# Model support for confidential service-oriented business processes

Andreas Lehmann and Niels Lohmann

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{andreas.lehmann, niels.lohmann}@uni-rostock.de

**Abstract.** A core motivation of service-oriented execution of business processes is the opportunity to reduce costs by outsourcing certain tasks to third-party service providers. For legal or economic reasons, it might be undesirable that delicate information (e. g., customer data, trade secrets, or financial details) "leak" to the involved third parties. The absence of such leaks — called *noninterference* — can be checked automatically. To this end, a model is required in which each task is assessed as either confidential or public. A drawback of this method is that (1) this distinction has to be made for each task prior to the verification and that (2) an unsuccessful check requires a new confidentiality assessment followed by another verification step.
This paper introduces a full-automatic technique to complete partial confidentiality assessments while guaranteeing noninterference. The proposed technique can be integrated into the design phase of a service-oriented business process and help the modeler choose which tasks can be safely outsourced.

## 1 Introduction

Service-oriented computing aims at reducing complexity and costs by replacing large monolithic systems by interacting components, called *services*. Such services are offered by service providers and can be flexibly reused in service compositions. As a result, business owners can focus on their core business and outsource other tasks to (possibly cheaper) third-party service providers according to their needs. This trend has led to paradigms such as software as a service, infrastructure as a service, or platform as a service.

The service-oriented execution of a business process adds new challenges, as a business process is usually a very sensitive asset of each company. Though the interplay with third parties can be regulated by contracts, a business owner should never entirely trust other agents. Consequently, only uncritical tasks may be outsourced. To ensure *noninterference* (i. e., the absence of information leaks) in a service-oriented business process, three steps need to be taken: First, the modeler needs to assess each task whether it is confidential or public. This assessment may be straightforward given the nature of the tasks (e. g., processing financial data), but can also be arbitrary for noncritical tasks. Second, the assessment needs to be checked for information leaks. In the context of this paper, we speak of an information leak if a third party can derive confidential runtime information of the business process (e. g., the outcome of choices). Recently [1], we investigated

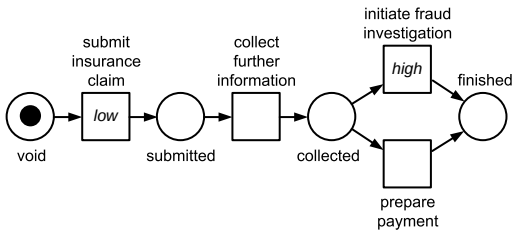



**Fig. 1.** Petri net model of the insurance business process

noninterference in terms of Petri net models and showed that modern model checking techniques [7] allow to check noninterference of industrial models in fractions of seconds. Finally, the public tasks of the business process can be delegated to third-party service providers, whereas the confidential tasks remain in the responsibility of the business process owner. Apparently, an information leak can be avoided by assessing more tasks as confidential and hence by reducing the number of outsourced tasks. This would, however, contradict the idea of service-orientation.

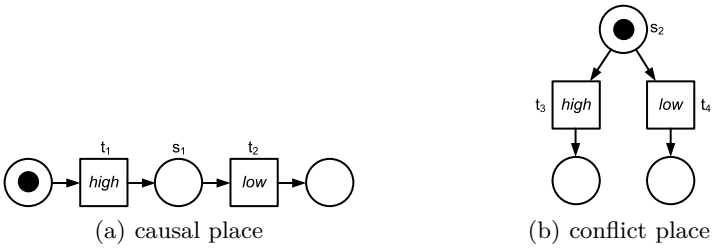
*Contribution.* The contribution of the paper is twofold. Instead of requiring a complete confidentiality assessment, we first present an approach that *completes a partial assessment while guaranteeing noninterference*. As a second contribution, we provide a *characterization of all valid assessments*. This enables the modeler to interactively assess tasks by automatically removing any invalid choices. Furthermore, a characterization of all possible assessments can be seen as a first step toward finding a cost-optimal assessment assuming given costs for each transition that cannot be outsourced.

