Decentralized Consistency Checking in Cross-organizational Workflows

Andreas Wombacher
University of Twente, 7500 AE Enschede, The Netherlands
A.Wombacher@utwente.nl

Abstract

Service Oriented Architectures facilitate loosely coupled composed services, which are established in a decentralized way. One challenge for such composed services is to guarantee consistency, i.e., deadlock-freeness. This paper presents a decentralized approach to consistency checking, which utilizes only bilateral views of the composed service based on a synchronous communication model.

1 Introduction

Service Oriented Architectures (SOA) are based on services, which are independently maintained by service providers, without a centralized coordination of these services. An essential of SOA is service composition, that is, the combination of services such that output of one service is used as input for another service to provide more complex functionality.

Service composition of simple or stateless services is quite easy because services are used as a method call and calling a service does not influence other services. In case a service itself is a state dependent service, that is, internally maintains a state, the service composition gets more complicated, because these states may influence other services as a side effect and therefore have to be considered during the service composition. Since these side effects are not visible to all parties no centralized coordinator of the service composition can check consistency of the composed service, that is, the guarantee that the execution of the composed service does not run into a deadlock. As a consequence a decentralized consistency checking of composed services is required.

In this paper such a decentralized consistency checking based on supported execution sequences provided by the services is presented. In particular, state dependent services are considered as public workflows, which are abstracted from the private, internal workflows of the corresponding service. The combination of the public workflows specifies the cross-organizational workflow, that is, the supported execution sequences of the composed service. In case the cross-organizational workflow is deadlock-free it is considered to be consistent [1]. While several approaches exist for centralized consistency checking in this paper a decentralized consistency checking is proposed and its equivalence to consistency of the corresponding cross-organizational workflow is shown.

This paper focus on acyclic workflows without considering parameter constraints. These assumptions are reasonable since (i) business processes are designed to terminate, thus unlimited cyclic workflows are impractical and cycles can be unfolded resulting in acyclic workflows, and (ii) business processes do not reveal mission critical information usually contained in constraints like the highest price a party is willing to pay, thus parameter constraints are less applicable. However, an outlook on extending the presented approach to workflows with parameter constraints is presented in Section 6 and a complete description of the approach is available in [10].

The paper starts with an example (Section 2) and an introduction of the used formal model (Section 3). Next, the centralized and decentralized consistency definitions are introduced (Section 4) and the correctness of the proposed approach is shown (Section 5). Finally, related work (Section 7) and conclusions (Section 8) are presented.

2 Scenario

The example used throughout this paper is a simple procurement workflow within a virtual enterprise comprising a buyer, an accounting and a logistics department. The public workflows of the involved service providers further called parties are depicted in Figure 1 represented in a kind of a Finite State Automaton (FSA)[6] notation [2]. In this notation states are represented as circles, transitions represent message exchanges denoted as arcs where an arc is labeled

[1] A consistency definition including parameter constraints is contained in [10].

[2] In this paper annotated FSA notation is used to represented workflows, although other models like Workflow Nets (WF-Net) [1] or statecharts [5] could also have been used.

by the sender, the recipient, and the message name of the exchange. The termination state of an FSA is represented by a final state denoted by a circle with a thick line.

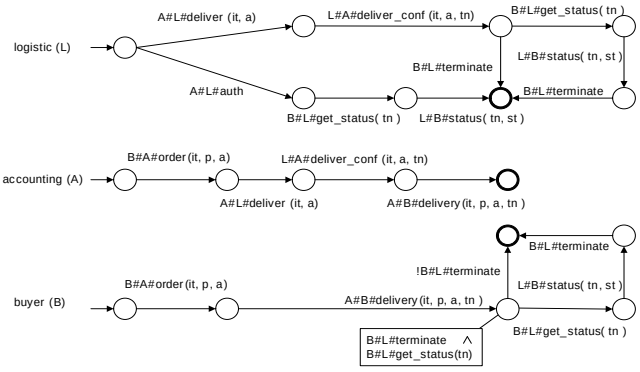

Figure 1. Public Workflow Models

The process is started by buyer *B* sending a *B#A#order* message to accounting *A* with the parameters item number *it*, price *p*, and amount *a*. Accounting *A* informs logistics *L* via a *A#L#deliver* message to deliver the ordered goods without forwarding the price parameter *p* of the order. The receiving *L* confirms the order with a *L#A#deliver_conf* message providing an additional tracking number (*tn* parameter). *A* forwards the delivery details of the order (*A#B#delivery* message) to *B*. Afterwards, *B* can either terminate the process (*B#L#terminate* message) or ask *L* for the status of the delivery by sending a *B#L#get_status* message containing a tracking number parameter *tn* answered by a *L#B#status* message with an additional status parameter *st*. While *B* must have received the *A#B#delivery* message before tracking parcels *L* allows parcel tracking at any time after receiving an authentication message from *A* (*A#L#auth* message).

