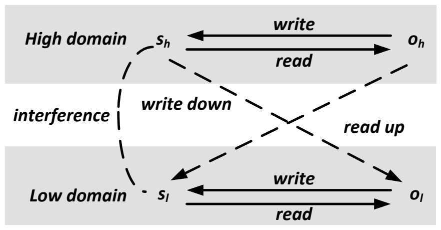


Figure 1: Multi-level security model with levels *high* and *low*.

different strategies may be employed to automatically suggest labels to activities, depending on the goal and scope of analysis. The running example used to illustrate the approach employs, e.g., a strategy for the analysis of concurrent process executions (see [28] for strategies and their definitions). *Secondly*, Anica takes the labeled Petri net and automatically creates a series of reachability problems, whereas the underlying model-checking tool LoLA solves these problems and returns the corresponding witnesses. Figure 8 illustrates the approach.

Organization. For completeness, Sect. 2 revisits the concept of PBNI and a small example. Section 3 presents the approach to detect non-interference based on reachability, and shows its main properties. Section 5 introduces Anica and Sect. 6 presents its evaluation. Section 7 compares our contribution with related work and Sect. 8 summarizes the lessons learnt and indicates further work.

2. Information Flow Control in Petri Nets

This section provides the formal basis necessary to reduce the detection of interferences in BPs to a reachability problem. We firstly introduce the multi-level security model [29] used to reason about interferences. Secondly, we define the necessary Petri net background, then revisit *place-based non-interference* (PBNI) [13] to capture leaks as interferences.

Multi-level Security Model. We employ a multi-level security (MLS) model to capture and detect leaks as interferences in BP. In the MLS model, objects and subjects of the system are divided in *security levels* (e.g. Confidential, Secret, TopSecret) with their usual semantics. That is, an object labeled with "Secret" is less sensitive than a "TopSecret" object. A subject with *clearance* to access secret documents is not authorized to access top secret documents. Policies in the MLS model fall under what is know as *lattice-based access control systems*.

As a particular instance of the MLS, information flow control employs a simple lattice model consisting of two security levels, namely *high* and *low*, as depicted in Fig. 1. Here, a *direct information leak* happens if a high subject s_h writes a classified information into a lower-level object o_l (so-called "write down") or if a low subject s_l reads on a classified object o_h ("read up"). Furthermore, an interference characterizes an *indirect information leak* in which a subject s_l can observe the behavior of s_h and infer information. The absence of direct and indirect information leaks indicates that the security levels are *isolated* from each other. Hence, the MLS model provides "end-to-end" guarantees, being thereby stronger than "point-to-point" guarantees provides by encryption or discretionary access control.

A common criticism on the binary model is that, in practical systems, a more complex levels appear. Benantar [30, p. 136] and Anderson [31, Chap. 8] show that *linear* lattices with more than two levels can be reduced without loss of generality to a lattice consisting of high and low levels. We exemplify this below. Overall, the binary MLS is a suitable abstraction to reason about leaks in information systems [32, Chap. 9].

Petri Nets. We assume *sound* [33] and *safe* Petri nets; that is, all places contain at most one token in all reachable markings. Soundness is a fundamental sanity check for BP models. It requires a BP model to reach a dedicated final state from all reachable states (thereby ruling out deadlocks and livelocks) and demands that every modeled activity can be executed (ruling out "dead code"). Consequently, a security check for an unsound BP is not useful, because such a BP has more fundamental problems than security issues. Safeness itself is no restriction, because sound nets are bounded and can be expressed as safe Petri nets as well.

Definition 1 (Petri net). A *Petri net* $N = [P, T, F, m_0]$ consists of two finite and disjoint sets P of *places* and T of *transitions*, a *flow relation* $F \subseteq (P \times T) \cup (T \times P)$, and an *initial marking* m_0. A *marking* $m \subseteq P$ represents a state of the Petri net and is visualized as a distribution of tokens on the places. Let $x \in (P \cup T)$. The *preset* of x is the set ${}^\bullet x = \{y \mid [y, x] \in F\}$, the *postset* of x is $x^\bullet = \{y \mid [x, y] \in F\}$. Transition t is *enabled* in marking m iff, ${}^\bullet t \subseteq m$. An enabled transition t can *fire*, transforming m into the new marking m' with $m' = (m \setminus {}^\bullet t) \cup t^\bullet$. The firing of one transition is denoted as $m \xrightarrow{t} m'$ and a sequence $\sigma \in T^*$ of transitions transforming m into m' is denoted as $m \xrightarrow{\sigma} m'$. A marking m' is *reachable* from m, if $m \xrightarrow{*} m'$ (where $*$ stands for an arbitrary sequence).

Running example. We employ a simple example to illustrate the main insights on the information flow analysis. Figure 2 denotes a simple linear hierarchy of security levels for a hospital department. It consists of three levels (i.e. TopSecret, Secret and Confidential) which are assigned to the roles Chief Physician, Physician, and Nurse, respectively. Assuming that one wants to analyze the flow of information between the roles Physician and Nurse, the corresponding two-level lattice considers subjects in the role Nurse as *low* whereas the roles Chief Physician and Physician are considered *high*.

The Petri net in Fig. 3, which serves as input to the information flow analysis approach, represents two domains (for instance Physician and Nurse) or swim lanes of a process. This net is obtained with the labeling strategy to discover potential interferences between instances of the process running in parallel [28]. Roughly speaking, in this strategy the process is "cloned", whereby the activities of one instance are labeled with high and those activities in the other with low.

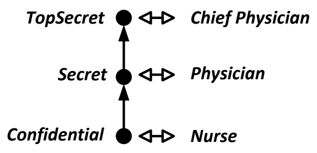


Figure 2: Security levels and role assignment.

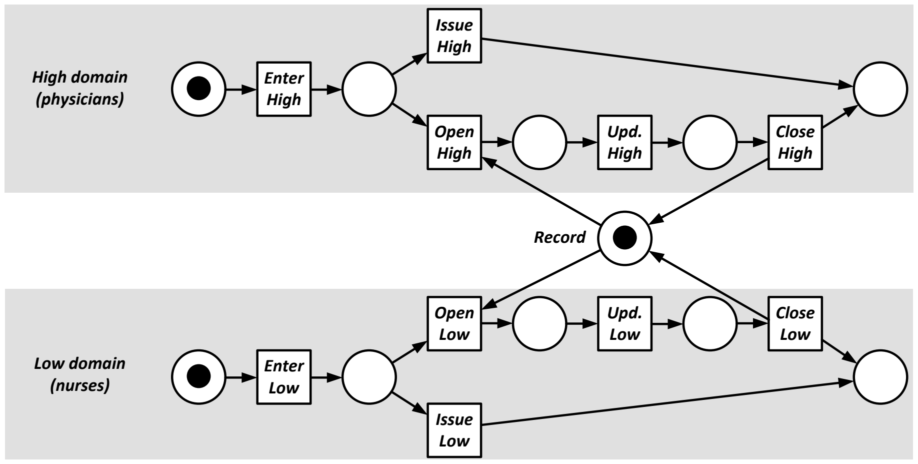
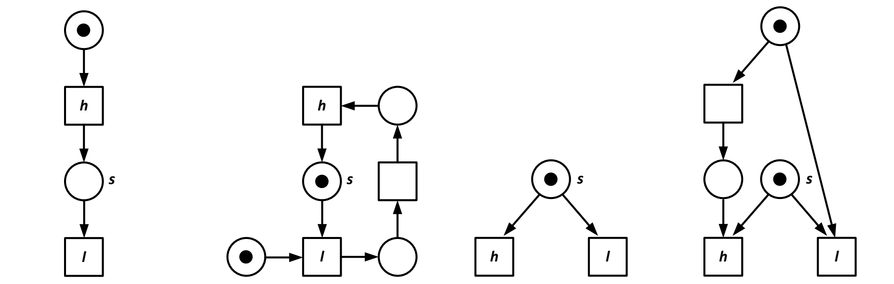


Figure 3: Petri net model of the Update Patient Record process.

The domain classified high is depicted in the upper part; the domain classified low is depicted in the lower part. Each domain realizes the update of a patient's record in a hospital. Given an existing patient ID, the corresponding record is opened, updated, and closed. The record itself is represented with a token on the place "Record", which is shared between the two domains. The central security requirement for the process is that information about the patients and the patient record remain secret. This is usually achieved by using discretionary access control techniques. At the end of this section, we shall show that information leaks can occur.

