# Decompositional Computation of Operating Guidelines Using Free Choice Conflicts

Niels Lohmann*

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

**Abstract.** An operating guideline (OG) for a service $S$ finitely characterizes the (possibly infinite) set of all services that can interact with $S$ without deadlocks. This paper presents a decompositional approach to calculate an OG for a service whose underlying structure is acyclic and contains free-choice conflicts. This divide-and-conquer approach promises to be more efficient than the classical OG computation algorithm.

## 1 Introduction

In the paradigm of service-oriented computing, a *service* is a component that offers a functionality over a well-defined interface and is discoverable and accessible via a unique identifier. By composing several services, complex tasks (e. g., inter-organizational business processes) can be realized. Thereby, the correct interplay of distributed services is crucial to achieve a common goal.

Recent literature [1] proposed an *operating guideline* (OG) of a service $S$ as finite characterization of all (partner) services that communicate correctly (i. e., without deadlocks or livelocks) with $S$. Applications of OGs include the realization the "find" and "publish" operations of service brokering, as well as the analysis, construction, and correction of services. Unfortunately, the algorithm to calculate an OG for a service has exponential complexity in both the service's state space and the size of the interface. In this paper, we propose a decompositional divide-and-conquer approach to calculate OGs for service models that contain free choice conflicts.

In Sect. 2, we recall some necessary definitions. Section 3 introduces decomposition of service models and describes how OGs can be calculated decompositionally. In Sect. 4, we analyze which constructs of industrial specification languages meet the requirements of the decomposition. Section 5 concludes the paper and discusses future work.

## 2 Background

We use *open nets* [2] to model services. Open nets extend classical Petri nets [3] with an interface $I = (P_{in} \cup P_{out}) \subseteq P$ to explicitly model asynchronous message exchange and a set of final markings $\Omega$ modeling desired final states of the service. Two open nets $N$ and $M$ can be *composed* (denoted by $N \oplus M$) by merging their interfaces accordingly ($N$'s input places with $M$'s output places, and vice versa). Thereby, the *inner structures* of $N$ and $M$ (i.e., the open net without interface)

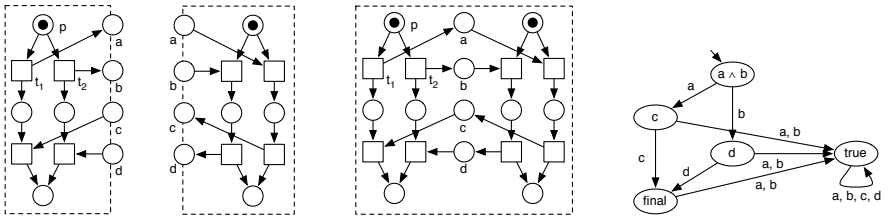
* funded by the DFG project "Operating Guidelines for Services" (WO 1466/8-1)

are assumed to be disjoint. An open net is *acyclic* if the reachability graph of its inner structure is acyclic. An open net *weakly terminates* if, from every reachable marking, a final marking is reachable.

**Definition 1 (Controllability, strategy).** *Let $N$ be an open net. $N$ is* controllable*, iff there exists an open net $M$ such that $N \oplus M$ is weakly terminating. Then $M$ is called a* strategy *for $N$. Denote the set of all strategies for $N$ by $Strat(N)$.*

In [1], the concept of an *operating guideline* (OG) was introduced. The operating guideline $OG_N$ for a service $N$ is a finite automaton whose states are annotated with Boolean formulae. It characterizes a (possibly infinite) set of services, denoted by $Comply(OG_N)$. In fact, it *exactly* characterizes the set of strategies of $N$.

**Theorem 1 ([1]).** *Let $OG_N$ be an operating guideline for an open net $N$. Then $Comply(OG_N) = Strat(N)$.*




(a) N (b) M (c) N ⊕ M (d) $OG_N$

**Fig. 1.** Open net N, strategy M, composition N ⊕ M, and operating guideline $OG_N$

*Example* Figure 1 depicts an open net N. The net is controllable, as there exists a strategy M which first receives either an a or a b message and then responds with a c or a d message, resp. This and all other strategies are characterized by the operating guideline $OG_N$. The conjunction a ∧ b annotated to the initial node states that a strategy must be ready to initially both receive an a message *and* a b message. The node with the true formula is a technical necessity. Though it will never be reached in a composition (e. g., after having received an a message, the further receipt of either a or b is impossible, because N will not send these messages), such respective branches may be still part of a strategy, because they do not jeopardize weak termination.

