

# Multiparty Contracts: Agreeing and Implementing Interorganizational Processes

Wil M.P. van der Aalst<sup>1</sup>, Peter Massuthe<sup>2</sup>,  
Christian Stahl<sup>2,\*</sup>, and Karsten Wolf<sup>3</sup>

<sup>1</sup> Department of Mathematics and Computer Science  
Technische Universiteit Eindhoven  
P.O. Box 513, 5600 MB Eindhoven, The Netherlands  
W.M.P.v.d.Aalst@tue.nl

<sup>2</sup> Humboldt-Universität zu Berlin, Institut für Informatik  
Unter den Linden 6, 10099 Berlin, Germany  
{massuthe, stahl}@informatik.hu-berlin.de

<sup>3</sup> Universität Rostock, Institut für Informatik  
18051 Rostock, Germany  
karsten.wolf@uni-rostock.de

**Abstract.** A *contract* specifies an interorganizational process together with a distribution of responsibilities for the activities among the parties involved. In this paper, we formally show how a party can implement its part of the contract such that the implementation accords with the contract. We propose a formal notion of a contract and give a criterion for *accordance* between a local implementation and a contract such that, if all local implementations accord with the contract, the overall process is deadlock-free and it is always possible to terminate properly. Then, we sketch a technique for automatically checking the proposed accordance criterion. Finally, we present accordance-preserving transformation rules. These rules can be used to implement a part of the contract while preserving the accordance criterion.

## 1 Introduction

Today's corporations often must operate across organizational boundaries. Phenomena such as e-commerce, extended enterprises, and service-oriented computing stimulate cooperation between organizations [1–8]. Therefore, the importance of workflows distributed over a number of organizations is increasing [9–12]. Interorganizational workflow support offers companies the opportunity to re-shape business processes beyond the boundaries of their own organizations. However, interorganizational workflows are typically subject to conflicting constraints. On the one hand, there is a strong need for coordination to optimize the flow of work in and between the different organizations. On the other hand, the organizations involved are essentially autonomous and have the freedom to create or modify

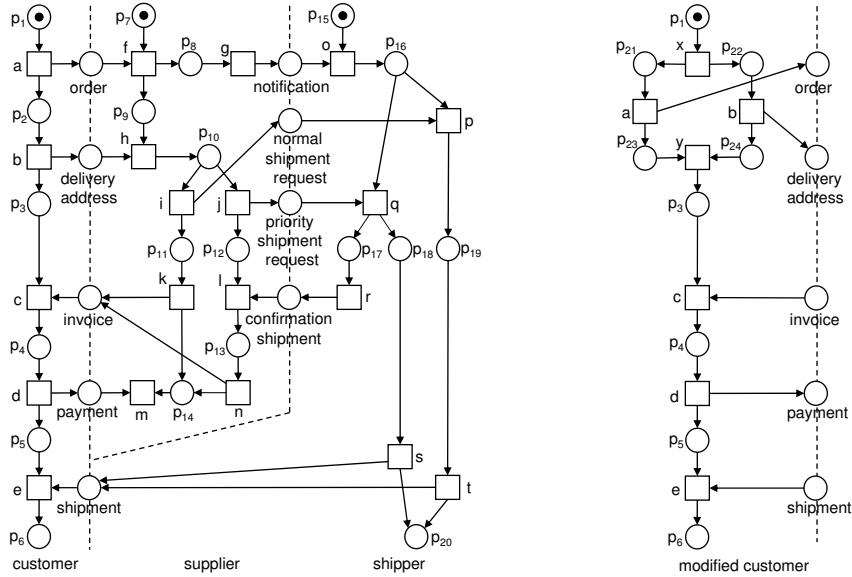
---

\* Funded by the DFG project “Substitutability of Services” (RE 834/16-1).

workflows at any point in time. To address this issue, the notion of a *contract* in the form of an agreed upon process model was introduced in [9, 10], for instance.

To illustrate the idea of having a contract, we use the example shown in Fig. 1(a). The example shows a contract expressed in terms of a Petri net [13].

The Petri net is partitioned over three *parties*: *customer*, *supplier*, and *shipper*. Each party has a part of the contract which can be seen as a *service*. Different services are connected through *interface places* that model asynchronous message passing. Interface places model message buffers and are depicted on dashed lines in Fig. 1(a). As a formalism, we use *open workflow nets* (oWFNs) [14] which extend the well known concept of *workflow nets* (WFNs) [15] with interface places. However, the concepts are not limited to oWFNs and can be translated into other languages using message passing as a communication paradigm.



(a) oWFN modeling the contract between customer ( $N_{\text{cust}}$ ), supplier ( $N_{\text{supp}}$ ), and shipper ( $N_{\text{ship}}$ ).

(b) oWFN of the modified customer ( $N'_{\text{cust}}$ ).

Fig. 1. The running example.

In our example, the customer creates an order (transition a). As a result, the supplier sends a notification to the shipper (transition g). After sending the order, the customer sends the delivery address. Then, the supplier makes a choice by selecting the desired form of shipment. This is signaled to the shipper by sending either a normal shipment request or a priority shipment request. In case of a priority shipment, the shipper sends a confirmation shipment to the supplier

and only then the supplier sends an invoice to the customer. For normal orders, there is no confirmation and the invoice can be sent immediately. Then, the customer pays (**payment**) and receives the **shipment**.

Assuming a contract as shown in Fig. 1(a), each party will implement its part (its *public view*) of the contract. Clearly, the implemented version, the *private view*, may deviate significantly from the public view. For example, the task represented by transition **a** may be implemented by a subprocess containing dozens of detailed activities that are only of local interest. Obviously, this local modification has to accord with the contract. In [10], however, it was shown that this local modification is a nontrivial task as it may cause global errors such as deadlocks. For example, changing the order of transitions **d** and **e** in Fig. 1(a) leads to a deadlock in the process. To guide the user during the modification process, in earlier work [16, 17], it has been proposed to use *projection inheritance* for WFNs for relating the actual realization of a contract to the contract itself [9, 10]. Projection inheritance (see [16, 17]) uses abstraction and branching bisimulation as a mechanism to establish subclass-superclass relationships between processes. In contrast to many other notions of inheritance, it primarily addresses the dynamic behavior rather than data types or method signatures. It was proven that if private and public view are related by projection inheritance, then the party can execute this private view and other parties are not effected by this change. Based on this notion, several inheritance-preserving transformation rules have been defined. These rules correspond to design patterns when extending a superclass to incorporate new behavior: (1) adding loops, (2) inserting methods in-between existing methods, and (3) putting new methods in parallel with existing methods. They can be used by a party to implement its part of the contract while preserving projection inheritance.

However, it turns out that in practice the notion of projection inheritance is too restrictive. This is mainly caused by the fact that projection inheritance looks at “methods” rather than the exchange of messages. For example, when messages are sent, their order does not really matter. This is caused by the fact that we consider asynchronous message passing; that is, messages may be consumed in a different order than they were produced. Nevertheless, projection inheritance will differentiate between the different orderings of sending messages. As an example,  $N'_{\text{cust}}$  in Fig. 1(b) is a valid implementation of the net  $N_{\text{cust}}$  in Fig. 1(a). In  $N_{\text{cust}}$ , the customer first sends the order and then the address information. In  $N'_{\text{cust}}$ , these two messages are sent concurrently. However, this change does not raise any problems for the other parties. Consequently, any environment that can cooperate with the public view can also cooperate with this private view. However,  $N_{\text{cust}}$  and  $N'_{\text{cust}}$  are not related by projection inheritance.

The contribution of this paper is threefold. Firstly, we present a more generic notion of a contract using oWFNs. In contrast to WFNs, oWFNs explicitly model the interface and have less syntactical restrictions. Secondly, we define a notion of accordance between an implementation of a contract and the contract itself. An oWFN  $N'$  *accords* with an oWFN  $N$  if it has the same set of interface places and any environment that can cooperate with  $N$  can also cooperate with

$N'$ . Moreover, we show how accordance can be computed for acyclic finite state oWFNs. Thirdly, we prove that projection inheritance implies accordance. As a consequence, inheritance-preserving transformation rules presented in [9, 10, 16, 17] imply accordance. However, we also provide additional (more powerful) transformation rules that guarantee accordance but do not preserve projection inheritance.

The remainder of this paper is structured as follows. Section 2 defines oWFNs and contracts. The notion of accordance is defined in Sect. 3 where it is applied to relate the private and public views of a contract. Section 4 presents an algorithm to decide accordance using operating guidelines. Section 5 discusses the relation between projection inheritance and accordance and accordance-preserving transformation rules. Related work is presented in Sect. 6. Finally, Sect. 7 draws the conclusion.

## 2 Formalizing Contracts

An interorganizational process couples interacting processes handled by different parties. To design the overall process, the involved parties specify a public view of this process together with a distribution of responsibilities for the activities among them (i.e., a partitioning). The public view and its partitioning serve as a *contract*. Each party will modify its part of the process; that is, it will implement a private view of its part. Obviously, this local modification has to stand to the contract, meaning, the behavior of this part has to be preserved. In [10], it was shown that this local modification is a nontrivial task, as local changes may cause global faults such as deadlocks. Therefore, a formal framework, which guides the user during the modification process, is needed.

For specifying the overall interorganizational process and the processes it consists of, we use the concept of *open workflow nets* (oWFNs) [14]. We consider the overall process as *self-contained*. This fact is reflected by its representation as an oWFN *with empty interface*. An immediate alternative to our approach would be to represent a process within the much more established framework of workflow nets. This framework would, however, cause more involved technicalities for the approach presented below. Thus, we use oWFNs in the technical parts of the paper and discuss the implications for workflow nets (e.g., to [18]) in a more pragmatic fashion later on.

We start with the definition of classical (place/transition) Petri nets (see [13], for instance) and define then oWFNs.

**Definition 1 (Petri net).** *A Petri net  $N = (P, T, F, m_0)$  consists of*

- *two finite and disjoint sets  $P$  and  $T$  of places and transitions,*
- *a flow relation  $F \subseteq (P \times T) \cup (T \times P)$ , for which we introduce the following notation to denote the preset and postset of places and transitions:  $\bullet x = \{y \mid (y, x) \in F\}$  and  $x^\bullet = \{y \mid (x, y) \in F\}$ , and*
- *an initial marking  $m_0$ , where a marking is a mapping  $m : P \rightarrow \mathbb{N}$ .*

As usual, places are depicted as circles, transitions as boxes, the flow relation as arrows, and markings as distributions of (black) tokens on the places. The behavior of Petri nets is defined next. A transition is enabled if each place of its preset holds at least a token. An enabled transition  $t$  can fire in a marking  $m$  by consuming tokens from the preset places and producing tokens for the postset places, yielding a marking  $m'$ .

**Definition 2 (Behavior of Petri nets).** *Let  $N = (P, T, F, m_0)$  be a Petri net. Transition  $t \in T$  is enabled in marking  $m$  if, for all  $p \in \bullet t$ ,  $m(p) > 0$ . If  $t$  is enabled, it can fire leading to marking  $m'$ , where  $m'(p) = m(p) - 1$  for  $p \in \bullet t \setminus t^\bullet$ ,  $m'(p) = m(p) + 1$  for  $p \in t^\bullet \setminus \bullet t$ , and  $m'(p) = m(p)$ , otherwise. The described firing relation is denoted  $m \xrightarrow{t} m'$ . A marking  $m'$  is reachable from a marking  $m$  if there is a sequence  $t_1, \dots, t_n$  of transitions and a sequence  $m = m_1, \dots, m_{n+1} = m'$  of markings such that, for all  $i \in \{1, \dots, n\}$ ,  $m_i \xrightarrow{t_i} m_{i+1}$ . With  $RG_N(m)$  we denote the set of markings that can be reached from  $m$  by firing any number of transitions.*

Now we define oWFNs, a special class of Petri nets.

**Definition 3 (Open workflow net).** *An open workflow net is a Petri net  $N = (P, T, F, m_0)$  together with*

- *an interface defined as a set  $I \subseteq P$  of input places such that  $\bullet p = \emptyset$  for any  $p \in I$  and a set  $O \subseteq P$  of output places such that  $p^\bullet = \emptyset$  for any  $p \in O$  and  $I \cap O = \emptyset$ ,*
- *a set  $\Omega$  of final markings such that no transition of  $N$  is enabled in any  $m \in \Omega$ . We further require that  $m \in \Omega \cup \{m_0\}$  implies  $m(p) = 0$  for all  $p \in I \cup O$ ; that is, in the initial and the final markings the interface places are not marked.*

We use indices to distinguish the constituents of different oWFNs (e.g.,  $I_j$  refers to the set of input places of oWFN  $N_j$ ). As an example, the whole process shown in Fig. 1(a) represents an oWFN with  $I = O = \emptyset$ .  $N_{\text{cust}}$  in Fig. 1(a) is an oWFN with interface  $I = \{\text{invoice, shipment}\}$  and  $O = \{\text{order, delivery address, payment}\}$ .

In the definition of oWFNs, interfaces do not play a distinguishing role. This is possible because, when composing oWFNs, the interface places of one oWFN are connected to the interface places of other oWFNs. As a result, the interface places can be internalized and no special provision is needed for them. In other words, the semantics of an open system is only defined when the open system is closed by adding an appropriate environment. The role of the interface itself is clarified through the concept of *composition* further down in this section.

In order to assign a reasonable meaning to *final* markings, we restrict our approach to such oWFNs where a marking in  $\Omega$  does not enable any transition.

For the oWFN depicted in Fig. 1(a) we have  $m_0 = [\mathbf{p}_1, \mathbf{p}_7, \mathbf{p}_{15}]$  and we define  $\Omega = \{[\mathbf{p}_6, \mathbf{p}_{20}]\}$ . It is easy to check that, for any marking reachable from the initial marking shown in Fig. 1(a), the final marking is reachable. This means

that it is always possible to terminate properly. This property is formalized in the following definition.

**Definition 4 (Weak termination).** *Let  $N$  be an oWFN with empty interface ( $I = O = \emptyset$ ).  $N$  weakly terminates if, from every marking reachable from the initial marking, a final marking can be reached.*

For the composition of oWFNs, we assume that all constituents (except the interfaces) are pairwise disjoint. This requirement can be easily achieved by renaming. In contrast, the interfaces intentionally overlap. For a reasonable concept of composition of oWFNs it is, however, convenient to require that all communication is bilateral; that is, every interface place  $p \in I \cup O$  has only one party that sends into  $p$  and one party that receives from  $p$ . For a third party  $C$ , a communication taking place inside the composition of parties  $A$  and  $B$  is internal matter. These considerations lead to the following definitions of composable and composition.

**Definition 5 (Composable set of oWFNs).** *Let  $N_1, \dots, N_k$  be oWFNs with pairwise disjoint constituents, except for the interfaces.  $N_1, \dots, N_k$  are composable if, for all  $i \in \{1, \dots, k\}$ ,*

- $p \in I_i$  implies that there is no  $j \neq i$  such that  $p \in I_j$  and there is at most one  $j$  such that  $p \in O_j$ , and
- $p \in O_i$  implies that there is no  $j \neq i$  such that  $p \in O_j$  and there is at most one  $j$  such that  $p \in I_j$ .

