Fundamenta Informaticae 107 (2011) 1–19
DOI 10.3233/FI-2011-300
IOS Press

Compact Representations and Efficient Algorithms for Operating Guidelines

Niels Lohmann
Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
niels.lohmann@uni-rostock.de

Karsten Wolf*
Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
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. They are typically represented as an annotated transition system where the annotations are Boolean formulae attached to the states.

The core result of this article is to propose an alternative representation of operating guidelines where, instead of a Boolean formula, only a few bits need to be stored with a state. This way, we safe one order of magnitude in the space complexity of the representation. Moreover, we demonstrate that the new representation yields efficiency gains in several algorithms which involve operating guidelines. Finally we show that the new representation permits the translation of the transition system representing the operating guidelines into a Petri net which typically yields further gains concerning the space for storing operating guidelines.

1. Introduction

Services (e. g., Web services) are made for loosely coupled interaction [12]. Several aspects are crucial for their correct interaction: semantics (compatible interpretation of the meaning of exchanged data), behavior (compatible order of exchanged data), and non-functional ones (compatible process of exchanging data, including policies, quality of service, etc.).

*Address for correspondence: Universität Rostock, Institut für Informatik, 18051 Rostock, Germany

We contribute to the behavioral aspect of service interaction. In this realm, the concept of *operating guidelines* [19] plays a key role. An operating guideline OG_P of a service P is a finite characterization of the (typically infinite) set of all those services R that have a compatible behavior with respect to P. In this paper, we use a compatibility notion that consists of deadlock freedom of the composed system $P \oplus R$ and a given finite bound 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?) [2, 24] to service correction [16] and test case generation [14].

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; that is, 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 transitions. The formulae describe how the behavior may be restricted without destroying compatibility.

The main achievement in this article is an implicit representation of the formulae. Instead of formulae, we come up with three sets of states (in other words, three bits attached to each state) from which the original formula can be fully reconstructed by exploiting regularities in the annotated formulae and the structure of the most-permissive partner. This way, the space complexity for an operating guideline becomes linear in the size of the most-permissive partner. So far, the accumulated length of the attached formulae introduced another order of magnitude.

The new representation has a number of further advantages. First, several algorithms connected to operating guidelines turn out to react quite well to the new representation. It is indeed possible to deal with the new representation directly, without need to reconstruct the original formulae. This happens in particular to the substitutability problem and the minimization problem.

As another advantage, we can further compress the representation of operating guidelines by translating the transition system of the most-permissive partner into a Petri net. In the original representation, this translation was problematic since the formulae (attached to states) could not be assigned to Petri net elements. The compression obtained by translation into Petri nets is remarkable since other attempts for symbolic representation, for instance using BDD [7], brought some but not sufficient success [15]. We introduce the concept of operating guidelines in Sect. 2. The new representation of annotations is derived in Sect. 3.

In Sect. 4, we show that the matching algorithm (checking that a given service is represented by a given operating guideline) can be executed using the new representation. We further show in Sect. 5 that the substitutability problem (checking whether a service can be safely substituted by another service) and the minimization problem (calculating a minimal representation of an operating guideline) can be efficiently implemented using the new representation.

For translating the underlying structure into Petri nets, we use existing theory [11, 4] and an existing tool, Petrify [9]. Section 6 briefly explains our approach. In Sect. 7, we discuss the representation of the three sets of states (or markings) in the context of a 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 set of experiments. It gives evidence of the significant space reduction that can be obtained and is summarized in Sect. 8. In Sect. 9, 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 [3] or BPMN [22], semiformal languages like UML activity diagrams [23], or formal models like Petri nets, process algebras or state machines (automata). Existing translations between WS-BPEL and Petri nets in both directions [17, 18] show that formal models are capable of expressing the relevant behavioral features of services as understood by industry.

We model the behavior 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 2.1. (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 $x \in I$. Define $en(q) := \{x \mid [q, x, q'] \in \delta\}$. 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 behavior (non-communicating activities forever) is 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* [20], 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* [5]. The semantics of message passing is implicitly given in the following definition of service composition. 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). Addition of multisets is performed element-wise. 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 2.2. (Composition of service automata)
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:) $[[q_A, M + [c], q_B], [q'_A, M, q_B]]$, if $[q_A, c, q'_A] \in \delta_A$ and $c \in I_A$,
- (receive by B:) $[[q_A, M + [c], q_B], [q_A, M, q'_B]]$, if $[q_B, c, q'_B] \in \delta_B$ and $c \in I_B$.

