Petrifying operating guidelines for services

Petrifying Operating Guidelines for Services

Niels Lohmann and Karsten Wolf
Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
{niels.lohmann, karsten.wolf}@uni-rostock.de

Abstract

Operating guidelines characterize correct interaction (e.g. deadlock freedom) with a service. They can be stored in a service registry. So far, they have been represented as an annotated transition system.

For the sake of saving space in the registry, we want to translate operating guidelines into Petri nets. To make this possible, we carefully investigate regularities in the annotations.

1. Introduction

Services (e.g. Web services) are made for loosely coupled interaction [1]. Several aspects are crucial for correct interaction: semantics (compatible interpretation of the meaning of exchanged data), behavior (compatible order of exchanged data), and non-functional (compatible process of exchanging data, including policies, quality of service, etc.).

We contribute to the behavioral aspect of service interaction. In this realm, the concept of *operating guidelines* [2] plays a key role. An operating guideline OG_P of a service P is a finite characterization of the (possibly infinite) set of all those services R that have a compatible behavior w.r.t. P. In this paper, we use a compatibility notion that consists of deadlock freedom of the composed system $P \oplus R$ and a given limit for the number of pending messages in a communication channel.

Tasks like checking compatibility can be performed more easily with OG_P than with P. Such a compatibility check must be performed by a service broker who assigns a fitting (previously published) service P to a query for another service R. Other interesting applications of operating guidelines range from checking substitutability (can every user of the old service use the new service, too?) [3], [4] to scenarios of test case generation [5].

Structurally, operating guidelines exploit the fact that the set $Strat(P)$ of compatible partners (*strategies*) of a service P contains most permissive ones. A most permissive partner R^* can simulate every other partner, i.e. is a top element in the simulation preorder in $Strat(P)$. To characterize *all* partners, a most permissive partner is enhanced with Boolean annotations, one for each state. Their propositions correspond to outgoing edges. The formulae control how the behavior may be restricted without destroying compatibility.

Until now, we represented an operating guideline explicitly, as a transition system (of the used most permissive partner). Attempts for symbolic representation using BDD [6] brought some but not sufficient success [7]. It was thus natural to try to transform the transition system into a Petri net, particularly in the face of frequently occurring diamond structures in the transition system. However, the mentioned formulae are attached to states and cannot be naturally assigned to Petri net places or transitions. We introduce the concept of operating guidelines in Sect. 2

In essence, our new approach relies on an implicit representation of the attached formulae. Exploiting regularities in the formulae, we show that they can be reconstructed from two sets of states. This way, the space complexity of an operating guideline is reduced from $|P||R^*|$ to $|R^*|$ where $|X|$ is the number of states and edges of transition system X. This result is presented in Sect. 3.

In Sect. 4, we show that the matching problem (checking that a given service is represented by a given operating guideline) can be executed using the new representation.

The simpler representation of the attached formulae enables us to condense the state space into a Petri net. For this task, we use existing theory [8], [9] and an existing tool, Petrify [10]. Section 5 briefly explains our approach. In Sect. 6, we discuss the representation of the two sets of states (or markings) in the context of an Petri net representation of a most permissive partner.

We evaluated the Petri net representation of the state space and the new representation of the formulae in a case study. It gives experimental evidence of the significant space reduction that can be obtained and is summarized in Sect. 7. In Sect. 8, we discuss ideas for a further optimization of the Petri net output.

2. Services and operating guidelines

In this section, we present the background of our approach.

2.1. Services

The behavior of services can be specified in various formalisms, including industrial languages like WS-BPEL [11] or BPMN [12], semiformal languages like UML activity diagrams [13], or formal models like Petri nets, process algebras or state machines (automata). Existing translations

1550-4808/09 $25.00 © 2009 IEEE
DOI 10.1109/ACSD.2009.11

between WS-BPEL and Petri nets in both directions [14], [15] show that formal models are capable of expressing the relevant behavioral features of services as understood by industry.

We model the behaviour of a service as an automaton. Communication actions (sending or receiving messages) of the service are attached to the transitions. For internal (non-communicating) transitions, we attach the symbol τ.

Definition 1 (Service automaton)**.** A *service automaton* A consists of an input alphabet I, an output alphabet O such that $I \cap O = \emptyset$ and $\tau \notin (I \cup O)$, a finite set of states Q with an initial state $q_0 \in Q$ and a set $\Omega \subseteq Q$ of final states, and a transition relation $\delta \subseteq Q \times (I \cup O \cup \{\tau\}) \times Q$ such that $q \in \Omega$ and $[q, x, q'] \in \delta$ implies $x \in I$. We write $q\delta q'$ if there is an x such that $[q, x, q'] \in \delta$. Define $en(q) := \{x \mid \exists q' : [q, x, q'] \in \delta\}$. A service automaton is *deterministic* iff, for all q, x, q', q'', $[q, x, q'] \in \delta$ and $[q, x, q''] \in \delta$ implies $q' = q''$. A service automaton is *responsive* iff, for each state q, a state q' is reachable which is final or enables a transition with a label other than τ.

In the sequel, we consider only responsive service automata which does not restrict practical applicability as non-responsive behaviour (non-communicating activities forever) is very unusual for services.

An element $[q, x, q'] \in \delta$ is referred to as a transition leaving q, arriving at q', and labeled x. In contrast to *I/O automata* [16], we assume an asynchronous model of message passing. Furthermore, messages can overtake each other and are not queued on the receiver as, for instance, in the case of *communicating finite-state machines* [17]. The semantics of message passing is implicitly defined in the following definition of composition of services. For this purpose, we use the concept of multisets $M : I \cup O \to \mathbb{N}$. We use list notations for describing multisets (e.g., $[a, a, b]$ is the multiset with $[a, a, b](a) = 2$, $[a, a, b](b) = 1$, and $[a, a, b](x) = 0$, for all other x). $a \in M$ is true if $M(a) > 0$, and the binary operations $+$ and $-$ are performed elementwise. Denote $Bags(I \cup O)$ the set of all multisets M over I and O. The result of composing two services is a transition system.