**Definition 6 (Composition of oWFNs).** *Let  $N_1, \dots, N_k$  be a composable set of oWFNs. The composition  $N = N_1 \oplus \dots \oplus N_k$  is the oWFN with the following constituents:*

- $P = P_1 \cup \dots \cup P_k$ ,
- $T = T_1 \cup \dots \cup T_k$ ,
- $F = F_1 \cup \dots \cup F_k$ ,
- $I = (I_1 \cup \dots \cup I_k) \setminus (O_1 \cup \dots \cup O_k)$ ,
- $O = (O_1 \cup \dots \cup O_k) \setminus (I_1 \cup \dots \cup I_k)$ ,
- $m_0 = m_{0_1} \oplus \dots \oplus m_{0_k}$ , and
- $\Omega = \{m_1 \oplus \dots \oplus m_k \mid m_1 \in \Omega_1, \dots, m_k \in \Omega_k\}$ .

*For markings  $m_1 \in N_1, \dots, m_k \in N_k$  which do not mark the interface places, their composition  $m = m_1 \oplus \dots \oplus m_k$  is defined by  $m(p) = m_i(p)$  if  $p \in P_i$ .*

Fig. 1(a) can be interpreted as a single oWFN with an empty interface or as three oWFNs:  $N_{\text{cust}}$ ,  $N_{\text{supp}}$ , and  $N_{\text{ship}}$ . Places on the dashed lines are the interface places of each of the three oWFNs. We assume the final marking of the oWFNs being  $[p_6]$ ,  $[\ ]$  (i.e., the empty marking), and  $[p_{20}]$ , respectively. Clearly, the three oWFNs are composable:  $N = N_{\text{cust}} \oplus N_{\text{supp}} \oplus N_{\text{ship}}$  is the overall contract shown in Fig. 1(a).

Note that any subset of a set of composable oWFNs is composable as well. Furthermore, we have  $N_1 \oplus N_2 \oplus N_3 = (N_1 \oplus N_2) \oplus N_3 = N_1 \oplus (N_2 \oplus N_3)$ , and

$N_1 \oplus N_2 = N_2 \oplus N_1$ . In other words, the composition of oWFNs is associative and commutative. Thus, composition of a set of oWFNs can be broken into single steps without affecting the final result.

Basically, we see a contract as an oWFN where every activity is assigned to one of the involved parties. We impose only one restriction: If a place is accessed by more than one party, it should act as a directed bilateral communication place; that is, one party produces tokens, another party consumes tokens, and there is no third party accessing the place. In the following,  $|X|$  denotes the cardinality of a set  $X$ .

**Definition 7 (Contract).** *Let  $\mathcal{A}$  be a set representing the parties involved in a contract. Then, a contract  $[N, r]$  consists of an oWFN  $N = (P, T, F, I, O, m_0, \Omega)$  with an empty interface ( $I = O = \emptyset$ ) (the agreed public view of the process) and a mapping  $r \in T \rightarrow \mathcal{A}$  (the partitioning) such that, for all places  $p \in P$ ,  $|\{r(t) \mid t \in \bullet p\}| \leq 1$  and  $|\{r(t) \mid t \in p^\bullet\}| \leq 1$ . For technical purposes, we further require that  $N$  has only one final marking,  $\Omega = \{m_f\}$ .*

The oWFN shown in Fig. 1(a) is an example of a contract involving  $\mathcal{A} = \{\text{customer, supplier, shipper}\}$ . The dashed lines in the figure show the partitioning of transitions over the parties involved in the contract;  $r(a) = \text{customer}$ ,  $r(f) = \text{supplier}$  and  $r(l) = \text{shipper}$ , for instance.

A contract can be cut into parts, each representing the agreed share of a single party. Every part is an oWFN, this time typically with a nonempty interface. In accordance with terminology of service-oriented computing [8], we consider the contribution of a party to an interorganizational business process as a *service*. Correspondingly, the agreed version (specification) of the service is called *public view* while an actual local implementation is called *private view* of the service.

**Definition 8 (Public view).** *Let  $[N, r]$  be a contract with  $N = (P, T, F, I, O, m_0, \Omega)$ ,  $\Omega = \{m_f\}$ , and  $r \in T \rightarrow \mathcal{A}$ , and let  $A \in \mathcal{A}$  be a party. The public view of  $A$ 's share in the contract is the oWFN  $N_A$  where*

- $P_A = \{p \in P \mid \exists t \in \bullet p \cup p^\bullet : r(t) = A\}$ ,
- $T_A = \{t \in T \mid r(t) = A\}$ ,
- $F_A = F \cap ((P_A \times T_A) \cup (T_A \times P_A))$ ,
- $I_A = \{p \in P_A \mid \exists t \in \bullet p : r(t) \neq A\}$ ,
- $O_A = \{p \in P_A \mid \exists t \in p^\bullet : r(t) \neq A\}$ ,
- $m_{0_A} = m_{0|P_A}$  (i.e., the restriction of  $m_0$  to the places in  $P_A$ ), and
- $\Omega_A = \{m_{f|P_A}\}$ .

The contract shown in Fig. 1(a) involves three parties. Hence, there are three public views on the contract:  $N_{\text{cust}}$ ,  $N_{\text{supp}}$ , and  $N_{\text{ship}}$ .  $N'_{\text{cust}}$  shown in Fig. 1(b) is an example of a private view of  $N_{\text{cust}}$  in Fig. 1(a). The private views  $N'_{\text{supp}}$  and  $N'_{\text{ship}}$  are not provided here.

For a set  $\mathcal{A} = \{A_1, \dots, A_k\}$  of parties and a contract  $[N, r]$ , it is easy to see that  $N_{A_1} \oplus \dots \oplus N_{A_k} = N$ . In this respect, the restriction that  $\Omega$  contains only one element is indeed crucial, as otherwise  $N_{A_1} \oplus \dots \oplus N_{A_k}$  could have final

markings which result from recombining final markings of different parties but which are not final markings of  $N$ .

It should be noted that the definition of a contract is much more generic than the definition given in [9, 10]. In [9, 10], both the public and private view need to be workflow nets [15]; that is, Petri nets with a unique source and sink place, and with every node on a path from source to sink. In this paper, we do not impose such syntactical restrictions. Nevertheless, it may be wise to “massage” the contract such that each of the resulting public views is in fact connected. This can easily be done by adding implicit places as shown in [9, 10]. The implicit places do not change the behavior of the contract. However, when partitioning the contract into public views, the resulting oWFNs more clearly show the responsibilities of each partner.

### 3 Accordance Between Public and Private View

In this section, we define the notion of *accordance*. This criterion is used to compare the public view (agreed specification of the service) and the private view (actual implementation of the service) on a party’s share of a contract. The goal of the accordance notion is to preserve weak termination (see Def. 4) of the overall process  $N$ . Formally, weak termination of  $N$  and accordance of each private view  $N'_A$  with the corresponding public view  $N_A$  should imply weak termination of  $N'_{A_1} \oplus \dots \oplus N'_{A_k}$  which models the overall process as actually implemented. Consider for example the weakly terminating contract shown in Fig. 1(a) which is split into three public views:  $N_{\text{cust}}$ ,  $N_{\text{supp}}$ , and  $N_{\text{ship}}$ . If each of these public views is replaced by a private view such that the private view *accords* with the public view, then the composition of these private views  $N'_{\text{cust}} \oplus N'_{\text{supp}} \oplus N'_{\text{ship}}$  should yield a weakly terminating oWFN.

To define a suitable notion of accordance, we introduce the concept of *strategies*.

**Definition 9 (Strategy).** *An oWFN  $N$  is a strategy for an oWFN  $N'$  if  $N \oplus N'$  is weakly terminating.  $\text{Strat}(N)$  denotes the set of all strategies of  $N$ .*

Note that  $\text{Strat}(N)$  may correspond to a large (in fact infinite) set of oWFNs; that is, it is the set of all potential partners of  $N$ .  $N_{\text{cust}}$  in Fig. 1(a) and  $N'_{\text{cust}}$  in Fig. 1(b) are two examples of strategies for  $N_{\text{supp}} \oplus N_{\text{ship}}$ .

If  $[N, r]$  is a contract with  $\mathcal{A} = \{A_1, \dots, A_k\}$  and  $N$  is weakly terminating, then  $N_{A_1} \oplus \dots \oplus N_{A_{i-1}} \oplus N_{A_{i+1}} \oplus \dots \oplus N_{A_k}$  is a strategy for  $N_{A_i}$ . These properties of the strategy concept justify the following definition of accordance.

**Definition 10 (Accordance).** *An oWFN  $N'$  (private view) accords with an oWFN  $N$  (public view) if it has the same interface ( $I' = I$  and  $O' = O$ ) and has at least the strategies that  $N$  has; that is,  $\text{Strat}(N') \supseteq \text{Strat}(N)$ .*

$N_{\text{cust}}$  can be seen as the public view and  $N'_{\text{cust}}$  as the private view. Clearly,  $N'_{\text{cust}}$  accords with  $N_{\text{cust}}$  (and vice versa). The following theorem shows that  $N_{\text{cust}}$  can be substituted by  $N'_{\text{cust}}$  without jeopardizing weak termination in the contract.

**Theorem 1 (Implementation of a contract).** *Let  $[N, r]$  be a contract between parties  $\{A_1, \dots, A_k\}$  where  $N$  is weakly terminating. If, for all  $i \in \{1, \dots, k\}$ ,  $N'_{A_i}$  (the private view of  $A_i$ ) accords with  $N_{A_i}$  (the public view of  $A_i$ ), then  $N' = N'_{A_1} \oplus \dots \oplus N'_{A_k}$  (the actual implementation) is weakly terminating.*

**Proof.**

Let  $\{A_1, \dots, A_k\}$  be the set of involved parties and  $\mathcal{N}(j) = N'_{A_1} \oplus \dots \oplus N'_{A_j} \oplus N_{A_{j+1}} \oplus \dots \oplus N_{A_k}$  for  $j \in \{0, \dots, k\}$ . Note that  $\mathcal{N}(0) = N_{A_1} \oplus \dots \oplus N_{A_k} = N$  and  $\mathcal{N}(k) = N'_{A_1} \oplus \dots \oplus N'_{A_k} = N'$ .

We show by induction that  $\mathcal{N}(j)$  is weakly terminating for any  $j \in \{0, \dots, k\}$ .

Clearly, this holds for  $j = 0$ :  $\mathcal{N}(0) = N$  is weakly terminating. Assume that  $\mathcal{N}(j)$  is weakly terminating and  $0 \leq j < k$ . Let  $\mathcal{N}' = N'_{A_1} \oplus \dots \oplus N'_{A_j} \oplus N_{A_{j+2}} \oplus \dots \oplus N_{A_k}$ ; that is,  $\mathcal{N}(j)$  without  $N_{A_{j+1}}$ .  $\mathcal{N}'$  is a strategy for  $N_{A_{j+1}}$  since  $\mathcal{N}(j) = \mathcal{N}' \oplus N_{A_{j+1}}$  is weakly terminating; that is,  $\mathcal{N}' \in \text{Strat}(N_{A_{j+1}})$ . Since  $N'_{A_{j+1}}$  (the private view) accords with  $N_{A_{j+1}}$  (the public view),  $\text{Strat}(N_{A_{j+1}}) \subseteq \text{Strat}(N'_{A_{j+1}})$ . Hence,  $\mathcal{N}' \in \text{Strat}(N_{A_{j+1}}) \subseteq \text{Strat}(N'_{A_{j+1}})$ , indicating that  $\mathcal{N}'$  is a strategy for  $N'_{A_{j+1}}$ . Therefore,  $\mathcal{N}' \oplus N'_{A_{j+1}} = \mathcal{N}(j+1)$  is weakly terminating. By induction this implies that  $\mathcal{N}(j)$  is weakly terminating for any  $j$  including  $j = k$ . Hence,  $N' = \mathcal{N}(k)$  is weakly terminating.  $\square$

The value of the theorem is that it gives each party a criterion (accordance of  $N'_{A_i}$  with  $N_{A_i}$ ) that can be locally verified for asserting a global property (weak termination of the overall process as actually implemented). For example, any combination of arbitrary private views  $N''_{\text{cust}}$ ,  $N''_{\text{supp}}$ , and  $N''_{\text{ship}}$  according with the corresponding public view (i.e.,  $N''_{\text{cust}}$  accords with  $N_{\text{cust}}$ ,  $N''_{\text{supp}}$  accords with  $N_{\text{supp}}$ , and  $N''_{\text{ship}}$  accords with  $N_{\text{ship}}$ ) yields a weakly terminating realization of the contract shown in Fig. 1(a).

## 4 Checking Accordance

In this section, we demonstrate that the property of *accordance* can be verified automatically, at present time subject to restrictions. Checking accordance relies on the concept of *operating guidelines* [19, 20]. The original purpose of an operating guideline of a service  $N$  is to characterize the set of all services  $M$  such that the composition of  $N$  and  $M$  behaves “correctly”. Thereby, “correctly” means weak termination in [19] or deadlock-freedom in [20]. The results in [19] are restricted to services with acyclic and finite behavior while there are only marginal restrictions for the approach in [20]. As we are interested in weak termination in this paper, we use the approach in [19] thus inheriting the restriction to acyclic finite state services. Such a service may have *while* loops in its operational description as long as every iteration produces states that are different from other iterations, for example, different counter values. In ongoing research, we work on an extension of the approach in [20] to weak termination, so the current restriction to acyclic finite state services may be only temporary.

An operating guideline  $OG_N$  of an oWFN  $N$  is basically an annotated automaton; that is, a transition system where transitions are labeled with interface places of  $N$  representing send or receive actions to  $N$ , respectively. Each state of the transition system is annotated with a Boolean formula which has places  $p \in I \cup O$  of  $N$  as propositions.

**Definition 11 (Annotated automaton).**  $[Q, C, \delta, q_0, \Phi]$  is an annotated automaton iff  $Q$  is a nonempty set of states,  $C$  is a set of message channels,  $\delta : Q \times C \dashrightarrow Q$  is a (partial) transition relation such that every state  $q \in Q$  is reachable from  $q_0$  via transitive applications of  $\delta$ ,  $q_0 \in Q$  is the initial state, and  $\Phi$  is an annotation function, where, for all  $q \in Q$ ,  $\Phi(q)$  is a Boolean formula with propositions in  $C$ .

The goal of an annotated automaton is to represent a *set of automata*. Concrete automata are created by removing nodes, arcs, and annotations. Function  $\Phi$  provides a Boolean formula for each state indicating which combinations of outgoing arcs are allowed to be present in a concrete automaton. Note that each oWFN corresponds to an automaton. Hence, an annotated automaton can describe a set of oWFNs (e.g.,  $Strat(N)$ ).