PBNI. The *place-based non-interference* (PBNI) [13] is an approach to reason about structural non-interference in Petri nets. Specifically, PBNI tackles the characterization and detection of places that correspond to a violation of non-interference. Previous work focuses on the characterization of *bisimulation non-deducibility on composition* (BNDC) [34]. BNDC is a strong structural non-interference property which guarantees that the high and the low parts of the net do not interfere.

To this end we employ *labeled Petri nets* to encode the two security levels in the Petri nets modeling BPs. There, the set of transitions T is partioned into two disjoint subsets L and H, capturing, respectively, the low and high levels.



(a) Active causal. (b) Potential causal. (c) Active conflict. (d) Potential conflict.

Figure 4: Patterns for causal and conflict places s.

Definition 2 (Labeled Petri Net). A *labeled Petri net* $N = [P, L, H, F, m_0]$ with $L \cap H = \emptyset$ is a Petri net $[P, L \cup H, F, m_0]$.

Given a labeled Petri net, the main insight in PBNI is that some places in a net characterize interferences between a high and a low domain, thereby denoting information flows and violations of confidentiality requirements. Specifically, these places are the *causal places* and *conflict places* (labeled s in Fig. 4).

Definition 3 (Causal and conflict places [13]). Let $N = [P, L, H, F, m_0]$ be a labeled Petri net. Let $s \in P$ be a place of N such that $s^\bullet \cap L \neq \emptyset$.

- s is a *potential causal place* if ${}^\bullet s \cap H \neq \emptyset$. A potential causal place s is an *active causal place* if the following condition holds: there exists (1) a transition $l \in s^\bullet \cap L$, (2) a transition $h \in {}^\bullet s \cap H$, and (3) a transition sequence $\sigma \in (H \cup L)^*$ and a reachable marking m such that $m \xrightarrow{h\sigma l} m'$ and $s \notin t^\bullet$ for all $t \in \sigma$.

- s is a *potential conflict place* if $s^\bullet \cap H \neq \emptyset$. A potential conflict place is an *active conflict place* if the following condition holds: there exists (1) a transition $l \in s^\bullet \cap L$, (2) a transition $h \in s^\bullet \cap H$, and (3) a transition sequence $\sigma \in (H \cup L)^*$ and a reachable marking m such that $m \xrightarrow{h} m'$, $m \xrightarrow{\sigma l} m''$ and $s \notin t^\bullet$ for all $t \in \sigma$.

For the sake of simplicity, we use the term *potential (active) place* for both potential (active) causal and potential (active) conflict places.

Running example (cont.). The labeled Petri net for the Petri net in Fig. 3 follows through the two domains. All transitions in the upper part (the high domain) belong to the set H and all other transitions, those in the lower part (for the low domain), are in the set L. Due the same transition names in both domains, we distinguish identically named transitions with there domain in front of the transition name. The place *Record* is a potential causal one (*Low.Open* followed by *High.Close*) and also a potential conflict one (conflict between *High.Open* and *Low.Open*).

Causal and conflict places capture the following interferences: in case of an *active causal* place s (cf. Fig.4(a)) transition l *always* depends on transition h. By observing the firing of l, the fact that transition h has fired leaks to a low user. In case of an *active conflict* place s (cf. Fig.4(c)), a high and a low transition are in conflict for a common token on place s; if transition l is not activated, a low user may deduce that transition h has fired (i.e., the firing of h leaks). Both, active causal and conflict places capture the BNDC property. Its equivalent formalization for Petri nets is called *positive place-based non-interference* (PBNI+) [13]. If active causal and conflict places do *not* occur in a net, PBNI+ holds. In other words, active causal and conflict places encode all possible leaks. Nevertheless, PBNI+ is not a structural property; that is, it cannot be decided solely on the net structure. Of course, potential places are identified by the net *structure*, but the decision whether a potential place is an active place also depends on the *behavior* of the net (viz. the presence of a corresponding transition sequence σ, cf. Def. 3). Figure 4(b) depicts a net with a potential causal place s, which is not active, because after h has fired, l cannot fire any more. In Fig. 4(d), an example in which the place s is a potential but not an active conflict place, is given: the structural conflict between h and l is not decided by s, but by the other initially marked place.

Frau et al. [16] propose a two-step algorithm for the verification of PBNI+. The *static* step analyzes the structure of the net to determine whether *potential* causal or conflict places exist; if so, the *dynamic* step generates and investigates the state space of the net to determine whether they are *active*; that is, on an execution path in the net. To this end, the algorithm constructs the *complete* state space, which makes tool support based upon these algorithms inefficient. In fact, Petri nets representing complex BPs cannot be analyzed using state space exploration (cf. Sect. 6 for details.)

Running example (cont.). Being used in a hospital, the BP model in Fig. 3 can be executed by different users, including nurses (typically low) and physicians (high). The Petri net is designed to analyze the concurrent interaction of subjects in these two domains with the BP to identify possible interferences. The net in Fig. 3 violates PBNI+ and, thereby, the confidentiality requirements. The analysis identifies place *Record* as potential causal and potential conflict place. The dynamic analysis further indicates that place *Record* is active in both cases.

Technically, place *Record* induces a *storage covert channel* [35] by which users in the low domain may deduce information about the high domain. This happens, because users share the same storage resource. Specifically, the hospital information system encompassing this BP allows, for instance, the deduction which patients were hospitalized (e.g., the low part observes which records are being processed) and the kind of treatment for a patient (the low part observes the timespan necessary to update a record, associating the elapsed time to a particular treatment).

3. Verification of PBNI+ as Reachability Problems

In this section, we show how PBNI+ verification can be realized by reducing the question to the reachability problem of Petri nets [17]. To this end, we introduce *objectives* which we use to encode potential interferences. To decide whether such a potential interference, encoded as objective, is an active one, we create an *extended net* based on the objective. Subsequently, we perform a reachability check on each extended net. In case a dedicated place in the extended net can be marked, the potential interference is an active one. The provided proofs guarantee that our translation into reachability is correct (i.e., the approach is sound) and that we capture all interferences (i.e., the approach is complete).

3.1. Creating Reachability Problems

The central concepts in our new definitions are the objective and the set of *undesired transitions*. An objective is a triple $[s, h, l]$ consisting of a place s, a high labeled transition h and a low labeled transition l. With these three nodes we can describe each potential and active place (cf. Def. 3). The difference between a potential and an active place, expressed as objective, lies in the dynamic behavior of their labeled Petri net. This difference is based on a transition sequence σ in which some transitions are prohibited (cf. Def. 3) for each particular objective $[s, h, l]$, these transitions are therefore *undesired*.

Definition 4 (Objective, undesired transitions). An *objective* $[s, h, l]$ is a triple for both a potential causal or a potential conflict place with a place $s \in P$, a transition $l \in s^\bullet \cap L$ and a transition $h \in {}^\bullet s \cap H$ (in the causal case) or a transition $h \in s^\bullet \cap H$ (in the conflict case). We call $[s, h, l]$ an *active objective* (or *potential objective*) if place s is an active (or potential) causal or conflict place for the transitions h and l. Let $U_{[s,h,l]} = \{t \in T \mid s \in t^\bullet\}$ be the set of *undesired transitions* in the transition sequence σ (cf. Def. 3), necessary to decide whether place s is active.

The set of undesired transitions contains all transitions which would be necessary for an alternative explanation of the observed behavior. One possible explanation could be another transition t' producing a token on place s, after h has produced a token on place s. In that case an observer cannot deduce for sure, that transition h was the reason for the activation of transition l. A transition t', which produces a token on place s, could either be labeled low or high. If t' is labeled low, its firing is observable and if t' is labeled high, it will be checked in the additional objective $[s, t', l]$. Consequently, those transitions are undesired from the perspective of objective $[s, h, l]$ for active place. In case the restricted transition sequence σ is sufficient for the flow from high to low, this flow can be observed by a low user, which, by assumption, knows the complete BP.

Running example (cont.). We already identified the place *Record* as potential causal and as potential conflict place. According to the involved transitions the objective for the potential causal case is $[\mathit{Record}, \mathit{High.Close}, \mathit{Low.Open}]$. The set of undesired transition for this objective is $\{\mathit{High.Close}, \mathit{Low.Close}\}$, because those transitions produce a token on the place *Record*. For the potential conflict case it is $[\mathit{Record}, \mathit{High.Open}, \mathit{Low.Open}]$ with the same set of undesired transitions.

