The Projected TAR and its Application to Conformance Checking

Johannes Prescher[1], Matthias Weidlich[2], Jan Mendling[1]

[1]Institute for Information Business, Wirtschaftsuniversität Wien, Austria
[johannes.prescher|jan.mendling]@wu.ac.at
[2]Technion - Israel Institute of Technology, Haifa, Israel
weidlich@tx.technion.ac.il

Abstract: Relational semantics of business process models have seen an uptake in various fields of application. As a prominent example, the Transition Adjacency Relation (TAR) has been used, for instance, to conduct conformance checking and similarity assessment. TAR is defined over the complete set of transitions of a Petri net and induces order dependencies between pairs of them. In this paper, we consider TAR in a more general setting, in which the order dependencies shall be derived only for a subset of projected transitions. We show how to derive this projected variant of TAR from the reachability graph of a reduced Petri net. We elaborate the projected TAR for conformance checking in a case from industry.

1 Introduction

Business process models capture activities of a business process and the way their execution is coordinated to achieve a certain goal [Wes07]. Process models are an important means for process improvement and process conformance. In fact, the analysis of business processes is often traced back to an analysis of process models. Conclusions drawn for process models then allow for more effective process support.

Recently, relational semantics of process models have seen a particular uptake for answering analysis questions. For instance, behavioural relations have been the basis for checking conformance between a process model and a process log [WPD+11], to assess the similarity of process models [ZWW+10], or to manage process model variants [SWMW12]. Most relational semantics capture order dependencies for pairs of actions, or transitions in Petri net terms. Such order may be grounded on direct successorship of transitions, as proposed by the Transition Adjacency Relation (TAR) [ZWW+10], also referred to as a footprint [vdA11]. TAR captures direct successorship based on all transitions. Once a business process is captured, however, only a subset of the transitions of a model may have actual business semantics, i.e., represent activities of the business process. Other transitions may be considered to be silent steps, needed purely for syntactically reasons. Still, these transitions affect the order dependencies between transitions that have business semantics. As a consequence, any analysis, conformance analysis in the setting of this paper, that is

based on the standard notion of TAR is biased by the influence of these silent transitions.

In this paper, we provide a solution to this problem by presenting a projection of TAR. Given a set of transitions of a Petri net system, it lifts the order dependencies of TAR to these transitions, neglecting transitions that are not part of the projection. Our contribution is threefold. First, we define the novel variant of TAR, called *projected TAR (pTAR)*. Second, we show how it is derived by exploiting the state space of a Petri net once existing reduction rules have been applied. As such, we provide the basis for using TAR-based techniques in a broader context. Third, we present an experimental evaluation based on an industry case.

The remainder of the paper is structured as follows. Section 2 illustrates the problem of applying TAR defined over all transitions with an example and Section 3 gives formal preliminaries. In Section 4, we define the projected TAR and elaborate on its derivation. We present experimental results on applying the projected TAR to conformance checking in Section 5. We review related work in Section 6, before Section 7 concludes the paper.

2 Background

Business process models are typically defined using conceptual modelling languages such as BPMN or EPCs. These languages tend to be well accepted by business professionals due to their intuitive representation of process semantics. As a downside though these languages are often not readily equipped with execution semantics. Therefore, formal analysis is typically conducted using the theoretical concepts of Petri nets. One specific instance of such analysis is conformance checking. This type of analysis is concerned with the problem to both a) quantify the degree of execution conformance of an execution log with the behaviour as defined by the process model and to b) point to those parts of the model which are violated by the cases.

On a technical level, the application of Petri net concepts requires the mapping from, for instance, BPMN as a conceptual model to a corresponding Petri net. In general, this mapping can be performed for each BPMN construct in an automated way yielding semantically equivalent Petri net components. In their work [DDO08], Dijkman et al. define such a mapping for a subset of BPMN constructs. Although basic elements such as activities can be easily mapped, more advanced constructs such as error subprocesses can result in Petri net components of considerable complexity. Further, the mapping of many BPMN constructs produces silent transitions which will not be observable in the execution log of the process. As an an example, consider the AND-split of BPMN. It is typically translated using a silent transition that produces tokens of each place representing the start of the branches diverging from the AND-split. Such transitions on the Petri net level do not have business semantics. More complicated components with silent transitions stem from, for instance, exception handling in BPMN.