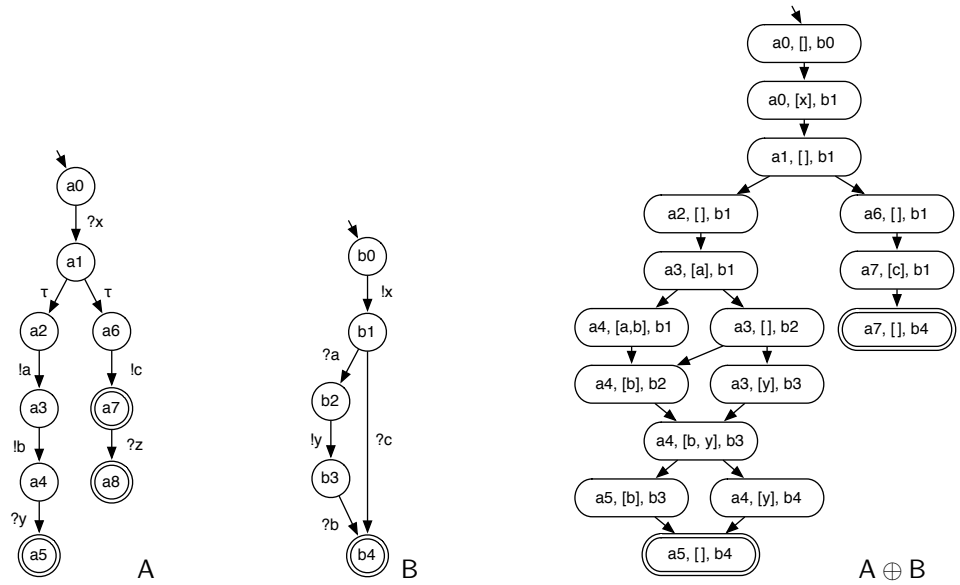

Figure 1. Two service automata and their composition.

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. We thereby assume the existence of a finite bound k and do not consider service compositions with unbounded communication.

Example 2.1. Figure 1 depicts two service automata A and B together with their composition A $\oplus$ B (only states 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 2.3. (Annotated automaton [27])
An *annotated automaton* consists of a deterministic service automaton B without final states 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 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 $\beta(\mathit{final})$ be true iff $q_C \in \Omega_C$. For other propositions x, let $\beta(x)$ be true iff $x \in en(q_C)$.

Annotated automata do not have final states. According to the definition, annotations to states define requirements about the presence of outgoing transitions and the status of a state as final state.

Definition 2.4. (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 2.5. (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 2.2. An operating guideline $[\mathsf{C}, \Phi]$ of the service automaton A (Fig. 1) is depicted in Fig. 2. For instance, the annotation $\Phi(\mathsf{c0}) = !\mathsf{x}$ states that any strategy must send an x-message in the initial state. To increase legibility, we employ the following shorthand notations: (1) all transitions without target state are assumed to have node $\mathsf{c10}$ as target, and (2) a transition with multiple labels represents one transition for each label. A service that initially sends an x-message *and* is able to receive an a-message is also a valid strategy of A. In fact, the transition for receiving a is never enabled since the a-message is never sent by A. However, dead transitions are not part of our correctness criterion, so there is no problem with such a transition. To characterize a strategy with dead transitions (and thus unreachable states), virtually every operating guideline contains a "sink state" (state $\mathsf{c10}$ in Fig. 2) which represents a situation that cannot be reached in a composition with any strategy. A sink state is always entered by a receive activity (send activities or internal activities are always enabled) and has a self-loop with the whole alphabet. The annotation of such a sink state is always true.

In [19], 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 Wendy[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 2.1. (Structure of OG matches)
The identity function id is a complete matching between B and $[B, \Phi]$ and, for all $q_B \in Q_B$, $q_B \models \Phi(q_B)$.

[1] Available as free software for download at `http://service-technology.org/wendy`.

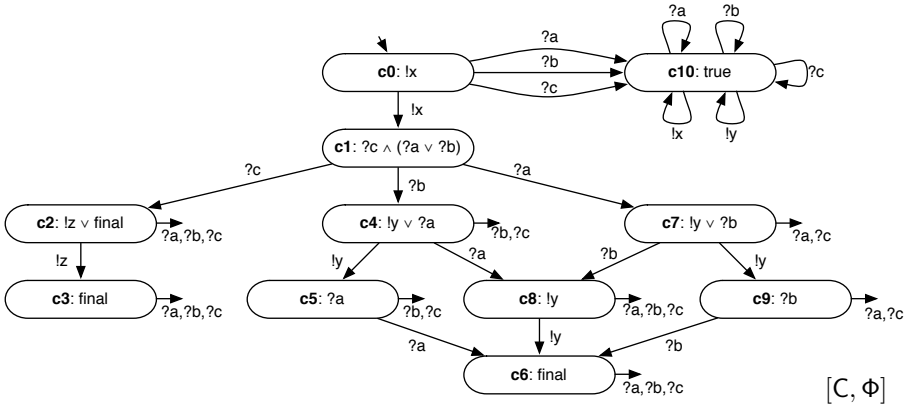

Figure 2. A 1-operating guideline of A.

The second observation enlightens the origin of the annotations. It shows that violations of state annotations in the operating guideline exactly characterize deadlocking interactions with the corresponding service

Proposition 2.2. (Unsatisfied formulae yield deadlocks)
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. We shall later exploit these regularities and present a compact bit-encoding of the formulae which in turn allows for more efficient algorithms. Furthermore, these regularities lead us to an alternative representation for the annotations 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; that is, $\varphi = \{C_1, \ldots, C_n\}$ for a set of clauses C_i, where a clause C_i is a set of literals (negated or plain propositions).
- φ is reduced; that is, no clause can be replaced by a proper subset, and no clause can be removed, without changing the represented function.