For each potential place s, there exists at least one objective $[s, h, l]$, which we use to construct a Petri net $N_{[s,h,l]}$. The next definition shows how we construct this extended net based on the given net N and the objective $[s, h, l]$.

Definition 5 (Extended net). Let $N = [P, L, H, F, m_0]$ be a labeled Petri net and $[s, h, l]$ be an objective of N. The *extended net* $N_{[s,h,l]} = [P', T, F', m_0']$ is a Petri net, which is constructed based on N as follows:

(1) $P_T := \{p_t \mid t \in U_{[s,h,l]}\}$,

(2) $P'' := P \cup \{\mathit{fired}, \mathit{goal}\} \cup P_T$,

(3) $T := L \cup H \cup \{h_C, l_C\}$,

(4) $F'' := F \cup \{[p, h_C] \mid p \in {}^\bullet h\} \cup \{[p, l_C] \mid p \in {}^\bullet l\} \cup \{[h_C, \mathit{fired}], [\mathit{fired}, l_C], [l_C, \mathit{goal}]\} \cup \{[p_t, t], [t, p_t] \mid t \in U_{[s,h,l]}, p_t \in P_T\} \cup \{[p_t, h_C] \mid p_t \in P_T\}$,

(5) $m_0'' := m_0 \cup P_T$.

For the causal case, the following additional changes are necessary:

(6) $F' := F'' \cup \{[h_C, p] \mid p \in h^\bullet\}$,

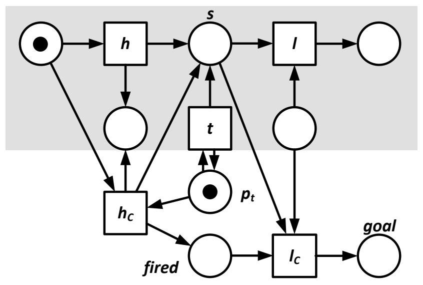
(7) $P' := P'' \cup \{\mathit{enabled}\}$.

In case of a conflict, the following changes are additionally required:

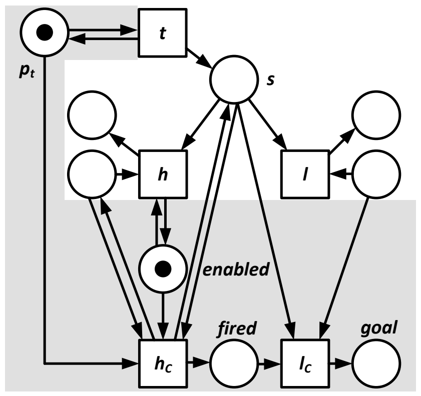
(8) $F' := F'' \cup \{[h_C, p] \mid p \in {}^\bullet h\} \cup \{[\mathit{enabled}, h_C], [h, \mathit{enabled}], [\mathit{enabled}, h]\}$,

(9) $m_0' := m_0'' \cup \{\mathit{enabled}\}$.

Extended nets are unlabeled Petri nets (cf. Def. 1) as well, because the labeling of the original transitions is not considered any more, so we disregard the labels in $N_{[s,h,l]}$. The extended nets are safe by construction. Regarding Def. 3, we need to disable all undesired transitions, therefore we create, for each undesired transition $t \in U_{[s,h,l]}$, an additional place p_t (1). Self loops between the undesired transitions $t \in U_{[s,h,l]}$ and their new places $p_t \in P_T$ (4), together with their initial marking (5) provide the desired behavior. The places of the extended net (2) consist of those from (1) and these new places *fired* (indicating whether h_C has fired) and *goal* (marked, if the objective $[s, h, l]$ is active). The new transitions (3) of the extended net are h_C and l_C. Think of them as copies of h and l. To make use of the additional nodes in the extended Petri net, we



(a) Causal case.



(b) Conflict case.

Figure 5: Extension patterns for causal and conflict case.

add arcs (4) from the preset of h to h_C, from the preset of l to l_C and the flow between *fired*, *goal*, h_C and l_C.

In the *causal* case, we need the effect of firing h_C (instead of h), so we add the postset of h to the postset of h_C (6). In the *conflict* case, we need an extra place *enabled* (7); this place ensures that after firing h_C (instead of h) h cannot fire any more, even though all preconditions for h remain. This is necessary because, in the conflict case, h and l need the same token on s. In contrast to the causal case, we do not need the effect of firing h or h_C. Instead, we must notice that h was activated. We thus add the preset of h to the postset of h_C and use *enabled* to ensure that this happens only once (8). The initial marking must hence contain an additional token on *enabled* (9).

In Fig. 5, the extension patterns for both cases are shown exemplarily. The extended parts of $N_{[s,h,l]}$ (over N) are shaded gray. For simplicity, only the vicinity of the interesting nodes s, h, and l are shown with just one element.

3.2. Completeness and Soundness Proofs

The main property to be proven is stated in Theorem 1: A leak encoded in an objective is active if and only if it is comprised in a reachable marking.

Theorem 1. *The objective $[s, h, l]$ is active in N if and only if a marking m with goal $\in m$ is reachable in $N_{[s,h,l]}$.*

To prove Theorem 1, it suffices to show completeness and soundness. We state each as lemma, proving them separately.

Lemma 1 (Completeness). *The objective $[s, h, l]$ is active in N if a marking m with goal $\in m$ is reachable in $N_{[s,h,l]}$*

To prove Lemma 1, we shall rely an additional proposition.

$$\begin{tikzcd}
N : & m_0 \arrow[r, "\sigma_0"] & m \arrow[r, "h\sigma"] & s \cup m' \arrow[r, "l"] & m''
\end{tikzcd}$$

$$\begin{tikzcd}
N_{[s,h,l]} : & m_0 \cup P_T \arrow[r, "\sigma_0"] & m \cup P_T \arrow[r, "h_C\sigma"] \arrow[dr, "h\sigma"] & s \cup m' \cup \mathit{fired} \arrow[r, "l_C"] & m'' \cup \mathit{goal} \\
& & & s \cup m' \cup P_T \arrow[r, shift left, "l"] \arrow[r, shift right] & m'' \cup P_T
\end{tikzcd}$$

(a) Firing sequences for the causal case.

$$\begin{tikzcd}
N : & m_0 \arrow[r, "\sigma_0"] & m \arrow[r, "\sigma l"] \arrow[dr, "h"] & m' \\
& & & m''
\end{tikzcd}$$

$$\begin{tikzcd}
N_{[s,h,l]} : & m_0 \cup P_T \cup \mathit{enabled} \arrow[r, "\sigma_0"] & m \cup P_T \cup \mathit{enabled} \arrow[r, "h_C"] \arrow[dr, "h"] \arrow[ddr, "\sigma l"] & m \cup \mathit{fired} \arrow[r, "\sigma l_C"] & m'' \cup \mathit{goal} \\
& & & m' \cup P_T \cup \mathit{enabled} \arrow[r, shift left, "\sigma l"] \arrow[r, shift right] & m'' \cup P_T \cup \mathit{enabled} \\
& & & m'' \cup P_T \cup \mathit{enabled} &
\end{tikzcd}$$

(b) Firing sequences for the conflict case.

Figure 6: Proof ideas with interesting firing sequences highlighted.

Proposition 1 (Same firing sequences). *Let N be a labeled Petri net, $[s, h, l]$ a potential objective of N and $N_{[s,h,l]}$ be the extended net for N and $[s, h, l]$. Then $N_{[s,h,l]}$ contains all firing sequences of N.*

Proof (Proposition 1). Observations: No transition and no arc is removed but only additional places are added in some presets. Thus, we only consider these new places in this proof.

In both cases, for each transition $t \in U_{[s,h,l]}$, an additional place p_t is added to its preset. In the initial marking m_0, these places are marked, hence their enabling is not restricted. Every time a transition $t \in U_{[s,h,l]}$ fires, it produces the token on p_t again due to the selfloop. The only transition which consumes all tokens from P_T is h_C, which is a new transition.

In the conflict case, an additional place *enabled* is added to the preset of h. In the initial marking m_0, *enabled* is marked, hence the enabling of h is not restricted. Every time h fires, it produces the token on *enabled* again. The only transition which consumes the token from *enabled* is h_C, which is a new transition.

Consequently, all firing sequences of N, which do not contain any new transitions, are also possible firing sequences in $N_{[s,h,l]}$. $\square$

With Proposition 1, we can prove Lemma 1 for completeness.