To illustrate this mapping, consider the BPMN process model shown in Figure 1. It represents the way in which IT Service Management is performed at a center of an IT service provider. The process starts with the creation of an issue. Subsequently, we observe two parallel branches. The first path contains the customer extension which will be executed

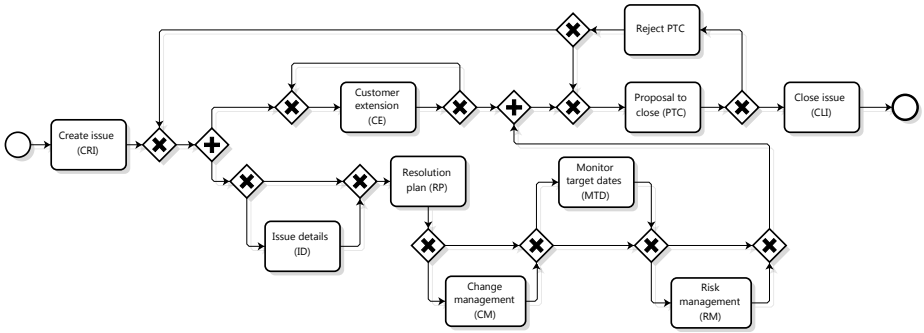


Figure 1: BPMN model of the SIMP process

at least once, optionally multiple times. The second path contains several optional activities: issue details may be updated, a resolution plan may be created, change management activities may be executed, target dates may be monitored and risk management tasks may be documented. After these optional activities, the parallel paths are synchronised and a proposal to close the issue has to be filed. If the issue is resolved, it may be closed. Otherwise, the proposal to close has to be rejected. In case of rejection, either a new proposal to close has to be stated or the process starts again before the two parallel paths are introduced (except the creation of an issue).

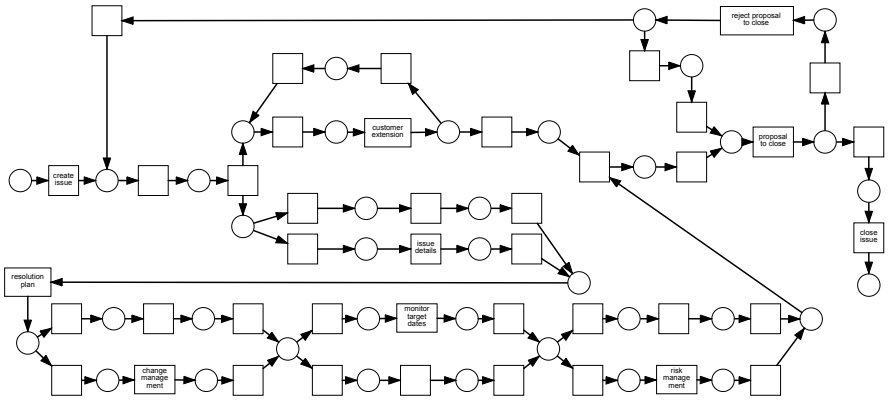


Figure 2: Petri net model of the SIMP process

Using the mapping rules defined in [DDO08], we can construct a Petri net that exactly captures the behaviour represented in the BPMN model. The net is depicted in Figure 2. We observe the following. First, each activity of the BPMN process is represented as a single transition. Second, each gateway of the BPMN model is represented by a component of silent transitions and places which represent its semantics. For instance, the exclusive-OR gateway prior to *issue details* is represented by two silent transitions which either activate the *issue details* path or the silent path. The transitions are silent, as they represent an implicit decision which is, in case of BPMN, not modelled explicitly.

There is a problem if we aim at using the constructed Petri net for analysis that builds on the Transition Adjacency Relation (TAR). TAR defines the set of ordered pairs of transitions (t_1, t_2) that can be executed one after another. Two transitions are part of this relation if there is an execution sequence of the process model, in which t_1 is directly followed by t_2. If we would simply calculate the TAR relation for the Petri net depicted in Figure 2, we would have to define unique identifiers for each of the silent transitions. The TAR relation would then represent, among others, that *resolution plan* can be followed by a silent transition, and that this same silent transition can lead to *change management*. This is problematic in case we want to analyse execution logs, because the option to execute *resolution plan* first and then directly next *change management* is not visible in the TAR relation. Hence, it is not simply possible to neglect TAR relations that refer to silent transitions. On the other hand, it is also not possible to derive new TAR relations in a transitive way. The idea here would be to conclude on (t_1, t_3) in TAR if we observe (t_1, t_2) and (t_2, t_3) in TAR. Such a derivation rule, however, requires assumptions on the absence of behavioural anomalies and does not work for short loops and non-free-choice constructs.

Accordingly, we need a technique to derive the notion of a projected TAR (pTAR). The calculation of the pTAR has to take into account that we are interested only in a particular subset of transitions (projection set). We want to characterise the traces of the Petri net as if occurrences of transitions that are not in the projection set would have been deleted.

3 Preliminaries

We first clarify notions and notations for Petri net systems. Second, we present the existing definition of the Transition Adjacency Relation (TAR).