Our first observation is that all send events enabled in a state q appear positive in every clause.

Lemma 3.1. (Sending satisfies formulae)
Let A be a service, $[B, \Phi]$ its operating guideline, $q \in Q_B$, and $x \in en(q) \cap O_B$. 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 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. 2.2, this means that $q \models \Phi(q)$. □

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

Lemma 3.2. (Formulae are positive)
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 $\Phi' := (\Phi(q) \backslash \{C_i\}) \cup \{C'\}$ 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. 2.2, $\beta \models \Phi(q)$ and $\beta' \not\models \Phi(q)$ means that there is no deadlock involving q if an outgoing transition labeled x in q is absent while there is a deadlock involving q if such a transition is present. Since an additional transition 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. 2.2. □

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

Lemma 3.3. (*final* excludes sending)
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. 2.2 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, $\Phi(q)$ also evaluates to true again. This would, however, not help to leave the deadlock as $M = \emptyset$. Thus, our assumption contradicts Prop. 2.2. □

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 3.4. (Receiving does not restrict possibilities (1))
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 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. Reception of the last (i. e., the kth) value, 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 transitions 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 3.3, 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 transition 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 Propositiony 2.2, $\Phi(q')$ 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 3.1. For every clause C in $\Phi(q)$ there exists a path π 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 3.4.

Lemma 3.5. (Receiving does not restrict possibilities (2))
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 3.4, 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. Proposition 2.2 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 three sets of states.

Definition 3.1. (State sets S, F, and T)
Let $[B, \Phi]$ be the operating guideline of some service A. Define the following sets.

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

The set S contains states whose formulae can only be satisfied by a sending action, the set F contains states whose formulae contains final, and the set T contains states whose formulae is equivalent to true. In fact, the original formulae can be retrieved from B, S, F, and T as follows.

Theorem 3.1. (Reconstruction of formulae)
Let $[B, \Phi]$ be the operating guideline of some service A. Let S, F, and T be as in Def. 3.1. Then, for all states q of B, $\Phi(q)$ is determined as follows. If $q \in T$ then $\Phi(q) = \emptyset$. If $q \in S$ then $\Phi(q) = \{O_B \cap en(q)\}$. Otherwise,

$$\Phi(q) \equiv \{(O_B \cap en(q)) \cup lab(\pi) \mid q \xrightarrow{\pi} q', length(\pi) > 0, q' \in S \cup F, lab(\pi) \subseteq I_B\} \cup \mathcal{F},$$

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 $C \in \Phi(q)$. This means that $q \notin T$, so Φ is built according to the second case mentioned in Theorem 3.1. By Lemma 3.2, C does not contain negated literals. By Lemma 3.1, $C \cap O_B = O_B \cap en(q)$, as in all constructed clauses. If $\mathit{final} \in C$ then, by Lemma 3.3, $C \cap I_B = \emptyset$. 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. 3.1 asserts that there is a path π with $lab(\pi) \subseteq C \cap I_B$. Using this particular path, Lemma 3.5 states that $\Phi(q)$ contains a clause C' with $C' \cap I_B \subseteq lab(\pi)$. With Lemma 3.1, 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 $\Phi \subseteq \Phi(q)$. If $q \in T$, $\Phi = \emptyset$, so the inclusion holds trivially. Assume now that $q \notin T$ and let $C \in \mathcal{F}$. Then Lemma 3.1, Lemma 3.3, 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 with respect to set inclusion (other paths would lead to redundant clauses and have thus no impact on the semantics of Φ). Lemma 3.5 states that $\Phi(q)$ contains a clause C' with $C' \cap I_B \subseteq lab(\pi)$. For this C', Cor. 3.1 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 $[B, S, F, T]$ has a complexity which is linear in $|B|$. In fact, S, F, and T are disjoint and can thus be represented by 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 3.1. The formulae of the operating guideline $[\mathsf{C}, \Phi]$ (Fig. 2) can be represented by the sets $S = \{\mathsf{c0}, \mathsf{c8}\}$, $F = \{\mathsf{c2}, \mathsf{c3}, \mathsf{c6}\}$, and $T = \{\mathsf{c10}\}$.