The main idea for the following proofs are based on the firing sequence in the input net N which leads to an insecure information flow. This firing sequence is depicted in Fig. 6 for both cases in the lines beginning with N. Reasoning about this firing sequence in the extended nets $N_{[s,h,l]}$ is part of the proofs. In Fig. 6 the corresponding firing sequence which leads to the identification of the insecure information flow in the extended net $N_{[s,h,l]}$ is highlighted as well. In addition to that interesting firing sequence, all other possible firing sequences (derived from N) are also depicted to see the effect of Proposition 1. In both proofs, we start with the causal case and afterwards we handle the conflict case.

Proof (Lemma 1).
Causal case. Assume $[s,h,l]$ is active in N. Then a marking m^* with $goal \in m^*$ must be reachable in $N_{[s,h,l]}$. N can fire $m_0 \xrightarrow{\sigma_0} m \xrightarrow{h\sigma} (s \cup m') \xrightarrow{l} m''$, so we have to show that $N_{[s,h,l]}$ can fire $m'_0 = (m_0 \cup P_T) \xrightarrow{*} m^*$, where m'_0 is the initial marking of $N_{[s,h,l]}$.

Based on Proposition 1, the extended net $N_{[s,h,l]}$ can fire the same firing sequences as N, especially σ_0 from m_0 to m. After this firing sequence, the marking of $N_{[s,h,l]}$ is $m \cup P_T$. In this marking, $N_{[s,h,l]}$ can fire h_C instead of h, which has the same effect to the original part of the net as firing h. Because $[s,h,l]$ is active, σ contains no transitions of $U_{[s,h,l]}$. Therefore disabling them by h_C does not disable any transition of σ. So instead of $(m \cup P_T) \xrightarrow{h\sigma} (s \cup m' \cup P_T)$ (by firing h), we obtain $(m \cup P_T) \xrightarrow{h_C\sigma} (s \cup m' \cup \mathit{fired})$ (by firing h_C). In both cases, the net $N_{[s,h,l]}$ can either fire l or l_C, in case of l_C the place *goal* is marked.

Conflict case. Assume $[s,h,l]$ is active in N. Then a marking m^* with $goal \in m^*$ must be reachable in $N_{[s,h,l]}$. N can fire $m_0 \xrightarrow{\sigma_0} m$ and from that m two fire sequences are possible: (1) $m \xrightarrow{h} m'$ and (2) $m \xrightarrow{\sigma l} m''$. Consequently we have to show that $N_{[s,h,l]}$ can fire $m'_0 = (m_0 \cup P_T \cup \mathit{enabled}) \xrightarrow{*} m^*$, where m'_0 is the initial marking of $N_{[s,h,l]}$.

Based on Proposition 1, the extended net $N_{[s,h,l]}$ can fire the same firing sequences as N, especially σ_0 from m_0 to m. After this firing sequence, the marking of $N_{[s,h,l]}$ is $m \cup P_T \cup \mathit{enabled}$. In this marking, $N_{[s,h,l]}$ can fire h_C instead of h, which has no effect to the original part of the net, except the disabling of h. Because $[s,h,l]$ is active, σ contains neither h nor any transitions of $U_{[s,h,l]}$. Therefore disabling them by h_C does not restrict σ. So instead of $(m \cup P_T \cup \mathit{enabled}) \xrightarrow{h} (m' \cup P_T \cup \mathit{enabled})$ (by firing h), we obtain $(m \cup P_T \cup \mathit{enabled}) \xrightarrow{h_C} (m \cup \mathit{fired})$ (by firing h_C). In both cases, the net $N_{[s,h,l]}$ can either fire σl or σl_C, in case of σl_C the place *goal* is marked. □

For now, we proved that every active objective $[s,h,l]$ leads to a reachable marking m in which *goal* is marked in the extended net $N_{[s,h,l]}$. Next we show the reverse; that is, in all cases *goal* is reachable in $N_{[s,h,l]}$ the objective $[s,h,l]$ is active in N.

Lemma 2 (Soundness). *The objective $[s,h,l]$ is active in N only if a marking m with $goal \in m$ is reachable in $N_{[s,h,l]}$.*

The proof of Lemma 2 is analogous to the proof of Lemma 1, except for the fact that we use an indirect approach. Similar to the Lemma 1 proof, we start with the causal case.

Proof (Lemma 2).
Causal case. Assume that a marking m^* with $goal \in m^*$ is reachable in $N_{[s,h,l]}$ and $[s,h,l]$ is not active in N.

Place *goal* can only be marked by firing l_C. If l_C was able to fire, l was also able to fire ($^\bullet l \subsetneq {^\bullet l_C}$). Before l_C can fire, firing h_C is necessary. If h_C was able to fire, h was also able to fire ($^\bullet h \subsetneq {^\bullet h_C}$). A possible firing sequence between h_C and l_C cannot contain transitions from $U_{[s,h,l]}$, because h_C consumes all tokens on all places $p \in P_T$. This means σ contains no undesired transitions. Consequently $h\sigma l$ is a valid firing sequence in N starting in some marking reachable from m_0 in contradiction to the assumption.

Conflict case. Assume that a marking m^* with $goal \in m^*$ is reachable in $N_{[s,h,l]}$ and $[s, h, l]$ is not active in N.

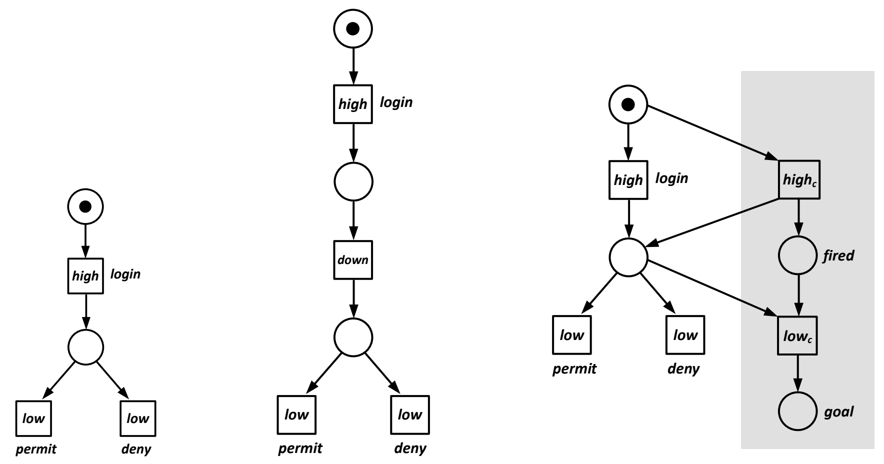
Place *goal* can only be marked by firing l_C. If l_C was able to fire, l was also able to fire ($^\bullet l \subsetneq {^\bullet l_C}$). Before l_C can fire, firing h_C is necessary. If h_C was able to fire, h was also able to fire ($^\bullet h \subsetneq {^\bullet h_C}$) in a marking m. Due the firing of h_C instead of h the marking m is preserved for l and l_C. A possible firing sequence from m to l_C cannot contain transitions from $U_{[s,h,l]}$ nor h, because h_C consumes all tokens on all places $p \in P_T$ and on the place *enabled*. This means σ contains no undesired transitions. Consequently h is able to fire in m and σl is a valid firing sequence in N starting in that marking m in contradiction to the assumption. $\square$

By proving Theorem 1 (due Lemma 1 and Lemma 2) we ensure that our approach detects all possible violations (completeness) and that all violations detected by our approach are indeed violations (soundness). We implemented the introduced constructions and shall report on the implementation and an evaluation in Sect. 5 and Sect. 6, respectively.

4. Verification of PBNID as Reachability Problems

In the previous section we demonstrated how PBNI+ verification is decided using reachability [17]. We therefore introduced objectives to encode potential interferences. To decide whether such an objective, is a security violation, we create an extended net based on the objective. Subsequently, we perform a reachability check on each extended net. In case a dedicated place in the extended net can be marked, the potential interference is an active violation. With the provided proofs we guarantee that our translation to reachability is sound and complete.

In practice, information flow control is often too strict, because every interference is considered as bad. In particular, there is no differentiation among any identified interference (i.e., its badness), also one could imagine that there are different levels of risks according to the specific information flow. Think, for instance, of a login check where in the end one bit distinguishes whether the login attempt is successful. Here, an authorized flow from a high domain (the user name and its password) to the low domain (does it match) happens, even though such a flow violates the non-interference property. In Fig. 7(a), such a simple login process is depicted. The concept of *declassification* is used to controllably *downgrade* flows from high to low. Busi and Gorrieri [36] extended the concept



(a) Simple one. (b) Secure one. (c) Extended net.