3.1 Petri Net Systems

Petri nets, in particular classes such as workflow nets [vdA98], are often used to capture process models. We mentioned earlier that many process description languages, such as BPMN and EPCs, may be at least partly be transformed to Petri net systems, cf. [LVD09].

Definition 1 (Net)**.** A *net* is a tuple $N = (P, T, F)$ with P and T as finite disjoint sets of places and transitions, and a flow relation $F \subseteq (P \times T) \cup (T \times P)$.

We write $X = P \cup T$ for all nodes. For $x \in X$, $\bullet x := \{y \in X \mid (y, x) \in F\}$ is the pre-set, $x\bullet := \{y \in X \mid (x, y) \in F\}$ is the post-set, and $\bullet(x\bullet) := \{z \in X \mid y \in X \wedge\ (x, y) \in F \wedge (z, y) \in F\}$.

To define semantics, we need notations for sequences. A *sequence* over a set S of length $n \in \mathbb{N}$ is a function $\sigma : \{1, \ldots, n\} \to S$. If $\sigma(i) = s_i$ for $i \in \{1, \ldots, n\}$, we write $\sigma = \langle s_1, \ldots, s_n \rangle$. The set of all finite sequences over S is denoted by S^*.

Let $N = (P, T, F)$ be a net. $M : P \mapsto \mathbb{N}$ is a *marking* of N, $\mathbb{M}$ denotes all markings of N.

$M(p)$ returns the number of *tokens* in place p. For any transition $t \in T$ and any marking $M \in \mathbb{M}$, t is *enabled* in M, denoted by $(N, M)[t\rangle$, iff $\forall\, p \in {\bullet}t\ [\ M(p) \geq 1\ ]$. Further, we identify the flow relation F with its characteristic function on the set $(P \times T) \cup (T \times P)$. Then, marking M' is reachable from M by *firing* of t, denoted by $(N, M)[t\rangle(N, M')$, such that $M'(p) = M(p) - F(p,t) + F(t,p)$, $p \in P$, i.e., one token is taken from each input place of t and one token is added to each output place of t.

A sequence of transitions $\sigma = \langle t_1, \ldots, t_n \rangle$, $n \in \mathbb{N}$, is a *firing sequence*, iff there exist markings $M_0, \ldots, M_n \in \mathbb{M}$, such that for all $i \in \mathbb{N}$, $1 \leq i \leq n$ it holds $(N, M_{i-1})[t_i\rangle(N, M_i)$. We say that σ is enabled in M_0, denoted by $(N, M_0)[\sigma\rangle$. For any two markings $M, M' \in \mathbb{M}$, M' is reachable from M in N, denoted by $M' \in [N, M\rangle$, if there exists a firing sequence σ leading from M to M'. Firing of σ in M is denoted by $(N, M)[\sigma\rangle(N, M')$. A *net system*, or *system*, is a pair $S = (N, M_i)$, where N is a net and M_i is the *initial marking* of N.

3.2 Transition Adjacency Relation

The Transition Adjacency Relation captures behavioural characteristics of a net system by means of an ordering relation defined over the Cartesian product of transitions [ZWW+10]. It captures direct succession of two transitions in some firing sequence of a Petri net system.

Definition 2 (TAR)**.** Let $S = (N, M_i)$ be a system with $N = (P, T, F)$. The *TAR* $> \subseteq T \times T$ contains a pair (x, y), iff there exists a firing sequence σ with $(N, M_i)[\sigma\rangle$ such that $\sigma(i) = x$ and $\sigma(i+1) = y$ for some $1 \leq i$.

Note that, by definition, TAR and the inverse relation $>^{-1} = \{(x, y) \mid (y, x) \in >\}$ partition the Cartesian product of transitions.

4 Conformance Checking with the Projected TAR

In this section, we present the notion of the *projected TAR (pTAR)*. It captures behavioural characteristics while projecting transitions that are given as a projection set. We first define the pTAR. Then, we present a derivation algorithm which uses reduction rules and state space search. Finally, we apply pTAR for conformance checking.

4.1 The Projected TAR

The projected TAR defines the set of transition pairs that follow each other directly in some projected firing sequence of the net system.

Definition 3 (pTAR)**.** Let $S = (N, M_i)$ be a system with $N = (P, T, F)$. Let $T' \subseteq T$ be a set of transitions called projection set. The *projected TAR* induced by T', $>_{T'} \subseteq T' \times T'$ contains a pair (x, y), iff there exists a firing sequence σ with $(N, M_i)[\sigma\rangle$, such that

$\sigma(i) = x, \sigma(j) = y$ for some $1 \leq i < j$ and $\sigma(k) \notin T'$ for all $i < k < j$.

Considering only the Cartesian product of transitions in the projection set, we see that pTAR actually extends TAR. That is, for transitions in the projection set additional successorships may be identified.

