Issues on Decentralized Consistency Checking of Multi-lateral Collaborations

Andreas Wombacher*
University of Twente, Enschede, The Netherlands
a.wombacher@utwente.nl

Abstract

Decentralized consistency checking of multi-lateral collaborations is based on propagating local information to trading partners until a fixed point has been reached. However, which information has to be propagated and how to represent this information is a challenge, because the local consistency decisions should derive consistency only if the collaboration is consistent. In this paper two scenarios are discussed arguing that history information about reaching a state must be propagated and that messages must be named uniquely in this history information to achieve the aimed equivalence of local consistency and collaboration consistency.

1. Introduction

Web Services are one technology supporting Service Oriented Architectures (SOAs). One major aspect of SOAs is its loosely coupling of services, which is supported by a service discovery infrastructure. Web Services can handle this loosely coupling quite well as long as the used services are stateless. In case the services are state dependent then this loosely coupling can be understood as a consistency problem of cross-organizational workflows representing multi-lateral collaborations or collaborations for short.

For this problem centralized solutions exist. However, the SOA does not fit to such centralized solutions, but is more appropriate with decentralized approaches. Thus, decentralized collaborations require also a decentralized consistency checking. However, decentralized approaches are less investigated.

Decentralized consistency checking must rely on local information that is the own local workflow and the relevant parts of the trading partner’s public workflows. Thus, consistency of the collaboration has to be decided on bilateral consistency checking and therefore requires propagation of information to support a sufficient consistency definition.

In [12] an overview of an approach is described which addresses the issues presented in [11], that is, the effect of parameter constraints and irrelevant parts of local workflows on the decentralized consistency checking. In particular, these issues have been discussed and a solution based on constraint and occurrence graph propagation has been proposed. In this paper additional problems and their solutions are presented. In particular, the effect of two uncoordinated sequential transitions each representing receiving messages and the usage of repeating message names in a single path are investigated and a solution using history information is presented.

The paper starts by introducing a bilateral consistency notion, which is afterwards extended to a multi-lateral consistency illustrating decentralized consistency checking of collaborations and investigating two special cases. Afterwards related work is discussed and conclusions and future work are presented.

2. Bilateral Consistency

As stated above, decentralized consistency checking has to be based on bilateral decisions, thus, a bilateral consistency definition is introduced intuitively followed by a formal model used throughout this paper.

2.1. Intuitive Explanation

Workflows can be represented in various ways, like e.g. Petri Net or Finite State Automata [4]. In the following an automaton notation is used due to the model’s simplicity. An automaton is represented by states (circles) representing business tasks and transitions connecting states representing a message exchange, where transition labels are of the form $s\#r\#msg$ representing sender $s$ and receiver $r$ of the message as well as its message name $msg$. The start state is indicated by a state with a small incoming arc and final states are represented by states with solid lines. A final state means that the workflow execution can correctly terminate in this state.

Figure 1 depicts two example workflows representing party A’s and B’s workflow respectively. The two workflows are inconsistent, because after message $B\#A\#msg0$
Figure 1. Automaton Representation of Example Local Workflows

has been sent by party B, B can choose whether to continue with sending message $B\#A\#msg_1$ or $B\#A\#msg_2$. However, party A can only handle message $B\#A\#msg_2$. Since party A has no influence on party B’s decision, the collaboration fails in case party B selects message $B\#A\#msg_1$. Be aware that the choice supported by party A in receiving either message $B\#A\#msg_0$ or $B\#A\#msg_1$ at the first step does not influence the consistency, because a party may be able to handle additional messages which are never send by the trading partner, since this does not cause the collaboration to fail, that is, making it inconsistent.

As a consequence of this example all messages at a certain state being sent by a party itself must be supported by a trading partner, therefore can be considered to be mandatory for the trading partner, while messages being received by the party are considered to be optional for the trading partner. The standard Finite State Automata model does not support this differentiation. Therefore, in the following a formal is introduced providing the required expressiveness.

2.2. Formal Model

Different models have been proposed supporting mandatory and optional messages, which can be classified in accordance to the underlying communication model: asynchronous communication is e.g. supported by v.d.Aalst [1] and Martens et.al. [5], while synchronous communication is supported by Wombacher et.al. [13]. Because Web Services use often synchronous communication based on the HTTP protocol, in the following the annotated Finite State Automata model [13] has been selected and is introduced next.