*Organization.* The rest of this paper is organized as follows. The next section introduces the fundamental concepts of noninterference and a running example we shall use throughout the paper. Section 3 presents our completion approach and a compact representation of all noninterfering assessments. We further discusses several optimizations to avoid combinatorial explosion. In Sect. 4, we provide first experimental results using 559 industrial business process models. Section 5 concludes the paper and sketches a research agenda of future extensions.

## 2 Background

We consider the Petri net representation of business process models as a basis for the analysis. For this, mappings from common modeling languages, such as WS-BPEL, BPMN, and EPC, exist [6]. To express the confidentiality requirements, we separate the tasks — modeled by Petri net transitions — into two logical security domains: *high* for confidential and *low* for public.

The Petri net in Fig. 1 models a service-oriented insurance claim business process. After submitting the claim, further information is collected and decided whether to initiate a fraud investigation or to prepare the resulting payment before the process finishes. In this example the submitting task is public, because claims can be submitted via a Web site or a call center. The tasks can be




(a) causal place (b) conflict place

**Fig. 2.** Patterns for potential causal and conflict places $s$

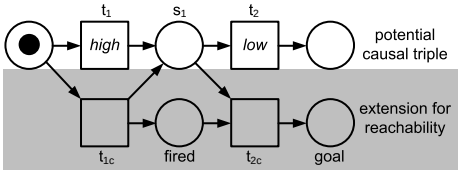



**Fig. 3.** Pattern for the reachability problem for the causal pattern from Fig. 2(a)

outsourced and the respective transition is labeled *low*. The submission process may contain no confidential data but must only be used to establish the first contact between the insurant and his insurance. The task that initiates a fraud investigation is, however, confidential yielding a *high* labeling.

An undesired leak happens whenever information meant to remain in the high domain *leaks* to the low domain. The analysis of noninterference for such Petri net models is carried out with *positive place-based noninterference* (PBNI+) [4]. PBNI+ is an approach to encode and reason about *structural noninterference* (and hence information flow control) in Petri nets. The idea is that some specific places in the net encode different noninterference properties which are leaks from the high to the low domain. In our example "collected" could be such a place, because the following decision depends on it. So in case "collect" is a *high* labeled transition, the transitions "initiate" and "prepare" should also be labeled *high*. In demonstrating the absence of such places in the net, one proves noninterferences.

Figure 2 depicts the two types of possible interference places, the causal case (a) and the conflict case (b). In the causal case, the *low* labeled transition $t_2$ can only fire after the *high* labeled transition $t_1$ has fired, so the fact that $t_1$ (and its corresponding confidential task) has fired is leaked. In the conflict case the two transitions $t_3$ and $t_4$ are mutually exclusive, which means that from firing of the *low* labeled transition $t_4$ one may deduce that the *high* labeled transition $t_3$ has not fired. Both cases can be expressed as a triple $(s, h, l)$ of a place $s$, a *high* labeled transition $h$, and a *low* labeled transition $l$. In our running example "collected" is both a causal and a conflict place and the triples are ("collected", "collect", "prepare") and ("collected", "initiate", "prepare").

A labeled Petri net is secure in terms of PBNI+ if it contains no such places. Although it appears like a structural property, the behavior of the net needs to be considered to decide PBNI+, because there must be a behavioral dependency

between the creation or consumption of the token on the place $s$ by the involved transition $h$ and $l$. This dependency can only be checked by taking the behavior of the net (i. e., its state space) into account. Based on our previous work [1], these checks can be expressed as independent reachability problems instead of an examination of the whole state space. Therefore, all checks can be done locally for each specific triple $(s, h, l)$. Figure 3 depicts the pattern for the causal case (cf. 2(a)) in which the place "goal" is interesting according to reachability. The interested reader is referred to [1].

## 3 Completion of partial confidentiality assessment

The PBNI+ check has several drawbacks: First, it requires a complete confidentiality assessment; that is, each transition has to be labeled with either *high* or *low*. This means that the modeler needs to make a manual decision for each transition whether the modeled task is confidential or public. Such choices can be very arbitrary, yet still affect overall noninterference. That said, if an information leak was detected, the assessment has to be manually corrected and re-checked.