Figure 7: A simple login process.

of PBNI+ to intransitive non-interference with downgrading transitions, so-called PBNID. Mainly they add, to the existing domains of high and low, a third domain *downgrading* to partition the set of transitions. Using a downgrade labeled transition between a high and a low labeled transition controllably downgrades the flow from high to low. Right to the simple process (cf. Fig. 7(a)) almost the same process which uses a downgrading transition in order to permit the necessary flow is depicted (cf. Fig. 7(b)). Therefore, we excluded a leak from h to l.

To this end the flow from high to low respects the security property. To be more exactly and to rule out the opinion that this may be transitive: PBNID prohibits flows from high to low as well. The only way to allow a flow from high to low is a downgrade between those two domains. This means, that flows from high to downgrade are secure and flows from downgrade to low respect the property as well. Nevertheless direct flows from high to low are still a security violation because the property is intransitive.

4.1. Formal background

To express PBNID Busi and Gorrieri [13] modified Def. 3: they add a set D of downgrading transitions and discard any $d \in D$ from the transition sequence σ. This means for the definition of labeled Petri nets [2], that we need a third partition to reason about the downgrading domain. The remainder of this section considers this type of Petri net.

Definition 6 (Labeled Petri Net for PBNID). Let $N = [P, L, H, F, m_0]$ be a labeled Petri net and D a set of transitions such that $L \cap H \cap D = \emptyset$. A *labeled Petri net for PBNID* is a Petri net $[P, L \cup H \cup D, F, m_0]$.

This extension to a third domain (viz. transition set) requires a review of the definitions of causal and conflict places. In Def. 3, we already required the transition sequence σ to only contain transitions from $H \cup L$. Though this requirement is artificial in the context of Sect. 3, it is necessary for PBNID: for both the causal and the conflict places, a low transition that is preceded by a downgrading transition cannot yield an active place. To this end, downgrading transitions are excluded in the definition of potential places.

4.2. Creating Reachability Problems

Considering the differences between the initial definitions (for PBNI+) and the new definitions (for PBNID) the approach to reduce PBNID to reachability is straightforward. An active violation for PBNI+ is also an active violation for PBNID, with just one exception: In case a downgrade happens between the occurrence of a high and a low transition, the flow (between high and low) is secure in terms of PBNID in contrast to PBNI+. Therefore also the definitions of the central new concepts objective and the undesired transitions can be used again.

The main difference and idea is to add all downgrading transitions $d \in D$ in $N_{[s,h,l]}$ to the undesired transitions $U_{[s,h,l]}$. By adding the downgrading transitions to the undesired transitions, there use is prohibited after the occurrence of h. In case the place *goal* can be marked, the objective is also active for PBNID.

Definition 7 (Undesired transitions for PBNID).
Let $U_{[s,h,l]} = \{t \in T \mid s \in t^\bullet\} \cup D$ be the set of *undesired transitions* in the transition sequence σ (cf. Def. 3), necessary to decide whether s is active.

Definition 8 of the extended net for the verification of PBNID is almost the same as for PBNI+ (cf. Def. 5). The only difference is in point (3), in which the transitions D must be considered as well.

Definition 8 (Extended net for PBNID). Let $N = [P, L, H, F, m_0]$ be a labeled Petri net and $[s, h, l]$ be an objective of N. The *extended net* $N_{[s,h,l]} = [P', T, F', m_0']$ is a Petri net, which is constructed based on N as follows:

(1) $P_T := \{p_t \mid t \in U_{[s,h,l]}\}$,

(2) $P'' := P \cup \{\mathit{fired}, \mathit{goal}\} \cup P_T$,

(3) $T := L \cup H \cup D \cup \{h_C, l_C\}$,

(4) $F'' := F \cup \{[p, h_C] \mid p \in {}^\bullet h\} \cup \{[p, l_C] \mid p \in {}^\bullet l\} \cup \{[h_C, \mathit{fired}], [\mathit{fired}, l_C], [l_C, \mathit{goal}]\} \cup \{[p_t, t], [t, p_t] \mid t \in U_{[s,h,l]}, p_t \in P_T\} \cup \{[p_t, h_C] \mid p_t \in P_T\}$,

(5) $m_0'' := m_0 \cup P_T$.

For the causal case the following additional changes are necessary:

(6) $F' := F'' \cup \{[h_C, p] \mid p \in h^\bullet\}$,

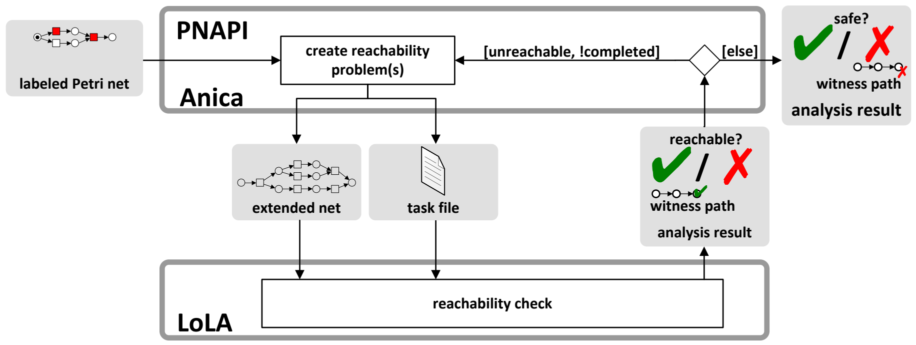


Figure 8: Architecture of Anica.

(7) $P' := P'' \cup \{enabled\}$.

In case of a conflict those changes are additionally required:

(8) $F' := F'' \cup \{[h_C, p] \mid p \in {}^{\bullet}h\} \cup \{[enabled, h_C], [h, enabled], [enabled, h]\}$,

(9) $m_0' := m_0'' \cup \{enabled\}$.

With this definition at hand, we are able to express the verification of PBNID in nearly the same manner as the verification of PBNI+. Figure 7(c) depicts the extended pattern for the simple login process depicted in Fig. 7(a). We refrain from providing proofs for soundness and completeness properties for PBNID because they are absolutely analogous to those in Sect. 3.

5. Anica — Tool Support for Detecting Information Flows

Anica (Automated Non-Interference Check Assistant) implements Theorem 1. Anica is a novel tool to check PBNI+ (and PBNID) much faster and with less memory consumption than existing tools (cf. Sect. 6 for details) by creating reachability problems automatically. As part of the open source *service-technology.org*-family [37], Anica uses as “reasoner” the general purpose model checker LoLA [18] and the Petri Net API [38].

Figure 8 depicts the modular architecture. Anica first reads a labeled Petri net as input, then creates pairs of extended nets and task files for the detection of causal and conflict places, as well as PBNID constraints. A task file contains the reachability statement for place *goal*. These pairs are passed to LoLA, which applies sophisticated state space reduction techniques to decide the reachability problem (task file). If the checked objective is not active, more pairs are created, otherwise the result is presented to the user, also describing the actual type of violation (e.g., causal or conflict place detected). Anica can be configured to detect all violations or stop after finding one counterexample. Eventually, if no objective is marked as active, the original Petri net representing the process does not exhibit leaks induced by causal and conflict places.

Algorithm 1 isNoninterfering(N): *Boolean*

```
1: for all $s \in P$ do
2:   if isActive($N, s$) then
3:     return false                              // PBNI+ or PBNID is violated
4: return true
```

Algorithm 2 isActive(N, s): *Boolean*

```
1: for all $l \in (L \cap s^\bullet)$ do
2:   for all $h \in (H \cap {}^\bullet s)$ do
3:     if isActiveObjective($N, [s, h, l]$) then   // $s$ is a potential causal place
4:       return true                              // $s$ is an active causal place
5:   for all $h \in (H \cap s^\bullet)$ do
6:     if isActiveObjective($N, [s, h, l]$) then   // $s$ is a potential conflict place
7:       return true                              // $s$ is an active conflict place
8: return false                                   // $s$ is not active
```

Algorithm 3 isActiveObjective(N, [s,h,l]): *Boolean*

```
1: construct extended net $N_{[s,h,l]}$                        // according to Def. 5
2: if $m_0 \xrightarrow{*} m$ in $N_{[s,h,l]}$ with $goal \in m$ then       // call LoLA
3:   return true
4: return false
```

The implementation of Anica follows the constructive Defs. 3–5. In this section, we focus on how to combine the definitions to a tool. The input of Anica is a labeled Petri net. The transitions of such a net are labeled with high ($h \in H$) or low ($l \in L$) and for PBNID possibly with downgrade ($d \in D$). A net violates the non-interference properties PBNI+ or PBNID if it contains at least one active place. To this end, we check for each place whether it is active (cf. Alg. 1). In case we are only interested in whether the whole net is secure, we can abort after the first active place (cf. Alg. 1, line 3). For a complete check of the whole net, we process all places, which can be done simultaneously.

