

Symbolic Representation of Operating Guidelines for Services

Kathrin Kaschner¹, Peter Massuthe², and Karsten Wolf¹

¹ Humboldt-Universität zu Berlin, Institut für Informatik,
Unter den Linden 6, 10099 Berlin, Germany
`massuthe@informatik.hu-berlin.de`

² Universität Rostock, Institut für Informatik,
18051 Rostock, Germany
{`kathrin.kaschner`, `karsten.wolf`}@informatik.uni-rostock.de

1 Introduction

Services are selfcontrolled executable software components. Generally they are not executed in isolation, but communicate with other services through message exchange via their interface. With the paradigm of service-oriented architectures (SOA) [1], a mechanism for the composition of services is provided. The *service provider* makes a service available. For his service to be found and used by a *service requester*, he publishes a description of his service in a central repository which is managed by a *service broker*. Hence, a service requester can attempt to find an adequate service in the repository. If that is the case, the services can bind and interact with each other.

SOA, however, only postulate a general protocol for the composition of services. It is intentionally left open how to be realized exactly. For instance, it is not described which information should be published. On the one hand, the service broker needs enough information about an offered service P to be able to decide if the service R of a requester fits to P . That means: beforehand it has to be guaranteed that the two services, while interacting, neither get into a deadlock nor send unexpected messages to each other. On the other hand, the whole service P cannot simply be published because the service provider may want to keep its internal structure secret.

In [2, 3] we proposed an approach in which the provider publishes an *operating guideline* (OG) of his service P . Operating guidelines describe how a potential partner has to behave. For a service R , the broker now only has to check if R fulfills the requirements of the operating guideline. This process is called *matching*. If the matching is successful, the interaction between R and P proceed without problems.

An operating guideline is a directed graph, whose nodes are enriched with annotations. In practice, however, the graph may become too big to be saved explicitly. Thus, it is necessary to look for a compact representation of operating guidelines. In this article we will propose an approach for this purpose. We will show how operating guidelines can be coded symbolically through binary decision diagrams (BDDs) and explain the matching based on BDDs.

This paper is organized as follows: first we introduce open workflow nets (oWFNs) as a formal model for services and define the interaction of two services based on oWFNs. In the third section we explain how the operating guideline is computed for an oWFN of a service. In the whole article we will restrict our considerations to acyclic services, meaning that they cannot return to a state that they have been in before. Then we will demonstrate how an operating guideline can be transformed into its BDD representation and how the matching can be done with BDDs.

In the following, we denote a service as P and R to emphasize the role of its owner as service provider and service requester, respectively.

2 Modeling of Services

To describe services, we use *open workflow nets* (oWFN) [4], a special class of Petri nets. They are similar to workflow nets introduced in [5], however, they own special communication places to model the sending and receiving of messages. We abstract from the content of messages and focus on the sending and receiving event only. Formally, an oWFN is a Petri net $N = (S, T, F)$ with:

- there is no transition with an empty set of preplaces,
- a set $in \subseteq S$ of *input places* and a set $out \subseteq S$ of *output places*, such that $in \cap out = \emptyset$ and for all transitions $t \in T$ holds: if $p \in in$ ($p \in out$), then $(t, p) \notin F$ ($(p, t) \notin F$),
- an *initial marking* m_0 , and
- a set Ω of *final markings*.

The *interface* of N is $I = in \cup out$ and the set of the *internal places* is $J = S \setminus I$. The *inner* of N is the subnet $Inner_N =_{def} (J, T, F \cap ((J \times T) \cup (T \times J)))$. As a convention, we label a transition that is connected to an input place x (output place y) with $?x$ ($!y$). The receiving of a message via channel x is modeled through a transition $?x$ that consumes a marking from the input place x . In the same manner, sending a message through channel y produces a marking of the transition $!y$ to the output place y .

As an example for an provided service, we look at a vending machine which is described by the oWFN V represented in Fig. 1(b). The machine expects a customer to insert coins ($?\epsilon$) and to press a button for tea ($?T$) or coffee ($?C$). Afterwards, it returns the beverage ($!B$) as long as enough money has been paid. If not enough coins have been inserted, the customer does not receive the beverage, and instead, the money is returned ($!\epsilon'$). Additionally, we consider a customer who orders tea (Fig. 1(a)).