Definition 2 (Composition)**.** Service automata A and B are *composable* iff $I_A = O_B$ and $I_B = O_A$. The *composed system* $A \oplus B$ is a transition system consisting of the set of states $Q_A \times Bags(I_A \cup O_A) \times Q_B$, the initial state $[q_A, [\,], q_B]$, a set of final states $\Omega_A \times \{[\,]\} \times \Omega_B$ and the following transitions:

- (internal move in A:) $[[q_A, M, q_B], [q'_A, M, q_B]]$, if $[q_A, \tau, q'_A] \in \delta_A$,
- (internal move in B:) $[[q_A, M, q_B], [q_A, M, q'_B]]$, if $[q_B, \tau, q'_B] \in \delta_B$,
- (send by A:) $[[q_A, M, q_B], [q'_A, M + [c], q_B]]$, if $[q_A, c, q'_A] \in \delta_A$ and $c \in O_A$,
- (send by B:) $[[q_A, M, q_B], [q_A, M + [c], q'_B]]$, if $[q_B, c, q'_B] \in \delta_B$ and $c \in O_B$,
- (receive by A:) $[[q_A, M, q_B], [q'_A, M - [c], q_B]]$, if $[q_A, c, q'_A] \in \delta_A$, $c \in M$, and $c \in I_A$,
- (receive by B:) $[[q_A, M, q_B], [q_A, M - [c], q'_B]]$, if $[q_B, c, q'_B] \in \delta_B$, $c \in M$, and $c \in I_B$.

In $A \oplus B$, a non-final state without successors is called a *deadlock*. The composition of A and B respects *k-limited communication* iff, for all reachable states $[q_A, M, q_B]$ and all x, $M(x) \leq k$. A service automaton B is a *k-strategy* of a service automaton A iff $A \oplus B$ has k-limited communication and no deadlocks are reachable from the initial state.

For k-limited communication, the composition is finite. Throughout the paper, let k be arbitrary and fixed. In practice, the value of k may stem from capacity considerations on the channels, from static analysis of the message transfer, or be chosen just sufficiently large.

Example. Fig. 1 depicts two service automata $\mathsf{S_A}$ ($I_{\mathsf{S_A}} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$, $O_{\mathsf{S_A}} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$) and $\mathsf{S_B}$ ($I_{\mathsf{S_B}} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $O_{\mathsf{S_B}} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$). We precede transition names with "!" to denote sending transitions and with "?" to denote receiving transitions. In the composition $\mathsf{S_A} \oplus \mathsf{S_B}$, only states that are reachable from the initial state are depicted.

2.2. Operating guidelines

Structurally, an operating guideline OG_A of a service automaton A is an annotated service automaton.

Definition 3 (Annotated automaton [18])**.** An *annotated automaton* consists of a deterministic service automaton B and an annotation Φ that assigns a Boolean formula to every state $q \in Q_B$. The formula $\Phi(q)$ is built upon $en(q)$ and an additional proposition *final*. Given an annotated automaton $[B, \Phi]$, another service automaton C with the same alphabet

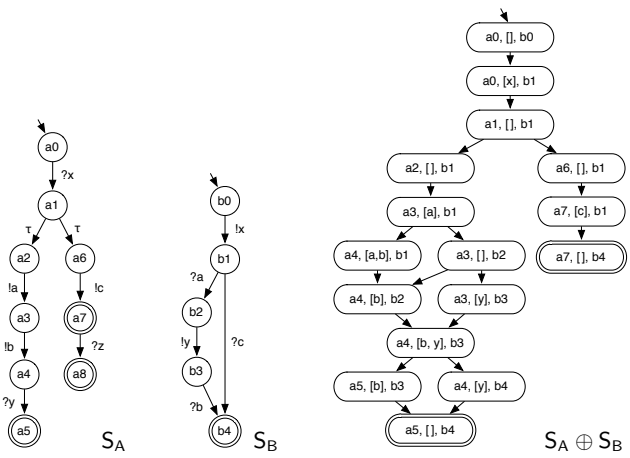

Figure 1: Two service automata and their composition.

as B, a state $q_B \in Q_B$, and a state $q_C \in Q_C$, we say that q_C models $\Phi(q_B)$ (denoted $q_C \models \Phi(q_B)$) iff $\Phi(q_B)$ evaluates to true in the following assignment β to the propositions. Let $\beta(final)$ be true iff $q_C \in \Omega_C$. For other propositions x, let $\beta(x)$ be true iff $x \in en(q_C)$.

According to the definition, annotations to states define requirements about the presence of outgoing edges and the status of a state as final state.

Definition 4 (Matching)**.** A *matching* between two service automata A and B is a relation $\rho_{AB} \subseteq Q_A \times Q_B$, inductively defined as follows:

- $q_{0A}\ \rho_{AB}\ q_{0B}$,
- If $q_A\ \rho_{AB}\ q_B$ and $[q_A, \tau, q'_A] \in \delta_A$ then $q'_A\ \rho_{AB}\ q_B$,
- If $q_A\ \rho_{AB}\ q_B$, $[q_A, x, q'_A] \in \delta_A$, $[q_B, x, q'_B] \in \delta_B$, and $x \neq \tau$, then $q'_A\ \rho_{AB}\ q'_B$.