Definition 1 (annotated Finite State Automata (aFSA))
An annotated Finite State Automaton $A$ is represented as a tuple $A = (Q, \Sigma, \Delta, q_0, F, QA)$ where $Q$ is a finite set of states, $\Sigma$ is a finite set of messages, $\Delta : Q \times \Sigma \times Q$ 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.

The terms in $E$ are standard Boolean formulas:

Definition 2 (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, (iii) if $\phi$ is a formula, so is $\neg \phi$, (iv) if $\phi$ and $\psi$ are formulas, so is $\phi \land \psi$ and $\phi \lor \psi$.

The graphical representation of an annotated Finite State Automaton (aFSA) is based on the one of automata as introduced in the previous section, extended by annotations of states, which are given within a square connected to the corresponding state.

Figure 2. aFSA Representation of Example Local Workflows

Figure 2 contains the aFSA representation of the example used in the previous section (see Figure 1). As described above, the mandatory messages of a choice are represented by a conjunction, while the optional messages of a choice are not explicitly annotated meaning a disjunction of the messages. A formal definition of generating an aFSA model from a workflow modeling language can be found in [14] where a mapping of a subset of the Business Process Execution Language for Web Services (BPEL) to aFSA is provided.

Based on the aFSA definition, intersection and emptiness operations have been defined in [13], which are quite similar to the ones of standard automata. In particular, intersection is based on the usual cross product construction of automata intersection, where the annotations of states are combined by conjunction. The emptiness test representing the reachability of a final state from the start state is extended by requiring all transitions contained in a conjunction to support a path to a final state. As a consequence, two automata are consistent, if the intersection of the automata is non-empty, that is, if there exists at least one path from the start state to a final state, where each formula annotated to a state being contained in this path evaluates to $true$. A variable evaluates to $true$, if the automaton using that state as the start state is non-empty.

With regard to the above example, the intersection automaton of party A and B as depicted in Figure 3 is empty,
Figure 3. aFSA Representation of Intersection Automaton of Local Workflows from Figure 2

because the automaton does not contain the mandatory transition labeled $B\#A\#msg1$. In particular, the variable $B\#A\#msg2$ of the annotation evaluates to true, because there is a path to a final state, but the variable $B\#A\#msg1$ of the annotation is evaluated to false because there is no such transition available at that state providing a path to a final state.

3. Multi-lateral Consistency

Multi-lateral consistency can be checked in a centralized way similar to bilateral consistency checking. In particular, each local workflow has to be extended by all messages he is not directly involved in. Based on this extension, the collaboration is consistent, if the intersection of the extended local workflows is non-empty.

Figure 4. aFSA Representation of Local Workflows

To illustrate this, the example depicted in Figure 4 is used containing three parties $A$, $B$ and $C$, where the messages exchanged between party $A$ and party $B$ are $A\#B\#msg1$ and $A\#B\#msg3$, while the message exchanged between party $B$ and party $C$ is $B\#C\#msg2$. Party $A$ and party $C$ are not directly exchanging messages. The extended versions of the local workflows are depicted in Figure 5. Party A’s workflow is extended by message $B\#C\#msg2$, party B’s workflow does not require any extension, while party C’s workflow is extended by messages $A\#B\#msg1$ and $A\#B\#msg3$. The resulting intersection automaton equals the automaton of part B, which is non-empty, thus, indicating a consistent workflow.

Figure 5. aFSA Representation of Extended Local Workflows

3.1. Issue of Sequentially Received Messages

The above example is considered to be consistent due to the non-empty intersection automaton. However, this decision is wrong for synchronous communication model. The intersection automaton, which equals party B’s workflow in Figure 5 expects message $B\#C\#msg2$ to be sent by party B before message $A\#B\#msg3$ is received by party B. Due to party A being independent of party B sending message $A\#B\#msg3$, party A may send the message $A\#B\#msg3$ before party B has sent message $B\#C\#msg2$. Thus, if party A sends its message before party B does the execution of the collaboration fails. In particular, such unsynchronized dependencies of message exchanges make a collaboration inconsistent, thus, must result in an empty intersection automaton.