3 Formal Model

As stated in the introduction, service composition of state dependent services can be restated as combining public workflows and deciding consistency, i.e., deadlock-freeness. Combining workflows result in a so called cross-organizational workflow and different formal models have been proposed. These models can be classified in accordance to the underlying communication model: asynchronous communication like e.g. supported by [1, 7], or synchronous communication like e.g. supported by [13]. In the following the focus is on synchronous communication since the same order of sent and received messages is guaranteed, which simplifies the problem.

The used workflow model is called annotated Finite State Automata (aFSA) [13], which extends classical Finite State Automata (FSA) [6] by supporting different types of transitions affecting consistency checking only. In particular, mandatory and optional transitions are distinguished: mandatory transitions are those where **all** transitions must be supported by a trading partner, while optional transitions **could** be supported by a trading partner. Classical Finite State Automata (FSA) are extended by annotating states with logical formulas based on transition labels: mandatory transitions are represented as conjunctions reflecting that each transition must be supported, while optional transitions are given as disjunctions.

The annotations are standard Boolean formulas, which are defined similar as in [3]:

Definition 1 ***(definition of formulas)***
The syntax of the supported logical formulas is given as follows: (i) the constants $true$ and $false$ are formulas, (ii) the variables $v \in \Sigma$ are formulas, where Σ is a set of messages that is transition labels (iii) if ϕ is a formula, so is $\neg\phi$, (iv) if ϕ and ψ are formulas, so is $\phi \wedge \psi$ and $\phi \vee \psi$.

The set of all formulas is denoted as E. Based on these formulas, annotated Finite State Automata (aFSA) can be defined.

Definition 2 *(aFSA)*
An aFSA $A := (Q, \Sigma, \Delta, q_0, F, QA)$, where Q is a finite set of states, Σ is a finite set of messages denoted $s\#r\#msg$ with sender $s \in \mathcal{P}$ sending message msg to recipient $r \in \mathcal{P}$, with $\mathcal{P}$ being the set of all parties, $\Delta : Q \times \Sigma \times Q$ represents labeled transitions, q_0 a start state with $q_0 \in Q$, $F \subseteq Q$ a set of final states, and $QA : Q \times E$ is a finite relation of states and logical terms within the set E of propositional logic terms.

Graphically, states are represented as circles, while transitions are depicted as arcs connecting states labeled with sender, recipient and message information. The extended annotations of states are given within a square connected to the corresponding state. Optional annotations are usually not depicted since they are considered to be default. Examples of aFSAs are depicted in Figure 1.

Public workflows represent interactions with several trading partners. Thus, to check bilateral consistency with a single trading partner it is necessary to abstract from the public workflow the relevant part representing the interaction with this particular partner. In particular, all messages, which are unrelated to the abstracting party's public workflow, are relabeled by an empty word ϵ.

Definition 3 *(abstraction)*
An aFSA $A' = (Q, \Sigma', \Delta', q_0, F, QA')$ with $A' = \tau_p(A)$ is an abstraction of an aFSA $A = (Q, \Sigma, \Delta, q_0, F, QA)$ with regard to a party p, where

$$\tau_p(s\#r\#msg) := \begin{cases} s\#r\#msg & \text{if } (s = p) \vee (r = p) \\ \epsilon_{s\#r\#msg} & otherwise \end{cases}$$

with $\Sigma' := \{\tau_p(\alpha) \mid \alpha \in \Sigma\}$, $\Delta' := \{(q, \tau_p(\alpha), q') \mid (q, \alpha, q') \in \Delta\}$, $QA' := \{(q, \tau_p(e)) \mid (q, e) \in QA\}$ *with* $\tau_p(e)$ *being a classical string replacement in a logical expression.*

The resulting aFSAs of applying the logistics abstraction ($\tau_{logistics}$) on the buyer public workflow and the buyer abstraction (τ_{buyer}) on the logistics public workflow are depicted in Figure 2(a).

Next, intersection and emptiness operations are defined being the basis for the used notion of consistency. In particular, intersection is based on the cross product of transitions and states, while the corresponding annotations are combined by conjunctions respectively. Due to the introduction of silent transitions ϵ in the abstraction the definition of intersection must copy the ϵ transitions of each aFSA to the intersection automaton.

Definition 4 *(intersection)*
The intersection $A_1 \cap A_2$ *of two aFSA with* $A_i = (Q_i, \Sigma_i, \Delta_i, q_{i0}, F_i, QA_i)$ *(*$i \in \{1, 2\}$*) is an aFSA* $A = (Q, \Sigma, \Delta, q_0, F, QA)$ *based on the usual cross-product construction with* $Q = Q_1 \times Q_2$, $\Sigma = \Sigma_1 \cap \Sigma_2$, $q_0 = (q_{10}, q_{20})$, $F = F_1 \times F_2$, *and*