A matching ρ_{AB} is *complete* if, for all $q_A, q'_A \in Q_A$, $q_B \in Q_B$ and $x \neq \tau$, $q_A\ \rho_{AB}\ q_B$ and $[q_A, x, q'_A] \in \delta_A$ implies that there is an q'_B such that $[q_B, x, q'_B] \in \delta_B$.

A complete matching ρ is actually a particular weak simulation relation. We disregarded τ-steps in B as we will use matching relations only for τ-free automata B as the following definition suggests.

Definition 5 (Operating guideline)**.** A deterministic τ-free annotated automaton $[B, \Phi]$ is an *k-operating guideline* for a service automaton A iff the following statement is true: C is a k-strategy of A if and only if there is a complete matching ρ_{CB} between C and B such that for all $q_B \in Q_B$ and $q_C \in Q_C$, $q_C\ \rho_{CB}\ q_B$ implies $q_C \models \Phi(q_B)$.

Example. The operating guideline $[\mathsf{S_C}, \Phi]$ of the service automaton $\mathsf{S_A}$ (Fig. 1) is depicted in Fig. 2. It is easy to see that there exists a complete matching between $\mathsf{S_B}$ and $[\mathsf{S_C}, \Phi]$.

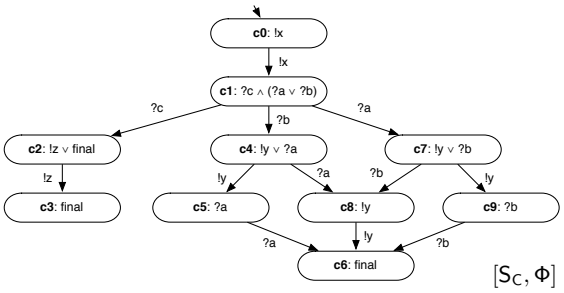

Figure 2: A 1-operating guideline of $\mathsf{S_A}$.

In [2], we proved that every finite state service automaton that has at least one k-strategy, has a k-operating guideline, too, and presented a construction algorithm that is implemented in the tool Fiona[1]. The proof is based on two observations which we exploit in the sequel. For understanding the results of this article, it is sufficient to consider these observations as granted properties.

The first observation states that the service B underlying the operating guideline $[B, \Phi]$ for A is actually a strategy of A.

Proposition 1 ([2])**.** Let $[B, \Phi]$ be an operating guideline. The identity function id is a complete matching between B and $[B, \Phi]$ and, for all $q_B \in Q_B$, $q_B \models \Phi(q_B)$.

The second observation enlightens the origin of the annotations.

Proposition 2 ([2])**.** Let A be a service and $[B, \Phi]$ be its operating guideline. Let C be composable with A and assume that C has a complete matching ρ_{CB} with $[B, \Phi]$, the operating guideline for A. Then, for a state $q_C \in Q_C$, there exist $q_A \in Q_A$ and $M \in Bags(I_A \cup O_A)$ such that $[q_C, M, q_A]$ is a reachable deadlock in the composed system $C \oplus A$ if and only if there is a $q_B \in Q_B$ such that $q_C\ \rho_{CB}\ q_B$ and $q_C \not\models \Phi(q_B)$.

In the sequel, we use these observations for deriving our implicit representation of Φ.

3. Implicit representation of formulae

In this section, we study a number of regularities in the formulae. These regularities lead us to an alternative representation of operating guidelines.

Throughout this section, we consider an arbitrary operating guideline. We assume that every attached formula φ meets the following structural restrictions which do not restrict generality.

- φ is represented in conjunctive normal form. i.e. $\varphi = \{C_1, \ldots, C_n\}$ for a set of clauses C_i, where a clause C_i is a set of literals (negated or plain propositions).
- φ is reduced, i.e. no clause can be replaced by a proper subset, and no clause can be removed, without changing the represented function.

Our first observation says that all send events enabled in q appear positive in every clause.

Lemma 1. Let A be a service, $[B, \Phi]$ its operating guideline, $q \in Q_B$, and $x \in O_B$. Then the assignment β_x assigning true to x and false to all other variables satisfies $\Phi(q)$.

Intuition. If a send event x is enabled in state q_C of service C, then there cannot be a deadlock involving q_C as occurrence of x is not restricted. Thus, the sole presence of x must satisfy a corresponding annotation, regardless of other enabled events.

Proof: (Sketch) Remove all transitions from B that leave q, except for the transition labeled x. Further remove all states and transitions that become unreachable from the

1. Available for download at http://service-technology.org/fiona.

initial state this way. Let the resulting service be B'. It can be shown that $id_{B'}$ is a complete matching between B' and B. As the remaining transition in q is a send transition of B, it is enabled in every state $[q, M, q_A]$ of $B' \oplus A$. Thus, there cannot be a deadlock of the composed system that involves q. By Prop. 2, this means that $q \models \Phi(q)$. □

Our second observation is that an operating guideline of a service does never contain negated literals.

Lemma 2. Let A be a service, $[B, \Phi]$ its operating guideline, $q \in Q_B$, and C_i a clause in $\Phi(q)$. Then C_i does not contain negated literals.

Intuition. In a reduced formula, a negated literal corresponds to a non-monotonous formula. Annotations in operating guidelines are, however, intrinsically monotonous as an additional option (another event or turning a non-final state final) can never turn a non-deadlock into a deadlock.