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 $[B, S, F, T]$. In principle, we proceed as suggested in Def. 2.5. Using a coordinated depth-first search through B and C (i. e., a traversal of the states of C that is mimicked by B), we create a complete matching $\rho \subseteq Q_C \times Q_B$. As far as B is concerned, the coordinated depth-first search does not introduce backtracking, because B is deterministic. 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** $q_B \in T$ **then**
  true
2: **if** $en(q_C) \cap O_B \neq \emptyset$ **then**
  true
3: **if** $q_B \in F$ and $q_C \notin \Omega_C$ **then**
  false
4: **if** $q_B \in S$ **then**
  false
5: **if** exists $\pi : q_B \xrightarrow{\pi} q', q' \in S \cup F, \emptyset \neq lab(\pi) \subseteq I_B \setminus en(q_C)$ **then**
  false
  true

Theorem 4.1. 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 $q_B \in T$. This means that $\Phi(q_B)$ trivially evaluates to true. Then consider the case $en(q_C) \cap O_B \neq \emptyset$ (line 3) and let $x \in en(q_C) \cap O_B$. Since ρ is a complete matching, $x \in en(q_B)$. By Lemma 3.1, x occurs in every clause of $\Phi(q_B)$, so $\Phi(q_B)$ evaluates to true, as returned by $\mathsf{match}(q_B, q_C)$. Consider next the case $q_B \in F$ and $q_C \notin \Omega_C$ (line 5). As the test of line 3 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 $C \subseteq O_B \cup \{final\}$. This clause evaluates to false, so the returned value in line 6 is correct. If, in line 7, $q_B \in S$ then $\Phi(q_B)$ contains a clause $C \subseteq O_B$. Since the test in line 3 already failed, this clause evaluates to false and the returned value is correct. In the case specified in the condition of line 9, existence of a nonempty path with labels in $I_B \setminus en(q_C)$ implies existence of a clause C where $C \subseteq O_B \cup (I_B \setminus en(q_C))$. By the failed test of line 3, 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 10 is correct. In line 11, we may conclude the following situation. By the failed test in line 7, 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 5, *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 9 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 9. 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 1–8 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. Checking substitutability and minimization

Substitutability

Substitutability is a decision problem. Given two service automata S_A and S_B, we ask whether every k-strategy of S_A is a k-strategy of S_B, too. If the answer to this problem is "yes", it is possible to substitute service S_A with S_B such that correctness in the interaction is preserved. Then S_B *accords* to S_A. The accordance preorder corresponds to the stable failures preorder [6]. However, to the best of our knowledge, there exists no tool support to decide either preorder.

As operating guidelines characterize correctly interacting partners, it is not very surprising that accordance can be decided through an inspection of the involved operating guidelines.

Proposition 5.1. ([24, 21])
Let $[A, \Phi]$ and $[B, \Psi]$ be operating guidelines of services with identical interfaces (i. e., $O_A = O_B$ and $I_A = I_B$). Then the following two statements are equivalent:

(1) For all services C: If C matches with A then C matches with B.

(2) There is a complete matching ϱ between A and B such that, for all $[q_A, q_B] \in \varrho$, $\Phi(q_A) \implies \Psi(q_B)$ is a tautology.

In this section, we contribute to this matter by rephrasing condition (2) in terms of the previously introduced sets S, F, and T. Assuming a representation of these sets as bits attached to the states of an operating guideline, this

means that we replace the check for tautology of the formula $\Phi(q_A) \implies \Psi(q_B)$ by a reasoning on 4 bits (viz. 2 bits per operating guideline). The efficiency gain is evident.

The main result of this section is:

Theorem 5.1. (New accordance check)
Let $[A, \Phi]$ and $[B, \Psi]$ be operating guidelines of services with identical interfaces. If there exists a complete matching ϱ between A and B, then the following two statements are equivalent:

(1) For all $[q_A, q_B] \in \varrho$, the formula $\Phi(q_A) \implies \Psi(q_B)$ is a tautology.

(2) For all $[q_A, q_B] \in \varrho$, the following conditions hold:

(a) $q_A \in T_A$ implies $q_B \in T_B$,

(b) $q_B \in S_B$ implies $q_A \in S_A$,

(c) $q_B \in F_B$ implies $q_A \in S_A \cup F_A$.

For the proof of this theorem, we exploit the following observation on negation-free formulae in CNF.

Proposition 5.2. (Implication between negation-free formulae)
Let ϕ and ψ be formulae in CNF (i. e. sets of clauses) in reduced form (cf. Sect. 3) where all clauses contain only positive literals. Then $\phi \implies \psi$ if and only if, for all $C_\psi \in \psi$, there is a $C_\phi \in \phi$ such that $C_\phi \subseteq C_\psi$.

By Lemma 3.2, this proposition is applicable throughout this section. We separate the proof of Theorem 5.1 into two lemmas representing the two implications to be proven.

Lemma 5.1. Let the assumptions be as in Theorem 5.1. Then (1) implies (2).

Proof:
Assume (1), let ϱ be a complete matching between A and B, and consider arbitrary q_A and q_B.

Ad (a). If $q_A \in T_A$ then $\Phi(q_A) \equiv \top$. As $\Phi(q_A) \implies \Psi(q_B)$, we have $\Psi(q_B) \equiv \top$. This means $q_B \in T_B$.

Ad (b). If $q_B \in S_B$ then, by Def. 3.1, there is a $C_B \in \Psi(q_B)$ such that $C_B \subseteq O_B$. By Prop. 5.2, there is a $C_A \in \Phi(q_A)$ where $C_A \subseteq C_B \subseteq O_B = O_A$. By Def. 3.1, $q_A \in S_A$.