$$\begin{aligned}\Delta := \quad & \{((q_{11}, q_{21}), s\#r\#msg, (q_{12}, q_{22})) \mid \\ & (q_{11}, s\#r\#msg, q_{12}) \in \Delta_1 \wedge (q_{21}, s\#r\#msg, q_{22}) \in \Delta_2\} \\ & \cup\{((q_{11}, q), \epsilon, (q_{12}, q)) \mid (q_{11}, \epsilon, q_{12}) \in \Delta_1 \wedge q \in Q_2\} \\ & \cup\{((q, q_{21}), \epsilon, (q, q_{22})) \mid q \in Q_1 \wedge (q_{21}, \epsilon, q_{22}) \in \Delta_2\}\end{aligned}$$

$$QA = \bigcup_{\substack{q_1 \in Q_1,\\ q_2 \in Q_2}} \begin{cases} ((q_1, q_2), e_1 \wedge e_2) & if (q_1, e_1) \in QA_1, \\ & (q_2, e_2) \in QA_2 \\ ((q_1, q_2), e_1) & if (q_1, e_1) \in QA_1, \\ & q_2 \in Q_2, \\ & \nexists e'.(q_2, e') \in QA_2 \\ ((q_1, q_2), e_2) & if (q_2, e_2) \in QA_2, \\ & q_1 \in Q_1, \\ & \nexists e'.(q_1, e') \in QA_1 \\ \emptyset & otherwise \end{cases}$$

To illustrate the intersection definition, the logistics and buyer abstracted workflows (as introduced above) are reused and the minimized intersection automaton is depicted in Figure 2(a). Based on the intersection automaton, it can be checked whether the accepted language is empty or not. Again emptiness test is based on standard automaton emptiness test, where the automaton is checked whether it contains a single path to a final state. Regarding aFSAs this emptiness test has to be extended by requiring that all transitions of a conjunction associated to a single state by an annotation are available in the automaton and a final state can be reached following each of these transitions, where the labels represent the variables in the logical term.

As a consequence, two automata are consistent, if their intersection is non-empty; i.e., there is at least one path from the start state to a final state, where each formula annotated

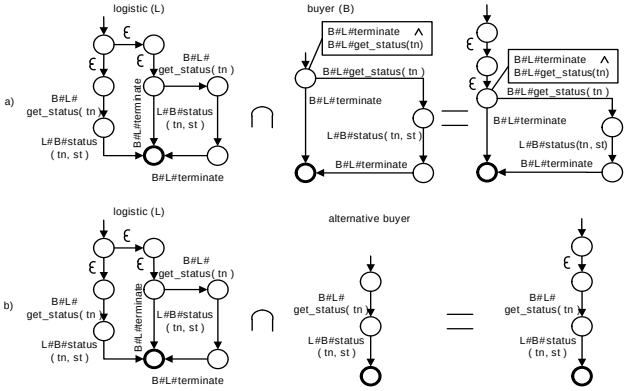

Figure 2. aFSA Intersection Examples: (a) Buyer and Logistic (b) Alternative Buyer and Logistics

to a state on this path evaluates to $true$. In particular, a variable becomes $true$, if there is a transition labeled equally to the variable from the current state to another state where the annotation evaluates to $true$. Finally the automaton is non-empty, if the annotation of the start state is $true$.

For the above example the intersection automaton depicted in Figure 2(a) is non-empty since there exist a path to a final state for both variables in the conjunction. Thus, the logistics and buyer abstracted workflows are consistent.

4 Multi-lateral Collaboration Consistency

A multi-lateral collaboration can be checked for consistency from a global point of view knowing all public workflows involved which are forming a cross-organizational workflow. However the aim is to do the checking in a decentralized way while providing the same decision. Thus, we start with introducing consistency of cross-organizational workflows and a decentralized consistency definition. In Section 5 it is shown that the two consistency definitions are equivalent.

4.1 Cross-organizational Consistency

Consistency of a cross-organizational workflow can be specified as the non-empty intersection of the public workflows forming the cross-organizational workflow. A message sequence accepted by an intersection automaton must be contained in each public workflow. However, a public workflow contains only messages, where the party providing the public workflow is either sender or recipient. As a consequence, the public workflows have to be extended by messages for which the local party is neither sender nor re-

cipient to get a non-empty intersection, where the additional messages might occur in arbitrary order.

The regular expression representing an arbitrary order of messages Σ_M contained in a cross-organizational workflow A_M without the set of messages Σ_k contained in the public workflow A_k is specified by $(\Sigma_M \backslash \Sigma_k)^*$, which corresponds to an automaton with a single start state being also a final state having one transition per message $\alpha \in \Sigma_M \setminus \Sigma_k$ from the start state to the start state. In the following, the regular expression notation $(\Sigma_M \setminus \Sigma_k)^*$ is used to specify the equivalent automaton.