Property 1. For a system $S = (N, M_i)$ with $N = (P, T, F)$ and $T' \subseteq T$ holds $(> \cap (T' \times T')) \subseteq >_{T'}$.

The property follows directly from the definition of the relations. Every pair of transitions in the projection set that is part of TAR is, by definition, also part of pTAR.

4.2 Derivation

We approach the derivation of pTAR in two stages. First, we reduce the original Petri net, then we identify the pTAR using the state space techniques. Figure 3 illustrates the set of reduction rules we consider. They were adapted in [vdADO$^+$08] from the liveness and boundedness preserving reduction rules by Murata [Mur89]. The rules eliminate transitions that are not included in the projection set as follows.

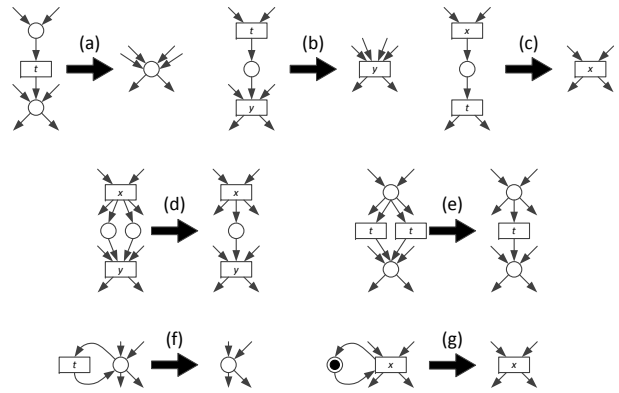


Figure 3: Illustration of reduction rules, t is not part of the projection set, as adapted in [vdADO$^+$08].

Definition 4 (Reduction Rules)**.** Let $S = (N, M_i)$ be a system with $N = (P, T, F)$. Let $T' \subseteq T$ be a projection set. For system S, a reduced system is derived by one of the following rules:

(a) If there exists $p \in P$ and $t \notin T'$ such that $p\bullet = \{t\}$ and $\bullet t = \{p\}$, then $S_a = (N_a, M_i)$ with $N_a = (P_a, T_a, F_a)$ such that
- $P_a = P \setminus \{p\}$,
- $T_a = T \setminus \{t\}$,
- $F_a = F \cup \{(n_1, n_2) | n_1 \in \bullet p \wedge n_2 \in t\bullet\} \setminus \{(n_1, n_2) | n_1 \in \{p, t\} \vee n_2 \in \{p, t\}\}$.

(b) If there exists $p \in P$, $t \notin T'$ and $y \in T$ such that $\bullet p = \{t\}$ and $p\bullet = \{y\}$, then $S_b = (N_b, M_i)$ with $N_b = (P_b, T_b, F_b)$ such that

- $P_b = P \setminus \{p\}$,
- $T_b = T \setminus \{t\}$,
- $F_b = F \cup \{(n_1, y) | n_1 \in \bullet t\} \setminus \{(n_1, n_2) | n_1 \in \{p, t\} \vee n_2 \in \{p, t\}\}$.

(c) If there exists $p \in P$, $t \notin T'$ and $x \in T$ such that $\bullet p = \{x\}$,$\bullet t = \{p\}$ and $p\bullet = \{t\}$, then $S_c = (N_c, M_i)$ with $N_c = (P_c, T_c, F_c)$ such that

- $P_c = P \setminus \{p\}$,
- $T_c = T \setminus \{t\}$,
- $F_c = F \cup \{(x, n_2) | n_2 \in t\bullet\} \setminus \{(n_1, n_2) | n_1 \in \{p, t\} \vee n_2 \in \{p, t\}\}$.

(d) If there exists $p_1, p_2 \in P$ such that $\bullet p_1 = \bullet p_2$ and $p_1\bullet = p_2\bullet$, then $S_d = (N_d, M_i)$ with $N_d = (P_d, T_d, F_d)$ such that

- $P_d = P \setminus \{p_2\}$,
- $T_d = T$,
- $F_d = F \setminus \{(n_1, n_2) | n_1 = p_2 \vee n_2 = p_2\}$.

(e) If there exists $t_1, t_2 \notin T'$ such that $\bullet t_1 = \bullet t_2$ and $t_1\bullet = t_2\bullet$, then $S_e = (N_e, M_i)$ with $N_e = (P_e, T_e, F_e)$ such that

- $P_e = P$,
- $T_e = T \setminus \{t_2\}$,
- $F_e = F \setminus \{(n_1, n_2) | n_1 = t_2 \vee n_2 = t_2\}$.