Proof: (Idea) Assume the contrary. Let C_i be a clause that contains a negated literal $\neg x$. Since $\Phi(q)$ is assumed to be reduced, there must be an assignment β where $\Phi(q)$ yields another value than $\Phi' := (\Phi(q) \setminus \{C_i\}) \cup \{C'\}$ where $C' := C_i \setminus \{\neg x\}$. Clearly, $\beta(x) = \mathit{false}$, $\beta \models C_i$, $\beta \not\models C'$ and thus $\beta \models \Phi(q)$. Let $\beta'(x) = \mathit{true}$ and $\beta'(y) = \beta(y)$, for all $y \neq x$. We have $\beta' \not\models C_i$ and thus $\beta' \not\models \Phi(q)$.

Let first $x \in I_B \cup O_B$. By Prop. 2, $\beta \models \Phi(q)$ and $\beta' \not\models \Phi(q)$ means that there is no deadlock involving q if an outgoing edge labeled x in q is absent while there is a deadlock involving q if such an edge is present. Since an additional edge cannot convert a non-deadlock into a deadlock, this is a contradiction.

Assume second that $x = \mathit{final}$. However, changing a state from non-final to final can only turn a deadlock into a final state (that is, a non-deadlock), but never a non-deadlock into a deadlock. So again, the occurrence of $\neg\mathit{final}$ contracts the observation in Prop. 2. □

The following lemma deals with the appearance of *final* in $\Phi(q)$.

Lemma 3. Let A be a service, $[B, \Phi]$ its operating guideline, $q \in Q_B$, and C_i a clause in $\Phi(q)$. If *final* appears in C_i then no elements of I_B appear in C_i.

Intuition. Getting final makes only sense in situations without pending messages as otherwise the composed system would not get into a final state. In a situation without pending messages, however, events in I_B are all disabled and cannot help to avoid a deadlock.

Proof: (Sketch) Assume the contrary and consider an assignment β to $\Phi(q)$ where all propositions not appearing in C_i are set to true and all propositions in C_i are set to false. As $\Phi(q)$ is false in this assignment, Prop. 2 states that there is a deadlock $[q, M, q_A]$ in $B' \oplus A$ where B' is obtained from B by removing transitions labeled x that leave q where $\beta(x) = \mathit{false}$, and by letting q be a non-final state in B'. As turning the assignment to *final* from false to true will let $\Phi(q)$ evaluate to true, that state must be a final state, that is $q_A \in \Omega_A$ and $M = \emptyset$. If any element of I_B is switched to true instead of *final*, $\Phi(q)$ evaluates to true again. This would, however, not help to leave the deadlock as $M = \emptyset$. Thus, our assumption contradicts Prop. 2. □

In the next two observations, we establish a relation between paths in B and the appearance of elements of I_B in clauses of $\Phi(q)$.

Lemma 4. Let $[B, \Phi]$ be the operating guideline of service A, q a state of B, C a clause appearing in $\Phi(q)$. If C contains an element $x \in I_B$ then there is a path starting in q, taking only transitions labeled x, and leading to a state q' where $\Phi(q')$ contains a clause C' with $C' \cap I_B \subseteq (C \cap I_B) \setminus \{x\}$.

Intuition. Receiving x by a strategy C can only help escaping a deadlock as long as messages are pending in channel x. This is possible at most k times, for the value k fixed throughout this article. Receipt of k, however, does not enable any further event in A, so the same problem as in q recurs in the x-successors of q, but ultimately without the opportunity of receiving x.

Proof: Let B' be obtained from B by removing all edges from B that leave q and have a label that is contained in C. As in previous arguments, id is a complete matching between B' and B. However, $\Phi(q)$ evaluates to false as all propositions in C are false (by Lemma 3, C cannot contain *final* as it contains an element of I_B). Thus, $A \oplus B'$ contains a deadlock $[q_A, M, q]$. This means that $M(y) = 0$ for all $y \notin C$. If any edge with a label $y \in C \cap I_B$ is inserted into B', C is satisfied. Moreover, as $\Phi(q)$ is assumed to be reduced, all other clauses are satisfied as well. Consequently, $M(y) > 0$, for all $y \in C \cap I_B$. Especially we have $M(x) > 0$. It is easy to see that receipt of x in q is an activity that is enabled in B. Following x-transitions $M(x)$ times, we see that some state $[q_A, M', q']$ is reachable in $A \oplus B$ where $M'(y) = M(y)$ for $y \neq x$, and $M(x) = 0$. In this state, transitions with labels in I_B help to prevent a deadlock iff they are contained in $(C \cap I_B) \setminus \{x\}$. Using Lemma 2, $\Phi(q')$ evaluates to false in every assignment which assigns false to all propositions in $(O_B \cup \{\mathit{final}\} \cup C) \setminus \{x\}$. Consequently, at least one clause in $\Phi(q')$ must contain $(C \cap I_B) \setminus \{x\}$. □

For a path π in B, define $lab(\pi)$ to be set of labels at transitions taken along π.

Corollary 1. For every clause C in $\Phi(q)$ there exists a path from q to some state q' where $\Phi(q')$ contains a clause C' with $C' \cap I_B = \emptyset$ and $C \cap I_B \supseteq lab(\pi)$.

The last observation establishes some kind of reversal of Lemma 4.

Lemma 5. Let $x \in I_B$, $[q, x, q']$ be a transition in B and $[B, \Phi]$ the operating guideline of some service A. For every $C' \in \Phi(q')$, there is a clause $C \in \Phi(q)$ where $C \cap I_B \subseteq (C' \cap I_B) \cup \{x\}$.

Intuition. If receipt of x leads to a situation with certain opportunities to resolve deadlocks, then the same problem is present in the corresponding predecessor state, but now with the additional opportunity of receiving x.