Ad (c). If $q_B \in F_B$ then, by Def. 3.1, there is a $C_B \in \Psi(q_B)$ where $\mathit{final} \in C_B$. By Lemma 3.3, $C_B \subseteq O_B \cup \{\mathit{final}\}$. Using Prop. 5.2, there is a C_A where $C_A \subseteq C_B \subseteq O_B \cup \{\mathit{final}\} = O_A \cup \{\mathit{final}\}$. So either $\mathit{final} \in C_A$ and $q_A \in F_A$, or $\mathit{final} \notin C_A$ and $q_A \in S_A$. □

The reverse direction requires another observation on the structure of operating guidelines.

Lemma 5.2. (Receive is always possible)
Let $[B, \Phi]$ be an operating guideline, $q \in Q_B$, and $x \in I_B$. Then there is a q' with $[q, x, q'] \in \delta_B$.

Proof:
This is a consequence of the fact that an operating guideline represents *all* strategies of some service. In fact, an input activity can not be wrong in any state as it cannot enable new deadlocks. So, by Def. 2.5, B must offer the corresponding transition in every state. □

Now we are ready to approach the second implication of Theorem 5.1.

Lemma 5.3. Let the assumptions be as in Theorem 5.1. Then (2) implies (1).

Proof:
Assume (2), let ϱ be a complete matching between A and B, and consider arbitrary q_A and q_B.

If $\Psi_B \equiv \top$ then the CNF of $\Psi(q_B)$ does not contain clauses. In this case, $q_B \in T_B$, $q_B \notin S_B$, and $q_B \notin F_B$. This means that conditions (a), (b), and (c) are trivially satisfied.

Otherwise, let C_B be an arbitrary clause of $\Psi(q_B)$. We show that there is a $C_A \in \Phi(q_A)$ such that $C_A \subseteq C_B$. By Prop. 5.2, this is sufficient to support the claim of the lemma. We distinguish three cases concerning C_B.

First case: $C_B \subseteq O_B = O_A$. Then $q_B \in S_B$. Using assumption (b), $q_A \in S_A$ which means that there is a clause $C_A \in \Phi(q_A)$ where $C_A \subseteq O_A = O_B$. By Lemma 3.1, the members of C_B are exactly those elements of O_B which are enabled in q_B. Likewise, the members of C_A are exactly those elements of O_A which are enabled in q_A. Since ϱ is a simulation relation and $[q_A, q_B] \in \varrho$, every enabled element in q_A corresponds to an enabled element in q_B. Thus $C_A \subseteq C_B$.

Second case: $\mathit{final} \in C_B$. By Lemma 3.3, $C_B \cap I_B = \emptyset$. Furthermore, $q_B \in F_B$ and therefore, using assumption (c), $q_A \in S_A \cup F_A$. If $q_A \in S_A$ then there is a clause $C_A \in \Phi(q_A)$ with $C_A \subseteq O_A$. Using the same argument as in the first case, we obtain $C_A \subseteq C_B \cap O_B$ and thus $C_A \subseteq C_B$. If $q_A \in F_A$ then there is a clause $C_A \in \Phi(q_A)$ with $\mathit{final} \in C_A$. For this clause, Lemma 3.3 asserts that $C_A \cap I_A = \emptyset$. Using again the same argument as in the first case, $C_A \cap O_A \subseteq C_B \cap O_B$ and consequently $C_A \subseteq C_B$.

Third case: $C_B \cap I_B \neq \emptyset$. Applying Theorem 3.1, there is a path π in B that starts at q_B, leads to some $q'_B \in F_B \cap S_B$ and where $lab(\pi) = C_B \cap I_B$. By Lemma 5.2, there is a a path in A starting from q_A, leading to some q'_A where the same labels appear in the same order as in B. Since ϱ is a simulation relation, $[q'_A, q'_B] \in \varrho$. If $q_A \in S_A$ then $C_A = O_A \cap en(q_A) \in \Phi$, so the inclusion holds. Otherwise, Theorem 3.1 yields that $(O_A \cap en(q_A)) \cup (C_B \cap I_B)$ or some subset making this clause redundant is in Φ. This is a subset of C_B since the elements of $O_A \cap en(q_A)$ are a subset of C_B, as argued in the first case. □

The more efficient check for accordance extends to checking equivalence. Two operating guidelines are equivalent iff they represent the same set of matching services. Clearly, equivalence corresponds to accordance in both directions.

Corollary 5.1. (New equivalence check)
Two operating guidelines $[A, \Phi]$ and $[B, \Psi]$ are equivalent if and only if there is a complete matching ϱ such that, for each $[q_A, q_B] \in \varrho$,

- $q_A \in T_A$ if and only if $q_B \in T_B$,
- $q_A \in S_A$ if and only if $q_B \in S_B$,
- $q_A \in F_A$ if and only if $q_B \in F_B$.

This corollary is a Boolean logic consequence of conditions (a), (b), and (c) of Theorem 5.1, applied in both directions.