To take this observation into account, a notion of history is introduced, which represents the set of messages already exchanged when reaching a state. Thus, a state can only be reached by a transition if the sending party history subsumes the history of the receiving party. Based on this additional constraint the above observed issue can be resolved. The required extension of the aFSA model is the support of guard expressions on transitions comparable to the ones used in coloured Petri Nets. The used notion of subsumption is a classical subset relation. The constraints are represented in the graphical representation in curled brackets below the transition label. The constraint represents the label of the transition itself as well as the messages which have been used to get to that transition. With regard to the above example (see Figure 4), the adapted local workflows are depicted in Figure 6.

Applying this additional constraint to the different messages the following subsumptions can be observed, where the message is indicated first, followed by the receiving and
sending histories:

\[ A#B#msg1 : \{ A#B#msg1 \} \subseteq \{ A#B#msg1 \} \]
\[ B#C#msg2 : \{ B#C#msg2 \} \subseteq \{ A#B#msg1, B#C#msg2 \} \]
\[ A#B#msg3 : \{ A#B#msg1, B#C#msg2, A#B#msg3 \} \not\subseteq \{ A#B#msg1, A#B#msg3 \} \]

Therefore, the transition labeled \( A#B#msg3 \) is omitted in the intersection automaton making the automaton empty indicating that the collaboration is inconsistent.

This paper contains the basic ideas to overcome this issue, while the formal specification of the solution will require description logic with concrete domains [6] with history being one domain. The extension to description logic is necessary because there might be several path resulting in the same state, thus, there are several independent histories, which have to be represented and being considered independently. Further, the description logic with the concrete domain representing histories will be used, because this logic is restrictive enough and provides polynomial computational complexity results for subsumption calculation [6]. However, due to a limitation of space, the formal extension of the model can not be provided in this paper.

### 3.2. Decentralization

As stated in detail in [12], the above mentioned consistency definition can be decentralized, that is, the decision on the collaboration consistency can be derived based on local decisions for acyclic workflow models in the following way: the constraints on history information and occurrence graphs have to be propagated through all local workflows. In particular, that means that

- histories assigned to a sending transition have to be combined with the history of receiving transitions by a simple set union in case the sending transition’s history subsumes the receiving one, and
- transitions contained in a local workflow specification, which will never be used during the execution of the collaboration, that is, which are not contained in the occurrence graph, are discarded.

This propagation of constraints changes local workflow specifications and is performed until a fixed point has been reached resulting in local workflows being relevant for the collaboration. Based on this fixed point workflow models the local decisions on the consistency are derived. The local decisions can be calculated based on the bilateral consistency introduced in Section 3.1, where the intersection of the local workflow and the combination (so called shuffle product) of the relevant parts of the trading partners workflows is calculated and checked for emptiness.

The local decision making is illustrated on behalf of party A based on the local workflows depicted in Figure 6. Party A’s workflow remains unchanged, while those messages in party B’s and C’s workflow, which are not related to an interaction with party A are relabeled by an empty transition ε in accordance to standard automaton theory. The resulting relabeled workflows of part B and C and the combination of those are depicted in Figure 7. The intersection of party A’s workflow and the combination of the relabeled workflows of party B and C equals the workflow of party A without the transition labeled \( A#B#msg3 \). Thus, the intersection automaton is empty indicating inconsistency.

Finally, a consensus making is initiated, whether all parties consider a collaboration to be consistent or not. The consensus making problem is well known from distributed systems.

However, the decentralization and the propagation of history constraints causes a new issue due to repeating message names along a single path.
3.3. Repeating Message Names

The following example illustrates this issue by providing a collaboration, which is considered to be consistent, although it is not. The reason for the wrong consistency decision is that history constraints disregard order and repeating message occurrences within a single path. To illustrate the effect the example depicted in Figure 8 is discussed.