In order to simplify the presentation, we make the assumption that each transition of an oWFN is connected to at most one interface place. This assumption does, however, not restrict generality as every oWFN can be transformed into an equivalent one that obeys this restriction [20].

A service described in terms of an oWFN *matches* with  $OG_N$  iff its behavior can be embedded into  $OG_N$  such that the annotations touched by this embedding evaluate to true.

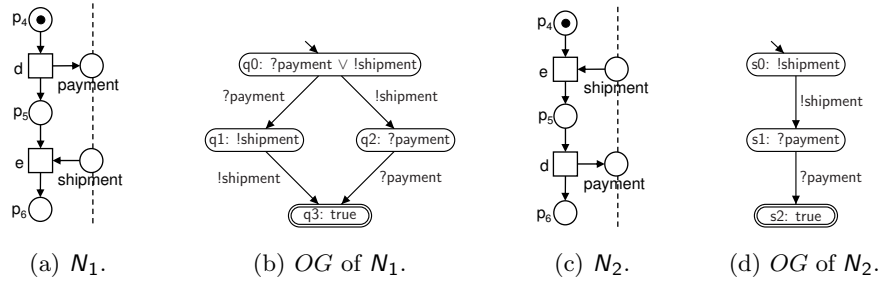
**Definition 12 (Matching).** Let  $M$  be an oWFN that obeys the restriction stated above and let  $A^\Phi = [Q, C, \delta, q_0, \Phi]$  be an annotated automaton with  $C = I_M \cup O_M$ . Let  $X$  be the set of all reachable markings of the Petri net obtained by removing all interface places of  $M$ . Then  $M$  matches with  $A^\Phi$  iff there is a mapping  $\rho$  from the set  $X$  to  $Q$  such that the following conditions hold:

1.  $\rho(m_{0_M}) = q_0$ ;
2. If  $t$  is a transition of  $M$  not connected to any interface place,  $m, m' \in X$ , and  $m \xrightarrow{t} m'$ , then  $\rho(m') = \rho(m)$ ;
3. If  $t$  is a transition of  $M$ ,  $c \in I_M$ ,  $c \in \bullet t$ ,  $m, m' \in X$ , and  $m + [c] \xrightarrow{t} m'$ , then  $(\rho(m), c) \in \text{dom}(\delta)$  and  $\rho(m') = \delta(\rho(m), c)$ ;
4. If  $t$  is a transition of  $M$ ,  $c \in O_M$ ,  $c \in t^\bullet$ ,  $m, m' \in X$  and  $m \xrightarrow{t} m' + [c]$ , then  $(\rho(m), c) \in \text{dom}(\delta)$  and  $\rho(m') = \delta(\rho(m), c)$ ;
5. For all  $m \in X$ , at least one of the following properties holds:
  - A transition not connected to any interface place is enabled in  $m$ ;
  - $\Phi(\rho(m))$  evaluates to true for the following assignment  $\beta$  to propositions (i.e., channels): let  $\beta(c) = \text{true}$  iff there is a transition with  $c \in \bullet t$  that is enabled in  $m + [c]$ , or a transition with  $c \in t^\bullet$  that is enabled in  $m$ .

In the formal definition,  $\rho$  represents the informally described embedding. The assignment used for evaluating an annotation represents transitions that leave the considered marking in  $M$ .

**Definition 13 (Operating guideline).** *An annotated automaton is an operating guideline  $OG_N$  of an oWFN  $N$  iff  $Strat(N)$  is exactly the set of all oWFNs matching with  $OG_N$ .*

If  $OG_N$  is an operating guideline of an oWFN  $N$ , then for any oWFN  $M$ :  $M$  matches with  $OG_N$  iff  $M \oplus N$  weakly terminates. The operating guideline of the oWFN  $N_{\text{cust}}$  of Fig. 1(a) consists of 25 states and 61 transitions and is too large to be depicted here. As an example, we consider the final part of the customer's oWFN  $N_{\text{cust}}$  (oWFN  $N_1$  in Fig. 2(a)) and computed its  $OG$  (Fig. 2(b)). It consists of 4 states with the annotation depicted inside the states.



**Fig. 2.** Two oWFNs  $N_1$  and  $N_2$  and their operating guidelines. An annotation starting with “!” (“?”) refers to a sending (receiving) event. Any strategy for  $N_1$  ( $N_2$ ) matches with the operating guideline of  $N_1$  ( $N_2$ ).  $N_2$  does not accord with  $N_1$ , since it excludes the strategy that first expects the payment, followed by sending the shipment.

In [19], we presented an algorithm to compute, for each acyclic finite state oWFN  $N$ , an operating guideline  $OG_N$ . The algorithm is implemented in our tool Fiona [21]. Since an operating guideline represents the set of strategies, it is natural to use  $OG_N$  and  $OG_{N'}$  for comparing  $Strat(N)$  with  $Strat(N')$ . Informally,  $N'$  accords with  $N$  iff  $OG_{N'}$  can be embedded into  $OG_N$  such that the annotations in  $OG_{N'}$  imply the annotations of  $OG_N$ .

**Theorem 2 (Accordance check with OGs).** *Let  $N$  and  $N'$  be two acyclic finite state oWFNs with  $OG_N = [Q, C, \delta, q_0, \Phi]$  and  $OG_{N'} = [Q', C', \delta', q'_0, \Phi']$ . Then,  $Strat(N') \subseteq Strat(N)$  iff there is a mapping  $\xi : Q' \rightarrow Q$  such that*

1.  $\xi(q'_0) = q_0$ ;
2. if  $\xi(q') = q$  and  $\delta'(q', c) = q'_1$ , then there is a  $q_1$  such that  $\delta(q, c) = q_1$  and  $\xi(q'_1) = q_1$ ; and
3. for all  $q' \in Q'$ , the formula  $\Phi'(q') \Rightarrow \Phi(\xi(q'))$  is a tautology.

For the proof of this theorem, we rely on a fact about operating guidelines as constructed in [19]. As we cannot repeat the whole approach of [19], we just state this fact without proof.

**Proposition 1 ([19]).** *For every service  $N$  with operating guideline  $OG_N = [Q, C, \delta, q_0, \Phi]$  and all  $q \in Q$ , the formula  $\Phi(q)$*

1. *uses only propositions  $c$  where  $\delta(q, c)$  is defined, and*
2. *is satisfied for the assignment assigning true to all propositions.*

**Proof (of Thm. 2 (Sketch)).**

*Implication.* Let  $OG_{N'} = [Q', C, \delta', q'_0, \Phi']$  and  $OG_N = [Q, C, \delta, q_0, \Phi]$ , and let  $Strat(N') \subseteq Strat(N)$ .

*We can construct an oWFN  $M$  whose behavior corresponds exactly to the transition system  $[Q', C, \delta', q'_0]$ . This can be achieved by using  $Q' \cup C$  as set of places (with  $C$  being the interface of  $M$ ), and having, for each  $q'_1, c$ , and  $q'_2$  with  $\delta'(q'_1, c) = q'_2$  a transition  $t_{q'_1, c, q'_2}$  that moves a token from  $q'_1$  to  $q'_2$ , and removes (produces, resp.) a token from (on)  $c$  if  $c$  is an output (input) place of  $N'$ . Let  $m_{q'}$  denote a marking of  $M$  where there is a token on place  $q'$  and no token elsewhere. Let  $m_{q'_0}$  be the initial marking of  $M$ . By induction, it can be shown that, for all  $q' \in Q'$ ,  $m_{q'}$  is reached by Def. 12, with  $\rho'(m_{q'}) = q'$ .*

*Since there is a transition for each  $c$  where  $\delta'(q', c)$  is defined, we can derive from Prop. 1 that all annotations evaluate to true when  $M$  is evaluated according to Def. 12. Consequently,  $M$  matches with  $OG_{N'}$  and hence  $M$  is a strategy for  $N'$  and thus, by assumption, a strategy for  $N$ .*

*Being a strategy for  $N$ , there is a mapping  $\rho$  from the markings of  $M$  to  $Q$ . Define  $\xi : Q' \rightarrow Q$  such that  $\xi(q') = q$  iff  $\rho(m_{q'}) = q$ . By the structural similarity of Def. 12 and Thm. 2, it is easy to see that  $\xi$  satisfies the first two items required in Thm. 2. For verifying the third item, let  $q' \in Q'$  and let  $\beta$  be an arbitrary assignment to propositions occurring in  $\Phi'(q')$  where  $\Phi'(q')$  is true. Remove from  $M$  all those transitions  $t_{q'_1, c, q'_2}$  where  $\beta(c)$  is false. By Def. 12, the resulting oWFN is still a strategy for  $N'$  and thus a strategy for  $N$ , too. Using Def. 12 again, we can see that  $\Phi(\xi(q'))$  is true as well. Thus,  $\Phi'(q') \Rightarrow \Phi(\xi(q'))$  is a tautology.*

*Replication.* Consider a mapping  $\xi$  as required and let  $M$  be a strategy for  $N'$ . We show that  $M$  is a strategy for  $N$ , too. By Def. 12, there is a mapping  $\rho'$  from the markings of  $M$  to  $Q'$ . Let  $\rho(m) = \xi(\rho'(m))$ . For all markings reached by Def. 12,  $\Phi'(\rho'(m))$  evaluates to true for the assignment described in Def. 12, and by the third item of Thm. 2, so does  $\Phi(\rho(m))$ . Consequently,  $M$  is a strategy for  $N$ .  $\square$

With the help of Thm. 2, we can easily verify accordance of  $N'_{\text{cust}}$  with  $N_{\text{cust}}$  of Fig. 1. In fact, the two corresponding operating guidelines are equal. Therefore, we conclude that  $Strat(N'_{\text{cust}}) = Strat(N_{\text{cust}})$ . As a counterexample, we consider the oWFN  $N_1$  shown in Fig. 2(a).  $N_1$  represents the final part of the customer of Fig. 1(a). The oWFN  $N_2$  of Fig. 2(c) reverses the order of the transitions d and e. If  $N_1$  is the public view, then  $N_2$  is a wrong implementation since it excludes

strategies. This is reflected by the corresponding operating guidelines, depicted in Fig. 2(b) and Fig. 2(d). Applying the mapping  $\xi$ , the state  $q1$  of the  $OG$  of Fig. 2(b) has no counterpart in the  $OG$  of Fig. 2(d). This violates the second item of Thm. 2.

For an implementation of the criteria in Thm. 2, finding the mapping  $\xi$  is the crucial task. As both  $OG_N$  and  $OG_{N'}$  are deterministic (i.e., in each state  $q$  there is at most one  $c$ -successor), this task actually amounts to a depth-first search through  $OG_{N'}$  which is mimicked in  $OG_N$ . The time and space required for finding  $\xi$  is thus linear in the number of states and edges of  $OG_{N'}$ . This size, in turn, is equal to the number of states and edges of a particular strategy for  $N$  [19]. The accordance check based on Thm. 2 has been implemented in our tool Fiona [21].

## 5 Derive a Private View From a Public View

In the previous section, we presented a method to check accordance. Using operating guidelines we can check whether some oWFN representing the private view accords with the oWFN representing the public view. However, instead of checking accordance after creating the private view, it is also possible to guarantee accordance by using *transformation rules*. This idea is inspired by the earlier work on projection inheritance [9, 10, 16, 17]. Accordance is a weaker notion than projection inheritance. This was illustrated already using Fig. 1 where  $N'_{\text{cust}}$  accords with  $N_{\text{cust}}$  but  $N'_{\text{cust}}$  and  $N_{\text{cust}}$  are not related by projection inheritance. In the following we explain the general approach of using transformation rules (cf. Sect. 5.1). Then, we give a retrospection of projection inheritance in Sect. 5.2. In Sect. 5.3, we prove that projection inheritance implies accordance, and therefore, all inheritance-preserving transformation rules presented in [17] also preserve accordance. We will show these rules by reformulating them to fit into the setting of this paper. Afterwards, in Sect. 5.4, we will formulate dedicated transformation rules that take the sending and receiving of messages into account while still guaranteeing accordance. Finally, the applicability of the presented transformation rules is demonstrated by help of a case study (cf. Sect. 5.5).

### 5.1 The Transformation Approach

According to Thm. 1 in Sect. 3, every party of a contract can implement its public view and finally it has to check accordance between the private and the public view. In the following, we present a different approach: The public view is incrementally transformed into a private view. To this end, fragments of the public view are incrementally replaced by other fragments until the private view is designed. In this approach, a fragment  $N'$  of a party is called a *pattern* and will be replaced by another fragment  $N''$ . We will prove that if  $N''$  accords with  $N'$ , then replacing  $N'$  by  $N''$  preserves weak termination of the overall contract.

First of all, we formally define an oWFN pattern  $N'$  of an oWFN  $N$ . Therefore, the set of interface places of  $N'$  is divided into two sets: one set contains all

places that are interface places of  $N$  for communicating with other parties (i.e., subsets of  $I$  and  $O$ ) and the other set,  $R \cup S$ , contains all places that serve as an interface to the rest of  $N$ .  $R$  is the set of input places from the other parts of  $N$ , and  $S$  is the set of output places. We first define a *subnet* of an oWFN and then an *oWFN pattern* which is a restricted subnet.

**Definition 14 (Subnet).** *Let  $N = (P, T, F, I, O, m_0, \Omega)$  be an oWFN. An oWFN  $N' = (P', T', F', I', O', m'_0, \Omega')$  is a subnet of  $N$  iff*

- $P' \subseteq P$ ,
- $T' \subseteq T$ ,
- $F' = F \cap ((P' \times T') \cup (T' \times P'))$ ,
- $I' = I|_{P'}$ ,
- $O' = O|_{P'}$ ,
- $m'_0 = m_{0|P'}$ , and
- $\Omega' = \{m'_f \mid m'_f = m_{f|P'}, m_f \in \Omega\}$ .

**Definition 15 (oWFN pattern).** *Let  $N = (P, T, F, I, O, m_0, \Omega)$  be an oWFN and  $N' = (P', T', F', I', O', m'_0, \Omega')$  a subnet of  $N$ .  $N'$  is an oWFN pattern of  $N$  iff*

- $m'_0 = []$ ,
- $I' = I|_{P'} \cup R$  with  $R \subseteq P' \setminus I$ ,
- $O' = O|_{P'} \cup S$  with  $S \subseteq P' \setminus O$ ,
- $\Omega' = \{[]\}$ ,
- for all  $p \in P' \setminus R$ , there is no  $t \in T \setminus T'$ ,  $(t, p) \in F$ ,
- for all  $p \in P' \setminus S$ , there is no  $t \in T \setminus T'$ ,  $(p, t) \in F$ , and
- for all  $t \in T'$ , there is no  $p \in P \setminus P'$ ,  $(p, t) \in F$  or  $(t, p) \in F$ .

The next theorem states that if the public view of a party participating in a contract has an oWFN pattern  $N'$  and there is another oWFN pattern  $N''$  with  $N''$  accords with  $N'$ , then we can replace  $N'$  by  $N''$  and the modified contract is still weakly terminating. Such transformations can be applied incrementally and thus we can derive a private view from a public view just by transforming the public view and the resulting private view is correct by construction.