A necessary condition for an active place is that it must be a potential (causal or conflict) place (cf. Def. 3). Checking whether a place is potential can be decided by the structure of the input net, which has linear complexity in terms of nodes. If the input net contains no potential place, it is secure. If a place s is a potential place, we generate all its objectives (cf. Alg. 2). For causal and conflict places we consider the transitions labeled low ($l \in L$) in the postset of s. Which transitions labeled high ($h \in H$) are necessary depends on the case: we use the preset of s for potential causal places and the postset of s for potential conflict places. Whereas one active objective $[s, h, l]$ (tested by Alg. 3) is enough to make a potential place s active, we abort incase on the first active objctive (Alg. 2, lines 4 and 7). All checks of the objectives are independent (Alg. 2, lines 2-4 and 5-7).

6. Evaluation

We evaluated Anica to demonstrate that information flow analysis becomes feasible with our approach in contrast to the existing ones. We use a library of 735 industrial BPs from different business branches, including financial services, ERP, supply-chain, and online sales. These BPs were modeled in the IBM WebSphere Business Modeler format and correspond to real business processes found in industry. Of 735 models, 559 fulfill the soundness criteria [33], which is a necessary requirement for analysis. These remaining sound models build the core set of BPs for the evaluation. Fahland et al. [39] translated this set of BPs to role annotated Petri nets.

The 559 sound BPs contain no semantic information with respect to the security domains. That is, the activities in these models are not labeled with "high" and "low" for security analysis, nor there are policies from which these labels could be derived from. However, it is a large, representative collection of industrial BP models and adding security aspects would not dramatically alter the structure of the processes. To this end, we treat the given process library as a valid approximation of concrete BP models, because the corresponding state spaces would have similar sizes.

To overcome the absence of security information and annotate the activities with security domains, we use the roles of the BPs (called swim lanes) for labeling. For each role r in a BP B, we create one labeled Petri net N_{B_r}. In N_{B_r}, all transitions labeled with r are labeled high. This means we check PBNI+.

Figure 9 gives an overview on this approach. Intuitively, with this strategy we can check the interference between the different roles within a process, (e.g., interference between the back- and front office of a financial service). We cannot judge the resulting interferences, because they are constructed by us, instead of derived from practice. But we believe this approach is comprehensible, acceptable, and more realistic than creating randomized BPs and security domains. This yields 1,206 labeled Petri nets which build the base for the evaluation. Table 1 summarizes their structural and behavioral properties.

Intentionally and for the sake of exhaustiveness, all the 1,206 processes violate the non-interference property: in each process, there exist at least one, at most 25 and in average 3 potential causal places. All of them are also active causal places and none of the 52,751 examined places is a potential conflict place. The absence of conflict places can be explained by the construction of the security

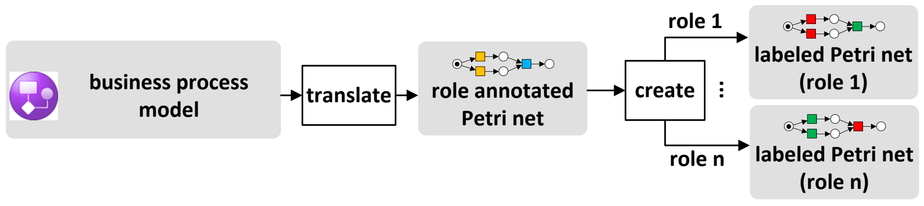


Figure 9: Testsuite generation.

Table 1: Structural and behavioral properties of the 1,206 labeled Petri nets.

	Min	**Average**	**Max**
Places	5	44	234
Transitions (labeled high)	4 (2)	28 (5)	100 (60)
Arcs	8	93	476
Full state space ($<$ 10,000,000 states)	5	1,777	588,508
Full state space ($>$ 10,000,000 states)	28,451,334	$>$ 1,074,797,069	$>$ 2,214,007,203

Table 2: Comparing other tools (state space) with Anica (reachability).

	Other tools [15, 16]	**Anica**
Labeled Petri nets verified	1,178	1,206
Minimum states computed	5	5
Average states computed	1,777	51
Maximum states computed	588,508	735
Overall states computed	2,088,135	62,049

domains: Labeling along the swim lanes results in a clear responsibility between two roles. After a role has finished a specific task, it provides its results to exactly one successor role. The successor role is chosen by the current role; that is, the choice is in the initial swim lane. Overall, 8 percent of all places violate the non-interference property.

State space and runtime. Using existing techniques to analyze PBNI+ [13] or PBNID [36], the whole state space must be created. As 28 of the 1,206 labeled Petri nets (see Tab. 2) contain more than 10 million states (see Tab. 1), those are infeasible using common desktop hardware (8 GB RAM). The existing state space approaches require about 2 million states to verify the remaining 1,178 nets and fail for the 28 big nets. For those, we used a server with 128 GB RAM, computed more than 30 billion states in several days and were able to verify 15 of 28 nets.

Our reachability approach instead verifies *all* 1,206 labeled Petri nets with about 0.06 million states in about 30 seconds on common desktop hardware. The proofs provided in Sect. 3.2 ensure the correctness of our approach. We performed all tests in a sequential manner (i.e., used no parallelism) and the 30 seconds include the time consumed by LoLA for deciding the reachability problems. This leads to an average time consumption of about 24 ms for each net (212 ms for the biggest net), without exploiting the inherent parallelism.

Running example (cont.). Anica verifies the net in Fig. 3 by checking 12 states, instead of 32 states for the full state space. Place *Record* is identified as an active causal and an active conflict place. The objective for the causal case is $[\mathit{Record}, \mathit{High.Close}, \mathit{Low.Open}]$ and the conflict case it is $[\mathit{Record}, \mathit{High.Open}, \mathit{Low.Open}]$.

The testsuite shows that, by reducing PBNI+ to Petri net reachability, (1) we can handle more complex inputs from industry and (2) we can check them independently. Especially the latter aspect is fundamental: Instead of calculating one large state space and check, for each potential place, whether it is active, we generate one Petri net for every potential place. As a result, we can use reduction techniques that aim at generating only a small fraction of the state space to proof or disproof the reachability of the goal place. Experiments [18] from several domains show that this modular approach is very effective and even though dozens of state spaces need to be constructed, this usually saves both time and memory.

Note that Anica follows a modular architecture and does not depend on LoLA. All results can be achieved with any other analysis tool for Petri nets capable of state-of-the-art reduction techniques for reachability.

7. Related Work

Business process security focuses on the enforcement of confidentiality, integrity and availability requirements in BP models [40]. The focus of information flow analysis is confidentiality and, dually, integrity [35]. While approaches employing formalisms other than Petri nets exist (e.g., [6, 8, 9, 41]), the vast majority of proposed technologies build upon Petri nets. This is due to the extensive use of Petri net models to reason about BPs and the availability of mature tool support. The state of the art addresses: (1) the detection of *explicit* information flows (i.e., data flows), both in terms of discretionary access control (DAC) and mandatory access control (MAC) based upon multi-level security; and (2) the detection of *implicit* flows over covert channels (i.e., interferences). Today's focus is on (1), whereas the tool Anica and the formal results presented here focus on (2).

DAC. Tackling DAC, Shafiq et al. [42] present a colored Petri net (CPN) framework for verifying the consistency of role based access control (RBAC) policies. Similarly, Armando and Ranise [5] provide a SAT-based tool support for analyzing BPs with RBAC. Sun et al. [43] employ hierarchical CPN to capture RBAC and analyze BP collaboration. This model is lifted by Sun et al. [44] to manage the service-based BP authorization. Moving to more sophisticated contraints, the Chinese Wall policy, together with the Strict Integrity Policy, is considered by Zhang et al. [45]. Here, policies are modeled with CPN, whereas the verification is based upon a coverability graph technique. Huang and Kirchner [46] use CPN to model policies in general, and conflict of interest in particular, thereby focusing on the modularity aspects of Petri net-based policies. Lu et al. [47] employ CPN to model and analyze separation of duty constraints in collaborative BP. Katt et al. [48] and ourselves [49] employ CPN to model usage control policies and regulatory compliance contraints. However, focusing on monitoring systems, their objective is not the static analysis. Operating within the context of the DAC model, the aforementioned analysis technologies merely cover "point-to-point" security guarantees. While this is sufficient to detect data leaks, these approaches cannot detect information leaks originating from interferences.