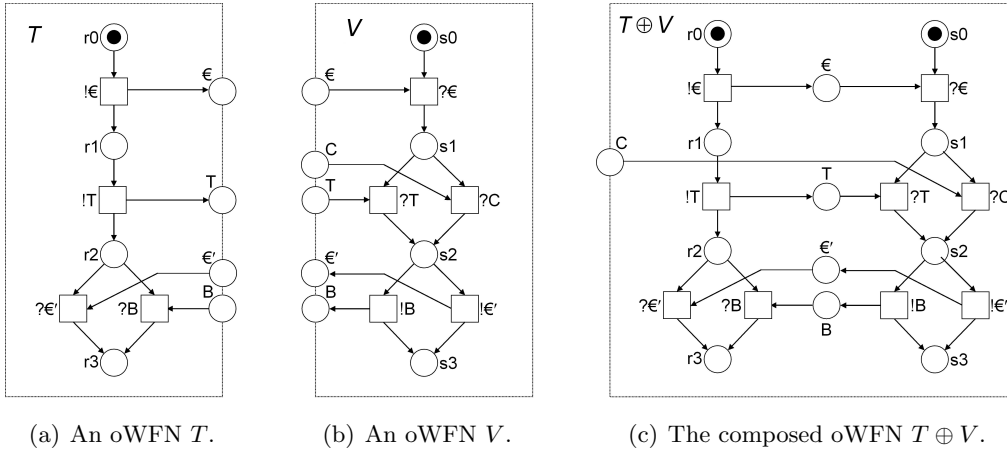


Fig. 1. Two oWFNs and their composition.

The interaction of two services can be described by the *composition* of their oWFNs. The composition of two oWFNs N and M results in the oWFN $M \oplus N =_{def} (S_M \cup S_N, T_M \cup T_N, F_M \cup F_N)$ with $in_{M \oplus N} =_{def} (in_M \setminus out_N) \cup (in_N \setminus out_M)$, $out_{M \oplus N} =_{def} (out_M \setminus in_N) \cup (out_N \setminus in_M)$ and the initial marking $m_{0_{M \oplus N}} = m_{0_M} + m_{0_N}$. A marking $m_{M \oplus N} = m_M + m_N$ is a final marking ($m_{M \oplus N} \in \Omega_{M \oplus N}$) iff $m_M \in \Omega_M$ and $m_N \in \Omega_N$. Places in $out_M \cap in_N$ or in $in_M \cap out_N$ become internal places in $M \oplus N$.

The composed system $T \oplus V$ of the two oWFN T and V is shown in Fig. 1(c). The only interface place in $T \oplus V$ is $C \in in_{T \oplus V}$. The other interface places ϵ , ϵ' , T and B were

melted into an internal place in $T \oplus V$. The only final marking is m with $m(r3) = m(s3) = 1$ (all other places are not marked in m).

Not every arbitrary service is appropriate to interact with a given service. The communication between two oWFN R and P proceeds without errors iff all deadlocks are final markings in $R \oplus P$. In this case, we call R a *strategy* for P . In our example, T is a strategy for V .

3 Operating Guidelines

If a provider publishes all strategies for his service, the broker only needs to examine if the service of a requester conforms with one of the strategies. In the following we will give a compact description for the set of all strategies of a service. We call this *operating guideline*.

The operating guideline OG_P of an oWFN P describes how an oWFN R has to behave in order to be a strategy for P . The *behavior* of an ordinary Petri net can be represented by its reachability tree. The subnet $Inner_P$ of P is a Petri net that we compute the reachability tree for. Since we consider only acyclic services, its reachability tree is finite.

In addition, edges in the reachability tree whose corresponding transitions in P are connected with an output or input place are labeled with $!x$ respectively $?x$. All other edges get the label τ (internal transitions). If the labels of outgoing edges of each node are pairwise different and there are no edges labeled with τ , the *behavior* is deterministic. For simplification we will only look at oWFN with deterministic behavior in the following.