**Theorem 3 (Justification of transformation rules).** *Let  $[N, r]$  be a contract between parties  $\{A_1, \dots, A_k\}$  where  $N = N_{A_1} \oplus \dots \oplus N_{A_k}$  is weakly terminating. Let  $N'_p$  be an oWFN pattern of  $N_{A_i}$ ,  $1 \leq i \leq k$ , such that there exists an oWFN  $N_{rest}$  with  $N_{A_i} = N'_p \oplus N_{rest}$ . Let further  $N''_p$  be an arbitrary oWFN. Then, if  $N''_p$  accords with  $N'_p$ , the modified contract  $N' = N_{A_1} \oplus \dots \oplus N_{A_{i-1}} \oplus (N''_p \oplus N_{rest}) \oplus N_{A_{i+1}} \oplus \dots \oplus N_{A_k}$  is weakly terminating.*

**Proof.**

*The theorem is an application of Thm. 1. In contrast to Thm. 1, we do not replace a party's public view by a private view but an oWFN pattern of a party's public view by another oWFN pattern.*

*Since  $N''_p$  accords with  $N'_p$  and the rest of the contract; that is,  $N_{A_1} \oplus \dots \oplus N_{A_{i-1}} \oplus N_{rest} \oplus N_{A_{i+1}} \oplus \dots \oplus N_{A_k}$  remains unchanged, the modified contract  $N'$  is by Thm. 1 weakly terminating.  $\square$*

During the next sections we will present several transformation rules whose correctness is justified by Thm. 3.

## 5.2 Projection Inheritance

Inheritance is one of the key concepts of object-orientation. In object-oriented design, inheritance is typically restricted to the static aspects (e.g., data and methods) of an object class. In many cases, however, the dynamic behavior of services is of prime importance. Therefore, in [17], four inheritance notions focusing on the dynamics have been defined. One of these notions is *projection inheritance*. The four notions of inheritance allow for comparing two process models: the subclass and the superclass. The subclass process is indeed a subclass if it inherits particular dynamic properties of its superclass. Although these four inheritance notions are notation-independent, in [17], they have been defined in terms of Petri nets and process algebra. The four inheritance relations use branching bisimulation [22] (to compare processes) in combination with the notions of *encapsulation* and *abstraction*. Encapsulation corresponds to blocking tasks, whereas abstraction corresponds to hiding tasks. In this paper, we only focus on projection inheritance. Projection inheritance is based on branching bisimulation and abstraction. The assumption is that the subclass adds methods to the superclass such that after hiding the additional methods both are equivalent. The basic idea of projection inheritance can be characterized as follows: “If it is not possible to distinguish the behaviors of  $x$  and  $y$  when arbitrary methods of  $x$  are executed, but when only the effects of methods that are also present in  $y$  are considered, then  $x$  is a subclass of  $y$ ” [17].

Projection inheritance was defined for workflow nets, but in this definition projection inheritance refers to “methods” rather than the “sending and receiving of messages”. However, it is easy to reformulate projection inheritance in terms of the setting of this paper by the following mapping: A method present in both the superclass and subclass corresponds to a transition that is connected to an interface place and these are the only transitions that are connected to an interface place. That way, we reformulate projection inheritance for open workflow nets. As branching bisimulation compares the transition systems of two oWFNs, we need to consider closed systems; that is, oWFNs with empty interface. For this purpose, we define the notion of the inner of an oWFN  $N$ . To compare transitions of two oWFNs, transitions connected to interface places need to have distinguishable labels. Obviously, the labeling function has to guarantee that two transitions connected to the same set of interface places have the same label. In contrast, transitions not connected to an interface place (i.e., internal transitions) are labeled with  $\tau$ . This leads to the following definition of a labeled oWFN.

**Definition 16 (Labeled oWFN).** *Let  $N = (P, T, F, I, O, m_0, \Omega)$  be an oWFN,  $t \in T$ ,  $p \in (I \cup O)$  an interface place of  $N$ , and  $Int_t = \{p \mid p \in \bullet t \cup t \bullet\}$  be the set of interface places connected to  $t$ . Then,  $l : T \rightarrow \mathcal{P}(I \cup O)$  is a function which labels every  $t$  with its interface places  $Int_t$ . If  $Int_t \neq \emptyset$ , then*

$l(t) = \{g \circ p \mid g \in \{!, ?\}, p \in \text{Int}_t, g = ? \text{ if } p \in I, g = ! \text{ if } p \in O\}$  (where  $\circ$  is the concatenation operator), else  $l(t) = \tau$ . Then,  $N' = (P, T, F, I, O, m_0, \Omega, l)$  is a labeled oWFN.

To distinguish the labels of transitions connected to an input place  $a$  and an output place  $a$ , we extend the label with “?” and “!”, respectively. For example, the labeled oWFN of the modified customer in Fig. 1(b) had labels  $l(a) = \{!order\}$ ,  $l(c) = \{?invoice\}$ ,  $l(x) = l(y) = \tau$ , etc.

Obviously, the definitions we presented for oWFNs can be easily extended to labeled oWFNs. We only define the composition of labeled oWFNs.

**Definition 17 (Composition of labeled oWFNs).** Let  $N_1, \dots, N_k$  be a composable set of labeled oWFNs. The composition  $N = N_1 \oplus \dots \oplus N_k$  is defined as for oWFNs (see Def. 6) and the labeling function  $l$  is  $l : T \rightarrow \mathcal{P}(I \cup O)$ .

Next, we define the inner of a labeled oWFN  $N$  that results from eliminating all interface places from  $N$ .

**Definition 18 ( $inner_N$ ).** Let  $N = (P, T, F, I, O, m_0, \Omega, l)$  be a labeled oWFN and let  $J = P \setminus (I \cup O)$  the set of internal places of  $N$ . Then,  $inner_N = (J, T, F \cap ((J \times T) \cup (T \times J)), m_0, \Omega, l)$ .

As an example, consider the inner of the customer  $N_{\text{cust}}$  (see Fig. 1(a)) and the modified customer  $N'_{\text{cust}}$  (see Fig. 1(b)) depicted in Fig. 3.

To compare the behavior of two labeled oWFNs  $N$  and  $N'$  with respect to branching bisimulation, we have to check whether  $inner_N$  and  $inner_{N'}$  are branching bisimilar. The following definition formalizes the notion of branching bisimulation. This definition takes into account that, for each final marking of  $inner_N$ , there exists a final marking in  $inner_{N'}$  and both markings are related by branching bisimulation.

**Definition 19 (Branching bisimulation).** Let  $N, N'$  be two labeled oWFNs and  $N_1 = inner_N, N_2 = inner_{N'}$  the inner of these two nets. Let further  $m_{0_{N_1}}, m_{0_{N_2}}$  be the initial marking of  $N_1$  and  $N_2$ , respectively.  $N_1$  and  $N_2$  are branching bisimilar, denoted  $N_1 \approx_{bb} N_2$ , iff there exists a symmetric relation  $R$  such that  $m_{0_{N_1}} R m_{0_{N_2}}$  and for all  $m_{N_1}, m_{N_2}$  holds:

- If  $m_{N_1} R m_{N_2}$  and  $m_{N_1} \xrightarrow{\alpha} m'_{N_1}$ , then either
  - $\alpha = \tau$  and  $m'_{N_1} R m_{N_2}$  or
  - there are  $m'_{N_2}, m''_{N_2}$  such that  $m_{N_2} \xrightarrow{\epsilon} m'_{N_2} \xrightarrow{\alpha} m''_{N_2}$ ,  $m_{N_1} R m'_{N_2}$ , and  $m'_{N_1} R m''_{N_2}$  (where  $\epsilon$  is a (possible empty) sequence of  $\tau$  transitions).

Furthermore, for each final marking  $m_{f_{N_1}} \in \Omega_{N_1}$  holds: If  $m_{f_{N_1}} R m_{g_{N_2}}$ , then either  $m_{g_{N_2}} \in \Omega_{N_2}$  or every transition sequence starting from  $m_{g_{N_2}}$  contains a state  $m'_{g_{N_2}} \in \Omega_{N_2}$  with  $m_{f_{N_1}} R m'_{g_{N_2}}$ .

The nets shown in Fig. 3 are not branching bisimilar. Intuitively, the inner of  $N'_{\text{cust}}$  has more behavior than the inner of  $N_{\text{cust}}$ . As an example, for marking  $[p_{21}, p_{24}]$  in Fig. 3(b) there is no corresponding marking in Fig. 3(a).

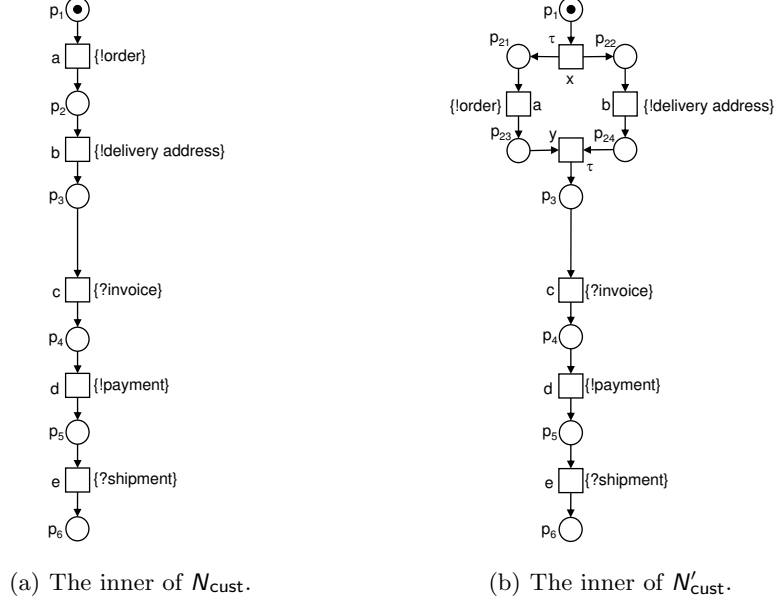


Fig. 3. The inner of the customer and the modified customer (cf. Fig. 1).

To decide whether two labeled oWFNs are related by projection inheritance, it is sufficient to check if the inner of these oWFNs are branching bisimilar. In contrast to [17], we do not need to define an abstraction operator. In our mapping, the comparison of the two oWFNs is restricted to the transitions that are connected to an interface place. We abstract from all other transitions by labeling them with  $\tau$ . The labeling, however, is fixed in the definition of a labeled oWFN and thus no additional definition of an abstraction is necessary. Consequently, we can define projection inheritance of two oWFNs as follows:

**Definition 20 (Projection inheritance).** *Let  $N, N'$  be two oWFNs, let  $N_I, N'_I$  their respective labeled oWFNs, and let  $N_1 = \text{inner}_{N_I}, N_2 = \text{inner}_{N'_I}$ .  $N$  and  $N'$  are related by projection inheritance, denoted  $N \approx_{pj} N'$ , iff  $N_1 \approx_{bb} N_2$ .*

Classical projection inheritance, as defined in [17], specifies a subclass-superclass relation between two WFNs. This relation, however, restricts the approach to either removing transitions (i.e., the resulting WFN is a superclass) or adding transitions (i.e., the resulting WFN is a subclass). In Def. 20, we have defined projection inheritance for oWFNs in a more generalized way, allowing for adding *and* removing transitions. So the resulting oWFN might be neither a subclass nor a superclass. Consequently, we do not have a subclass-superclass relation. However, in our setting this generalization is necessary, because it is important to know whether adding or removing internal transitions to/from an oWFN changes the interactional behavior of the resulting oWFN.

Consider again the two nets depicted in Fig. 3. Since they are not branching bisimilar, we can conclude that  $N_{\text{cust}}$  and  $N'_{\text{cust}}$  in Fig. 1 are not related by projection inheritance.

From the definition of projection inheritance we can conclude the following proposition.

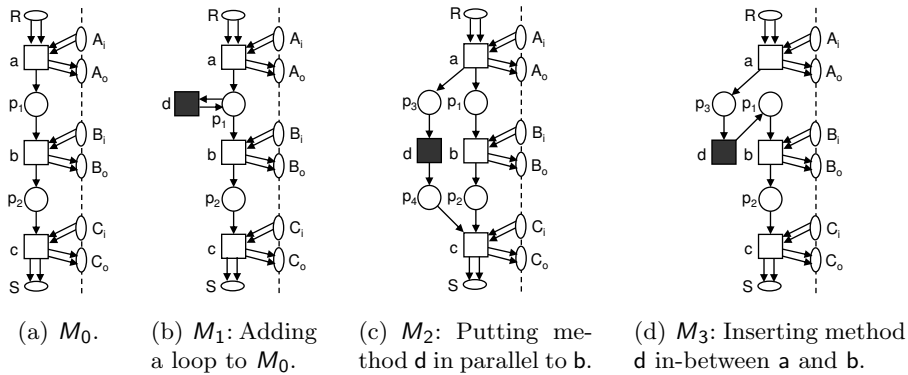
**Proposition 2.** *Let  $N, N'$  be two oWFNs. If  $N$  and  $N'$  are related by projection inheritance, the following necessary but not sufficient conditions hold:*

- (1)  $N$  and  $N'$  have the same set of interface places ( $I = I'$  and  $O = O'$ ).
- (2) If a transition of  $N$  ( $N'$ ) is connected to a set of interface places, then a transition connected to the same set of interface places is present in  $N'$  ( $N$ ).

### 5.3 Inheritance-Preserving Transformation Rules

Based on the notion of projection inheritance, three *inheritance-preserving transformation rules* have been defined in [17]. These rules correspond to design patterns for extending a superclass to incorporate new behavior: (1) adding loops, (2) inserting methods in-between existing methods, and (3) putting new methods in parallel with existing methods. In [9, 10, 16, 17], these rules are defined and/or applied.

Instead of redefining these rules formally, they are exemplified in Fig. 4. Figure 4(a) represents an oWFN pattern  $M_0$  of an oWFN  $M$ .  $M_0$  contains transitions  $a, b,$  and  $c$ . By Def. 15, there are no other connections of  $a, b, c, p_1,$  and  $p_2$  than those shown in Fig. 4(a).  $A_i = (\bullet a) \cap I_M$  is the set of input places of  $a$ ,  $A_o = (a \bullet) \cap O_M$  is the set of output places of  $a$ , etc.  $A_i, A_o, B_i, B_o, C_i, C_o$  do not need to be disjoint.  $R = (\bullet a) \setminus I_M$  and  $S = (c \bullet) \setminus O_M$  are (by Def. 15) the places connecting  $M_0$  to the rest of  $M$ . Similar remarks hold for the other three oWFN patterns  $M_1, M_2,$  and  $M_3$ . For example,  $M_1$  is obtained by adding transition  $d$  to  $M_0$ .