## 3 Decomposition

In a Petri net, a place with more than one transition in its postset models a conflict.

**Definition 2 (Conflict cluster, free choice).** *Let $x \in P \cup T$ be a node of a net. The* conflict cluster *$x$, denoted by $[x]$, is the minimal set of nodes such that:*

- $x \in [x]$.

– *If* $p \in [x]$ *for a place* $p \in P$*, then* $p^\bullet \subseteq [x]$.
– *If* $t \in [x]$ *for a transition* $t \in T$*, then* ${}^\bullet t \subseteq [x]$.

$[x]$ *is* free choice *if for all* $t, t' \in [x] \cap T$ *holds: either* ${}^\bullet t \cap {}^\bullet t' = \emptyset$ *or* ${}^\bullet t = {}^\bullet t'$.

Given a marking of a net, this marking either enables all transitions in a free choice conflict cluster or none of them. We exploit this property by using the transitions of a free choice conflict cluster to decompose the net. For each possible outcome of the conflict, we define one net in which only this transition is present and all others are removed together with their adjacent arcs.

**Definition 3 (Decomposition).** *Let* $N = [P, T, F, m_0, \Omega]$ *be an open net and* $C$ *a free choice conflict cluster of* $N$ *with* $C \cap T = \{t_1, \ldots, t_m\}$*. The* decomposition *of* $N$ *w.r.t.* $C$ *is the set* $\{N_1, \ldots, N_m\}$ *with* $N_i = [P, T \setminus \{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m\}, F \setminus ((P \times \{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m\}) \cup (\{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m\} \times P)), m_0, \Omega]$*, for* $i \in \{1, \ldots, m\}$.

**Theorem 2.** *Let* $N$ *be a safe acyclic open net and* $\{N_1, \ldots, N_m\}$ *be its decomposition w.r.t. a free-choice conflict cluster* $C$ *of* $N$ *with* $C \cap T = \{t_1, \ldots, t_m\}$*. Then* $\mathit{Strat}(N) = \bigcap_{i=1}^{m} \mathit{Strat}(N_i)$.

*Proof.* We prove the equality by showing mutual set inclusion.

$\subseteq$: Let $M \in \mathit{Strat}(N)$. We will show by contradiction that $M \in \bigcap_{i=1}^{m} \mathit{Strat}(N_i)$. Assume $M \notin \bigcap_{i=1}^{m} \mathit{Strat}(N_i)$. Then $M \oplus N_i$ contains a deadlock for a net $N_i$. Let $m_0 \xrightarrow{\sigma} m_d$ be a transition sequence to this deadlock in $M \oplus N_i$. This sequence is also realizable in $M \oplus N$. There, $m_d$ might activate a transition not present in $M \oplus N_i$, which can only be a transition in $(C \cap T) \setminus \{t_i\}$. As $C$ is a free choice conflict cluster, $m_d$ also activates $t_i$ in $M \oplus N_i$ which contradicts the assumption that $M \oplus N_i$ contains a deadlock. Consequently, $M \oplus N_i$ is deadlock free and $M \in \mathit{Strat}(N_i)$. Repeating the arguments, we can conclude $M \in \bigcap_{i=1}^{m} \mathit{Strat}(N_i)$.

$\supseteq$: Let $M \in \bigcap_{i=1}^{m} \mathit{Strat}(N_i)$. We will show by contradiction that $M \in \mathit{Strat}(N)$. Assume $M \notin \mathit{Strat}(N)$. Then $M \oplus N$ contains a deadlock. Let $m_0 \xrightarrow{\sigma} m_d$ be a transition sequence to this deadlock in $M \oplus N$. There are two cases:
  – $\sigma$ contains a transition of $C \cap T$. Then, by definition of the decomposition, there exists a net $N_i$ such that $\sigma$ is realizable in $M \oplus N_i$, because due to safeness and acyclicity, transitions in $C \cap T$ can occur at most once in $\sigma$.
  – $\sigma$ contains no transition of $C$. Then $\sigma$ is realizable in $M \oplus N_i$, for any $1 \leq i \leq n$.

  Both cases would contradict the assumption that $M \in \bigcap_{i=1}^{m} \mathit{Strat}(N_i)$. Hence, $M \oplus N$ is deadlock free and $M \in \mathit{Strat}(N)$.