MAC. Turning to MAC-based data-flow analysis, existing approaches are based upon the multi-level security (MLS) model, in particular the model of Denning [29] and Bell and LaPadula [50]. Juszczyszyn [51] uses CPN for the MLS specification of policies and CPN-Tool to realize the verification as a reachability problem. Röhrig and Knorr [10] define task based access control as a dynamic BP and then specify the BPs with Petri nets; the approach is unclear as to the realization of the analysis, though. Barkaoui et al. [7] employ the security rules of Bell and LaPadula for the verification of workflows expressed as Workflow nets. The reasoning reduces to a soundness check with Maude. While these technologies employ MLS to capture information flows, they do not detect interferences.

Information flow analysis. The information flow analysis of BPs models focusing on interferences is a young research strand in the business process management community [52]. The approach is inspired by the formalization of non-interference properties as Petri net patterns given by Busi and Gorrieri [13]. On these grounds, we [14, 28] present with InDico information flow nets to capture BP transformations to automatically label the model with security classes, and Petri net patterns capturing non-interference. We also consider other properties, including data flow-based properties and industrial constraints, such as separation of duties and declassification. This approach has been successfully employed to detect leaks in BP models; for example in e-auction [14], e-health [28, 53] and publishing [54]. However soundness, completeness and decidability guarantees are not provided. The approach presented in this paper is thus the first to use reachability to address the verification problem and to provide soundness and completeness properties. Decidability of non-interference properties over unbounded Petri nets, was first shown by Best et al. [55]. With the reduction to the reachability problem of Petri nets, we presented an alternative decidability result, even though our constructions currently rely on safe Petri nets. A relaxation of these prerequisites should be straightforward.

Some results of this paper have been published earlier [1]. This article extends the results in that abstract by a reduction of PBNID to reachability, more details and illustrations of the proofs in Sect. 3, and a discussion how our approach can be used to investigate multi-level security models.

Tools. Turning to tools for automatic information flow analysis of BP models, Frau et al. [16] implemented the general-purpose Petri Net Security Checker (PNSC) and we implemented InDico in the Security Workflow Analysis Toolkit (SWAT) [15]. Both realizations follow the state space exploration strategy proposed in [13]. They thus require the construction and exploration of the whole state space, so that the tools do not scale well for big Petri net models. The tool Anica presented in this paper provides a faster alternative.

8. Summary and Future Work

We introduced a novel approach for the automated verification of information flow control for BPs. The main insight in the paper is the treatment of PBNI+ and PBNID verification as a set of reachability problems, which opens up the possibility of employing efficient, automated tool-support for the analysis. The approach is proven to be sound and complete, and is implemented in the Anica tool. A thorough evaluation demonstrates the speed to verify complex BPs.

Lessons learnt. From a complexity-theoretic point of view, our reachability approach has the same worst case complexity than existing approaches building the complete state space of a safe net: it is PSPACE-complete. However, experimental results showed that in industrial examples, the reachability problems solved by Anica yields much faster results by less memory consumption.

This confirms experiences reported by Fahland et al. [39] which also employed LoLA [18] to analyze industrial business processes. The employed state space reduction techniques — in particular partial order reduction [56] — shows very efficient reduction in BP models which typically exhibit a lot of concurrent behavior. As a result, information flow security checks for industrial, complex BPs become efficient, in the sense that they only take split seconds. This opens up the possibility of information flow analysis in background during the process modeling, thereby contributing to reliably secure BP design. Such a guarantee was not possible with previous techniques. We recently integrated Anica into a BP modeling tool [19] and demonstrated how formerly expensive security checks can be offered to the modeler similar to spell checks as push-button technique.

Another interesting observation is that Anica produces faster results by replacing the construction of a *single state space* by *several reachability problems*. This shows that even the sum of several smaller state spaces constructed during the reachability checks is smaller than one complete state space. Furthermore, these checks could be parallelized to obtain further speed ups.

Future work. Future work aims at four directions: *firstly*, augment the set of properties to be analyzed. In particular, we plan to integrate the properties considered in InDico [28]; for instance parameter confidentiality and resource conflicts. We firmly believe that further properties considerably extend the expressive power of the certification, thereby contributing to reliably secure BP modeling from the outset. *Secondly*, spell out the consequences for more complex security models beyond the high and low classes. This allows one to capture fine-grained properties which are especially useful for data-flow reasoning. *Thirdly*, provide some further information on found violations. So far the modeler can identify the cause during the modeling (when using background checks) or use provided witness paths. Nevertheless the question is raised whether it is possible to rate found violations automatically. Based on these ideas, it seems possible to help the modeler to fix the BP, therefore he could use for instance downgrading transitions (PBNID) which are already supported by Anica. With regard to declassification, we *fourthly* plan to investigate the practibility of extensions to PBNID for selective declassification provided by Best and Darondeau [57] to provide a more fine-grained analysis of and control over declassification.

Acknowledgments. The authors cordially thank Dirk Fahland and Harro Wimmel for their valuable feedback and the anonymous reviewers of the paper at the Conference Business Process Management 2012 for their constructive reviews. The work is partially funded by the German Research Foundation in the project WS4Dsec of the priority program Reliably Secure Software Systems (SPP 1496).

References

[1] R. Accorsi, A. Lehmann, Automatic information flow analysis of business process models, in: BPM 2012, LNCS 7481, Springer, 2012, pp. 172–187.

[2] L. Lowis, R. Accorsi, Vulnerability analysis in SOA-based business processes, IEEE T. Services Computing 4 (3) (2011) 230–242.

[3] ISO/IEC, ISO/IEC Information Security Management System 27001, www.27000.org/iso-27001.htm (2005).

[4] DoD, Trusted computer security evaluation criteria, http://csrc.nist.gov/publications/histroy/dod85.pdf (1983).

[5] A. Armando, S. Ranise, Automated analysis of infinite state workflows with access control policies, in: C. Meadows, C. Fernandez-Gago (Eds.), Security and Trust Management, Vol. 7170 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 157–174.

[6] V. Atluri, S. A. Chun, P. Mazzoleni, A chinese wall security model for decentralized workflow systems, in: Proceedings of the 8th ACM conference on Computer and Communications Security, CCS '01, ACM, New York, NY, USA, 2001, pp. 48–57.

[7] K. Barkaoui, R. Ben Ayed, H. Boucheneb, A. Hicheur, Verification of workflow processes under multilevel security considerations, in: Risks and Security of Internet and Systems, 2008. CRiSIS '08. Third International Conference on, 2008, pp. 77 –84.

[8] W. R. Harris, N. A. Kidd, S. Chaki, S. Jha, T. Reps, Verifying information flow control over unbounded processes, in: Proceedings of the 2nd World Congress on Formal Methods, FM '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 773–789.

[9] M. Kovács, H. Seidl, Runtime Enforcement of Information Flow Security in Tree Manipulating Processes, in: G. Barthe, B. Livshits, R. Scandariato (Eds.), Engineering Secure Software and Systems, Vol. 7159 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2012, pp. 46–59.

[10] S. Röhrig, K. Knorr, Security analysis of electronic business processes, Electronic Commerce Research 4 (2004) 59–81.

[11] D. E. Denning, P. J. Denning, Certification of programs for secure information flow, Commun. ACM 20 (7) (1977) 504–513.

[12] N. Lohmann, H. Verbeek, R. M. Dijkman, Petri net transformations for business processes – a survey, LNCS ToPNoC II (5460) (2009) 46–63.

[13] N. Busi, R. Gorrieri, Structural non-interference in elementary and trace nets, Mathematical Structures in Computer Science 19 (6) (2009) 1065–1090.

[14] R. Accorsi, C. Wonnemann, Strong non-leak guarantees for workflow models, in: W. C. Chu, W. E. Wong, M. J. Palakal, C.-C. Hung (Eds.), SAC, ACM, 2011, pp. 308–314.

[15] R. Accorsi, C. Wonnemann, S. Dochow, SWAT: A security workflow toolkit for reliably secure process-aware information systems, in: ARES 2011, IEEE, 2011, pp. 692–697.

[16] S. Frau, R. Gorrieri, C. Ferigato, Petri net security checker: Structural non-interference at work, in: FAST 2008, LNCS 5491, Springer, 2008, pp. 210–225.