Minimization

In the thesis [21], a procedure was described for transforming an operating guideline into an equivalent one with a minimal number of states. The procedure involves checking equivalence between states in the given operating guideline:

Definition 5.1. (Equivalent states [21])
Two states q_1 and q_2 of an operating guideline $[B, \Phi]$ are *equivalent* iff (1) the formula $\Phi(q_1) \iff \Phi(q_2)$ is a tautology, (2) $en(q_1) = en(q_2)$, and (3) for all $x \in en(q_1)$ with $[q_1, x, q'_1] \in \delta$ and $[q_2, x, q'_2] \in \delta$ holds: q'_1 and q'_2 are equivalent.

Iterative merging of equivalent states then yields a minimized operating guideline which represents the same set of strategies. With the representation $[B, S, F, T]$ and Corollary 5.1, we can replace the equivalence check (1)

by a check whether the states q_1 and q_2 are annotated with the same bits. This check can be straightforwardly implemented in the standard procedure to minimize finite automata [13]. First experiments unsurprisingly show that the bit representation allows for a speed up especially when the interface of the operating guideline is large. In these cases, the Boolean formulae usually have many variables which in turn slows down equivalence checks.

In consequence, the implicit representation of the annotations yields clear efficiency gains in various constructions and algorithms related to operating guidelines. This way of representing annotations is thus natural and elegant.

6. Transforming an operating guideline 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); that is, situations where transitions may occur in any order and lead to the same state. As already mentioned in Sect. 4, 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* [11, 4]. It 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 [10]. It 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 [9]. Using this approach, we could, however, achieve a significant reduction in size for the representation of B, as the experimental results (see Sect. 8) suggest.

As already sketched in the explanation of Fig. 2, an operating guideline with $I \neq \emptyset$ typically has a specific sink state q^* with annotation *true* that characterizes unreachable but harmless behavior. q^* is reached by receiving events (cf. state c10 in Fig. 2). By Lemma 5.2, state q^* has a high in-degree (cf. Fig. 2) which has a negative impact on the heuristics employed by the tool Petrify. We therefore decided to pass a transition system to Petrify where q^* is removed. In the matching procedures, we know that we enter q^* whenever a receive event is disabled in some state. The only successor of q^* is q^* itself, so we do not lose any information.

7. Representing the implicit annotations in a Petri net context

Assume that the underlying structure B of an operating guideline $[B, S, F, T]$ is represented as a Petri net. There are two fundamental approaches to the representation of the sets S, F, and T 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 Booleans or a Boolean-encoded 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 [25]) and the normalization of binary decision diagrams [7]. 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, F, or T. In particular, we observed:

- a state without successors must be either in T or in F;
- a state without successors in I_B is most likely a member of S;
- the sets S, F, and T are pairwise disjoint.

These observations suggest to explicitly list three sets of states:

Definition 7.1. (Reduced state sets S', F')
Let $[B, \Phi]$ be the operating guideline of some service A. Define the following sets.

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

A state without successors must be either in T or F. Therefore, we can remove it from the set F, arriving at a smaller set F'. Likewise we remove from S those states that only have outgoing sending transitions yielding S'. The original sets can straightforwardly be reconstructed from the set T and the reduced sets F' and S'.

Corollary 7.1. The original sets S and F can be reconstructed from the structure of B and the reduced sets S', F', and T as follows.

- $F = F' \cup (\{q \mid q \in Q_B, en(q) = \emptyset\} \setminus T)$.
- $S = S' \cup (\{q \mid q \in Q_B, en(q) \subseteq O_B\} \setminus (T \cup F))$.

In Def. 7.1, we did not reduce the set T. This set is usually a singleton and only contains the state q^* modeling the unreachable behavior. We already motivated that we do not need to store this node explicitly, because it can be reconstructed from the structure of the remaining annotated automaton. The operating guideline $[\mathsf{S_E}, \Phi]$ (Fig. 3) of the service automaton $\mathsf{S_D}$ is an example with two $true$-annotated nodes.

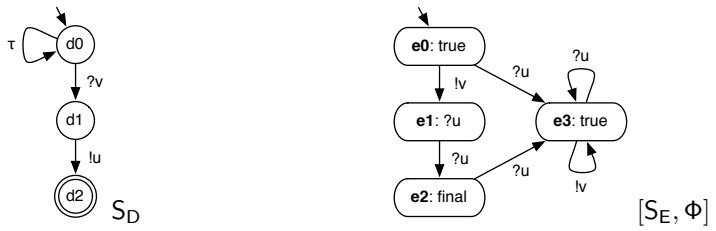

Figure 3. Example for an operating guideline with more than one *true*-annotated state.

Example 7.1. Figure 4 depicts the result of applying region theory to the structure of the operating guideline of Fig. 2. Node c10 is not stored. The formulae are represented by the marking sets $T = \emptyset$, $S = \{[\mathsf{p0}], [\mathsf{p4}, \mathsf{p6}]\}$, and $F = \{[\mathsf{p5}], [\mathsf{p6}, \mathsf{p7}], [\mathsf{p8}]\}$. Exploiting further regularities, the latter two sets can be alternatively represented by $S' = \emptyset$, and $F' = \{[\mathsf{p5}]\}$.