□

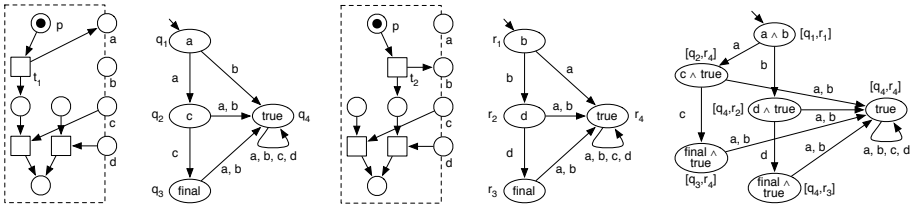
Theorem 1 describes the relationship between the strategy set of a service and its OG. The intersection of strategy sets can be related to OGs using the *product operator* [4]. The product of two operating guidelines $OG_N$ and $OG_M$, denoted by $OG_N \otimes OG_M$, is constructed similar to the product automaton for classical finite automata. In addition, the formula annotated to a state $[q_1, q_2]$ of the product is defined to be the conjunction of the formula annotated to $q_1$ and that of $q_2$.

**Theorem 3 ([4]).** *Let $OG_N$, $OG_M$ be operating guidelines. Then $Comply(OG_N \otimes OG_M) = Comply(OG_N) \cap Comply(OG_M)$.*

This result allows us to express Theorem 2 in terms of operating guidelines:

**Corollary 1.** *Let $N$ be a safe acyclic open net and $\{N_1, \ldots, N_m\}$ be its decomposition w.r.t. a free-choice conflict cluster $C$ of $N$ with $C \cap T = \{t_1, \ldots, t_m\}$. Then $Comply(OG_N) = Comply(OG_{N_1} \otimes \cdots \otimes OG_{N_m})$.*

We are now able to calculate an operating guideline for $N$ by calculating the operating guidelines for the decomposition of $N$, followed by calculating the product of the operating guidelines. Note that Theorem 2 does not require the whole net to be free choice, but only the conflict cluster under consideration.




(a) $\mathsf{N}_1$ (b) $OG_{\mathsf{N}_1}$ (c) $\mathsf{N}_2$ (d) $OG_{\mathsf{N}_2}$ (e) $OG_{\mathsf{N}_1} \otimes OG_{\mathsf{N}_2}$

**Fig. 2.** Decomposed open nets $\mathsf{N}_1$ and $\mathsf{N}_2$ with operating guidelines $OG_{\mathsf{N}_1}$ and $OG_{\mathsf{N}_2}$ and the product operating guideline $OG_{\mathsf{N}_1} \otimes OG_{\mathsf{N}_2}$

*Example (cont.)* The net $\mathsf{N}$ in Fig. 1 contains a free choice conflict cluster $\{\mathsf{p}, \mathsf{t}_1, \mathsf{t}_2\}$ and can be decomposed into the nets $\mathsf{N}_1$ and $\mathsf{N}_2$, depicted in Fig. 2. The respective OGs characterize the strategies for the decomposed nets. To determine the intersection of these strategy sets, the product $OG_{\mathsf{N}_1} \otimes OG_{\mathsf{N}_2}$ needs to be constructed. As described earlier, it is the product of the underlying automata, and each state is annotated with the conjunction of the respective formulae. For example, the annotation of state $[\mathsf{q}_1, \mathsf{r}_1]$ is the conjunction of the formulae of $\mathsf{q}_1$ ($\mathsf{a}$) and $\mathsf{r}_1$ ($\mathsf{b}$). The resulting product $OG_{\mathsf{N}_1} \otimes OG_{\mathsf{N}_2}$ is equivalent to $OG_{\mathsf{N}}$ (cf. Fig. 1); that is, it characterizes the same set of strategies.

The advantage of the decompositional OG calculation is the reduced complexity of the intermediate results. Though the OGs of the decomposed nets might have more nodes, the state space of the decomposed nets is usually much smaller. Furthermore, the product operator's associativity allows to interleave the OG calculation and the product construction.

## 4 Applications

Though the requirements of Theorem 2 (safeness, acyclicity) are very restrictive, the decompositional approach to calculate an OG can still be used for industrial specification languages. In the following, we evaluate which features of the languages BPEL [5], BPMN [6], and UML2 [7] activity diagrams may be used to while still meeting the requirements of Theorem 2.

**BPEL** For BPEL there exists a feature-complete Petri net semantics [8] which allows to translate a BPEL process into safe open nets. The net is cyclic only if activities for repetitive execution (`while`, `repeatUntil`, and sequential `forEach`) or event handlers are used in the process. Several patterns contain conflicts of which many are not free-choice. However, the following are:

- decisions modeled with the `if` activity (in case XPath errors are not modeled),
- transition conditions to set control links within a `flow` activity, and
- leaving the process's positive control flow (throwing a the first fault).