Figure 2 depicts the behaviors B_T and B_V of the oWFN T and V of Fig. 1. B_T as well as B_V are deterministic.

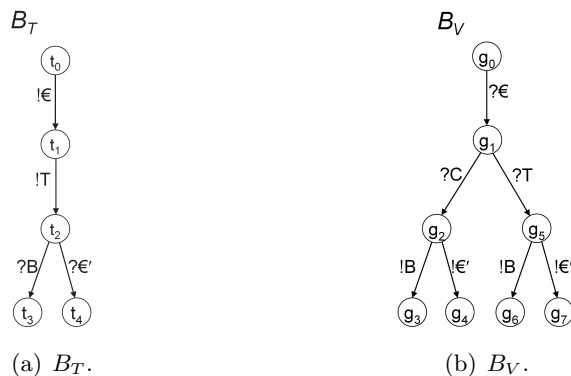


Fig. 2. The behaviors B_T of the oWFN T and B_V of the oWFN V .

We call a strategy with deterministic behavior a *deterministic strategy*. Let \mathcal{B}_P be the set of behaviors of all deterministic strategies for P . A behavior B is more *permissive* than a behavior B' if B' is isomorphic to one of the subtrees of B that starts at the root. We have shown in [6] that there exists a unique most *permissive behavior* B^* in \mathcal{B}_P for each oWFN. Hence, we call each oWFN R whose behavior is equal to the most permissive behavior of P ($B_R = B^*$), the *most permissive strategy* for P . In [7] we have further shown that the behavior of each (deterministic) strategy for P is isomorphic to a subtree starting in the root of the most permissive behavior B^* . Thus B^* provides the basis for the operating guideline.

Unfortunately, not every subtree of B^* is the behavior of a strategy for P . To detect and exclude invalid subtrees, we annotate every node of B^* with a Boolean formula. The annotation to node q expresses conditions on edges leaving q . If edges are removed from B^* , then the resulting subtree is the behavior of a strategy of P iff the assignment that assigns *true* just to the still present q leaving edges evaluates the annotation to *true*.

The operating guideline $OG_P = (B^*, \Sigma)$ of the oWFN P is thus compounded by two parts: the behavior B^* , that contains the structure of the operating guideline, and the function Σ , that assigns an annotation to every node in B^* . In [7] we described the algorithm for the automatic generation of operating guidelines in detail and proved its correctness.

Figure 3(a) shows the operating guideline OG_V for the oWFN V . We can now easily recognize that the oWFN T is a strategy for V . Firstly, the behavior B_T (see Fig. 2) is a subtree of OG_V starting at the root. Secondly, B_T owns all edges required by the annotations. The behavior of the second customer who inserts coins, presses the button $?T$, and receives the ordered beverage is also a subtree of OG_G . However, he is not a strategy for the vending machine because he cannot take back his inserted coins and therefore violates the annotation $?B \wedge ?\epsilon'$ with his behavior. The graph in Fig. 3(b) can be used as operating guideline for the oWFN V , too. The nodes with equal continuation are merged with each other.

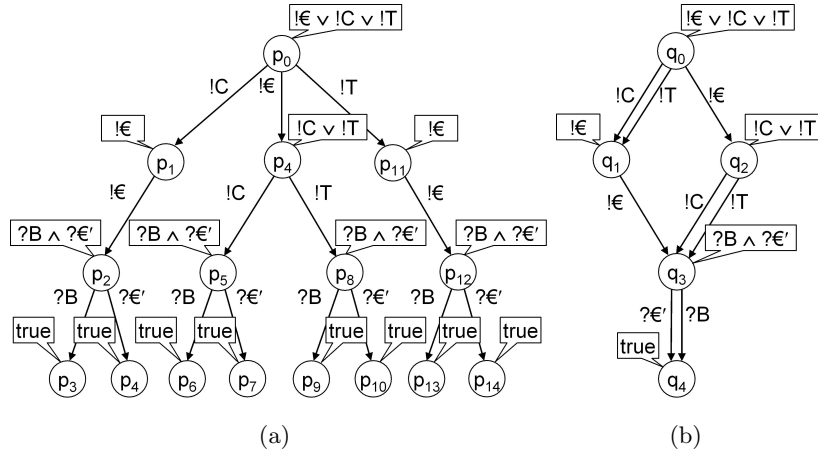


Fig. 3. The operating guideline OG_V of V as a tree (a) and as a graph (b).