Figure 8. aFSA Representation of Local Workflows with History Annotation

Three parties A, B, and C are involved in a collaboration. The workflow starts with B sending message B#C#msg2 to party C. Afterwards, party B has to decide to continue the process either by sending message B#C#msg2 to party C again, or sending message B#A#msg1 to party A. However, party A finally has to send message A#C#msg3 to party C. The history constraints depicted are the ones after history constraint propagation as mentioned in the previous section. This can be observed at the history constraints of party A containing message B#C#msg2, which has been propagated by party B through sending message B#A#msg1. Within party A’s workflow this history constraint is also propagated. Making local decisions as discussed in the previous section the resulting intersection automata are equal to the ones depicted in Figure 8.

Since each local workflow is consistent, the collaboration is considered to be consistent, although it is not. The wrong decision making is caused by the double usage of message B#C#msg2. In particular, the history representation can not distinguish between the first and second occurrence of message B#C#msg2, thus, it seems to suffice if the message has been exchanged at least once. However, the number a message has been exchanged has to be equal in every path. As a consequence, different occurrences of equally labeled messages are annotated by a subscribed number of occurrence of this message in a path to reach the message. Applying parameter constraint propagation again results in a fixed point and applying the first occurrence graph constraint propagation results in the automata depicted in Figure 9.

Figure 9. Fixed Point Local Workflows With Distinguishing Messages

The last transition in party A’s workflow is omitted, because the subsumption

\[ \{ B#A#msg1, B#C#msg2 \}, \]

\[ B#C#msg2, A#C#msg3 \} \]

is not fulfilled. Similar for the last transition in party C’s workflow, where

\[ \{ B#C#msg2, B#C#msg2 \}, \]

\[ B#C#msg2, A#C#msg3 \} \]

is not fulfilled.

As a consequence, the aFSA of party A and C are empty, thus, indicating the multi-lateral collaboration to be inconsistent. Extending the originally sketched decentralized consistency checking approach for symmetric communication models by history constraints and considering the unique naming of message occurrences in history constraints the approach improves further. In particular, I am not aware of any further case which can not be handled by the presented approach.

4. Related Work

Multi-lateral collaborations have been investigated in the fields of workflow management theory, logic based approaches, and coordination theory. All these approaches have in common that they are limited to centralized multi-lateral collaboration establishment.

The workflow community has addressed the issue of direct coordination between workflow engines [3] rather than implementing coordination based on bilateral communication as presented in this paper. Corresponding approaches require a centralized coordination checking the consistent
execution of distributed workflow engines, thus do not support a decentralized decision and execution of multi-lateral collaborations.

Dynamic deontic logic [7] is an approach, where a transition named action represents a change from one propositional world to another. In addition, deontic operations express permission, prohibition and obligation of actions, that is corresponding transitions. Another logic based approach is Courteous Logic Programs [2] being a non-monotonic logic, that allows to change predicate truth assignments. Both approaches have been used to represent workflows, but have never been applied to decentralized collaboration establishment. In particular, the propagation as introduced in this paper can be represented by these logics, if all transitions in all workflows are deterministic. Non-determinism can not be handled due to insufficient support of backtracking.

Coordination theory has also been applied to workflow coordination [8] being data-driven rather than control-driven. An example of such an approach are place transition nets modeling state changes by events. The WorkSpace [9] approach is based on the notion of steps representing a transformation of one or several data elements. These approaches are all centralized, thus, can not be applied to decentralized decision making for multi-lateral collaborations establishment.

5. Conclusion and Future Work

In this paper two issues on decentralized consistency checking of multi-lateral collaborations using synchronous communication model have been discussed: the first one is the need to introduce history information of states to detect uncoordinated sequential receiving of messages at a party, while the second one is the need to differentiate repeated occurrences of message names by uniquely naming them because otherwise the history information of states introduced above can not differentiate between one and several occurrences. However these capabilities are important to decentrally derive appropriate consistency decisions as illustrated on behalf of examples.

The approach presented here has been implemented and parts of this work have already been published, like the bilateral consistency checking [15], the aFSA operations [13], or a collaboration establishment protocol [10]. Future work will formalize the complete approach and provide a complete implementation of it. Further, besides decentralized consistency checking the process of decentralized collaboration establishment is quite challenging. Besides the protocol presented in [10] strategies are required to effectively create new collaborations. Here innovative research is required.

References