(f) If there exists $t \notin T'$ such that $\bullet t = t\bullet$, then $S_e = (N_f, M_i)$ with $N_f = (P_f, T_f, F_f)$ such that

- $P_f = P$,
- $T_f = T \setminus \{t\}$,
- $F_f = F \setminus \{(n_1, n_2) | n_1 = t \vee n_2 = t\}$.

(g) If there exists $p \in P$ such that $M_i > [i]$ and $\bullet p = p\bullet$, then $S_g = (N_g, M_i)$ with $N_f = (P_g, T_g, F_g)$ such that

- $P_g = P \setminus \{p\}$,
- $T_g = T$,
- $F_g = F \setminus \{(n_1, n_2) | n_1 = p \vee n_2 = p\}$.

Once the system is reduced according to these rules, we can calculate the projected TAR based on the reachability graph as outlined in Algorithm 1.

The algorithm uses the auxiliary data structure $TxM \subseteq T \times \mathbb{M}$ to keep track of which transition $t \in T$ led to a certain marking $M \in \mathbb{M}$. Once such pair has been investigated it is added to the structure $vTxM \subseteq T \times \mathbb{M}$ representing visited pairs. We initialise the algorithm with all markings reachable from the initial marking by firing a single transition (line 2). Then, all pairs of transitions and markings are evaluated (lines 3 to 19). A pair is selected and the structures TxM and $vTxM$ are updated (lines 4 to 6). Then, for each transition that is enabled in the respective marking, we check whether it is part of

Algorithm 1: Derivation of projected TAR based on Reachability Graph

Input: $S = (N, M_i)$, a net system with $N = (P, T, F)$, $T' \subseteq T$, a projection set.
Output: $>_{T'}$, the projected TAR of S induced by T'.

1 $>_{T'}, TxM, vTxM \longleftarrow \emptyset$;
2 **foreach** $t \in T$ *with* $(N, M_i)[t\rangle$ **do** $TxM \longleftarrow TxM \cup \{(t, M)\}$ with $(N, M_i)[t\rangle(N, M)$;
3 **while** $TxM \neq \emptyset$ **do**
4 select $(t, M) \in TxM$;
5 $TxM \longleftarrow TxM \setminus \{(t, M)\}$;
6 $vTxM \longleftarrow vTxM \cup \{(t, M)\}$;
7 $T_e \longleftarrow \{t \in T \mid (N, M)[t\rangle\}$;
8 **foreach** $t_e \in T_e$ **do**
9 **if** $t_e \in T'$ **then**
10 **if** $t \in T'$ **then** $>_{T'} \longleftarrow >_{T'} \cup \{(t, t_e)\}$;
11 **if** $(t_e, M') \notin vTxM$ *with* $(N, M)[t_e\rangle(N, M')$ **then**
12 $TxM \longleftarrow TxM \cup \{(t_e, M')\}$ with $(N, M)[t_e\rangle(N, M')$;
13 **end**
14 **end**
15 **else if** $(t, M') \notin vTxM$ *with* $(N, M)[t_e\rangle(N, M')$ **then**
16 $TxM \longleftarrow TxM \cup \{(t, M')\}$ with $(N, M)[t_e\rangle(N, M')$;
17 **end**
18 **end**
19 **end**

the projection set (line 9). If so, the projected TAR relation may be updated (line 10) and we proceed by adding new pairs of transitions and reachable markings to TxM for investigation (line 12). If not, then we add the pair comprising the original and the marking reached by firing the transition that is not part of the projection set to structure TxM (line 16). Intuitively, this captures the fact that the marking may be reached by firing the original transition and a sequence of silent transitions, i.e., transitions that are not in the projection set. Further, whenever TxM is updated, we need to check whether a pair has been processed already to ensure termination of the algorithm.

4.3 Checking pTAR Conformance

Having introduced the pTAR, we turn to its application for conformance checking. For a transition t of a given net system, we define its conformance set and its violation set. The projected TAR defines the set of permissible successors of a transition, providing the basis for defining both these sets. Intuitively, the conformance set of t comprises all transitions that are allowed to occur in some observable execution sequence. In contract, the violation set contains the transitions that are not allowed to directly succeed t. Formally, we define:

Definition 5 (Conformance Set and Violation Set)**.** Let $S = (N, M_i)$ be a system with $N = (P, T, F)$. Let $T' \subseteq T$ be a projection set and $pTAR$ its projected TAR.

- The *conformance set* for $t \in T$ is defined as $conf(t, pTAR) = \{x | (t, x) \in pTAR\}$.
- The *violation set* for $t \in T$ is defined as $viol(t, pTAR) = T' \setminus conf(t, pTAR)$.