4 BDD Representation of Operating Guidelines

The size of operating guidelines grows exponentially in the size of the interface of the corresponding oWFN so that in practice they cannot be stored explicitly. Thus it is necessary to look for a compact representation. For this purpose we will use *binary decision diagrams* (BDDs) [8]. They enable a compact representation of Boolean functions and allow an efficient implementation of Boolean operations like conjunction and disjunction.

We suggest to separately code the structure and the annotation of the operating guideline in one BDD, respectively. Therewith, we get one $BDD_{structure}$ and one $BDD_{annotation}$ which, both together, contain the content of the operating guideline. In [8] it was shown that there is a BDD representation for every Boolean function. Hence, we only explain the

computation of the corresponding Boolean functions $f_{structure}$ and $f_{annotation}$ for the two BDDs.

In order to code the structure of the operating guideline uniquely it is sufficient to represent the set of all edges of B^* in the function $f_{structure}$. Each edge $[q_1, l, q_2]$ consisting of the two nodes q_1 and q_2 and the label l can be represented by the word " $q_1 l q_2$ ". If every node and every label of B^* gets a unique binary code, we can describe each edge by an string consisting of zeros and ones. This we interpret as a Boolean assignment. The Boolean function $f_{q_1 l q_2}$ that belongs to $[q_1, l, q_2]$ is therefore a conjunction of negated and non negated variables according to the assignment. Finally, to get the Boolean function $f_{structure}$ we construct the disjunction over all functions $f_{q_i l_j q_k}$ of the edges.

Figure 4 shows one possible coding for the nodes and labels of the operating guideline OG_V depicted in Fig. 3(b). For example, the edge $[q_0, !T, q_1]$ is represented by the Boolean function $f_{q_0 !T q_1} = \overline{x_1} \overline{x_2} \overline{x_3} \overline{x_4} x_5 \overline{x_6} \overline{x_7} \overline{x_8} x_9$ whereas the Boolean variables x_1, x_2, x_3 stand for the node q_0 , the Boolean variables x_4, x_5 stand for the label $!T$ and the Boolean variables x_7, x_8, x_9 stand for q_1 . The rest of the edges are coded in the same manner. Thus, with the function $f_{structure, G} = f_{q_0 !C q_1} \vee f_{q_0 !T q_1} \vee f_{q_0 !E q_2} \vee f_{q_1 !E q_3} \vee f_{q_2 !C q_3} \vee f_{q_2 !T q_3} \vee f_{q_3 ?E' q_4} \vee f_{q_3 ?E q_4}$ we get a binary code of the structure of OG_G .

node	q_0	q_1	q_2	q_3	q_4
binary representation	000	001	010	011	100

(a)

label	$!C$	$!E$	$!T$	$?E'$	$?B$
binary representation	000	001	010	011	100

(b)

Fig. 4. Coding of the nodes (a) and labels (b) of the operating guideline OG_V .

In the function $f_{annotation}$, the annotations of the operating guideline are coded together with the corresponding nodes. To represent the nodes we use same binary coding as for the function $f_{structure}$. This is how, for every node q , we obtain the node function f_q . As a next step, by conjunction of f_q and the annotation $\Sigma(q)$ we construct a Boolean function $f_{q, \Sigma(q)}$ representing the node q and its annotation. The labels now are not coded through a binary number anymore but are Boolean variables in $f_{q, \Sigma(q)}$ themselves. After all we receive the $f_{annotation}$ through the disjunction of all functions.