To this end, we propose to provide a characterization of all valid confidential assessments given a partial (or even empty) confidentiality assessment. Whereas previous work [1] showed that a noninterference check is quite fast, a naive enumeration of all possible assessments has two major downsides:

1. Assuming $t$ transitions in the net, $2^t$ assessments need to be considered. Even with an average checking time of 30 milliseconds the exponential blowup makes this enumeration not applicable to industrial models with hundreds of transitions.
2. Even if we can determine the valid assessments, an explicit representation is infeasible due to the same exponential blowup. However, only a complete list of all valid assessments gives the modeler maximal freedom to come up with an optimal outsourcing plan.

The rest of this section presents reduction ideas how to tackle each mentioned problem.

### 3.1 Reducing the number of checks

Considering all possible assessments, one would end up with checking $2^t$ assignments, if a net has $t$ transitions. For each assignment more than one check (triples in terms of the reachability problem) may be necessary. Therefore it is necessary to reduce the number of checks considerably. In our running example with 4 transitions we already start with $2^4 = 16$ possible assignments. In Tab. 1 all possible 16 assignments are listed.

Based on our observation, all checks are independent from each other, so they can be executed independently [1]. In fact, this does not reduce their number, but all potential critical assignments follow from the structure of the Petri net, because for PBNI+ only potential causal and conflict places are relevant. This means, that only specific parts (the triples) of the net are interesting, which are in $\mathcal{O}(p \cdot t \cdot (t-1))$ if a net has $t$ transitions and $p$ places. Consider a potential conflict place $s$ with two transition $t_1$ and $t_2$ in its postset. Without any assignment on $t_1$ and $t_2$ there are two possible triples ($s$,$t_1$,$t_2$) and ($s$,$t_2$,$t_1$). In the first triple $t_1$ is

**Table 1.** All assignments and their necessary checks of our running example.

| Assignments | | | | Triples | | | | | Checks |
|---|---|---|---|---|---|---|---|---|---|
| **submit** | **collect** | **initiate** | **prepare** | **1** | **2** | **3** | **4** | **5** | |
| low | low | low | low | | | | | | **0** |
| low | low | low | high | | × | | | | **1** |
| low | low | high | low | × | | | | | **1** |
| low | low | high | high | | | | | | **0** |
| low | high | low | low | | | × | × | | **2** |
| low | high | low | high | | × | × | | | **2** |
| low | high | high | low | × | | | × | | **2** |
| low | high | high | high | | | | | | **0** |
| high | low | low | low | | | | | × | **1** |
| high | low | low | high | | × | | | × | **2** |
| high | low | high | low | × | | | | × | **2** |
| high | low | high | high | | | | | × | **1** |
| high | high | low | low | | | × | × | | **2** |
| high | high | low | high | | × | × | | | **2** |
| high | high | high | low | × | | | × | | **2** |
| high | high | high | high | | | | | | **0** |
| **Sum** | | | | **4** | **4** | **4** | **4** | **4** | **20** |

labeled *high* and $t_2$ is labeled *low* ($[t_1 \mapsto \mathit{high}, t_2 \mapsto \mathit{low}]$) and in the second triple it is the other way around. Both other combinations ($[t_1 \mapsto \mathit{low}, t_2 \mapsto \mathit{low}]$ and $[t_1 \mapsto \mathit{high}, t_2 \mapsto \mathit{high}]$) are not interesting according to PBNI+. Each of these two possible triples will occur in $2^{t-2}$ of all possible assignments, because of fixing the assignment of the two transitions. In our running example one can identify 5 of these triples:

1. ("collected", "initiate", "prepare"): potential conflict place "collected",
2. ("collected", "prepare", "initiate"): potential conflict place "collected",
3. ("collected", "collect", "initiate"): potential causal place "collected",
4. ("collected", "collect", "prepare"): potential causal place "collected", and
5. ("submitted", "submit", "collect"): potential causal place "submitted".

Table 1 lists all these triples (same enumeration) for all possible assignments. For instance, in line 2, where just "prepare" is assigned *high*, only the second triple needs to be checked, resulting in a single check for this assignments.