Furthermore, conflicts can depend on each other. For example, whether or not to skip an activity during dead path elimination is a non-free choice, yet dependent on the setting of the respective control links, which in turn is a free choice decision. Hence, when decomposing the net using such a "dominant" conflict, several "dependent" decisions become deterministic.

**BPMN** Dijkman et al. [9] defined a Petri net semantics for a subset of BPMN. The resulting Petri net is safe if the control flow does not contain a *lack of synchronization*. This situation can arise if gateways are not nested properly (e.g., the control flow splits using an AND-gateway, but joins using an XOR-gateway). Such models contain obvious design flaws.

The nets are acyclic if the control flow is acyclic and no activities with explicit loop annotation are used. Again, many occurring conflicts are not free-choice, especially when exception flow is modeled. However, decision gateways can be translated into free choice conflicts.

**UML-AD** UML2 activity diagrams have a very close relationship to BPMN and Petri nets. An activity diagram can be translated into an acyclic Petri net if its control flow is acyclic. In case the diagram contains no lack of synchronization, the translation results a safe Petri net. Additionally, the whole net (i. e., every conflict cluster) is free choice if no pinsets are used.

**Table 1.** Language constructs for acyclic safe Petri nets and free choice conflicts.

| language | acyclic Petri net | safe Petri net | free choice conflicts |
|---|---|---|---|
| BPEL | ✘ `while` | ✔ always safe | ✔ `if` branches |
| | ✘ `repeatUntil` | | ✔ transition conditions |
| | ✘ sequential `forEach` | | ✔ throwing first fault |
| | ✘ event handlers | | ✘ `pick` |
| BPMN | ✘ loop activities | ✘ lack of synchronization | ✔ data-based gateways |
| | ✔ acyclic control flow | | ✔ inclusive gateways |
| | | | ✔ timeout event gateway |
| UML-AD | ✔ acyclic control flow | ✘ lack of synchronization | ✔ all, if no pinsets are used |

Table 1 summarizes the language constructs that are forbidden or that guarantee acyclic and safe Petri nets, and that yield free choice conflicts. The latter constructs can be used to discover free choice conflict clusters already

during the translation of a process described in BPEL, BPMN or UML-AD into Petri nets. This allows for avoiding an a-posteriori discovery of free choice conflict clusters.

We implemented the described decomposition approach in the compiler BPEL2oWFN [8] which translates a BPEL process into a set of decomposed open nets. For these nets, the OGs can be calculated using the tool Fiona [10], which also implements the calculation of product operating guidelines.[1]

## 5 Conclusion

We presented a decompositional approach that uses free-choice conflict clusters to decompose a safe acyclic open net. The operating guidelines for the resulting nets can be calculated independently and subsequently merged using the product operator. Hence, the calculation can be seen as a divide-and-conquer approach to calculate operating guidelines.

Both the calculation of the OGs for the decomposed nets and the product operators are currently implemented to cope with arbitrary nets and OGs, resp. In future work we plan to adjust these algorithms to exploit the simpler structure of the intermediate constructs. In particular, we plan to study free choice open net, because they can be decomposed into conflict free open nets for which the OG construction should be less complex.

In addition, the requirements of Theorem 2 might be relaxed. For example, the theorem still holds if every transition sequence marks the conflict cluster at most once. This requirement can also be fulfilled by open nets which are cyclic or non-safe.

## References


1. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: ICATPN 2007. Volume 4546 of LNCS., Springer (2007) 321–341
2. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the SOA. AMCT **1**(3) (2005) 35–43
3. Reisig, W.: Petri Nets. EATCS Monographs on Theoretical Computer Science edn. Springer (1985)
4. Lohmann, N., Massuthe, P., Wolf, K.: Behavioral constraints for services. In: BPM 2007. Volume 4714 of LNCS., Springer (2007) 271–287
5. Alves, A., et al.: Web Services Business Process Execution Language Version 2.0. OASIS Standard, 11 April 2007, OASIS (2007)
6. OMG: Business Process Modeling Notation (BPMN) Version 1.0. OMG Final Adopted Specification, OMG (2006)
7. OMG: Unified Modeling Language (UML). Version 2.1.2, OMG (2007)
8. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In: WS-FM 2007. Volume 4937 of LNCS., Springer (2008) 77–91
9. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology (2008) (Accepted for publication).
10. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting BPEL processes. In: BPM 2006. Volume 4102 of LNCS., Springer (2006) 17–32


[1] Both tools are available for download at `http://service-technology.org/tools`.