Proof: Consider a service B' that is obtained from B through inserting a copy q'' of state q'. Successors of q'' are the same as those of q'. q'' has only one predecessor, q. $Id \cup \{[q'', q']\}$ is a complete matching between B' and B. Consider q'' and $\Phi(q')$. As in Lemma 4, we may conclude that there is a state $[q_A, M, q'']$ in $A \oplus B'$ where $M(y) > 0$ for all $y \in C' \cap I_B$ and $M(y) = 0$ for $y \in I_B \setminus C'$. As q is the only predecessor of q'', $[q_A, M + [x], q]$ must be reachable as well. Lemma 2 provides that $\Phi(q)$ must be false if all propositions in $O_B \cup C' \cup \{x\}$ are false. $\Phi(q)$ must thus contain a clause C with $C \cap I_B \subseteq (C' \cap I_B) \cup \{x\}$. □

From the above observations, we may now derive our implicit representation of the attached Boolean formulae. We claim that we can reproduce all formulae from the following two sets of states.

Definition 6 (S and F)**.** Let $[B, \Phi]$ be the operating guideline of some service A.

- $S := \{q \mid q \in Q_B, \exists C \in \Phi(q), C \subseteq O_B\}$.
- $F := \{q \mid q \in Q_B, \exists C \in \Phi(q), \mathit{final} \in C\}$.

The original formulae can be retrieved from B, S, and F as follows.

Theorem 1. Let $[B, \Phi]$ be the operating guideline of some service A. Let S and F be as in Def. 6. Then, for all states q of B,

$$\Phi(q) \equiv \{(O_B \cap en(q)) \cup lab(\pi) \mid q \xrightarrow{\pi} q', q' \in S \cup F, lab(\pi) \subseteq I_B\} \cup \mathcal{F}$$

where $\mathcal{F} := \{(O_B \cap en(q)) \cup \{\mathit{final}\}\}$, if $q \in F$ and $\mathcal{F} = \emptyset$, otherwise.

Note that the constructed set of clauses is not necessarily reduced.

Proof: Let Φ be the constructed set of clauses. We show first $\Phi(q) \subseteq \Phi$. Let $C \in \Phi(q)$. By Lemma 2, C does not contain negated literals. By Lemma 1, $C \cap O_B = O_B \cap en(q)$, as in all constructed clauses. If $\mathit{final} \in C$ then, by Lemma 3, $C \cap I_B = \emptyset$. Thus, $C = (O_B \cap en(q)) \cup \{\mathit{final}\}$. Since, in this case, $q \in F$, $C \in \Phi$. If $\mathit{final} \notin C$ then Cor. 1 asserts that there is a path π with $lab(\pi) \subseteq C \cap I_B$. Using this particular path, Lemma 5 states that $\Phi(q)$ contains a clause C' with $C' \cap I_B \subseteq lab(\pi)$. With Lemma 1, we conclude $C' \setminus \{\mathit{final}\} \subseteq C$. As $\Phi(q)$ is reduced, we may conclude $C = C'$. Consequently, C is represented in Φ.

Second, we show that $\Phi \subseteq \Phi(q)$. Let $C \in \mathcal{F}$. Then Lemma 1, Lemma 3, and the definition of F assert that $C \in \Phi(q)$. Let finally $C \in \{((O_B \cap en(q)) \cup lab(\pi) \mid q \xrightarrow{\pi} q', q' \in S \cup F, lab(\pi) \subseteq I_B\}$. Assume that, for the path π used for constructing C, $lab(\pi)$ is minimal w.r.t. set inclusion (other paths would lead to redundant clauses and have thus no impact on the semantics of Φ). Lemma 5 states that $\Phi(q)$ contains a clause C' with $C' \cap I_B \subseteq lab(\pi)$. For this C', Cor. 1 states that there is a path to some state in S using only labels in $C' \cap I_B$. Since we assumed minimality of $lab(\pi)$, the set of labels along this path is $lab(\pi)$. We may conclude $C' \cap I_B = lab(\pi)$. Hence, $C \in \Phi(q)$. □

The new representation $[B, S, F]$ has a complexity which is linear in $|B|$. In fact, S and F can be represented as two bits attached to each state of B. In contrast, the original representation of an operating guideline $[B, \Phi]$ of a service A has a complexity of $\mathcal{O}(|B|\,|A|)$ since the length of a single formula can be linear in the number of states of A.

Example. The formulae of the operating guideline $[\mathsf{S_C}, \Phi]$ (Fig. 2) can be represented by the sets $S = \{\mathsf{c0}, \mathsf{c8}\}$ and $F = \{\mathsf{c2}, \mathsf{c3}, \mathsf{c6}\}$.

4. Matching with the new representation

In this section, we sketch a procedure that checks whether a given service C matches an operating guideline in its new representation $[B, S, F]$. In principle, we proceed as suggested in Def. 5. Using a coordinated depth-first search through B and C, we create a complete matching $\rho \subseteq Q_C \times Q_B$. Then, for all $[q_C, q_B] \in \rho$, we need to check for satisfaction of $\Phi(q_B)$. The procedure match sketched in Listing 1 implements this check.

Listing 1 Checking formula satisfaction with new representation

```
procedure match(q_B, q_C : states)
 1: if en(q_C) ∩ O_B ≠ ∅ then
 2:     return true
 3: if q_B ∈ F and q_C ∉ Ω_C then
 4:     return false
 5: if q_B ∈ S then
 6:     return false
 7: if exists π : q_B -π-> q', q' ∈ S ∪ F, ∅ ≠ lab(π) ⊆ I_B \ en(q_C)
    then
 8:     return false
 9: return true
```

Theorem 2. Procedure match correctly implements the calculation of $\Phi(q_B)$ for an element $[q_C, q_B]$ of a complete matching between B and C.