Combining these two observations it is not necessary to check all $2^t$ assignments (by performing $\mathcal{O}(2^t \cdot (p \cdot t \cdot (t-1)))$ checks), but it is enough to check only the potential critical triples which are in $\mathcal{O}(p \cdot t \cdot (t-1))$, because they are common through the net structure. Back to our running example: Each of these 5 triples occur $2^{4-2} = 4$ times over all 16 assignments yielding to the sum of 20 checks. However, it is not necessary to perform all 20 checks ($\times$ in Tab. 1), but is is sufficient to check each possible triple (columns in Tab. 1) once.

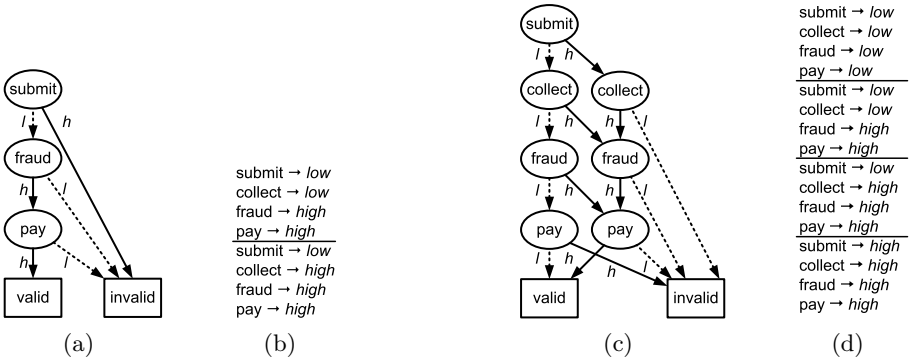



**Fig. 4.** BDD representation (a) and all valid assignments (b) of running example of Fig. 1. Without initial constraints, more assignments are possible (c, d).

In case the modeler has already assigned some confidentiality, the set of potential critical triples decrease and further triples can be ruled out. In fact our running example has two preassigned tasks ("submit" $\mapsto$ *low* and "initiate" $\mapsto$ *high*), so two triples (columns 1 and 4) are left to decide for all 16 assignments whether they are noninterfering.

To summarize, the main idea is to identify structural causal and conflict triples (columns in terms of the table) once for the net which has polynomial complexity in the net size. Afterwards perform these polynomial many checks (locally and independently) also once and represent all valid assignments in a compact way, which is the content of the following subsection.

### 3.2 Compact characterization of valid assessments

To fight the exponential blowup of the number of the valid assignments, we employ a *symbolic* representation, namely *binary decision diagrams* (BDDs) [2]. BDDs are successfully used in verification [3] as they can represent sets of bit vectors very compactly.

Figure 4(a) depicts an example of a BDD that represents all valid confidentiality assessments of the running example. The oval nodes are labeled with transition names and represent decisions whether to assess the transition as *high* (continuous outgoing arrow) or *low* (dashed outgoing arrow). After a sequence of decisions, either the node "valid" or "invalid" is reached which describes the status of the resulting assessment. Note that Fig. 4(a) does not mention the "collect" transition: This means that either label is valid for this transition, resulting in 2 valid assessments (cf. Fig. 4(b)). We can further derive that the pay task must be confidential in any case. In case no initial assessment is given (i. e., no transition is initially labeled high or low), the resulting BDD (cf. Fig. 4(c)) characterizes 2 additional valid assessments: setting all transitions to *high* or all transitions to *low* (cf. Fig. 4(d)).

The construction of the BDDs from the noninterference verification results use standard BDD operations for which efficient algorithms exist. In particular,

**Algorithm 1** Overall algorithm

**Require:** Petri net $N$
1: BDD $\leftarrow$ true
2: **for all** relevant potential causal/conflict triples $(s, h, l)$ **do**
3: create net $N_{(s,h,l)}$ and perform reachability check
4: **if** place "goal" can be marked (i.e., $s$ is an active causal/conflict place) **then**
5: BDD $\leftarrow$ BDD $\wedge \neg(h \wedge \neg l)$)
6: **end if**
7: **end for**
8: **return** BDD

the addition of further constraints (e. g., further assessments of the modeler) can be realized at modeling time and be used to guide the confidentiality assessment.