In the operating guideline of the oWFN V in Fig. 3, e.g., the annotation $\Sigma(q_0)$ of the root node q_0 is coded by the function $f_{q_0, \Sigma(q_0)} = \overline{x_1} \overline{x_2} \overline{x_3} (!E \vee !C \vee !T)$. Accordingly, with the function $f_{annotation, V} = f_{0, \Sigma(0)} \vee f_{1, \Sigma(1)} \vee f_{2, \Sigma(2)} \vee f_{3, \Sigma(3)} \vee f_{4, \Sigma(4)}$ we get a binary coding of the annotations of OG_V .

Now, we can transform the operating guideline into its BDD representation by creating the respective BDDs from the two functions $f_{structure}$ and $f_{annotation}$.

In the following we will demonstrate the efficiency of BDDs. Examined are the operating guidelines of oWFNs whose behavior only consists of sending a definite number of messages and operating guidelines of oWFNs which model the philosopher problem. Figure 5 shows how much memory our tool Fiona³ [9] requires to represent the operating guidelines in the BDD representation and in the explicit representation. For all large operating guidelines, the BDD representation is much more compact than the explicit representation.

³ Available at <http://www.informatik.hu-berlin.de/top/tools4bpe1>.

oWFN	required memory (in KB)		ratio
	BDD	explicit	BDD/explicit
sequence3	288	188	1.53
sequence6	312	248	1.26
sequence9	384	840	0.46
sequence12	408	7084	0.06
sequence15	456	68876	0.01
philosophers3	376	224	1.68
philosophers5	500	1424	0.35
philosophers7	3976	19776	0.20

Fig. 5. Comparison of operating guidelines in explicit and BDD representation.

5 Matching

To decide whether an oWFN R is a strategy for a given oWFN P , it only has to be tested whether the behavior B_R is a subgraph of the operating guideline OG_P and B_R owns all edges given by the annotation in OG_P (see Sect. 3). If the operating guideline is coded as a Boolean formula (or represented by BDDs), then the edges and annotations cannot simply be read as an annotated graph in the explicit form. To still be able to execute the matching, we transform B_R into a binary representation as well. As in the case of operating guidelines, we use two functions to represent B_R : the structure is coded in a function f_S , whereas the second function f_L represents for every node k the labels of edges leaving k .

The nodes and labels in B_R contain a binary number again so that F_S can be computed with the same procedure as the function $f_{structure}$. To execute the matching efficiently, the coding is not chosen arbitrarily but goes by the operating guideline OG_P that has to be matched: same labels and nodes get the same binary number. A node k in B_R is equal to a node q in OG_P if the path leading to k is also a path leading to q . The coding of the labels can easily be retrieved from the table that is used to code the labels of the operating guideline. For the nodes this procedure is not possible. By a depth-first search through B_R beginning at the starting node though the coding of the nodes can be determined by using Boolean operators on $f_{structure}$ which takes time linear in the size of R [10].

To compute the function F_L , we annotate every node k with a Boolean function $\Psi(k)$ in B_R as well. This function is always dependent on the labels in the operating guideline. The labels (negated or non negated) are tied to each other through conjunction. A label l in $\Psi(k)$ is non negated if there is an edge $[k, l, k']$ in B_R , otherwise l is negated. The nodes together with the annotation can now be represented by the function f_L , like in the case of operating guidelines.

As an example we will code the behavior B_T from Fig. 2a concerning the operating guideline OG_P from Fig. 3(b). For the labels in B_T , we apply the table from Fig. 4(b) which was used already for coding the labels of the operating guideline. The coding of the nodes is shown in Fig. 6. For the edge $[t_0, !\in, t_1]$, e.g., the Boolean function $f_{t_0! \in t_1} = \overline{x_1} \overline{x_2} \overline{x_3} \overline{x_4} \overline{x_5} x_6 \overline{x_7} x_8 \overline{x_9}$ results. Analogously, the other edges can be coded and we finally obtain the function $f_{S,T} = f_{t_0! \in t_1} \vee f_{t_1! \top t_2} \vee f_{t_2? \text{B} t_3} \vee f_{t_2? \in' t_4}$. The annotation for the starting node t_0 is, e.g., $\Psi(t_0) = !\text{C} !\in !\top ?\text{B} ?\in'$ since only one edge labeled with $!\in$ leaves t_0 . In this manner the function $f_{t_0, \Psi(t_0)} = \overline{x_1} \overline{x_2} \overline{x_3} !\text{C} !\in !\top ?\text{B} ?\in'$ results and we can compute the function $f_L = f_{t_0, \Psi(t_0)} \vee f_{t_1, \Psi(t_1)} \vee f_{t_2, \Psi(t_2)} \vee f_{t_3, \Psi(t_3)} \vee f_{t_4, \Psi(t_4)}$.