Assume that we are given a net system $S = (N, M_i)$, $N = (P, T, F)$, such that the projection set $T' \subseteq T$ comprises all transitions that have a business meaning. Hence, those transitions are expected to occur in the execution log that captures the observed behaviour of the business process. Formally, such an observed execution sequence is a finite sequence $\sigma \in T'^*$ over the transitions in the projection set.

To detect deviations of σ from the behaviour as defined in S, we proceed as follows. For the observed execution sequence $\sigma = \langle t_1, \ldots, t_n \rangle$, let t_i be a transition with $1 \leq i < n$. Then, transition t_{i+1} succeeding t_i in σ is either in the conformance set $conf(t_i, pTAR)$ or in the violation set $viol(t_i, pTAR)$ of t_i. The conformance according to pTAR is achieved once all considered succeeding transitions are in the respective conformance sets.

Definition 6 (pTAR Conformance)**.** Let $S = (N, M_i)$ be a system with $N = (P, T, F)$. Let $T' \subseteq T$ be a projection set and $pTAR$ its projected TAR. An observed execution sequence $\sigma = \langle t_1, \ldots, t_n \rangle \in T'^*$ is valid according to *pTAR conformance* iff for all $1 \leq i < n$ it holds that $t_{i+1} \in conf(t_i, pTAR)$.

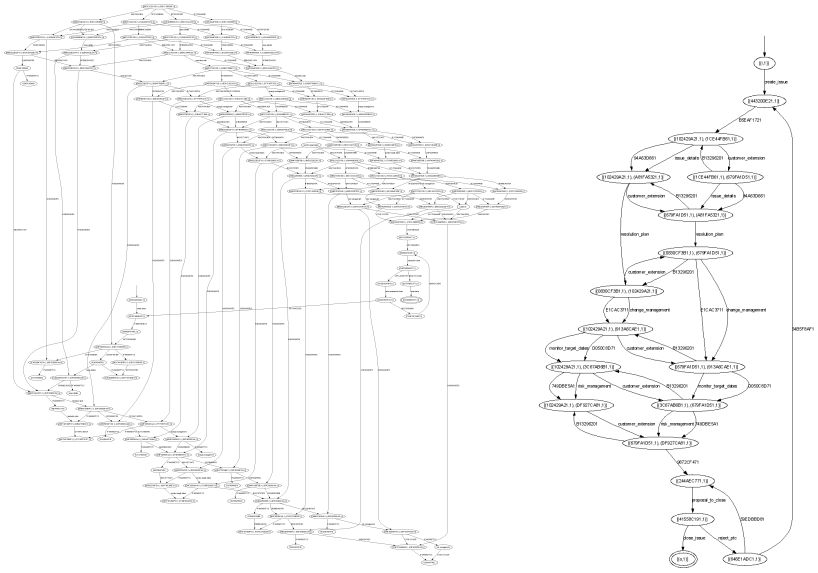
Once an observed execution sequence is not valid according to pTAR conformance, all non-empty violation sets of transitions of this sequence hint at the behavioural deviations.

5 Evaluation

In this section, we apply our conformance checking technique in an industry case. The data of this case relates to a Security Incident Management Process (SIMP). The process model and the corresponding execution sequences are taken from [WPDM10]. The process and its execution log were captured on-site at an IT service providers centre during five years. Overall, the study contains 852 cases of process execution for the SIMP process. First, we will investigate the effect on the state space of applying reduction rules. Then, we present the results of conformance analysis.

5.1 The Effect of Petri Net Reduction

The original Petri net shown in Figure 2 in Section 2 includes several silent transitions that do not carry any business semantics, i.e., they do not represent business activities. We reduce the net by considering all transitions that represent business activities as part of the projection set and apply the presented reduction rules. While the original net contains 33 silent transitions, the resulting net is reduced and shows solely nine silent transitions. Table 1 lists measures which allow for the comparison of both Petri nets. Altogether, the size of the Petri net shrinks from 43 to 19 transitions, 38 to 14 places and 88 to 40 arcs. Relatively speaking, the amount of the nets components decreases by more than 50%.



(a) (b)

Figure 4: Reachability graph (a) of original Petri net and (b) of reduced Petri net

The significance of this reduction relates to the state explosion problem of Petri nets with concurrency [Val96]. As our projected TAR calculation relies on the reachability graph, it is important to note that the number of transitions and states decreases by more than 85% (see Figure 5.1). This decrease leads to a smaller state space and a more efficient calculation of conformance sets.