Algorithm 1 describes how a complete characterization of all valid assessments can be calculated. We begin with a BDD that assigns true (viz. "high") to all transitions. Then, we check for each potential causal and conflict triple $(s, h, l)$ whether it is an actual violation of noninterference using the reachability check sketched in Fig. 2. In case a violation is found, the respective (partial) assignment is excluded by adding the constraint $\neg(h \wedge \neg l)$ to the BDD. This excludes assignments $[h \mapsto \mathit{high}, l \mapsto \mathit{low}]$.

## 4 Experimental results

The evaluation uses a library of 559 industrial business processes from different business branches, including financial services, ERP, supply-chain, and online sales [5]. They contain no semantic information with respect to the security domains; that is, they are not labeled for security analysis. To this end, this is a good start for our approach, because we can characterize all possible confidential assessments. Table 2 summarizes their experimental results.

As summarized in Tab. 2 we only need to perform 282 checks for the biggest process (no assignments) in contrast to more than $2^{100}$ checks, which takes 3 seconds on a desktop computer. For this process, the respective BDD has 1,054 nodes.

**Table 2.** Experimental results of the 559 industrial business processes.

| | **minimum** | **average** | **maximum** |
|---|---|---|---|
| transitions (exponent of problem size) | 1 | 20 | 100 |
| causal triples (cf. Fig. 2(a)) | 3 | 34 | 242 |
| conflict triples (cf. Fig. 2(b)) | 0 | 4 | 90 |
| possible assignments (main factor for checks) | 2 | 1.048.576 | $> 10^{30}$ |
| sum of triples (necessary checks) | 3 | 38 | 282 |

## 5 Conclusion

*Summary.* Confidentiality is important in service-oriented business processes, because business processes are sensitive asset of each company. To express such confidentiality requirements one can use PBNI+, which can be verified on the fly for business processes. So the next step after the verification of a complete assessed business process is to support the modeler in 2 ways: firstly by automatically complete a partial assessed business process and, secondly, by providing a complete characterization of all valid assessments. As shown in this paper, first numbers on runtime are very promising.

*Lessons learnt.* It is possible to derive all $2^t$ assessments with only polynomial many checks. The independence shown earlier is essential for this reduction. A polynomial number of checks is feasible for industrial business processes. In order to represent all $2^t$ assessments, necessary to provide a complete support for all assessments, existing model checking techniques (BDD) are used which proved their scalability in industrial settings.

*Future work.* Future work aims at two directions: Firstly, provide some interactive design support where only possible choices are offered and obvious ones are set automatically. One way could be an integration into an existing business process modeling tool with a graphical user interface. Second, enhance the approach with costs aspects. Based on the complete representation of all valid assessments one could reason about the costs for each assessment.

*Acknowledgement.* This work was partially funded by the DFG (German research foundation) in the project WS4Dsec in the priority program Reliably Secure Software Systems (SPP 1496).

## References

1. Accorsi, R., Lehmann, A.: Automated and fast information flow analysis for business process models (2012), unpublished manuscript available at `http://www.informatik.uni-rostock.de/~al357/reader.pdf`.
2. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. Computers C-35(8), 677–691 (1986)
3. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: $10^{20}$ states and beyond. Inf. Comput. 98(2), 142–170 (1992)
4. Busi, N., Gorrieri, R.: Structural non-interference in elementary and trace nets. Mathematical Structures in Computer Science 19(6), 1065–1090 (2009)
5. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on demand: Instantaneous soundness checking of industrial business process models. Data Knowl. Eng. 70(5), 448–466 (2011)
6. Lohmann, N., Verbeek, H., Dijkman, R.M.: Petri net transformations for business processes – a survey. LNCS ToPNoC II(5460), 46–63 (2009)
7. Wolf, K.: Generating Petri net state spaces. In: ICATPN 2007. pp. 29–42. LNCS 4546, Springer (2007)