**Fig. 4.** Accordance-preserving transformation rules based on projection inheritance.

$M_0$  in Fig. 4 may be replaced by any of the three other oWFNs  $M_1$ ,  $M_2$ , and  $M_3$  without changing the set of strategies; that is,  $M_1$  accords with  $M_0$ ,  $M_2$  accords with  $M_0$ ,  $M_3$  accords with  $M_0$ , and vice versa. More precisely, any of the four nets can be replaced by one of the other nets without violating accordancy. If one generates the labeled oWFNs and ignores the interface places (e.g., constructs the inner of the respective oWFNs), then  $M_0$ ,  $M_1$ ,  $M_2$ , and  $M_3$  are branching bisimilar. Thus,  $M_0$ ,  $M_1$ ,  $M_2$ , and  $M_3$  are related by projection inheritance. However, as an example,  $M_1$  and  $M_2$  are only in our setting related by projection inheritance but not in the setting of [17]. The reason is,  $M_1$  is neither a subclass nor a superclass of  $M_2$  and vice versa. It is easy to see that projection inheritance implies accordancy.

**Theorem 4 (Projection inheritance implies accordancy).** *Let  $N$  and  $N'$  be two oWFNs. If  $N$  and  $N'$  are related by projection inheritance, then  $N'$  accords with  $N$  and vice versa.*

The intuition behind this theorem is that the interface places of an oWFN  $N$  only restrict the behavior of  $N$ . Hence, the actual behavior after composing  $N$  with another net is included in  $inner_N$ . Consider the labeled oWFNs  $N_l$  and  $N'_l$  of  $N$  and  $N'$ , respectively. Since only visible transitions (i.e., transitions that are not labeled with  $\tau$ ) are connected to interface places, and the firing of any visible transition in  $N_l$  can be followed by the same transition in  $N'_l$  and vice versa, both  $N_l$  and  $N'_l$  are identical when it comes to the production or consumption of tokens in interface places. Moreover, since the final markings are related, the termination of  $N_l$  can be followed by  $N'_l$  and vice versa. For the proof of Thm. 4, we need the following two lemmata.

**Lemma 1.** *Let  $N$ ,  $N'$ ,  $M$  be labeled oWFNs. Let further  $N$  and  $M$  as well as  $N'$  and  $M$  be composable. Then,  $inner_N \approx_{bb} inner_{N'}$  implies  $inner_{N \oplus M} \approx_{bb} inner_{N' \oplus M}$ .*

**Proof.**

*We prove this lemma by contradiction. Assume  $inner_{N \oplus M}$  and  $inner_{N' \oplus M}$  are not branching bisimilar. We will show, this implies that  $inner_N$  and  $inner_{N'}$  are not branching bisimilar and thus contradicting the assumption.*

*Let  $N \oplus M = (P, T, F, I, O, m_0, \Omega, l)$  and  $N' \oplus M = (P', T', F', I', O', m'_0, \Omega', l')$ . Let further  $RG_{N \oplus M}(m_0)$  and  $RG_{N' \oplus M}(m'_0)$  be the reachable states of  $inner_{N \oplus M}$  and  $inner_{N' \oplus M}$ , respectively. From  $inner_N \approx_{bb} inner_{N'}$  it follows  $I_N = I_{N'}$  and  $O_N = O_{N'}$ . Thus,  $I = I'$  and  $O = O'$ , meaning, both compositions have the same interface. Consequently,  $inner_{N \oplus M}$  and  $inner_{N' \oplus M}$  have the same visible transitions. To check whether the transition systems of  $inner_{N \oplus M}$  and  $inner_{N' \oplus M}$  are branching bisimilar, one has to relate their respective initial states  $m_0$  and  $m'_0$  and then apply Def. 19.*

*Assume that  $inner_{N \oplus M}$  and  $inner_{N' \oplus M}$  are not branching bisimilar. Thus, applying Def. 19, there exist markings  $m \in RG_{N \oplus M}(m_0)$ ,  $m' \in RG_{N' \oplus M}(m'_0)$  violating branching bisimulation. Let  $m$  and  $m'$  be the first markings violating branching bisimulation. The transitions that cause this violation must be*

transitions of  $N$  and  $N'$  rather than transitions of  $M$ , because  $M$  can only behave differently in the two compositions if it is controlled differently by  $N$  and  $N'$ , respectively. This is, however, only possible if  $N$  and  $N'$  are not branching bisimilar and thus contradicts our assumption. Therefore, the assumption that  $inner_{N \oplus M}$  and  $inner_{N' \oplus M}$  are not branching bisimilar is wrong and the lemma holds.  $\square$

**Lemma 2.** *Let  $N, N'$  be two labeled oWFNs with empty interface; that is,  $inner_N = N$  and  $inner_{N'} = N'$ . If  $N \approx_{bb} N'$ , then  $N$  weakly terminates iff  $N'$  weakly terminates.*

**Proof.**

Let  $N \approx_{bb} N'$ .

*Implication:* Let  $N$  be weakly terminating. Thus, (by Def. 4) from every marking, a final marking can be reached. Let  $m' \in RG_{N'}(m_{0_{N'}})$  be an arbitrary marking in  $N'$ . Since  $N \approx_{bb} N'$ , there exists a marking  $m \in RG_N(m_{0_N})$  with  $m R m'$ . From  $N$  being weakly terminating we derive that there exists a transition sequence  $\sigma$  with  $m \xrightarrow{\sigma} m_f$ ,  $m_f \in RG_N(m_{0_N})$  and  $m_f \in \Omega_N$ . Since  $N \approx_{bb} N'$  we can derive that there is a marking  $m'_f \in RG_{N'}(m_{0_{N'}})$  with  $m_f R m'_f$  and a transition sequence  $\sigma'$  such that  $m' \xrightarrow{\sigma'} m'_f$ . By Def. 19,  $m'_f \in \Omega_{N'}$ . Thus,  $N'$  is also weakly terminating.

*Replication:* Same argumentation as in the implication.  $\square$

With the help of the two lemmata, we can prove Thm. 4.

**Proof (of Thm. 4).**

Let  $N_i$  and  $N'_i$  be the respective labeled oWFNs of  $N$  and  $N'$ . Since  $N$  and  $N'$  are related by projection inheritance,  $inner_{N_i} \approx_{bb} inner_{N'_i}$  holds.  $N'$  accords with  $N$  and vice versa if and only if  $Strat(N) = Strat(N')$ . To show that  $N$  and  $N'$  have the same set of strategies, it is sufficient to prove  $Strat(N) \subseteq Strat(N')$  and  $Strat(N') \subseteq Strat(N)$ . For the case  $Strat(N) \subseteq Strat(N')$ , let  $N_S \in Strat(N)$  be an arbitrary strategy. It is sufficient to show that  $N_S \in Strat(N')$ . Let  $N_{S_i}$  be the labeled oWFN of  $N_S$ .

From  $inner_{N_i} \approx_{bb} inner_{N'_i}$  and by Lemma 1 we derive  $inner_{N_i \oplus N_{S_i}} \approx_{bb} inner_{N'_i \oplus N_{S_i}}$ . Since  $S$  is a strategy for  $N$ , it follows  $N \oplus N_S$  and also  $N_i \oplus N_{S_i}$  is weakly terminating. By Lemma 2 we can conclude that  $N'_i \oplus N_{S_i}$  is weakly terminating, too. Therefore  $N' \oplus N_S$  is weakly terminating and hence,  $S$  is also a strategy for  $N'$ .

The proof for the case  $Strat(N') \subseteq Strat(N)$  follows the same argumentation. Hence,  $Strat(N) = Strat(N')$  and the theorem holds.  $\square$

Theorem 4 justifies that adding and removing transitions that are not connected to an interface place with respect to branching bisimulation preserves accordance.

In [9, 10, 16, 17], several inheritance-preserving transformation rules are provided. We would like to emphasize that the simplified examples shown in Fig. 4 do not do justice to these rules. The actual rules are much more generic. However, a detailed discussion of these is outside the scope of this paper.

## 5.4 Accordance-Preserving Transformation Rules

Projection inheritance implies accordance and, therefore, the inheritance-preserving transformation rules can be used to incrementally build a private view that accords with the public view of a service. However, inheritance-preserving transformation rules are limited in the sense that they do not allow to change the order of messages. In the following, we present seven *accordance-preserving transformation rules*. Five of these rules preserve accordance in both directions and two rules preserve accordance only in one direction. Given an oWFN  $N$ , each transformation rule specifies a pattern  $N'$  of  $N$  (see Def. 15) which can be replaced by another pattern  $N''$ , yielding a new oWFN  $N_{new}$ . Theorem 3 justifies that this replacement does not violate the overall contract.

The first oWFN pattern,  $N_3$ , we consider sequentially sends  $n$  messages  $a_1, \dots, a_n$  and is defined as follows.

**Definition 21 ( $N_3$ ).** *Let  $n > 0$ . Then,  $N_3 = (P, T, F, I, O, m_0, \Omega)$  is an oWFN pattern with*

- $P = I \cup O \cup \{p_1, \dots, p_{n-1}\}$ ,
- $T = \{t_1, \dots, t_n\}$ ,
- $F = \{(r, t_1) \mid r \in R\}$   
 $\cup \{(t_n, s) \mid s \in S\}$   
 $\cup \{(t_i, p_i), (p_i, t_{i+1}) \mid i = 1, \dots, n-1\}$   
 $\cup \{(t_i, a_i) \mid i = 1, \dots, n\}$ ,
- $I = R$ ,
- $O = \{a_1, \dots, a_n\} \cup S$ .

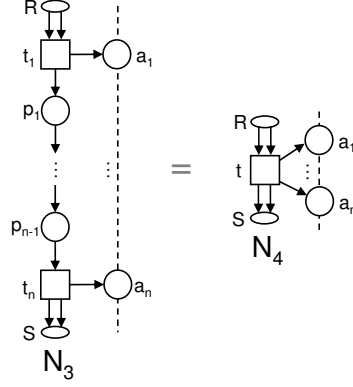
$N_3$  is depicted on the left hand side of Fig. 5. The next oWFN pattern,  $N_4$ , results from merging transitions  $t_1, \dots, t_n$  of  $N_3$  to a single transition  $t$ . Thus,  $N_4$  sends all messages  $a_1, \dots, a_n$  simultaneously.

**Definition 22 ( $N_4$ ).** *Let  $n > 0$ . Then,  $N_4 = (P, T, F, I, O, m_0, \Omega)$  is an oWFN pattern with*

- $P = I \cup O$ ,
- $T = \{t\}$ ,
- $F = \{(r, t) \mid r \in R\}$   
 $\cup \{(t, s) \mid s \in S\}$   
 $\cup \{(t, a_i) \mid i = 1, \dots, n\}$ ,
- $I = R$ ,
- $O = \{a_1, \dots, a_n\} \cup S$ .

$N_4$  is depicted on the right hand side of Fig. 5.

The first transformation rule is depicted in Fig. 5 and specifies that a sequence of sending events can be merged and the events can be sent simultaneously. Thus,  $N_3$  can be transformed into  $N_4$  and vice versa. Rule 1 preserves accordance in both direction. Consequently,  $Strat(N_3) = Strat(N_4)$ .



**Fig. 5.** Rule 1: A sequence of sending events ( $N_3$ ) can be sent simultaneously ( $N_4$ ).  $Strat(N_3) = Strat(N_4)$ .

**Lemma 3 (Rule 1: Merging of sending events).** *Let  $N_3$  and  $N_4$  be as defined.  $N_3$  accords with  $N_4$  and  $N_4$  accords with  $N_3$ .*

**Proof (Sketch).**

*It is sufficient to prove  $Strat(N_3) \subseteq Strat(N_4)$  and  $Strat(N_3) \supseteq Strat(N_4)$ .*

*$Strat(N_3) \subseteq Strat(N_4)$ : Let  $U = (P_U, T_U, F_U, I_U, O_U, m_{0_U}, \Omega_U) \in Strat(N_3)$  and  $N_3$  is not dead in  $N_3 \oplus U$ ; otherwise, we could trivially replace  $N_3$  by  $N_4$  without violating accordance. We have to show:  $U \in Strat(N_4)$ . Since  $N_3$  is not dead in  $N_3 \oplus U$ ,  $U$  has to mark places  $R$  in  $N_3$  in order to enable  $t_1$ . The final marking of  $N_3$  is the empty marking; thus, all transitions  $t_1, \dots, t_n$  in  $N_3$  have to be fired. Consequently,  $U$  has to consume all tokens from the output places of  $N_3$ . Consider now  $N_4$ .  $N_4$  has the same interface and the same final marking as  $N_3$  and  $t$  is enabled if and only if  $t_1$  is enabled (because  $\bullet t = \bullet t_1$ ). Therefore,  $U$  enables  $t$  in  $N_4 \oplus U$ . Since  $t$  has the same effect as  $t_1, \dots, t_n$  and there is no interaction between  $U$  and  $N_4$  (and  $U$  and  $N_3$ ) than sending,  $U$  can consume all tokens from the output places of  $N_4$ , too. The reason is, any transition  $t_U$  in  $U$  with  $p \in \bullet t_U$  and  $p \in I_U$  that can be enabled in  $N_3 \oplus U$  will also be enabled in  $N_4 \oplus U$ . Thus,  $U \in Strat(N_4)$ .*

*$Strat(N_3) \supseteq Strat(N_4)$ : Follows the same argumentation than above. Any transition  $t_U$  that is enabled in  $N_4 \oplus U$  will be eventually enabled in  $N_3$ . Therefore,  $U \in Strat(N_3)$  and hence the theorem holds.  $\square$*

As Rule 1 preserves accordance in both directions, we can derive that a sequence of sending events can also be reordered or it can be sent concurrently. Reordering of sending events and executing sending events concurrently also preserve accordance in both directions. It is worthwhile mentioning that  $N_3$  and  $N_4$  reflect exactly the Murata reduction rule Fusion of Series Transitions (see [23]).

Next, we redefine  $N_3$  and  $N_4$ ; that is, sending events are changed to receiving events. The corresponding oWFN pattern,  $N_5$  and  $N_6$ , are formalized as follows.

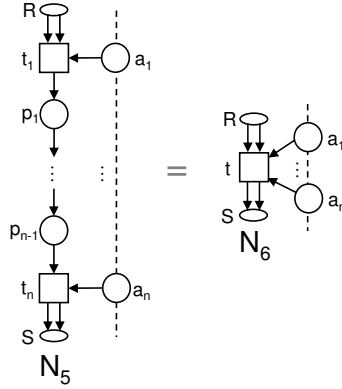
**Definition 23 ( $N_5$ ).** Let  $n > 0$ . Then,  $N_5 = (P, T, F, I, O, m_0, \Omega)$  is an oWFN pattern with

- $P = I \cup O \cup \{p_1, \dots, p_{n-1}\}$ ,
- $T = \{t_1, \dots, t_n\}$ ,
- $F = \{(r, t_1) \mid r \in R\}$   
 $\cup \{(t_n, s) \mid s \in S\}$   
 $\cup \{(t_i, p_i), (p_i, t_{i+1}) \mid i = 1, \dots, n-1\}$   
 $\cup \{(a_i, t_i) \mid i = 1, \dots, n\}$ ,
- $I = \{a_1, \dots, a_n\} \cup R$ ,
- $O = S$ .