Proof: Consider first the case $en(q_C) \cap O_B \neq \emptyset$ (line 1) and let $x \in en(q_C) \cap O_B$. Since ρ is a complete matching, $x \in en(q_B)$. By Lemma 1, x occurs in every clause of $\Phi(q_B)$, so $\Phi(q_B)$ evaluates to true, as returned by match(q_B, q_C). Consider next the case $q_B \in F$ and $q_C \notin \Omega_C$ (line 3). As the test of line 1 failed, all propositions in O_B take value false. Furthermore, $q_C \notin F$ implies that false is assigned *final*, too. However, with $q_B \in F$, $\Phi(q_B)$ contains a clause $C \subseteq O_B \cup \{\mathit{final}\}$. This clause evaluates to false, so the

returned value in line 4 is correct. If, in line 5, $q_B \in S$ then $\Phi(q_B)$ contains a clause $C \subseteq O_B$. Since the test in line 1 already failed, this clause evaluates to false and the returned value is correct. In the case specified in the condition of line 5, existence of a nonempty path with labels in $I_B \setminus C$ implies existence of a clause C where $C \subseteq O_B \cup (I_B \setminus en(q_C))$. By the failed test of line 1, all propositions in O_B get value false, and so do all propositions in $I_B \setminus en(q_C)$. Again, the returned value in line 8 is correct. In line 9, we may conclude the following situation. By the failed test in line 5, there is no clause that contains only propositions in O_B. Thus, every clause contains *final* or propositions in I_B. By the failure of the test in line 3, *final* does not appear in any clause, or proposition *final* is true. If a clause contains propositions in I_B, the failed test in line 7 provides that at least one of them is contained in $en(q_C)$, too. Since propositions in $en(q_C)$ get value true, all clauses evaluate to true which justifies the returned value. □

The only nontrivial computation in match is the search for a suitable path in line 7. It can be executed using any kind of graph search algorithm (e.g. depth first search). However, the search occurs only if all tests in lines 1–6 failed. Furthermore, the search space is restricted to transitions with labels in $I_B \setminus en(q_C)$. In addition, we may exploit further structural observations about operating guidelines. As a matter of fact, presence of a path starting in q with labels in I_B implies that there is a path for every permutation of the involved labels. Thus, it is sufficient to search only for paths where the sequence of labels is weakly monotonously ascending, for some total order on I_B. Such a search can be organized by considering transitions only if their label is equal to or greater than the label of the previously taken transition. With all these considerations in mind, procedure match should require only limited computational efforts.

5. Transforming B into a Petri net

The automaton B being part of an operating guideline is in fact a labeled transition system. This automaton exhibits a considerable number of so-called *diamond structures* (cf. Fig. 2), i.e. situations where transitions may occur in any order and lead to the same state. As already mentioned in the previous section, this behavior applies to all sequences of transition with labels in I_B but also to all sequences of transitions with labels in O_B. It is thus reasonable to consider a representation of B as a Petri net.

It is well known that some labeled transition systems can be transformed into a Petri net using a technique known as *theory of regions* [8], [9]. This technique creates a Petri net with exactly one transition per occurring label in the transition system whenever one exists. In our setting, we cannot assert that B can be represented by a Petri net with just one transition per label. Consequently, we use a generalization of the technique where labels are split into copies of the same label whenever the original technique fails [19]. This technique is intrinsically nondeterministic. So far, we did not explore a domain-specific heuristics for resolving the nondeterminism and just relied on the procedures available in the state-of-the-art tool Petrify [10]. Using this approach, we could, however, achieve a significant reduction in size for the representation of B, as the results of the case study (see Sect. 7) suggest.

6. Representing S and F in a Petri net context

Assume that the underlying structure B of an operating guideline $[B, S, F]$ is represented as a Petri net. There are two fundamental approaches to the representation of the sets S and F which replace the attached formulae in our approach. We can either represent them explicitly (as a plain enumeration of Petri net markings), or symbolically. A symbolic representation could, for instance, view a set of markings as a set of (Boolean or encoded as Boolean) vectors which can be translated into an implicit representation as a Boolean formula (assigning true to the vectors in the represented set). Approaches of this kind include the Quine/McCluskey algorithm (e.g., see [20]) and the normalization of binary decision diagrams [6]. Despite the availability of these (and many more) advanced techniques, we found that an explicit representation is quite competitive if the following optimizations are met.

We observed that some structural patterns in B give strong indications about presence or absence of states in S or F. In particular, we observed:

- a state without successors must be in F;
- a state in F is not a member of S;
- a state in S is not a member of F;
- a state without successors in I_B is most likely a member of S;
- a state with successors in I_B is most likely not a member of S.

These observations suggest to explicitly list three sets of states:

Definition 7 (F', S_1, S_2)**.** Let $[B, \Phi]$ be the operating guideline of some service A.

- $F' := F \cap \{q \mid q \in Q_B, en(q) \neq \emptyset\}$.
- $S_1 := S \cap \{q \mid q \in Q_B, en(q) \cap I_B \neq \emptyset\}$.
- $S_2 := \{q \mid q \in Q_B, en(q) \cap I_B = \emptyset\} \setminus (S \cup F)$.

Corollary 2. The original sets S and F can be reconstructed from the structure of B and the reduced sets S_1, S_2, and F' as follows.

- $F = F' \cup \{q \mid q \in Q_B, en(q) = \emptyset\}$.
- $S = S_1 \cup \big(\{q \mid q \in Q_B, en(q) \subseteq O_B\} \setminus S_2\big)$.