To combine the additional messages with the public workflow, the automaton theoretic shuffle product operation is used. In particular, the shuffle product of two message sequences results in a set of message sequences, where the order of messages contained in the original two message sequences remains unchanged in all constructed message sequences, while the interleaving of the message sequences is in arbitrary order. The formal definition of the shuffle product is:

Definition 5 *(shuffle operation)*
The shuffle product $A := A_1 \& A_2$ *of* $A_i = (Q_i, \Sigma_i, \Delta_i, q_{i0}, F_i, QA_i)$ *(*$i \in \{1,2\}$*) is* $A = (Q, \Sigma, \Delta, q_0, F, QA)$ *with* $Q := Q_1 \times Q_2$, $\Sigma := \Sigma_1 \cup \Sigma_2$, $q_0 := q_{10} \times q_{20}$, $F := F_1 \times F_2$,

$$\begin{aligned}\Delta := \; & \{((p,q_1),\alpha,(p,q_2)) \in (Q_1 \times Q_2) \times \Sigma_2 \times (Q_1 \times Q_2) \mid \\ & (q_1,\alpha,q_2) \in \Delta_2\} \\ \cup \; & \{((p_1,q),\alpha,(p_2,q)) \in (Q_1 \times Q_2) \times \Sigma_1 \times (Q_1 \times Q_2) \mid \\ & (p_1,\alpha,p_2) \in \Delta_1\}\end{aligned}$$

$$QA = \bigcup_{\substack{q_1 \in Q_1, \\ q_2 \in Q_2}} \begin{cases} ((q_1,q_2), e_1 \wedge e_2) & if (q_1,e_1) \in QA_1 \wedge \\ & \quad (q_2,e_2) \in QA_2 \\ ((q_1,q_2), e_1) & if (q_1,e_1) \in QA_1 \wedge \\ & \quad \nexists e'.(q_2,e') \in QA_2 \\ ((q_1,q_2), e_2) & if (q_2,e_2) \in QA_2 \wedge \\ & \quad \nexists e'.(q_1,e') \in QA_1 \\ \emptyset & otherwise \end{cases}$$

Now, the workflow of the cross-organizational workflow can be defined as the intersection of the public workflows extended by all messages, where the local party is neither sender nor recipient.

Definition 6 *(cross-organizational workflow)*
Let $A_0, \ldots, A_{n-1}$ *be a set of aFSA representing public workflows respectively, then the workflow* A_M *of the cross-organizational workflow* M *is defined as* $A_M := \bigcap_{0 \le j < n} A_j \& (\Sigma_M \setminus \Sigma_j)^*$ *where* $\&$ *is the shuffle product,* $*$ *is the Kleene Operator known from regular expressions, and* $\Sigma_M := \bigcup_{0 \le j < n} \Sigma_j$ *with* Σ_j *being the alphabet of automaton* A_j *provided by party* p_j *and* Σ_j *is complete. Completeness means that all messages of the cross-organizational workflow are contained, which are sent or received by a party being involved in messages used by automaton* A_j*, that is,* $\Sigma_j = \{s\#r\#msg \in \Sigma_M \mid s = p_j \vee r = p_j\}$*. The cross-organizational workflow is consistent if it is non-empty, that is,* $L(A_M) \neq \emptyset$*.*

With regard to the example described in Section 2 based on the public workflows depicted in Figure 1 the minimized cross-organizational workflow as depicted in Figure 3 is non-empty, thus it is consistent.

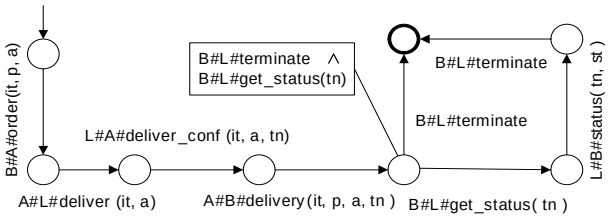

Figure 3. Minimized aFSA Representation of the Cross-organizational Workflow

According to the consistency definition of cross-organizational workflows, bilateral consistency can be defined quite similarly. In particular, bilateral consistency checking can be realized by extending a party's workflow and that of the trading partner, calculating the intersection, and checking the result for emptiness. The public workflows are bilaterally consistent, if the intersection automaton is non-empty. In the following the intersection automaton of two abstracted public workflows is called the corresponding bilateral workflow.

4.2 Decentralized Consistency Overview

In Service Oriented Architectures service composition does not guarantee that a single party knows all public workflows, thus, the consistency decision has to be derived in a decentralized way also. This decision can be derived for acyclic workflows by the following three steps [12]:

1. Propagation:
   Bilateral intersection can be used to modify the public workflows by removing irrelevant transitions. Thus, the relevant transitions are constraints for the trading partner therefore we call this step constraint propagation. This step has to be repeated until a fixpoint has been reached, that is, no further constraints can be propagated.

2. Decentralized Consistency Checking:
   Each party checks consistency of its bilateral and public workflows, i.e. does the emptiness test. If they are all consistent, then the party considers the cross-organizational workflow to be consistent until any

other party falsifies this decision by considering it to be inconsistent.

3. Consensus Making:
   A protocol is required to decentrally check whether all parties consider their public workflows to be consistent, and to inform all parties about the final consensus. One possible approach for this is to determine a leader election algorithm. The coordinator starts a minimal spanning tree algorithm setting up a hierarchical structure of the parties. Based on this structure a classical 2-Phase-Commit protocol can be used to collect intermediate results of the partners, deriving a decision and informing all parties on the result. This step is not further elaborated in this paper.