**Definition 24 ( $N_6$ ).** Let  $n > 0$ . Then,  $N_6 = (P, T, F, I, O, m_0, \Omega)$  is an oWFN pattern with

- $P = I \cup O$ ,
- $T = \{t\}$ ,
- $F = \{(r, t) \mid r \in R\}$   
 $\cup \{(t, s) \mid s \in S\}$   
 $\cup \{(a_i, t) \mid i = 1, \dots, n\}$ ,
- $I = \{a_1, \dots, a_n\} \cup R$ ,
- $O = S$ .

$N_5$  and  $N_6$  are depicted on the left and right hand side of Fig. 6, respectively.



**Fig. 6.** Rule 2: A sequence of receiving events ( $N_5$ ) can be merged ( $N_6$ ).  $Strat(N_5) = Strat(N_6)$

Like a sequence of sending events (see Lemma 3), a sequence of receiving events can be executed simultaneously, too. Thus,  $N_5$  can be transformed into  $N_6$  and vice versa, and this transformation preserves accordance in both directions. Consequently,  $Strat(N_5) = Strat(N_6)$ . This is specified by Rule 2 in Fig. 6.

**Lemma 4 (Rule 2: Merging receiving events).** *Let  $N_5$  and  $N_6$  be as defined.  $N_5$  accords with  $N_6$  and  $N_6$  accords with  $N_5$ .*

**Proof (Sketch).**

*It is sufficient to prove  $\text{Strat}(N_5) \subseteq \text{Strat}(N_6)$  and  $\text{Strat}(N_5) \supseteq \text{Strat}(N_6)$ .*

*$\text{Strat}(N_5) \subseteq \text{Strat}(N_6)$ : Let  $U = (P_U, T_U, F_U, I_U, O_U, m_{0_U}, \Omega_U) \in \text{Strat}(N_5)$  and  $N_5$  is not dead in  $N_5 \oplus U$ ; otherwise, we could trivially replace  $N_5$  by  $N_6$  without violating accordance. We have to show:  $U \in \text{Strat}(N_6)$ . Since  $N_5$  is not dead in  $N_5 \oplus U$  and the final marking of  $N_5$  is the empty marking, all transitions  $t_1, \dots, t_n$  in  $N_5$  have to be fired. Thus,  $U$  has to mark all input places in  $N_5$  and consume all tokens that are produced by  $N_5$  on the places  $S$ . Consider now  $N_6$ .  $N_6$  has the same interface and the same final marking as  $N_5$  and  $t$  is enabled if all input places of  $N_6$  are marked. Since  $U$  receives no response from  $N_5$  until all input places of  $N_5$  are marked (i.e., until  $t_n$  is enabled),  $U$  will also enable  $t$  in  $N_6$  and then consume tokens on  $S$  which are produced by  $N_6$ . Thus,  $U \in \text{Strat}(N_6)$ .*

*$\text{Strat}(N_5) \supseteq \text{Strat}(N_6)$ : Follows the same argumentation than above. Therefore,  $U \in \text{Strat}(N_5)$  and hence the theorem holds.  $\square$*

From Lemma 4 we can derive (as for a sequence of sending events) that a sequence of receiving events can be reordered or the events can be executed concurrently while preserving accordance in both directions.

For the third transformation rule, we consider an oWFN pattern  $N_7$  that first receives some events in sequence and then sends some events in sequence.  $N_7$  is defined as follows.

**Definition 25 ( $N_7$ ).** *Let  $n > 0$  and  $0 \leq j \leq n$ . Then,  $N_7 = (P, T, F, I, O, m_0, \Omega)$  is an oWFN pattern with*

- $P = I \cup O \cup \{p_1, \dots, p_{n-1}\}$ ,
- $T = \{t_1, \dots, t_n\}$ ,
- $F = \{(r, t_1) \mid r \in R\}$   
 $\cup \{(t_n, s) \mid s \in S\}$   
 $\cup \{(t_i, p_i), (p_i, t_{i+1}) \mid i = 1, \dots, n-1\}$   
 $\cup \{(a_i, t_i) \mid i = 1, \dots, j\}$   
 $\cup \{(t_i, a_i) \mid i = j+1, \dots, n\}$ ,
- $I = \{a_1, \dots, a_{k-1}\} \cup R$ ,
- $O = \{a_k, \dots, a_n\} \cup S$ .

*$n$  is the number of all messages being sent and received by  $N_7$  and  $j$  defines the number of messages being received.*

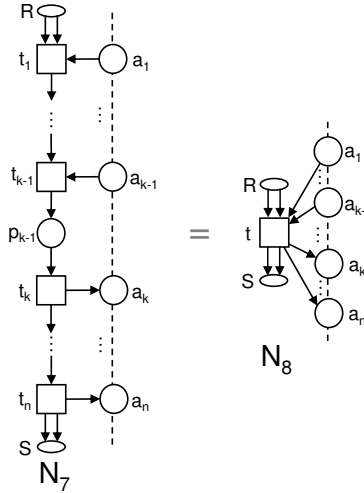
$N_7$  is depicted on the left hand side of Fig. 7. Next, we define  $N_8$  that results from merging all transitions  $t_1, \dots, t_n$  in  $N_7$  to a single transition  $t$ . Thus, all sending and receiving events are executed simultaneously in  $N_8$ .

**Definition 26 ( $N_8$ ).** *Let  $n > 0$  and  $0 \leq j \leq n$ . Then,  $N_8 = (P, T, F, I, O, m_0, \Omega)$  is an oWFN pattern with*

- $P = I \cup O$ ,
- $T = \{t\}$ ,
- $F = \{(r, t) \mid r \in R\}$   
 $\cup \{(t, s) \mid s \in S\}$   
 $\cup \{(a_i, t) \mid i = 1, \dots, j\}$   
 $\cup \{(t, a_i) \mid i = j + 1, \dots, n\}$ ,
- $I = \{a_1, \dots, a_{k-1}\} \cup R$ ,
- $O = \{a_k, \dots, a_n\} \cup S$ .

$n$  is the number of all messages being sent and received by  $N_8$  and  $j$  defines the number of messages being received.

$N_8$  is depicted on the right hand side of Fig. 7.



**Fig. 7.** Rule 3: A sequence of receiving events followed by a sequence of sending events ( $N_7$ ) can be executed simultaneously ( $N_8$ ).  $Strat(N_7) = Strat(N_8)$ .

The third accordance-preserving transformation rule, Rule 3, specifies that  $N_7$  can be transformed into  $N_8$  and vice versa without violating accordance; that is,  $Strat(N_7) = Strat(N_8)$ . Rule 3 is shown in Fig. 7 and its correctness is proven by the following lemma.

**Lemma 5 (Rule 3: Merging receiving and sending events).** *Let  $N_7$  and  $N_8$  be as defined.  $N_7$  accords with  $N_8$  and  $N_8$  accords with  $N_7$ .*

**Proof (Sketch).**

*Follows the same argumentation as Lemma 3 and Lemma 4.* □

It is worthwhile mentioning that Rule 1 and Rule 2 are special cases of Rule 3 with the set of input and output places being empty, respectively. More precisely,

if in  $N_7$  and  $N_8$  the parameter  $j$  is set to 0 ( $n$ ), the set of input (output) places is empty.

From Lemmata 3–5 we can derive that every oWFN pattern that has a transition connected to more than one interface place can be transformed into an equivalent oWFN pattern which has only transitions connected to a single interface place. In the following, without loss of generality, we therefore restrict ourselves to patterns where each transition is connected to at most one interface place.

So far, we excluded the possibility that a sending event is followed by a receiving event. For the next transformation rule, we define oWFN patterns  $N_9$  and  $N_{10}$ .  $N_9$  first sends a message  $a$  and then receives a message  $b$ . In  $N_{10}$ , in contrast, sending  $a$  and receiving  $b$  is executed concurrently.

**Definition 27 ( $N_9$ ).** Let  $N_9 = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with

- $P = I \cup O \cup \{p\}$ ,
- $T = \{t_a, t_b\}$ ,
- $F = \{(r, t_a) \mid r \in R\}$   
 $\cup \{(t_b, s) \mid s \in S\}$   
 $\cup \{(t_a, a), (t_a, p), (p, t_b), (b, t_b)\}$
- $I = \{b\} \cup R$ ,
- $O = \{a\} \cup S$ .

**Definition 28 ( $N_{10}$ ).** Let  $N_{10} = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with

- $P = I \cup O \cup \{p_1, p_2, p_3, p_4\}$ ,
- $T = \{t_a, t_b, t_1, t_2\}$ ,
- $F = \{(r, t_1) \mid r \in R\}$   
 $\cup \{(t_2, s) \mid s \in S\}$   
 $\cup \{(t_1, p_1), (t_1, p_2), (p_1, t_a), (p_2, t_b), (t_a, a), (b, t_b), (t_a, p_3), (t_b, p_4), (p_3, t_2),$   
 $(p_4, t_2)\}$ ,
- $I = \{b\} \cup R$ ,
- $O = \{a\} \cup S$ .

$N_9$  and  $N_{10}$  are depicted on the left and right hand side of Fig. 8, respectively.

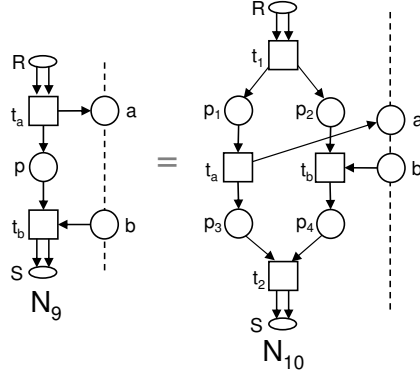
Rule 4 presented in Fig. 8 specifies that sending messages followed by receiving messages (transition  $t_a$  and  $t_b$  in  $N_9$ , respectively) can also be executed concurrently ( $N_{10}$ ) and vice versa. Rule 4 preserves accordance in both directions and hence  $\text{Strat}(N_9) = \text{Strat}(N_{10})$ .

**Lemma 6 (Rule 4: Send and receive in parallel).** Let  $N_9$  and  $N_{10}$  be as defined.  $N_9$  accords with  $N_{10}$  and  $N_{10}$  accords with  $N_9$ .

**Proof (Sketch).**

It is sufficient to prove  $\text{Strat}(N_9) \subseteq \text{Strat}(N_{10})$  and  $\text{Strat}(N_9) \supseteq \text{Strat}(N_{10})$ .

$\text{Strat}(N_9) \subseteq \text{Strat}(N_{10})$ : Let  $U \in \text{Strat}(N_9)$  and  $N_9$  is not dead in  $N_9 \oplus U$ ; otherwise, we could trivially replace  $N_9$  by  $N_{10}$  without violating accordance.



**Fig. 8.** Rule 4: Send and then receive ( $N_9$ ) can be executed concurrently ( $N_{10}$ ).  $\text{Strat}(N_9) = \text{Strat}(N_{10})$ .

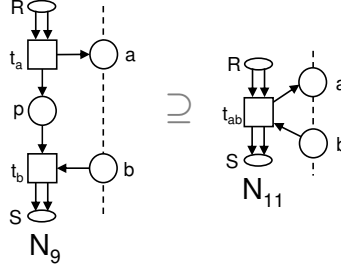
We have to show:  $U \in \text{Strat}(N_{10})$ . There is only one run in  $N_9$  possible (first  $t_a$  then  $t_b$ ) and this is a run of  $N_{10}$ , too. Thus, if  $U \in \text{Strat}(N_9)$ , then also  $U \in \text{Strat}(N_{10})$ .

$\text{Strat}(N_9) \supseteq \text{Strat}(N_{10})$ : Let  $U \in \text{Strat}(N_{10})$  and  $N_{10}$  is not dead in  $N_{10} \oplus U$ ; otherwise, we could trivially replace  $N_{10}$  by  $N_9$  without violating accordance. We have to show:  $U \in \text{Strat}(N_9)$ . Assume  $U \notin \text{Strat}(N_9)$ . We will show that such an  $U$  does not exist and thus we contradict the assumption. Obviously,  $U$  has to take advantage from the fact that  $t_a$  and  $t_b$  can be executed concurrently in  $N_{10}$ . In order that  $t_b$  can fire before  $t_a$ ,  $U$  has to send  $b$  before it receives  $a$ . However, in this case  $U$  is a strategy for  $N_9$ . As there is no other possibility for  $U$  to behave differently, every strategy for  $N_{10}$  is also a strategy for  $N_9$ . Hence,  $\text{Strat}(N_9) = \text{Strat}(N_{10})$  and the lemma holds.  $\square$

Sending and receiving simultaneously can also be transformed into first sending and then receiving. This is formalized in the next transformation rule, Rule 5. With it, we formalize oWFN pattern  $N_{11}$  which sends  $a$  and receives  $b$  simultaneously.  $N_{11}$  results from merging transitions  $t_a$  and  $t_b$  of  $N_9$  to a single transition  $t_{ab}$ .

**Definition 29** ( $N_{11}$ ). Let  $N_{11} = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with

- $P = I \cup O$ ,
- $T = \{t_{ab}\}$ ,
- $F = \{(r, t_{ab}) \mid r \in R\} \cup \{(t_{ab}, s) \mid s \in S\} \cup \{(t_{ab}, a), (b, t_{ab})\}$ ,
- $I = \{b\} \cup R$ ,
- $O = \{a\} \cup S$ .



**Fig. 9.** Rule 5: Send and then receive ( $N_9$ ) cannot be executed simultaneously ( $N_{11}$ ).  $Strat(N_{11}) \subseteq Strat(N_9)$ .

$N_{11}$  is depicted on the left hand side of Fig. 9.

Rule 5 illustrated in Fig. 9 only preserves accordance in one direction. This is formalized by the following lemma.