Table 1: Reduction of Petri nets complexity

Category	Original Net	Reduced Net
Petri net		
— transitions	43	19
— places	38	14
— arcs	88	40
Transition System		
— transitions	248	38
— states	121	18

5.2 Conformance Results

Turning to the conformance checking, we derive 68 distinct transition pairs representing undesirable sequences according to the violation sets of the respective transitions. The

sequencing of these pairs was violated 1453 times in the overall amount of 852 different cases of the SIMP process. Successorship according to TAR and pTAR does not only relate to distinct transitions, but may relate a transition to itself. Nine out of the 68 pairs relate to such self-relations, such that these transitions must not be a direct successor of themselves. Within our study these nine pairs (13.2%) relate to 1205 violations (82.9%), whereas the remaining 59 pairs (86.8%) relate to only 248 violations (17.1%).

Figure 5 illustrates the results. For each transition, we state the absolute amount of violations (V) related to this transition, and its relative share with respect to all detected violations. In general, a high amount of violations for a specific transition indicates that the execution of this activity and its context is worth to be investigated in detail. Note that the point of violation does not allow for any implications regarding the current state of the process. A violation might stem from progressing with non permissible other transitions or from forbidden multiple executions. For the later case, it is striking that the transition *proposal to close* relates to 466 violations out of which 403 were forbidden multiple executions.

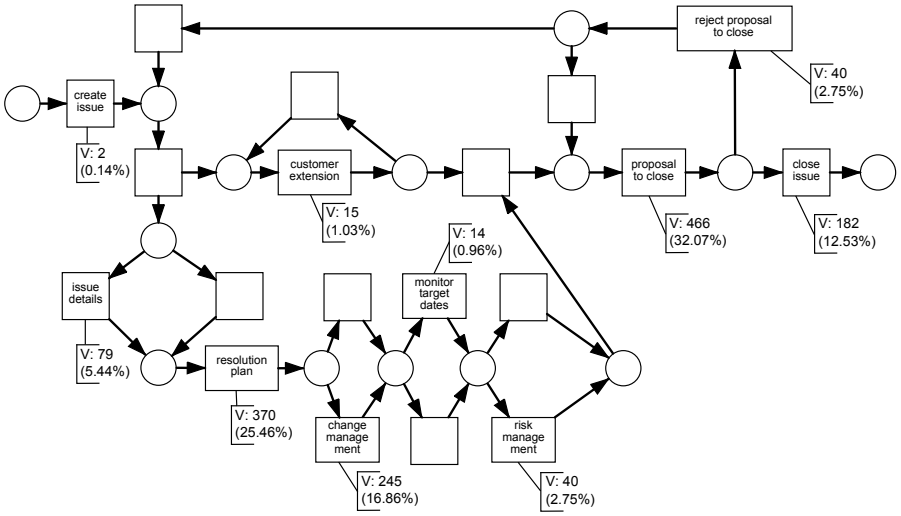


Figure 5: Reduced Petri net system of the SIMP

Altogether, we conclude that the projected TAR helps to discover activities that frequently relate to violations.

5.3 Results in Comparison to Existing Techniques

Albeit reduced, the net system for our case as visualised in Figure 5 still contains a several silent transitions. As such, application of the TAR to detect deviations between model and observed execution sequences will be biased as discussed in Section 2.

However, a similar yet different approach to conformance checking based on behavioural

relations has been proposed in [WPD+11]. It relies on the notion of behavioural profiles that do not capture direct successorship of transitions, but an indirect successorship. As such, they define a base relation that captures whether a transitions is *eventually* followed by another transition in some firing sequence of the net system, which avoids the problems induced by silent transitions. On the downside, this comes at the cost of lost precision. Relying on indirect successorship, e.g., means that transitions in cycles appear to be unordered since they may follow each other eventually in either order.

Table 2 provides further insights in this aspect by comparing the violations detected using the relations of the behavioural profile with the violations detected using pTAR. Here, the relative values are based on the joint result of both approaches. The provided ratio is supposed to visualise the effectiveness of the approaches.

Table 2: Detected violations for the case using different behavioural relations

Approach	Pairs with Violations	Violations	Violations/Pairs
BP	19 (21.84%)	201 (12.15%)	10.58
projected TAR	68 (78.16%)	1453 (87.85%)	21.37
$\sum$(BP + pTAR)	87	1654	19.01

According to Table 2 the technique presented in this paper is able to detect a significantly higher amount of process violations. A detailed look at the data of the case reveals the major reason for the observed deviations: 1205 out of 1453 violations are detected because our technique checks forbidden repetition of single transitions. Behavioural profiles, in turn, allow only for assessing whether a transition may occur only once or may be repeated. However, they lack the ability to check whether it single transitions may be repeated directly, without the occurrence of any other transition in between.