Step 1 is required because the bilateral workflow hides all constraints that are not immediately seen by the two parties involved. The bilateral workflow for buyer and logistics (Figure 2(a)) is consistent, because the intersection is non-empty. Further, the intersection of the logistics with an alternative buyer as depicted in Figure 2(b) is also consistent, however, the consistency is based on the usage of the logistics execution sequence requiring the *A#L#auth* message. Since this message is never send by the accounting, the bilateral workflow is consistent although the corresponding cross-organizational is inconsistent.

Therefore, the propagation of constraints is required to derive correct consistency decisions. In the following the propagation of constraints is formally introduced.

4.3 Propagation of Constraints

Propagation of constraints means that irrelevant transitions are removed from public workflows. In particular, propagation of constraints is based on the intersection of the extended public workflows of trading partners of a local party p and a final removal of all messages, which are unrelated to the party p's public workflow using the abstraction operation τ_p (see Definition 3). Thus, the propagation of constraints can be defined as follows:

Definition 7 *(constraint propagation)*
Let A_M be a cross-organizational workflow with respect to $A_0, \ldots, A_{n-1}$, where each A_i is an aFSA. Further, let party p having the public workflow A_k and $\{A_{i_p(0)}, \ldots, A_{i_p(m_p)}\} := \{A_j \mid 0 \leq j < n \wedge \Sigma_j \cap \Sigma_p \neq \emptyset\}$ be the set of party p's trading partner workflows. The propagated constraints of A_k result in an acyclic aFSA $A'_k := \Psi(A_k)$ with $A'_k := \tau_p\big(\bigcap_{0 \leq j < m_p} A_{i_p(j)} \& (\Sigma_{M_p} \setminus \Sigma_{i_p(j)})^*\big)$ where $\&$ is the shuffle product, $*$ is the Kleene Operator known from regular expressions, τ_p the abstraction τ_p, and $\Sigma_{M_p} := \bigcup_{0 \leq j < m_p} \Sigma_{i_p(j)}$ with $\Sigma_{i_p(j)}$ being the alphabet of automaton $A_{i_p(j)}$.

The iterative application of constraint propagation results finally in a fixed point defined as:

Definition 8 *(fixed point)*
Let A_M be a cross-organizational workflow A_M with respect to $A_0, \ldots, A_{n-1}$ public workflows, where each A_i is an aFSA. Then A_M is fixed point, if and only if
$\forall 0 \leq k < n . A_k = \Psi(A_k)$

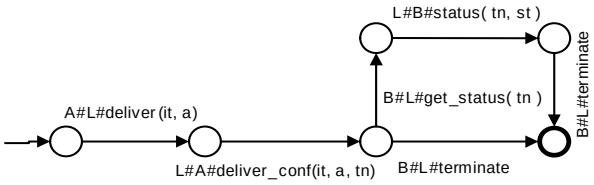

Figure 4. Minimized Logistics Department Propagated Occurrence Graph Constraints

With regard to the example, the logistics public workflow contains the transition labeled $A\#L\#auth()$ which is not supported by the accounting workflow. Thus, the message sequence starting with this transition is removed from the logistics workflow by applying constraint propagation. The resulting propagated logistics workflow is depicted in Figure 4. After this constraint propagation the fixed point has already been reached. This constraint propagation does not affect the remaining public workflows, thus they are already fixed point. Since the fixed point workflows of the three parties are locally consistent, the cross-organizational workflow is also consistent.

5 Correctness of the Approach

The correctness of the cross-organizational workflow consistency with the decentralized consistency decision is shown next. Based on the fact that the fixed points of the constraint propagation can always be reached and a reformulation of the decentralized consistency decision the correctness follows.

5.1 Constraint Propagation Convergence

The correctness proof is based on the fact that the public workflows are fixed point with regard to constraint propagation. Thus, it has to be shown that a fixed point can always been reached.

Lemma 1 *For all A_M being a cross-organizational workflow with respect to $A_0, \ldots, A_{n-1}$ public workflows, where each A_i is an aFSA with A_i being a fixed point, then all A_i always reach a fixed point $\forall 0 \leq k < n . A_k = \Psi(A_k)$*

Proof: Propagation of constraints is based on the intersection of extended automata of a public workflow and the trading partners corresponding public workflows. Since intersection of an automaton with other automata guarantees that the original one subsumes the intersection automaton, the propagation operation Ψ is monotonic, thus, the constraint propagation converges. □

5.2 Alternative Consistency Definition

Next, it has to be shown that the fixed point public workflows are equivalent to the corresponding party's projection of the cross-organizational workflow. To show this equivalence, the two implications forming the equivalence are separated by the following two lemmas. Based on this reformulation of the problem by means of locally available information the correctness of decentralized consistency can be shown.