**Lemma 7 (Rule 5: Sending and receiving simultaneously).** *Let  $N_9$  and  $N_{11}$  be as defined.  $N_9$  accords with  $N_{11}$ .*

**Proof.**

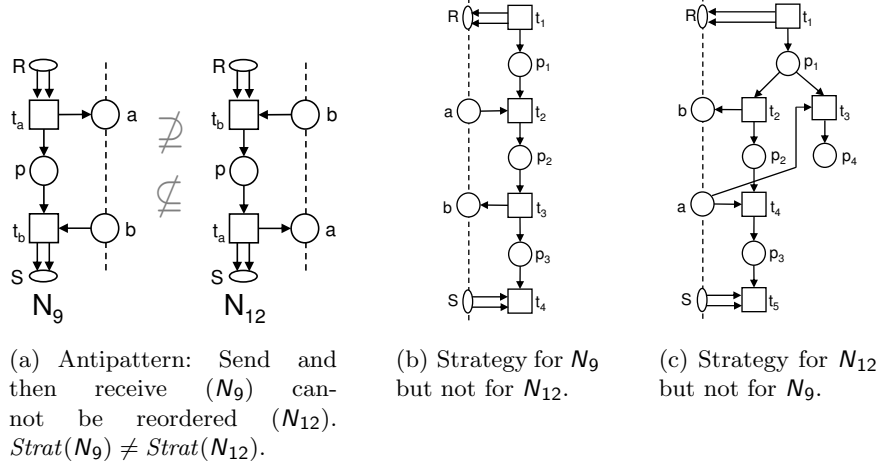
*It is sufficient to prove  $Strat(N_{11}) \subseteq Strat(N_9)$ . Let  $U \in Strat(N_{11})$  and  $N_{11}$  is not dead in  $N_{11} \oplus U$ ; otherwise, we could trivially replace  $N_{11}$  by  $N_9$  without violating accordance. We have to show:  $U \in Strat(N_9)$ . Since  $N_{11}$  is not dead in  $N_{11} \oplus U$  and the final marking of  $N_{11}$  is the empty marking,  $U$  has to enable  $t_{ab}$  (i.e., mark places  $R$  and  $b$ ) and after firing  $t_{ab}$  it has to consume all tokens from the output places of  $N_{11}$  (i.e., places  $a$  and  $S$ ).  $N_9$  and  $N_{11}$  have the same interface and the same final marking. Consequently, if we substitute  $N_{11}$  by  $N_9$ ,  $U$  enables transitions  $t_a$  and  $t_b$  in  $N_9 \oplus U$ . Furthermore,  $U$  consumes all tokens from the output places of  $N_9$ . Hence,  $U \in Strat(N_9)$ .  $\square$*

Although  $N_9$  accords with  $N_{11}$ ,  $N_{11}$  does not accord with  $N_9$ . The oWFN pattern depicted in Fig. 10(b) is a strategy for  $N_9$  but it is no strategy for  $N_{11}$ .

Now, let us consider  $N_{12}$  depicted on the right hand side of Fig. 10(a).  $N_{12}$  results from changing the order of sending and receiving in  $N_9$ . It is worthwhile mentioning that neither  $N_{12}$  accords with  $N_9$  nor  $N_9$  accords with  $N_{12}$ . The oWFN depicted in Fig. 10(b) is a strategy for  $N_9$  but no strategy for  $N_{12}$ . The oWFN depicted in Fig. 10(c), in contrast, is a strategy for  $N_{12}$  but not for  $N_9$ . From  $Strat(N_9) \neq Strat(N_{12})$  and Lemma 6 we can derive  $Strat(N_{10}) \neq Strat(N_{12})$ . That means, transforming  $N_{12}$  into  $N_{10}$  or transforming  $N_{10}$  into  $N_{12}$  violates accordance.

However, we can merge transitions  $t_a$  and  $t_b$  in  $N_{12}$  to a single transition and this transformation preserves accordance in both directions. This is, in fact, an application of Lemma 5.

The next rule, Rule 6, specifies how an alternative branch can be added to an oWFN pattern  $N_{13}$  depicted on the left hand side of Fig. 11. The pattern  $N_{13}$  first receives  $a$  and then enters either the left or the right branch. In each branch,



**Fig. 10.** Counterexample.

messages are sent (b and c, respectively), and then messages are received (d and e, respectively).

**Definition 30 ( $N_{13}$ ).** Let  $N_{13} = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with

- $P = I \cup O \cup \{p_1, p_2, p_3\}$ ,
- $T = \{t_a, t_b, t_c, t_d, t_e\}$ ,
- $F = \{(r, t_a) \mid r \in R\}$   
 $\cup \{(t_d, s) \mid s \in S\}$   
 $\cup \{(t_e, s) \mid s \in S\}$   
 $\cup \{(a, t_a), (t_a, p_1), (p_1, t_b), (p_1, t_c), (t_b, b), (t_c, c), (t_b, p_2), (t_c, p_3), (p_2, t_d), (p_3, t_e), (d, t_d), (e, t_e)\}$ ,
- $I = \{a, d, e\} \cup R$ ,
- $O = \{b, c\} \cup S$ .

The pattern  $N_{13}$  can be transformed into  $N_{16}$  (the net depicted on the right hand side of Fig. 11) by adding an alternative branch. In this branch, d is received, and then a message f is sent. Afterwards, this branch can be arbitrary; that is, there can be any continuation (including direct continuation in S) of this net illustrated by the frame.

First of all, we define an oWFN pattern  $N_{14}$  specifying the alternative branch to be added to  $N_{13}$ .  $N_{14}$  is very general. It is only required that  $N_{14}$  contains places  $R$  and  $S$ .  $N_{14}$  is concretized by defining its subnet  $N_{15}$ . Afterwards,  $N_{15}$ , the subnet contained in  $N_{14}$ , is defined. Finally,  $N_{16}$  is defined by merging  $N_{13}$  and  $N_{14}$ .

**Definition 31 ( $N_{14}$ ).** Let  $N_{14} = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with  $R, S \subset P$ , with  $R = R_{N_{13}}$  and  $S = S_{N_{13}}$ .

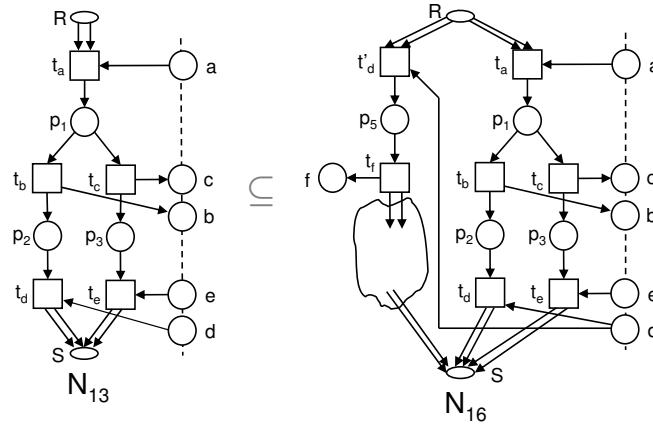
**Definition 32 ( $N_{15}$ ).** Let  $N_{15} = (P, T, F, I, O, m_0, \Omega)$  be a subnet of  $N_{14}$  with

- $P = R_{N_{14}} \cup \{p_5, d, f\}$ ,  $p_5 \notin S_{N_{14}}$
- $T = \{t'_d, t_f\}$ ,
- $F = \{(r, t'_d) \mid r \in R_{N_{14}}\} \cup \{(d, t'_d), (t'_d, p_5), (p_5, t_f), (t_f, f)\}$  and in  $N_{14}$  there are no other arcs connected to  $R_{N_{14}}$ ,  $p_5$ ,  $t'_d$ ,  $t_f$  except outgoing arcs for  $t_f$ ,
- $I = \{d\}$  and  $d \in I_{N_{13}}$ ,
- $O = \{f\}$ .

oWFN pattern  $N_{16}$  results from merging patterns  $N_{13}$  and  $N_{15}$  and is defined as follows.

**Definition 33 ( $N_{16}$ ).** Let  $N_{16} = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with

- $P = P_{N_{13}} \cup P_{N_{14}}$ ,
- $T = T_{N_{13}} \cup T_{N_{14}}$ ,
- $F = F_{N_{13}} \cup F_{N_{14}}$ ,
- $I = I_{N_{13}} \cup I_{N_{14}}$ ,
- $O = O_{N_{13}} \cup O_{N_{14}}$ .



**Fig. 11.** Rule 6: Adding an alternative branch to  $N_{13}$  starting with an receiving event results in  $N_{16}$ .  $Strat(N_{13}) \subseteq Strat(N_{16})$ .

Rule 6 specifies that  $N_{13}$  can be transformed into  $N_{16}$  while preserving accordance. More detailed,  $Strat(N_{13}) \subseteq Strat(N_{16})$ . The correctness of this rule is justified by the following lemma.

**Lemma 8 (Rule 6: Adding an alternative branch).** Let  $N_{13}$  and  $N_{16}$  be as defined.  $N_{16}$  accords with  $N_{13}$ .

**Proof (Sketch).**

It is sufficient to prove  $\text{Strat}(N_{13}) \subseteq \text{Strat}(N_{16})$ . Let  $U = (P_U, T_U, F_U, I_U, O_U, m_{0_U}, \Omega_U) \in \text{Strat}(N_{13})$  and  $N_{13}$  is not dead in  $N_{13} \oplus U$ ; otherwise, we could trivially replace  $N_{13}$  by  $N_{16}$  without violating accordance. We have to show:  $U \in \text{Strat}(N_{16})$ .

We will prove the lemma by contradiction. Assume that  $U$  is no strategy for  $N_{16}$ . This implies that the composition of  $U$  and  $N_{16}$  can deadlock because of choosing the (newly added) left branch (i.e.,  $N_{14}$ ) in  $N_{16}$ . Consider how  $U$  interacts with  $N_{13}$ :  $U$  cannot send  $d$  or  $e$  before it has received  $b$  or  $c$ . The reason is, the oWFN pattern  $N_{13}$  decides if  $b$  or  $c$  is sent. Only by knowing that decision,  $U$  can react to message  $b$  and  $c$  being sent by  $N_{13}$  by sending  $d$  and  $e$ , respectively. As a result,  $U$  cannot control (i.e., it cannot enter) the left branch of  $N_{16}$ . Thus,  $N_{16} \oplus U$  cannot deadlock and therefore  $U$  is also a strategy for  $N_{16}$ . This contradicts the assumption and hence,  $\text{Strat}(N_{13}) \subseteq \text{Strat}(N_{16})$  holds.  $\square$

However,  $N_{13}$  accords with  $N_{16}$  does not hold in general: Assume the left branch of  $N_{16}$  is controllable in isolation. Then there is at least one strategy  $U'$  for  $N_{16}$ .  $U'$  had to put a token on place  $d$  and thus to control the left branch. This is, in fact, not possible for any strategy for  $N_{13}$ . Thus,  $U' \notin \text{Strat}(N_{13})$ .

The intuition behind the next transformation rule (Rule 7) is the possibility to add (remove) “dead code” to (from) a service. To motivate this transformation rule, consider a party that wants to reuse an existing service in the contract. This service may provide functionality to other parties not involved in the current contract. Technically, in the first step, this party makes internal all interface places of this service that are not used and in the second step, it looks for transformation rules justifying the service to be a valid private view.

To formalize Rule 7, we have to define oWFN patterns  $N_{17}$ – $N_{21}$ .  $N_{17}$  receives a message  $a$ , then sends a message  $b$ , and finally it can behave arbitrarily. For this purpose,  $N_{17}$  is defined quite general and concretized by its subnet  $N_{18}$ .

**Definition 34 ( $N_{17}$ ).** Let  $N_{17}$  be an oWFN pattern. Let further  $N_{18} = (P, T, F, I, O, m_0, \Omega)$  be a subnet of  $N_{17}$  with

- $P = R_{N_{17}} \cup \{p_1, a, b\}$ ,
- $T = \{t_a, t_b\}$ ,
- $F = \{(r, t_a) \mid r \in R\} \cup \{(a, t_a), (t_a, p_1), (p_1, t_b), (t_b, b)\}$  and in  $N_{17}$  there are no other arcs connected to  $R_{N_{17}}$ ,  $p_1$ ,  $t_a$ ,  $t_b$  except outgoing arcs for  $t_b$ ,
- $I = \{a\}$ ,
- $O = \{b\}$ .

The oWFN pattern on the left hand side of Fig. 12 illustrates  $N_{17}$ . The arbitrary part of  $N_{17}$  is depicted by a frame.

The oWFN pattern  $N_{21}$  results from adding an alternative branch to  $N_{17}$ . This branch can be entered if place  $c$  is marked. Afterwards, the branch may behave arbitrarily. In the end, both branches are synchronized in  $S$ . However,  $c$  is an example of an internal place with empty preset (it is a former interface place). Thus, transition  $t_c$  will never be enabled.

First of all, we define oWFN pattern  $N_{19}$ , the branch that is added to  $N_{17}$ .  $N_{19}$  shares the initial and final marking with  $N_{17}$ .

**Definition 35 ( $N_{19}$ ).** Let  $N_{19} = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with  $R, S \subset P$ , with  $R = R_{N_{17}}$  and  $S = S_{N_{17}}$ .

$N_{19}$  contains a subnet  $N_{20}$  which concretizes  $N_{19}$ .

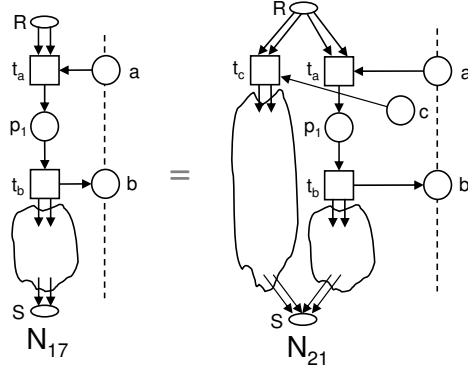
**Definition 36 ( $N_{20}$ ).** Let  $N_{20} = (P, T, F, I, O, m_0, \Omega)$  be a subnet of  $N_{19}$  with

- $P = R_{N_{19}} \cup \{c\}$ ,
- $T = \{t_c\}$ ,
- $F = \{(r, t_c) \mid r \in R_{N_{19}}\} \cup \{(c, t_c)\}$  and in  $N_{19}$  there are no other arcs connected to  $R_{N_{19}}$  and  $t_c$ , except outgoing arcs for  $t_c$ ,
- $I = \emptyset$ ,
- $O = \emptyset$ .

Finally,  $N_{21}$  depicted on the right hand side of Fig. 12 results from merging the patterns  $N_{19}$  and  $N_{17}$ .

**Definition 37 ( $N_{21}$ ).** Let  $N_{21} = (P, T, F, I, O, m_0, \Omega)$  be an oWFN pattern with

- $P = P_{N_{17}} \cup P_{N_{19}}$ ,
- $T = T_{N_{17}} \cup T_{N_{19}}$ ,
- $F = F_{N_{17}} \cup F_{N_{19}}$ ,
- $I = I_{N_{17}} \cup I_{N_{19}}$ ,
- $O = O_{N_{17}} \cup O_{N_{19}}$ .



**Fig. 12.** Rule 7: Adding a dead code to  $N_{17}$  results in  $(N_{21})$ .  $Strat(N_{17}) = Strat(N_{21})$

Rule 7 preserves accordance in both directions, meaning neither adding nor deleting “dead code” will change the set of strategies for  $N_{17}$  and  $N_{21}$ . Consequently,  $Strat(N_{17}) \subseteq Strat(N_{21})$ .