Without the set S_2, the set S cannot be correctly reconstructed. The operating guideline $[\mathsf{S_E}, \Phi]$ (Fig. 3) of the

service automaton $\mathsf{S_D}$ is an example in which the node e0 has only successors in O_E and is not annotated with *final*. This node's annotation can still be satisfied without setting the literal !v to true. Hence, e0 $\notin S$. Though the *true*-annotation characterizes non-responsive services such as $\mathsf{S_F}$, changing the e0's annotation to "!v $\vee$ *final*" would not help, because this would exclude the responsive service $\mathsf{S_G}$.

Example. Fig. 4 depicts the result of applying region theory to the structure of the operating guideline of Fig. 2. The formulae are represented by the marking sets $S = \{[\mathsf{p0}], [\mathsf{p4}, \mathsf{p6}]\}$ and $F = \{[\mathsf{p5}], [\mathsf{p6}, \mathsf{p7}], [\mathsf{p8}]\}$. Exploiting further regularities, these sets can be alternatively represented by $S_1 = \emptyset$, $S_2 = \emptyset$, and $F' = \{[\mathsf{p5}]\}$.

7. Case Study

To evaluate the Petri net representation of B and the new representation of the formulae, we applied the presented techniques to several real-life WS-BPEL services from industrial partners. To this end, we translated the WS-BPEL processes into service automata using the tool BPEL2oWFN [14]. Then we used Fiona to calculate the operating guidelines as annotated service automata.

Fiona is able to translate these automata into a Petri net representation using Petrify as backend. We measured the size of B as the sum of its states and edges and the size of B's Petri net representation as the sum of its places, transitions, and arcs. For the examples, the effect of the reduction dramatically increases with the size of the operating guideline B. For the largest examples, the size Petri net representation is less than $1\,\%$ of the automaton representation. Table 1 summarizes the results.

Additionally, we implemented the implicit encoding of the nodes' formulae using the sets S and F (Def. 6). The results (see Table 2) show that these sets are — compared to the size of B — relatively small. The reduction compares the size of the sets with the number of formulae of B. This reduction can be further improved using the sets S_1, S_2, and F' (Def. 7): in some cases, no markings had to be stored at all and the largest set contained only 7 markings. Note that for all services of our case studies, the set S_2 was empty, because situations such as depicted in Fig. 3 are very rare in practice. The data show that the number of elements in the represented sets is such small that a symbolic representation

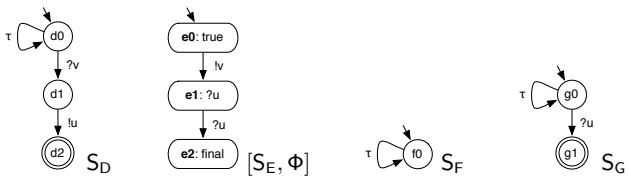

Figure 3: Justification for the set S_2.

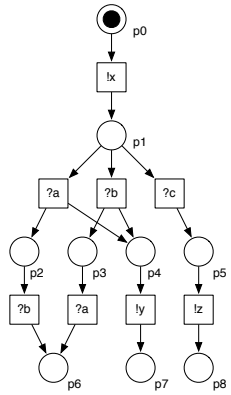

Figure 4: A Petri net representation of $\mathsf{S_C}$.

is not necessary.

The case study of this paper can be replayed using the Web-based implementation of the tool chain available at http://service-technology.org/live/pnog. At the same URL, the tools and the examples of the case study can be downloaded.

8. Conclusion and future work

We presented a compact representation of the formulae that are attached to the states of an operating guideline. This representation is not only very space-efficient, but also allows to translate the structure of the operating guideline into a Petri net representation which again is much smaller than the original operating guideline. Experiments show that the

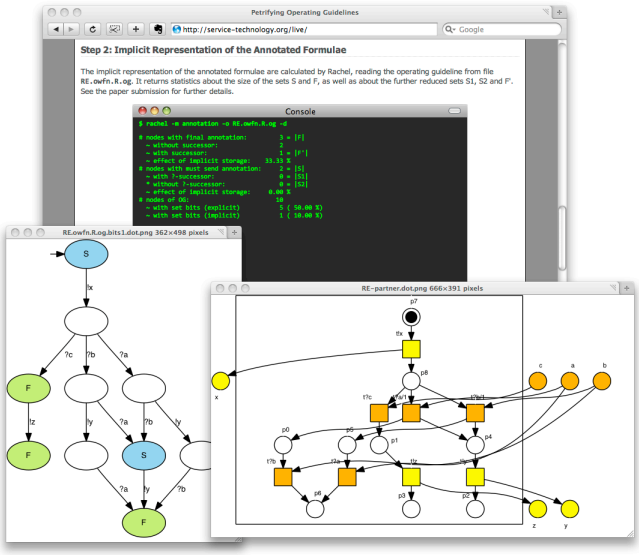

Figure 5: Screen shot of the Web-based implementation available at http://service-technology.org/live/pnog.