Lemma 2 *Let A_M be a cross-organizational workflow with respect to $A_0, \ldots, A_{n-1}$ with A_j being the public workflow of party j and A_j being fixed point. Then for every $0 \leq k < n$ and for every message sequence ω accepted by A_k, the message sequence can be represented as the shuffle product of the projections $\tau_j(\omega)$ with $0 \leq j < n$ and $j \neq k$, that is, $\forall 0 \leq k < n. \forall \omega \in L(A_k). \omega \in L(\&_{0 \leq j < n, j \neq k} \tau_j(\omega))$*

Proof: Since ω contains messages, which are exchanged between A_k and another party A_j, the messages used by different j in $\tau_j(\omega)$ are disjoint. As a consequence, ω is in the language created by the combination of these messages, where a potential order per trading partner is already considered, that is, $\omega \in L(\&_{0 \leq j < n, j \neq k} \tau_j(\omega))$. □

While the above lemma is quite straightforward, the following has to show that each word which can be created by the shuffle product implies that the word is contained in the public workflow.

Lemma 3 *Let A_M be a cross-organizational workflow with respect to $A_0, \ldots, A_{n-1}$ with A_j being the public workflow of party j and A_j being fixed point. Then for every $0 \leq k < n$ and for every message sequence ω accepted by $A'_k := \tau_k(A_M)$, all message sequences ω', which can be constructed by the shuffle product of the projections $\tau_j(\omega)$ with $0 \leq j < n$ and $j \neq k$ are contained in $L(A'_k)$. That is,*

$$\forall 0 \leq k < n. \quad \forall \omega \in L(A'_k). \\ \forall \omega' \in L\Big(\&_{0 \leq j < n, j \neq k} \tau_j(\omega)\Big). \omega' \in L(A'_k)$$

Proof: Be aware that A'_k is equivalent to A_k due to the fixed point definition (see Definition 8). Further, due to the definition of A'_k being the projection τ_k of the cross-organizational workflow (see Definition 6), that is,

$$A'_k := \tau_k\Big(\bigcap_{0 \leq j < n} A_j \& (\Sigma_M \setminus \Sigma_j)^*\Big) \tag{1}$$

and the requirement of $\omega \in L(A'_k)$ every public workflow A_j with $0 \leq j < n$ accepts a message sequence ω_j, which shares the same order of messages exchanged between A_k and A_j, that is,

$$\forall 0 \leq j < n. \exists \omega_j \in L(A_j). \tau_k(\omega_j) = \tau_j(\omega) \tag{2}$$

If this condition is not fulfilled, then the intersection in equation 1 would be empty and, thus, ω would not be contained in $L(A'_k)$ as stated in the requirement. As a consequence, such a message sequence ω_j exists for every public workflow A_j in the cross-organizational workflow.

Each message used in ω has a sender and a recipient, where party k is either but not both of them. As a consequence, the set of messages used in message sequences $\tau_k(\omega_j)$ and $\tau_k(\omega_i)$ are disjoint, where $0 \leq j < i < n, i \neq k, j \neq k$. Based on the construction of ω' (see equation 3) using the shuffle product of the disjoint message sequences $\tau_j(\omega)$

$$\omega' \in L\Big(\&_{0 \leq j < n, j \neq k} \tau_j(\omega)\Big) \tag{3}$$

and the fact that $\tau_j(\omega)$ equals $\tau_k(\omega_j)$ by equation 2 it follows that

$$\omega' \in L\Big(\&_{0 \leq j < n, j \neq k} \tau_k(\omega_j)\Big) \tag{4}$$

ω_j is contained in $L(A'_j)$, which is constructed in the same way as A'_k, and which is equal to $L(A_j)$ due to the fixed point assumption. Thus $L(A_j)$ is recursively given via the intersection calculation. Since $\omega_j \in L(A_j)$ the intersection of A_j with the remaining parties as specified in equation 1 accept the message sequence ω_j due to the fixed point assumption, otherwise ω_j would not be contained in $L(A_j)$. As a consequence, the intersection of the extended ω_j cannot be empty, thus,

$$\omega' := \tau_k\Big(\bigcap_{0 \leq j < n, j \neq k} \omega_j \& (\Sigma_M \setminus \Sigma_j)^*\Big) \tag{5}$$

which by equation 1 means that $\omega' \in L(A'_k)$, while ω' has been constructed by the shuffle product (see equation 3). □

Based on the previous lemmas, the initial aim of this section can be formally stated as a theorem:

Theorem 1 *Let A_M be a cross-organizational workflow with respect to $A_0, \ldots, A_{n-1}$ with A_j being the public workflow of party j and A_j being fixed point with $A_j = \tau_j(A_M)$. Then the public workflow resulting from the projection of the cross-organizational workflow is equivalent to the one constructed by the intersection of the public workflow with the shuffle product of the local party's projection of the remaining public workflows, that is,*
$\tau_k\Big(\bigcap_{0 \leq j < n} A_j \& (\Sigma_M \setminus \Sigma_j)^*\Big) \equiv A_k \cap \&_{0 \leq j < n, j \neq k} \tau_k(A_j)$

Proof: Two automata are equivalent if the corresponding languages are equivalent. Thus, every message sequence is

accepted by the right hand side if and only if the message sequence is also accepted by the left hand side.