**Lemma 9 (Rule 7: Adding / Removing dead code).** *Let  $N_{17}$  and  $N_{21}$  be as defined.  $N_{17}$  accords with  $N_{21}$  and  $N_{21}$  accords with  $N_{17}$ .*

**Proof (Sketch).**

*It is sufficient to prove  $\text{Strat}(N_{17}) = \text{Strat}(N_{21})$ .*

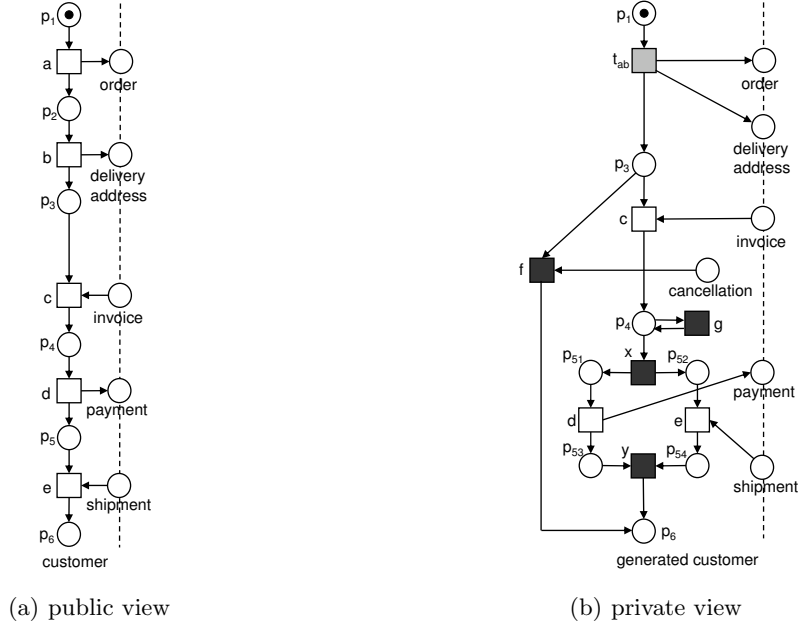
*Place  $c$  in  $N_{21}$  is an internal place and the preset of  $c$  is empty (i.e.,  $\bullet c = \emptyset$ ). Since  $c$  is not marked in the initial marking of  $N_{21}$ , there will never be a token on  $c$  and thus transition  $t_c$  is dead. Consequently,  $N_{19}$  is dead. Since “dead code” does not affect the set of strategies, we can conclude that  $\text{Strat}(N_{17}) = \text{Strat}(N_{21})$ .  $\square$*

The seven transformation rules presented in this section reflect the crucial impact of the order of sending and receiving messages. The first two rules show that sequences of sending events and sequences of receiving events can be executed simultaneously while preserving accordance in both directions. This was our motivation to consider only oWFNs where each transition is connected to at most one interface place. Transforming first-send-then-receive into send-and-receive-concurrently preserves accordance in both directions (Fig. 8). However, first-send-then-receive cannot be transformed into send-and-receive-simultaneously. In contrast, receive-and-send-simultaneously can be transformed into first-send-then-receive while preserving accordance (Fig. 9). Consider first-receive-then-send next. It can be transformed into receive-and-send-simultaneously (Fig. 7), but it cannot be transformed into receive-and-send-concurrently.

## 5.5 Case Study

In this section, we demonstrate how accordance-preserving transformation rules can be applied to derive a private view of the customer from its respective public view, taken from the running example in Fig. 1(a). On first sight, the generated customer depicted in Fig. 13(b) and the (original) customer (depicted again in Fig. 13(a) to ease the comparison) are not very similar. We will now show that the generated customer was derived from the original customer by applying the transformation rules defined in the last two sections:

- The messages order and delivery address can be sent simultaneously to the supplier by transition  $t_{ab}$  (transitions created by merging transitions of the public view are depicted in light gray). The merging of transition  $t_a$  and  $t_b$  is justified by Rule 1.
- Sending the payment and receiving the shipment can be done concurrently. Therefore we had to add two new internal transitions  $x$  and  $y$  (newly-added transitions are depicted in dark gray). This transformation is justified by Rule 4.
- After receiving the invoice the customer can check the invoice arbitrary times. This is modelled by the newly added loop transition  $g$ .
- Finally, a new branch was added to the customer, starting with transition  $f$ . Intuitively, this branch models additional behavior that is available when



**Fig. 13.** The public view (a) and a private view (b) of the agency of Fig. 1(a).

the **generated customer** is running in a different environment. When **cancellation** is an input place for messages sent from a (modified) supplier service, the newly-added branch can be triggered by messages. Thus, the **generated customer** can be reused in a different contract. However, place **cancellation** is not exposed as interface place, and as it is not marked, the branch is dead. Therefore, the addition is justified by Rule 7.

As all rules applied are accordance-preserving, the **generated customer** in Fig. 13(b) is a correct private view of the **customer** in Fig. 13(a), and thus accords with the running example contract (cf. Fig. 1(a)).

## 6 Related Work

As already mentioned in the introduction, the work presented in this paper mainly builds on results presented in [9, 10] where classical workflow nets [15] are used as a formal model and projection inheritance [16, 17] is used for relating the implementation of a contract to the original contract. In this paper, we presented a more generic notion of a contract using open workflow nets. Thus, a public workflow is not restricted to be a workflow net and therefore “massaging”; that is, transforming a possible unconnected net (i.e., a net that consists of several parts) into a workflow net is no longer necessary. Furthermore, our notion of

accordance is weaker than projection inheritance since it uses asynchronous message passing rather than synchronous communication as an interaction model. Consequently, we could define accordance-preserving transformation rules which are less restrictive than the rules presented in [9].

The accordance preserving-transformation rules presented in Figures 5–7 can be seen as an enhancement of existing rules for structural refinement and abstraction of Petri nets as presented in [23, 24].

The concept of contracts is also related to the problem when a service can be substituted by another service. Most of this work, however, is restricted to synchronous communication [25–27] whereas we consider asynchronous message passing. Benatallah et al. [27] present four notions of substitutability. In this paper, we cover two of them: *equivalence* and *subsumption*. Equivalence in our notion means that both services have the same set of strategies and subsumption means the inclusion of the set of strategies.

Fournier et al. present in [28] a refinement relation for CCS processes between a specification  $S$  and an implementation  $I$  of asynchronous message passing software components. This relation is called *stuck-free conformance* and formalizes (like weak termination) deadlock-freedom of the system. It further satisfies the substitutability property: If  $I$  conforms to  $S$  and  $E$  is an environment such that the composition of  $E$  and  $S$  is stuck-free, then the composition of  $E$  and  $I$  is stuck-free, too. In contrast to accordance, this relation does not allow (similar to inheritance) the re-ordering of sending or receiving events. To check conformance, the model checker Zing [29] is used.

The *ComFoRT* framework [30] analyzes whether a software component  $S$  implemented in the programming language C can be substituted by another software component  $S'$ .  $S$  can be substituted by  $S'$  if the following two criteria hold: (i) every behavior possible in  $S$  must also be a behavior of  $S'$ , and (ii) the new version of the software system must satisfy previously established correctness properties.

The idea of using annotated automata as a representation of a set of automata has been first published in [31]. However, to the best of our knowledge, there exists no approach which uses the novel concept of operating guidelines to characterize when a service  $N$  can be substituted by a service  $N'$ .

## 7 Conclusion

In interorganizational cooperation the involved parties specify a public version of the overall process which serves as a contract. Later on, each party implements its part of the contract (i.e., the public view). Such a local modification is nontrivial as it may cause global errors such as deadlocks. This shows the necessity for a formal framework which guides the user during the modification process.

In this paper, we proposed a formal notion of a contract based on open workflow nets. Our correctness criterion, weak termination, guarantees the ability to always be able to terminate properly (e.g., the overall process cannot run into

a deadlock or livelock). To decide if the public view  $N$  of a party can be substituted by a modified version, the private view  $N'$ , we presented the notion of accordance.  $N'$  accords with  $N$  if  $N$  and  $N'$  have the same interface and any environment that can cooperate with  $N$  can also cooperate with  $N'$ . The value of our accordance notion is that it can be checked locally and guarantees that the overall workflow preserves the weak termination property.

To check accordance automatically, we introduced our concept of operating guidelines. The operating guideline of an oWFN  $N$ ,  $OG_N$ , characterizes all oWFNs  $M$  (called strategies) such that the composition of  $M$  and  $N$  weakly terminates. We proved for acyclic finite state oWFNs  $N$  and  $N'$  that  $N'$  accords with  $N$  if the operating guideline of  $N'$  characterizes at least the strategies that are characterized by the operating guideline of  $N$ .

In addition, we also presented accordance-preserving transformation rules to derive  $N'$  from  $N$ . Accordance guarantees that the overall process will always terminate properly; that is, the overall process cannot run into a deadlock or livelock. We showed that some of the rules preserve accordance in both directions while other preserve accordance only in one direction. We discussed that the notion of accordance generalizes the notion of projection inheritance [9, 10, 16, 17]. As a consequence, we showed that projection inheritance implies accordance and therefore the inheritance-preserving transformation rules preserve accordance.

In ongoing work, we want to generalize our accordance check to cyclic oWFNs. For this purpose, our correctness criterion must also guarantee the absence of livelocks which is more challenging than checking deadlocks. Furthermore, we look for other correctness criteria than weak termination. Moreover, we want to relate the notion of accordance to other equivalence notions described in literature.

## References

1. Aalst, W.M.P.v.d., Anyanwu, K.: Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. In Dognac, A., Heck, E.v., Saarinen, T., et al., eds.: Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC 1999), Nashville, Tennessee (1999) 141–157
2. Benjamin, R., Wigand, R.: Electronic markets and virtual value chains on the information superhighway. Sloan Management Review (1995) 62–72
3. Kalakota, R., Whinston, A.B.: Frontiers of Electronic Commerce. Addison-Wesley, Reading, Massachusetts (1996)
4. Malone, T.W., Benjamin, R.I., Yates, J.: Electronic Markets and Electronic Hierarchies: Effects of Information Technology on Market Structure and Corporate Strategies. Communications of the ACM **30**(6) (1987) 484–497
5. Merz, M., Liberman, B., Muller-Jones, K., Lamersdorf, W.: Interorganisational Workflow Management with Mobile Agents in COSM. In: Proceedings of Conference on the Practical Application of Agents and Multiagent Systems (PAAM 1996). (1996)
6. The White House: A Framework for Global Electronic Commerce. <http://www.ecommerce.gov/framework.htm> (1997)

7. Zwass, V.: Electronic commerce: structures and issues. *International Journal of Electronic Commerce* **1**(1) (1996) 3–23
8. Papazoglou, M.P.: Agent-oriented technology in support of e-business. *Commun. ACM* **44**(4) (2001) 71–77
9. Aalst, W.M.P.v.d.: Inheritance of Interorganizational Workflows: How to agree to disagree without loosing control? *Information Technology and Management Journal* **4**(4) (2003) 345–389
10. Aalst, W.M.P.v.d., Weske, M.: The P2P approach to Interorganizational Workflows. In Dittrich, K.R., Geppert, A., Norrie, M.C., eds.: *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001)*. Volume 2068 of *Lecture Notes in Computer Science.*, Springer (2001) 140–156
11. Kumar, A., Zhao, J.L.: Workflow Support for Electronic Commerce Applications. *Decision Support Systems* **32**(3) (2002) 265–278
12. Sheth, A.P., Aalst, W.M.P.v.d., Arpinar, I.B.: Processes Driving the Networked Economy: ProcessPortals, ProcessVortex, and Dynamically Trading Processes. *IEEE Concurrency* **7**(3) (1999) 18–31
13. Reisig, W.: *Petri Nets*. EATCS Monographs on Theoretical Computer Science edn. Springer, Berlin, Heidelberg, New York, Tokyo (1985)
14. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. *Annals of Mathematics, Computing & Teleinformatics* **1**(3) (2005) 35–43
15. Aalst, W.M.P.v.d.: The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers* **8**(1) (1998) 21–66
16. Aalst, W.M.P.v.d., Basten, T.: Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science* **270**(1-2) (2002) 125–203
17. Basten, T., Aalst, W.M.P.v.d.: Inheritance of Behavior. *Journal of Logic and Algebraic Programming* **47**(2) (2001) 47–145
18. Aalst, W.M.P.v.d.: Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. *Electronic Commerce Research* **2**(3) (2002) 195–231
19. Massuthe, P., Schmidt, K.: Operating Guidelines – an Automata-Theoretic Foundation for the Service-Oriented Architecture. In Cai, K.Y., Ohnishi, A., Lau, M.F., eds.: *Proceedings of the Fifth International Conference on Quality Software (QSIC 2005)*, Melbourne, Australia, IEEE Computer Society (2005) 452–457
20. Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services. In: *Proceedings of the International Conference on Petri Nets and Other Models of Concurrency (ICATPN'07)*. (2007) accepted.
21. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing Interacting BPEL Processes. In Dustdar, S., Fiadeiro, J.L., Sheth, A., eds.: *Fourth International Conference on Business Process Management (BPM 2006)*. Volume 4102 of *Lecture Notes in Computer Science.*, Springer (2006) 17–32
22. Glabbeek, R.J.v., Weijland, W.P.: Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM* **43**(3) (1996) 555–600
23. Murata, T.: *Petri Nets: Properties, Analysis and Applications*. *Proceedings of the IEEE* **77**(4) (1989) 541–580
24. Berthelot, G.: Transformations and Decompositions of Nets. In Brauer, W., Reisig, W., Rozenberg, G., eds.: *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*. Volume 254 of *Lecture Notes in Computer Science.*, Springer (1987) 360–376

25. Bordeaux, L., Salaün, G., Berardi, D., Mecella, M.: When are Two Web Services Compatible? In Shan, M.C., Dayal, U., Hsu, M., eds.: Proceedings of the Fifth International Workshop on Technologies for E-Services (TES 2004). Volume 3324 of Lecture Notes in Computer Science., Springer (2004) 15–28
26. Beyer, D., Chakrabarti, A., Henzinger, T.A.: Web service interfaces. In Ellis, A., Hagino, T., eds.: Proceedings of the 14th international conference on World Wide Web (WWW 2005), ACM (2005) 148–159
27. Benatallah, B., Casati, F., Toumani, F.: Representing, analysing and managing Web service protocols. *Data Knowl. Eng.* **58**(3) (2006) 327–357
28. Fournet, C., Hoare, C.A.R., Rajamani, S.K., Rehof, J.: Stuck-Free Conformance. In Alur, R., Peled, D., eds.: Proceedings of the 16th International Conference on Computer Aided Verification (CAV 2004). Volume 3114 of Lecture Notes in Computer Science., Springer (2004) 242–254
29. Rajamani, S.K., Rehof, J.: Models for Contract Conformance. In Margaria, T., Steffen, B., eds.: First International Symposium on Leveraging Applications of Formal Methods (ISoLA 2004). Volume 4313 of Lecture Notes in Computer Science., Springer (2006) 181–196
30. Sharygina, N., Chaki, S., Clarke, E.M., Sinha, N.: Dynamic Component Substitutability Analysis. In Fitzgerald, J., Hayes, I.J., Tarlecki, A., eds.: Proceedings of the International Symposium of Formal Methods Europe (FM 2005). Volume 3582 of Lecture Notes in Computer Science., Springer (2005) 512–528
31. 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