8. Experimental results

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 [17]. Then we used Wendy to calculate the operating guidelines as annotated service automata.

We then translated these automata into a Petri net representation using Petrify as backend. We measured the size of B as the sum of its states and transitions 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 of the Petri net representation is less than $1\,\%$ of the automaton representation. Table 1 summarizes the results.

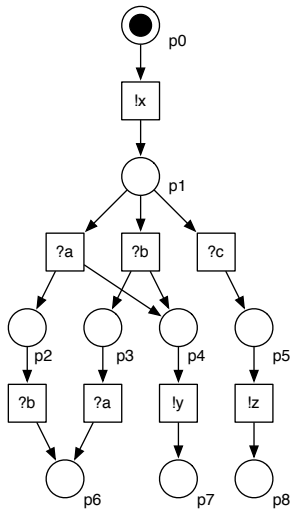

Figure 4. A Petri net representation of C.

Additionally, we implemented the implicit encoding of the nodes' formulae using the sets S and F (Def. 3.1). 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' and F' (Def. 7.1): 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 experiments, the set T 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 is not necessary.

The experiments 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 experiments can be downloaded.

9. 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 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 ones introduced in Sect. 7. Likewise, the synthesis algorithm can be adjusted to exploit such domain-specific knowledge. For instance, Carmona et al. [8] present an algorithm without label splitting to apply region theory in the area of *process mining* [1].

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* [26] might help to realize a matching algorithm that avoids to build the complete state space of the operating guideline.

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	28.78 %
Online Shop	484	316	65.28 %
Internal Order	692	249	35.98 %
Travel Service	776	95	12.24 %
Purchase Order	992	76	7.66 %
Reservations	1866	94	5.04 %
Contract Negotiation	5760	116	2.01 %
Deliver Finished Goods	7792	93	1.19 %
Passport Application	9472	65	0.68 %
Quotation Requisition	77056	167	0.22 %

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

service	**formulae of $[B, \Phi]$**	$\|S\|$	$\|F\|$	$\|T\|$	**reduction**	$\|S'\|$	$\|F'\|$	**reduction**
Ticket Reservation	110	8	2	0	9.09 %	0	1	0.91 %
Online Shop	153	11	21	0	20.92 %	0	7	4.56 %
Internal Order	184	7	8	0	8.15 %	1	4	2.72 %
Travel Service	192	47	2	0	25.52 %	0	1	0.52 %
Purchase Order	232	16	1	0	7.32 %	0	0	0.00 %
Reservations	369	3	6	0	2.43 %	1	5	1.63 %
Contract Negotiation	1152	26	2	0	2.43 %	0	0	0.00 %
Deliver Finished Goods	1376	18	2	0	1.45 %	1	1	0.15 %
Passport Application	1536	3	1	0	0.26 %	0	0	0.00 %
Quotation Requisition	11264	255	1	0	2.27 %	0	0	0.00 %

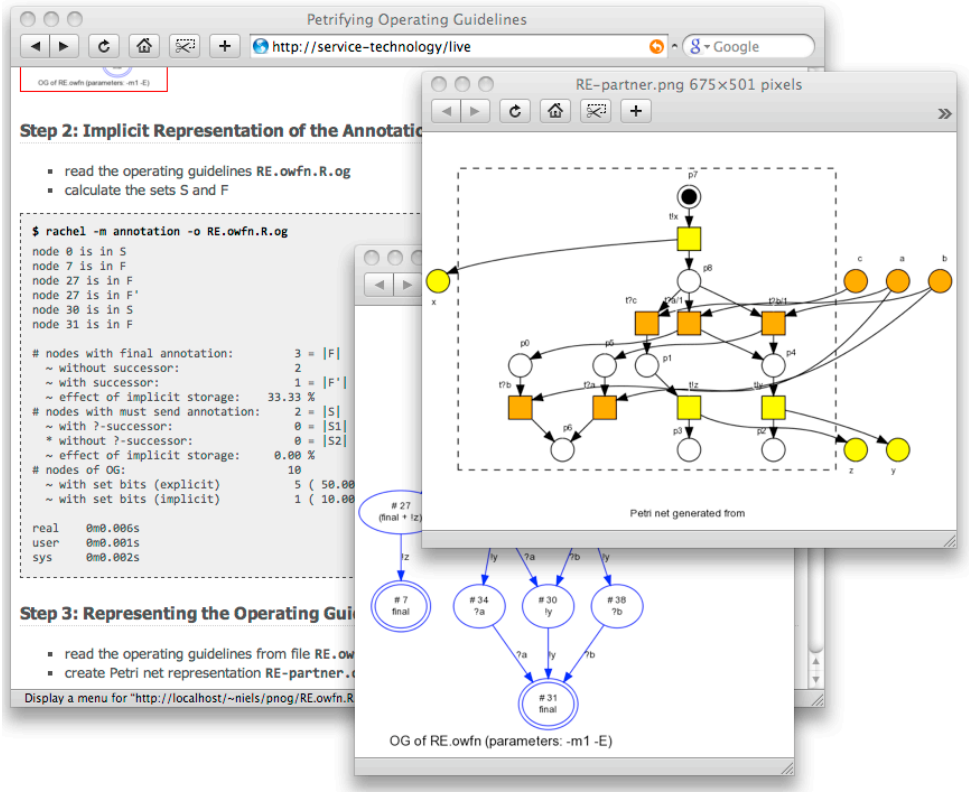

Figure 5. Screen shot of the Web-based implementation available at `http://service-technology.org/live/pnog`.