Because of the fact that B_R is coded concerning the operating guideline OG_P that has to be matched, the matching is reduced to the question if f_S is a subfunction of $f_{structure}$ and f_L is a subfunction of $f_{annotation}$ and therefore to the verification of the two

node in B_T	t_0	t_1	t_2	t_3	t_4
node in OG_P	q_0	q_2	q_3	q_4	q_4
binary representation	000	010	011	100	100

Fig. 6. Coding of the nodes in B_T .

equations $\overline{f_S} \vee f_{structure} = 1$ and $\overline{f_L} \vee f_{annotation} = 1$. This can be executed efficiently in $O(|f_L| \cdot |f_{annotation}|)$ on the BDD representation.

6 Conclusion

An operating guideline OG_P of a service P characterizes all strategies for P . Since OG_P may become very large for real-world services, a compact representation is needed. In this paper, we presented an approach to code operating guidelines as two Boolean functions: one function represents the structure of the OG , and the other function represents the Boolean annotations. These functions can then easily be transformed into BDDs. We have shown that matching a service R with OG_P given as BDDs amounts to representing R as two BDDs, too, and then deciding a subfunction relationship which can be done efficiently on the BDD representations. Two small case studies showed that the BDD representation is indeed more compact than the explicit representation.

References

1. Gottschalk, K.: Web Services architecture overview. IBM whitepaper, IBM developerWorks (2000) <http://ibm.com/developerWorks/web/library/w-ovr/>.
2. Massuthe, P., Schmidt, K.: Operating guidelines - An automata-theoretic foundation for the service-oriented architecture. In Cai, K.Y., Ohnishi, A., Lau, M., eds.: Proceedings of the Fifth International Conference on Quality Software (QSIC 2005), Melbourne, Australia, IEEE Computer Society (2005) 452–457
3. Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services. In: Petri Nets and Other Models of Concurrency – ICATPN 2007. (2007) accepted.
4. Massuthe, P., Reisig, W., Schmidt, K.: An Operating guideline approach to the SOA. Annals of Mathematics, Computing & Teleinformatics 1(3) (2005) 35–43
5. Aalst, W.: The application of petri nets to workflow management. Journal of Circuits, Systems and Computers 8(1) (1998) 21–66
6. Schmidt, K.: Controllability of Open Workflow Nets. In Desel, J., Frank, U., eds.: Enterprise Modelling and Information Systems Architectures. Volume P-75 of Lecture Notes in Informatics (LNI)., Bonn, Entwicklungsmethoden für Informationssysteme und deren Anwendung (EMISA, RWTH Aachen), Köllen Druck+Verlag GmbH (2005) 236–249
7. Massuthe, P., Wolf, K.: An Algorithm for Matching Nondeterministic Services with Operating Guidelines. In Leymann, F., Reisig, W., Thatte, S.R., Aalst, W.M.P.v.d., eds.: The Role of Business Processes in Service Oriented Architectures. Number 06291 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (2006)
8. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers C-35(8) (1986) 677–691
9. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing Interacting BPEL Processes. In Dustdar, S., Fiadeiro, J.L., Sheth, A., eds.: Forth International Conference on Business Process Management (BPM 2006), 5–7 September 2006 Vienna, Austria. Volume 4102 of Lecture Notes in Computer Science., Springer-Verlag (2006) 17–32
10. Kaschner, K.: BDD-basiertes Matching von Services. Diplomarbeit, Humboldt-Universität zu Berlin (2006)