[17] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (4) (1989) 541 –580.

[18] K. Wolf, Generating Petri net state spaces, in: J. Kleijn, A. Yakovlev (Eds.), ICATPN 2007, Vol. 4546 of LNCS 4546, Springer-Verlag, 2007, pp. 29–42.

[19] A. Lehmann, N. Lohmann, Modeling wizard for confidential business processes, in: R. Accorsi, R. Matulevicius (Eds.), 1st Joint Int. Workshop on Security in Business Processes (SBP'12), Tallinn, 3 September 2012, Proceedings, 2012, (To appear in a LNBIP volume).

[20] B. Lampson, A note on the confinement problem, Communications of the ACM 16 (10) (1973) 613–615.

[21] F. Gallegos, S. Senft, Information Technology Control and Audit, Auerbach Publications, 2004.

[22] T. Ristenpart, E. Tromer, H. Shacham, S. Savage, Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds, in: E. Al-Shaer, S. Jha, A. D. Keromytis (Eds.), ACM Conference on Computer and Communications Security, ACM, 2009, pp. 199–212.

[23] M. Pearce, S. Zeadally, R. Hunt, Virtualization: Issues, security threats, and solutions, ACM Comput. Surv. 45 (2) (2013) 17:1–17:39.

[24] S. Chen, R. Wang, X. Wang, K. Zhang, Side-channel leaks in web applications: A reality today, a challenge tomorrow, in: IEEE Symposium on Security and Privacy, IEEE, 2010, pp. 191–206.

[25] S. Subashini, V. Kavitha, A survey on security issues in service delivery models of cloud computing, J. Network and Computer Applications 34 (1) (2011) 1–11.

[26] A. Shabtai, Y. Elovici, L. Rokach, A survey of data leakage detection and prevention solutions, Springer, 2012.

[27] R. Accorsi, On process rewriting for business process security, in: International Symposium on Data-driven Process Discovery and Analysis, Vol. 1027 of CEUR Workshop Proceedings, CEUR-WS.org, 2013, pp. 111–126.

[28] R. Accorsi, C. Wonnemann, Indico: Information flow analysis of business processes for confidentiality requirements, in: J. Cuéllar, J. Lopez, G. Barthe, A. Pretschner (Eds.), STM, Vol. 6710 of Lecture Notes in Computer Science, Springer, 2010, pp. 194–209.

[29] D. E. Denning, A lattice model of secure information flow, Commun. ACM 19 (5) (1976) 236–243.

[30] M. Benantar, Access Control Systems, Springer, 2006.

[31] R. J. Anderson, Security engineering - a guide to building dependable distributed systems (2. ed.), Wiley, 2008.

[32] J. Biskup, Security in Computing Systems - Challenges, Approaches and Solutions, Springer, 2009.

[33] W. M. P. v. d. Aalst, The application of Petri nets to workflow management, Journal of Circuits, Systems and Computers 8 (1) (1998) 21–66.

[34] R. Focardi, R. Gorrieri, Classification of security properties (part i: Information flow), in: R. Focardi, R. Gorrieri (Eds.), FOSAD, Vol. 2171 of Lecture Notes in Computer Science, Springer, 2000, pp. 331–396.

[35] A. Sabelfeld, A. C. Myers, Language-based information-flow security, Selected Areas in Communications, IEEE Journal on 21 (1) (2003) 5–19.

[36] R. Gorrieri, M. Vernali, On intransitive non-interference in some models of concurrency, in: A. Aldini, R. Gorrieri (Eds.), Foundations of Security Analysis and Design VI, Vol. 6858 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 125–151.

[37] N. Lohmann, K. Wolf, How to implement a theory of correctness in the area of business processes and services, in: R. Hull, J. Mendling, S. Tai (Eds.), Business Process Management, 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 14–16, 2010, Proceedings, Vol. 6336 of Lecture Notes in Computer Science, Springer-Verlag, 2010, pp. 61–77.

[38] N. Lohmann, S. Mennicke, C. Sura, The Petri Net API: A collection of Petri net-related functions, in: M. Schwarick, M. Heiner (Eds.), Proceedings of the 17th German Workshop on Algorithms and Tools for Petri Nets (AWPN 2010), Cottbus, Germany, October 7-8, 2010, Vol. 643 of CEUR Workshop Proceedings, CEUR-WS.org, 2010, pp. 148–155.

[39] D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, K. Wolf, Analysis on demand: Instantaneous soundness checking of industrial business process models, Data Knowl. Eng. 70 (5) (2011) 448–466.

[40] V. Atluri, J. Warner, Security for workflow systems, in: M. Gertz, S. Jajodia (Eds.), Handbook of Database Security, Springer US, 2008, pp. 213–230.

[41] I. Attali, D. Caromel, L. Henrio, F. L. Del Aguila, Secured information flow for asynchronous sequential processes, Electron. Notes Theor. Comput. Sci. 180 (1) (2007) 17–34.

[42] B. Shafiq, A. Masood, J. Joshi, A. Ghafoor, A role-based access control policy verification framework for real-time systems, in: Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, WORDS '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 13–20.

[43] H. Sun, X. Wang, J. Yang, Y. Zhang, Authorization policy based business collaboration reliability verification, in: A. Bouguettaya, I. Krüger, T. Margaria (Eds.), International Conference on Service-Oriented Computing, Vol. 5364 of Lecture Notes in Computer Science, 2008, pp. 579–584.

[44] H. Sun, J. Yang, W. Zhao, S. Nepal, SOAC-Net: A model to manage service-based business process authorization, in: L. E. Moser, M. Parashar, P. C. K. Hung (Eds.), IEEE International Conference on Services Computing, IEEE, 2012, pp. 376–383.

[45] Z.-L. Zhang, F. Hong, H.-J. Xiao, Verification of strict integrity policy via Petri nets, in: Systems and Networks Communications, 2006. ICSNC '06. International Conference on, 2006, p. 23.

[46] H. Huang, H. Kirchner, Formal specification and verification of modular security policy based on colored Petri nets, IEEE Transactions on Dependable and Secure Computing 8 (2011) 852–865.

[47] Y. Lu, L. Zhang, J. Sun, Using colored Petri nets to model and analyze workflow with separation of duty contraints, Journal on Advances in Manufactory Technology (40) (2009) 179–192.

[48] B. Katt, X. Zhang, M. Hafner, Towards a usage control policy specification with Petri nets, in: R. Meersman, T. S. Dillon, P. Herrero (Eds.), OTM Conferences (2), Vol. 5871 of Lecture Notes in Computer Science, Springer, 2009, pp. 905–912.

[49] R. Accorsi, L. Lowis, Y. Sato, Automated certification for compliant cloud-based business processes, Business & Information Systems Engineering 3 (3) (2011) 145–154.

[50] D. E. Bell, L. J. LaPadula, Secure Computer Systems: Mathematical Foundations, Tech. rep., MITRE Corporation (Mar. 1973).

[51] K. Juszczyszyn, Verifying enterprise's mandatory access control policies with coloured Petri nets, in: Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on, 2003, pp. 184 – 189.

[52] R. Accorsi, C. Wonnemann, Detective information flow analysis for business processes, in: W. Abramowicz, L. A. Maciaszek, R. Kowalczyk, A. Speck (Eds.), Business Process, Services Computing and Intelligent Service Management, Vol. 147 of LNI, GI, 2009, pp. 223–224.

[53] S. Pfeiffer, S. Unger, D. Timmermann, A. Lehmann, Secure information flow awareness for smart wireless ehealth systems, in: Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on, 2012, pp. 1 –6.

[54] R. Accorsi, T. Stocker, On the exploitation of process mining for security audits: the conformance checking case, in: S. Ossowski, P. Lecca (Eds.), SAC, ACM, 2012, pp. 1709–1716.

[55] E. Best, P. Darondeau, R. Gorrieri, On the decidability of non interference over unbounded Petri nets, in: K. Chatzikokolakis, V. Cortier (Eds.), International Workshop on Security Issues in Concurrency, Vol. 51 of EPTCS, 2010, pp. 16–33.

[56] A. Valmari, Stubborn sets for reduced state space generation, in: APN, Vol. 483 of LNCS, Springer, 1990, pp. 491–515.

[57] E. Best, P. Darondeau, Deciding selective declassification of Petri nets, in: P. Degano, J. D. Guttman (Eds.), Conference on Principles of Security and Trust, Vol. 7215 of Lecture Notes in Computer Science, Springer, 2012, pp. 290–308.