The implication from left to right can easily be shown, since $\omega \in L(A_k)$ with using Lemma 2 and according to the argumentation in the proof of Lemma 3 (see equation 2 $\tau_k(A_j) = \tau_j(A_k)$) it follows that $\omega \in L\Big(\&_{0\leq j<n, j\neq k}\tau_k(A_j)\Big)$. As a consequence of $\omega \in L(A_k)$, ω is also contained in the language resulting from $A_k \cap \&_{0\leq j<n, j\neq k}\tau_k(A_j)$.

The implication from right to left can be shown using Lemma 3. Since ω is contained in the intersection language of A_k and $\&_{0\leq j<n, j\neq k}\tau_k(A_j)$, ω is contained in either language. As a consequence, it has been shown that $\omega \in L(A_k)$ (see Lemma 3) implies that $\omega \in L\big(\tau_k(\bigcap_{0\leq j<n} A_j \& (\Sigma_M \setminus \Sigma_j)^*)\big)$ □

The current definition of the fixed point calculation requires a public workflow A_p to recognize all messages of the trading partners to be able to extend its public workflow before doing the intersection calculation (see also Definition 7). This seems inappropriate since the decentralization requires to stick to local knowledge. As a consequence, an equivalent representation of this propagation rule is introduced.

Lemma 4 *Let A_M be a cross-organizational workflow with respect to $A_0, \ldots, A_{n-1}$ with A_j being the public workflow of party j and A_j being fixed point with $A_j = \tau_j(A_M)$. The trading partner's workflows of a party k are the subset of all public workflows, where the corresponding alphabets have at least a single message in common, that is, $\{A_{i_k(0)}, \ldots, A_{i_k(m_k)}\} := \{A_j \mid 0 \leq j < n \wedge j \neq k \wedge \Sigma_j \cap \Sigma_k \neq \emptyset\}$. The following equivalence holds:* $\&_{0\leq j<n, j\neq k}\tau_k(A_j) \equiv \&_{0\leq l<m_k, l\neq k}\tau_k(A_{i_k(l)})$

Proof: Due to Lemma 3 all public workflows A_j are non-empty in case A_M is non-empty. Based on $\&_{0\leq j<n, j\neq k}\tau_k(A_j)$ two cases have to be distinguished: In the first case, A_j represents a public workflow of a trading partner, that is, $\Sigma_j \cap \Sigma_k \neq \emptyset$ which maps to a workflow contained in the subset $\{A_{i_k(0)}, \ldots, A_{i_k(m_k)}\}$ with $j = i_k(l)$. Thus, the automaton is represented at either side of the equivalence.

In the second case, A_j represents a public workflow of a party being not a trading partner, that is, $\Sigma_j \cap \Sigma_k = \emptyset$. Thus, A_j is not contained in the subset $\{A_{i_k(0)}, \ldots, A_{i_k(m_k)}\}$. Therefore, the workflow A_j appears only on the left hand side of the equivalence. However, due to $\Sigma_j \cap \Sigma_k = \emptyset$ the abstraction $\tau_k(A_j)$ of the public workflow A_j results in an empty message sequence. Thus, the projection does not contribute any message to the construction of the workflow by the shuffle product and therefore can be neglected. □

Based on this lemma and Theorem 1 it follows that the fixed point public workflows A_j are equivalent to $\tau_j(A_M)$.

5.3 Decentralized Consistency

Finally, the decentralization aspect of deciding consistency is addressed. In Definition 6 consistency of a cross-organizational workflow has been defined as the non-empty intersection of the public workflows extended by all messages that the party is not directly involved in. Due to this definition, it has to be shown that emptiness of fixed point public workflows is equivalent to emptiness/inconsistency of the cross-organizational workflow.

Theorem 2 *Let A_M be a cross-organizational workflow with respect to the public workflows $A_0, \ldots, A_{n-1}$ with A_j being the public workflow of party j and A_j being fixed point with $A_j = \tau_j(A_M)$. A_M is consistent, if and only if all public workflows A_j with $0 \leq j < n$ are non-empty.*

Proof: Both directions of the equivalence have to be shown:

$\exists 0 \leq j < n. L(A_j) = \emptyset \longrightarrow L(A_M) = \emptyset$: Since the cross-organizational workflow is defined as the intersection of the extended public workflows the intersection with an empty public workflow results in an empty intersection language $L(A_M)$. Thus, the cross-organizational workflow A_M is inconsistent.

$\forall 0 \leq j < n. L(A_j) \neq \emptyset \longrightarrow L(A_M) \neq \emptyset$: Based on Lemma 4, Theorem 1 and the fixed point of A_j, each A_j is equivalent to the projection $\tau_j(A_M)$ of the cross-organizational workflow. Since all workflows are non-empty, the cross-organizational workflow A_M is also non-empty. □

Thus, it has been shown that based on the proposed propagation the centralized and the decentralized consistency definitions are equivalent.