Table 1: Size comparisons between the automaton and the Petri net representation of B.

service	size B (automaton representation)	size B (Petri net representation)	reduction
Ticket Reservation	403	116	71.22 %
Online Shop	484	316	34.72 %
Internal Order	692	249	64.02 %
Travel Service	776	95	87.76 %
Purchase Order	992	76	92,34 %
Reservations	1866	94	94.96 %
Contract Negotiation	5760	116	97.99 %
Deliver Finished Goods	7792	93	98.81 %
Passport Application	9472	65	99.32 %
Quotation Requisition	77056	167	99.78 %

Table 2: Size comparison between the number of formulae of B and the set representations.

service	formulae of $[B, \Phi]$	$\lvert S \rvert$	$\lvert F \rvert$	reduction	$\lvert S_1 \rvert$	$\lvert S_2 \rvert$	$\lvert F' \rvert$	reduction
Ticket Reservation	110	8	2	9.09 %	0	0	1	99.09 %
Online Shop	153	11	21	20.92 %	0	0	7	95.44 %
Internal Order	184	7	8	8.15 %	1	0	4	97.28 %
Travel Service	192	47	2	25.52 %	0	0	1	99.48 %
Purchase Order	232	16	1	7.32 %	0	0	0	100.00 %
Reservations	369	3	6	2.43 %	1	0	5	98.37 %
Contract Negotiation	1152	26	2	2.43 %	0	0	0	100.00 %
Deliver Finished Goods	1376	18	2	1.45 %	1	0	1	99.85 %
Passport Application	1536	3	1	0.26 %	0	0	0	100.00 %
Quotation Requisition	11264	255	1	2.27 %	0	0	0	100.00 %

approach is applicable to realistic services.

The size of the synthesized Petri net heavily depends on the level of concurrency in the operating guideline. By adding new states to the operating guideline, the concurrency can be increased and the resulting Petri net might be more compact. To still correctly characterize all strategies of a service, the added nodes then have to be listed in a set similar to the sets introduced in Sect. 6. Likewise, the synthesis algorithm can be adjusted to exploit such domain-specific knowledge. For instance, Carmona et al. [21] present a algorithm without label splitting to apply region theory in the area of *process mining* [22].

Another interesting direction for future work is the matching between a service automaton and an operating guideline represented by a Petri net. Applying state space reduction techniques such as *partial order reduction* [23] might help to realize a matching algorithm that avoids to build the complete state space of the operating guideline.

Finally, the concurrency revealed by the region theory might give a deeper insight in the nature of the services characterized by the operating guideline.

Acknowledgment

N. Lohmann and K. Wolf are supported by the DFG project "Operating Guidelines for Services" (WO 1466/8-1).

References

[1] K. Gottschalk, "Web Services Architecture Overview," IBM developerWorks, IBM Whitepaper, 2000, http://ibm.com/developerWorks/web/library/w-ovr.

[2] N. Lohmann, P. Massuthe, and K. Wolf, "Operating guidelines for finite-state services," in *PETRI NETS 2007*, ser. LNCS, vol. 4546. Springer, 2007, pp. 321–341.

[3] W. M. P. v. d. Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf, "Multiparty contracts: Agreeing and implementing interorganizational processes," *Comput. J.*, 2009, (In press).

[4] C. Stahl, P. Massuthe, and J. Bretschneider, "Deciding Substitutability of Services with Operating Guidelines," *LNCS Transactions on Petri Nets and Other Models of Concurrency*, 2009, (In press).

[5] K. Kaschner and N. Lohmann, "Automatic test case generation for interacting services," in *ICSOC 2008 Workshops*, ser. LNCS. Springer, 2008, (in press).

[6] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[7] K. Kaschner, P. Massuthe, and K. Wolf, "Symbolic representation of operating guidelines for services," *Petri Net Newsletter*, vol. 72, pp. 21–28, 2007.

[8] A. Ehrenfeucht and G. Rozenberg, "Partial (set) 2-structures. Part I, II," *Acta Inf.*, vol. 27, no. 4, pp. 315–368, 1989.

[9] E. Badouel, L. Bernardinello, and P. Darondeau, "Polynomial algorithms for the synthesis of bounded nets," in *TAPSOFT 1995*, ser. LNCS, vol. 915. Springer, 1995, pp. 364–378.

[10] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *Trans. Inf. and Syst.*, vol. E80-D, no. 3, pp. 315–325, 1997.

[11] A. Alves *et al.*, "Web Services Business Process Execution Language Version 2.0," OASIS, OASIS Standard, 2007.

[12] OMG, "Business Process Modeling Notation (BPMN) Version 1.0," OMG, OMG Final Adopted Specification, 2006.

[13] ——, "Unified Modeling Language (UML)," OMG, Version 2.1.2, 2007.

[14] N. Lohmann, "A feature-complete Petri net semantics for WS-BPEL 2.0," in *WS-FM 2007*, ser. LNCS, vol. 4937. Springer, 2008, pp. 77–91.

[15] N. Lohmann and J. Kleine, "Fully-automatic translation of open workflow net models into simple abstract BPEL processes," in *Modellierung 2008*, ser. LNI, vol. P-127. GI, 2008, pp. 57–72.

[16] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[17] D. Brand and P. Zafiropulo, "On communicating finite-state machines," *J. ACM*, vol. 30, no. 2, pp. 323–342, 1983.

[18] A. Wombacher, P. Fankhauser, B. Mahleko, and E. J. Neuhold, "Matchmaking for business processes based on choreographies," *Int. J. Web Service Res.*, vol. 1, no. 4, pp. 14–32, 2004.

[19] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev, "Deriving Petri nets from finite transition systems," *IEEE Trans. Computers*, vol. 47, no. 8, pp. 859–882, 1998.

[20] R. F. Tinder, *Engineering Digital Design*, revised second ed. Academic Press, 2000.

[21] J. Carmona, J. Cortadella, and M. Kishinevsky, "A region-based algorithm for discovering Petri nets from event logs," in *BPM 2008*, ser. LNCS, vol. 5240. Springer, 2008, pp. 358–373.

[22] W. M. P. v. d. Aalst, B. v. Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, "Workflow mining: A survey of issues and approaches," *Data Knowl. Eng.*, vol. 47, no. 2, pp. 237–267, 2003.

[23] A. Valmari, "Stubborn sets for reduced state space generation," in *PETRI NETS 1989*, ser. LNCS, vol. 483. Springer, 1989, pp. 491–515.