6 Related Work

The research presented in this paper is relates to the derivation of behavioural relations from process models and to conformance checking. Several sets of relations have been defined for capturing the behaviour of process models. The causality, conflict and concurrency relation have been proposed for Petri net systems based on unfoldings [McM95, ERV02]. The α-relations originally defined for mining processes [vdAWM04] have been adapted for process models, yielding the TAR [ZWW+10] that is the starting point for our work. The behavioural profile and an efficient calculation for sound free-choice workflow nets is presented in [WMW11], and extended with a causality relation in [WPMW11]. All these relations can be calculated at varying degrees of complexity. Behavioural profiles can be determined in cubic time for certain net classes. Here, we use a state space technique to determine the projected TAR. Unfolding techniques might be applicable to improve performance.

Several approaches have been defined for conformance checking. Rozinat and van der Aalst

introduce a fitness measure which builds on a state-based replay of execution sequences from a log [RvdA08]. The concept of a violation set shares some characteristics of negative events as discussed in [GDWM+11]. Earlier we mentioned that the relations of the behavioural profile may also be used for conformance checking [WPD+11]. This approach has been extended towards monitoring in [WZM+11]. We discussed that a downside of behavioural profiles is that they represent a behavioural abstraction, which has major implications for cyclic structures in particular. In contrast, our approach enables the monitoring of behaviour while relying on behavioural relations to precisely capture any behavioural deviation.

7 Conclusion

In this paper, we have presented an approach for conformance checking based on the projected TAR relation. Our contribution is the definition of the projected TAR and its calculation based on efficient reduction rules and the state space. We applied the technique for a service management process demonstrating its applicability. The advantage of our novel technique is a combination of an efficient representation of behaviour in terms of the projected TAR and higher precision in comparison to existing approaches.

In future research, we aim to improve the theoretical complexity of the calculation of the projected TAR. Several concepts including the process structure tree [VVK09] and the efficient calculation of the concurrency relation [Esp04] will be helpful to this end. We also aim to conduct further industry evaluations in the service management domain. The characteristics of this domain (process models available but not enforced, cases documented in ticketing systems) are perfect to challenge conformance checking techniques.

References

[DDO08] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Inf. & Software Techn.*, 50(12):1281–1294, 2008.

[ERV02] Javier Esparza, Stefan Römer, and Walter Vogler. An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.

[Esp04] Javier Esparza. A Polynomial-Time Algorithm for Checking Consistency of Free-Choice Signal Transition Graphs. *Fundam. Inform.*, 62(2):197–220, 2004.

[GDWM+11] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens. Process discovery in event logs: An application in the telecom industry. *Applied Soft Computing*, 11(2):1697–1710, 2011.

[LVD09] Niels Lohmann, Eric Verbeek, and Remco M. Dijkman. Petri Net Transformations for Business Processes - A Survey. *T. Petri Nets and Other Models of Concurrency*, 2:46–63, 2009.

[McM95] Kenneth L. McMillan. A Technique of State Space Search Based on Unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[RvdA08] Anne Rozinat and Wil M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008.

[SWMW12] Sergey Smirnov, Matthias Weidlich, Jan Mendling, and Mathias Weske. Action patterns in business process model repositories. *Computers in Industry*, 63(2):98–111, 2012.

[Val96] Antti Valmari. The State Explosion Problem. In *Petri Nets*, LNCS 1491, pages 429–528. 1996.

[vdA98] Wil M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[vdA11] Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[vdADO+08] Wil M. P. van der Aalst, Marlon Dumas, Chun Ouyang, Anne Rozinat, and Eric Verbeek. Conformance checking of service behavior. *ACM Trans. Internet Techn.*, 8(3), 2008.

[vdAWM04] Wil M. P. van der Aalst, A.J.M.M. Weijters, and Laura Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[VVK09] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. *Data Knowl. Eng.*, 68(9):793–818, 2009.

[Wes07] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[WMW11] Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. *IEEE Trans. Software Eng.*, 37(3):410–429, 2011.

[WPD+11] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, Jan Mendling, and Mathias Weske. Process compliance analysis based on behavioural profiles. *Inf. Syst.*, 36(7):1009–1025, 2011.

[WPDM10] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, and Jan Mendling. Process Compliance Measurement Based on Behavioural Profiles. In *CAiSE*, LNCS 6051, pages 499–514. 2010.

[WPMW11] Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, and Mathias Weske. Causal Behavioural Profiles - Efficient Computation, Applications, and Evaluation. *Fundam. Inform.*, 113(3-4):399–435, 2011.

[WZM+11] Matthias Weidlich, Holger Ziekow, Jan Mendling, Oliver Günther, Mathias Weske, and Nirmit Desai. Event-Based Monitoring of Process Execution Violations. In *BPM 2011*, LNCS 6896, pages 182–198. 2011.

[ZWW+10] Haiping Zha, Jianmin Wang, Lijie Wen, Chaokun Wang, and Jiaguang Sun. A workflow net similarity measure based on transition adjacency relations. *Computers in Industry*, 61(5):463–471, 2010.