6 Parameter Constraints

In this paper we do not discuss the effect of constraints on message parameters, since these parameters are less applicable due to their potential of revealing mission critical information. However, there exist an approach described in [10] handling also parameter constraints which can not be presented in this paper due to the lack of space. The underlying idea is to represent parameter constraints similar to the introduction of guard functions in Colored Petri Nets. Thus, the aFSA definition is extended by guard functions to represent parameter constraints. The guard function is used to introduce additional constraints on the enabling of transitions, thus, guard functions are annotated to transitions rather than to states as done by the annotations representing mandatory and optional transitions. Based on guarded aFSA the introduced operations and the construction of cross-organizational workflows can be extended coming to the same conclusion.

Further, based on using parameter constraint propagation the preconditions for the presented approach can be weakened. Currently a precondition is specified that a public workflow has to be an abstraction of the cross-organizational workflow. This condition is needed to cover very specific situations as e.g. discussed in [11] where otherwise circular dependencies may cause wrong decisions. By introducing history constraints, that is, history information on reaching a certain state in the public workflow this precondition can be eliminated. The used mechanism is the one of parameter constraint propagation.

7 Related Work

Checking consistency of a cross-organizational workflow can usually be defined based an the set of potential execution sequences, a straight forward approach is to check consistency on a centralized cross-organizational workflow model, which has to be split afterwards into several public ones. This approach has been applied to several workflow models, like for example by v.d.Aalst and Weske [2] to Workflow Nets (WF-Nets), by Fu et.al. to guarded Finite State Automata [4], by Yi and Kochut [15] to Colored Place/Transition Nets, or by Wodtke and Weikum [8] to statecharts. However, this represents the top-down approach based on a centralized consistency checking, which is different to what is addressed in this paper.

The bottom-up approach of constructing the cross-organizational workflow based on several public workflows has been investigated to a lesser extend. Approaches have been proposed e.g. in [1, 4]. However, these approaches require a centralized decision making and are not constructive, that is, they only specify criteria for various notions of consistency but do not provide an approach to adapt public workflows to make the cross-organizational workflow consistent. In addition, neither of the approaches addresses the synchronous communication model.

8 Conclusion and Future Work

In this paper a consistency definition of cross-organizational workflows and its decentralized version are defined. Further, the correctness/equivalence between the two definitions is shown. The paper is motivated based on service composition in Service Oriented Architectures. In particular, the basic operations of the presented approach have been implemented [13] and have been applied in a service discovery implementation [14] based on bilateral consistency. Finally, a protocol has been proposed to determine a potential service composition in a decentralized way resulting in a set of services as a basis for consistency checking [9]. As a consequence, the applicability of the presented approach and main parts of the implementation has been shown already. Future work focus on indexing of the bilateral consistency checking.

References

[1] W. Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.

[2] W. Aalst and M. Weske. The P2P approach to interorganizational workflows. In *Proc. of 13. Int'l. Conference on Advanced Information Systems Engeneering (CAISE)*, 2001.

[3] J. Chomicki and G. Saake, editors. *Logics for Database and Information Systems*. Kluwer, 1998.

[4] X. Fu, T. Bultan, and J. Su. Realizability of conversation protocols with message contents. In *Proc. IEEE Int'l. Conf. on Web Services (ICWS)*, pages 96–103. IEEE Computer Society, 2004.

[5] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[6] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.

[7] E. Kindler, A. Martens, and W. Reisig. Interoperability of workflow applications: Local criteria for global soundness. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 235–253, 2000.

[8] D. Wodtke and G. Weikum. A formal foundation for distributed workflow execution based on state charts. In F. N. Afrati and P. Kolaitis, editors, *Proc. Int'l. Conf. on Database Theory (ICDT)*, pages 230–246. LNCS 1186, 1997.

[9] A. Wombacher. Decentralized decision making protocol for service composition. In *Proc. IEEE Int'l. Conf. on Web Services (ICWS)*, pages 203–210, 2005.

[10] A. Wombacher. *Decentralized establishment of consistent, multi-lateral collaborations*. PhD thesis, Technical University Darmstadt, 2005.

[11] A. Wombacher. Issues on decentralized consistency checking of multi-lateral collaborations. In *Int'l. WS on Distributed and Mobile Collaboration (DMC)*, pages 93–98, 2005.

[12] A. Wombacher, P. Fankhauser, and K. Aberer. Overview on decentralized establishment of consistent multi-lateral collaborations based on asynchronous communication. In *Proc. IEEE Int'l. Conf. on e-Technology, e-Commerce and e-Service (EEE)*, pages 164–170, 2005.

[13] A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold:. Matchmaking for business processes based on choreographies. *Intl. Journal of Web Services*, 1(4):14–32, 2004.

[14] A. Wombacher, B. Mahleko, and E. Neuhold. IPSI-PF: A business process matchmaking engine. In *Proc. of Conf. on Electronic Commerce (CEC)*, pages 137–145, 2004.

[15] X. Yi and K. J. Kochut. Process composition of web services with complex conversation protocols: a colored petri nets based approach. In *Proc. of the Design, Analysis, and Simulation of Distributed Systems Symposium at Adavanced Simulation Technology Conf.*, pages 141–148, 2004.