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

Acknowledgements The authors would like to thank the anonymous referees for their valuable advice. N. Lohmann and K. Wolf are supported by the DFG project "Operating Guidelines for Services" (WO 1466/8-1).

References

[1] Aalst, W. M. P. v. d., Dongen, B. v., Herbst, J., Maruster, L., Schimm, G., Weijters, A. J. M. M.: Workflow mining: A survey of issues and approaches, *Data Knowl. Eng.*, **47**(2), 2003, 237–267.

[2] Aalst, W. M. P. v. d., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty Contracts: Agreeing and Implementing Interorganizational Processes, *Comput. J.*, 2009, (In press).

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

[4] Badouel, E., Bernardinello, L., Darondeau, P.: Polynomial Algorithms for the Synthesis of Bounded Nets, *TAPSOFT 1995*, 915, Springer, 1995.

[5] Brand, D., Zafiropulo, P.: On Communicating Finite-State Machines, *J. ACM*, **30**(2), 1983, 323–342.

[6] Brookes, S. D., Hoare, C. A. R., Roscoe, A. W.: A Theory of Communicating Sequential Processes, *J. ACM*, **31**(3), 1984, 560–599.

[7] Bryant, R. E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Computers*, **C-35**(8), 1986, 677–691.

[8] Carmona, J., Cortadella, J., Kishinevsky, M.: A Region-Based Algorithm for Discovering Petri Nets from Event Logs, *BPM 2008*, 5240, Springer, 2008.

[9] Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers, *Trans. Inf. and Syst.*, **E80-D**(3), 1997, 315–325.

[10] Cortadella, J., Kishinevsky, M., Lavagno, L., Yakovlev, A.: Deriving Petri nets from finite transition systems, *IEEE Trans. Computers*, **47**(8), 1998, 859–882.

[11] Ehrenfeucht, A., Rozenberg, G.: Partial (Set) 2-Structures. Part I, II, *Acta Inf.*, **27**(4), 1989, 315–368.

[12] Gottschalk, K.: *Web Services Architecture Overview*, IBM whitepaper, IBM developerWorks, 2000, `http://ibm.com/developerWorks/web/library/w-ovr`.

[13] Hopcroft, J.: *An $n \log n$ algorithm for minimizing states in a finite automaton*, Technical Report STAN-CS-71-190, Stanford University, 1971.

[14] Kaschner, K., Lohmann, N.: Automatic Test Case Generation for Interacting Services, *ICSOC 2008 Workshops*, 5472, Springer, 2009.

[15] Kaschner, K., Massuthe, P., Wolf, K.: Symbolic Representation of Operating Guidelines for Services, *Petri Net Newsletter*, **72**, 2007, 21–28.

[16] Lohmann, N.: Correcting Deadlocking Service Choreographies Using a Simulation-Based Graph Edit Distance, *BPM 2008*, 5240, Springer, 2008.

[17] Lohmann, N.: A Feature-Complete Petri Net Semantics for WS-BPEL 2.0, *WS-FM 2007*, 4937, Springer, 2008.

[18] Lohmann, N., Kleine, J.: Fully-automatic Translation of Open Workflow Net Models into Simple Abstract BPEL Processes, *Modellierung 2008*, P-127, GI, 2008.

[19] Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services, *PETRI NETS 2007*, 4546, Springer, 2007.

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

[21] Massuthe, P.: *Operating Guidelines for Services*, Dissertation, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Berlin, Germany; Eindhoven University of Technology, Eindhoven, The Netherlands, 2009.

[22] OMG: *Business Process Modeling Notation (BPMN) Version 1.0*, OMG Final Adopted Specification, OMG, 2006.

[23] OMG: *Unified Modeling Language (UML)*, Version 2.1.2, OMG, 2007.

[24] Stahl, C., Massuthe, P., Bretschneider, J.: Deciding Substitutability of Services with Operating Guidelines, *LNCS Transactions on Petri Nets and Other Models of Concurrency*, **II**(5460), 2009, 172–191.

[25] Tinder, R. F.: *Engineering Digital Design*, Revised second edition, Academic Press, 2000.

[26] Valmari, A.: Stubborn sets for reduced state space generation, *PETRI NETS 1989*, 483, Springer, 1989.

[27] Wombacher, A., Fankhauser, P., Mahleko, B., Neuhold, E. J.: Matchmaking for Business Processes Based on Choreographies, *Int. J. Web Service Res.*, **1**(4), 2004, 14–32.