Soundness and Equivalence of Petri Nets and annotated Finite State Automate: A Comparison in the SOA context

Andreas Wombacher[*] and Axel Martens[†]
[*]School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland, Email: andreas.wombacher@epfl.ch
[†]Component Systems Group, IBM TJ Watson Research Center, USA, Email: amarten@us.ibm.com

***Abstract*— A lot of work exists on notions of equivalence and soundness relations of different workflow models, which are used in different domains like e.g. Business Process Modeling, Software and Service Engineering. These definitions are based on different models, including Petri Nets and different versions of Finite State Automata, having different expressiveness and computational complexity classes. The aim of this paper is to compare the equivalence and soundness relations of the Petri Net and annotated Finite State Automaton models in the context of Service Oriented Architectures. It turns out that up to a certain expressiveness the relations are comparable and computable with reasonable effort. For these cases also mappings between the different models and relations are presented.**

I. Introduction

Service Oriented Architectures are based on a decentralized development, deployment, and maintenance of services. To find, compose, and engineer new services the support of finding existing services is a crucial element of the infrastructure. A lot of work exists proposing different ways of describing services based on the e.g. used message structures, behavior, semantics, quality of service parameters, or just proposing a classification of services. In this paper the focus is on the behavior aspects of service discovery.

However, even if the area is narrowed down to behavior based service discovery, there still exist a lot of different approaches. Some of them are comparing services based on a notion of equivalence, while others propose a guarantee of successful execution, i.e., soundness or consistency. Further, the different approaches are using different models, with different expressiveness and as a consequence different computational complexity. Finally, for the soundness case the different models make different assumptions on the underlying communication model mostly without stating them explicitly. Thus, comparing the different approaches is hard.

In this paper a comparison of two often cited approaches is made: an approach proposed by v.d.Aalst et.al.[1] based on Workflow Nets (WF-nets) and another approach proposed by Wombacher et.al.[2] based on annotated Finite State Automata. Although the proposed approaches aim at different communication models for the soundness comparison, they provide the expressiveness to represent also the other communication model. In particular, it is shown that the equivalence notion as well as the soundness notion in both models describe the same semantics independent of the communication model.

In the following the two workflow models are briefly introduced in Section II. Next the equivalence operation is investigated in Section III. Section IV discusses the soundness operation for synchronous and asynchronous communication models. Finally, the results are summarized and an outlook is presented in Section V.

II. Workflow Models

A. *Workflow Nets*

Petri nets are often used to model workflows based on an asynchronous communication model. A Petri net consists of places (circles) representing business tasks, transitions (rectangles) representing a message exchange, and arcs connecting places and transitions. Transitions are labeled with message names. Petri nets have an execution semantics using tokens stored in places, which are depicted as black dots. The distribution of tokens over all places is called a marking and represents the state of a Petri net. Tokens can be removed and added to places when a transition is fired, i.e., executed. In particular, tokens are removed from places connected by an incoming arc of a transition and are add to places connected by an outgoing arc of a transition. The reachable states of a Petri net can be denoted in an occurrence graph, where each vertex of the graph represents a marking, i.e., a state of the Petri net, and each labeled edge represents the name of the transition fired to get from one state to another. Examples of occurrence graphs are depicted in Figure 1(b) and (d), where the numbers in curly brackets mean that a token is contained in the place with the corresponding number.

Because of their generic structure, additional constraints have often be added to Petri nets to meet the requirements of a given domain. Hence, Workflow Nets (WF-Net) [3], [1] have been introduced to describe the behavior of a workflow or service. WF-Nets require a single initial and final place, and all net objects have to lie on a path between those places. Moreover, WF-Nets have been extended to Interorganizational Workflow Nets (IOWF-Nets). The idea is to relate transitions of different WF-Nets with each other by introducing communication channels and later on translate the IOWF-Net again into a WF-Net by translating communication channels into WF-Net constructs. In particular, in [1] communication channels representing synchronous communication are translated by merging the connected transitions of the two WF-Nets, while in asynchronous communication the communication channel is instantiated by a place keeping the directed connections to the sending and receiving transitions of the two WF-Nets. Open Workflow Nets (oWFN) [4] loosen the constraints on initial and final places while preserving the soundness notion of WF-Nets, and therefore seem the most appropriate way of modeling services and service compositions with Petri nets.

B. annotated Finite State Automata

Annotated Finite State Automata are an extension of Finite State Automata. A Finite State Automaton (FSA) consists of an alphabet, a set of states, an initial state, a set of final states, and a set of transitions representing a state change from a source state to a target state labeled with an element of the alphabet [5]. A language constitutes a set of words representing sequences of transitions from the start to a final state. The extension proposed in [2] to apply FSAs to service discovery is called annotated Finite State Automata (aFSA) and distinguishes optional and mandatory transitions, i.e., transitions that could be supported by a communication partner and transitions that must be supported by a communication partner. The differentiation is not possible in standard FSA therefore annotations of states are introduced where mandatory transitions are represented as a conjunction and optional transitions as a disjunction. In particular, if a protocol describes a set of alternative messages that are to be transmitted (e.g. acceptance vs. rejection), the associated transitions for the sending party of those messages are considered to be mandatory (i.e. to be supported by the receiving party), whether associated transitions of the receiving party are considered to be optional. In other words, each service has to fully respect an external decision of its partner while it is legal to restrict its own internal decisions. The annotations are maintained by all FSA operations and are evaluated during the emptiness test, i.e., an operation determining whether an aFSA represents a language containing no words at all. Figure 2 shows examples of aFSAs.

The application of aFSAs to service discovery is mainly corresponding to soundness in oWFNs which makes use of nonempty intersection of two aFSAs. Intersection is an operation constructing a new aFSA representing the common transition sequences of two aFSAs. Intersection is defined as the usual cross-product construction of states and transitions, while the annotations of two states are combined by conjunction and associated to the cross-product of the two states. The emptiness test evaluates the annotations to determine a truth value associated to the state the annotation is associated with. In particular, a message in an annotation is associated with the truth value of the state reachable by the transition with the message as a label. Final states are evaluated as true.

Be aware that the expressiveness of aFSAs does support choices and iterations, but no parallel nor recursive behavior. While parallel behavior can be simulated by enumerating all possible execution sequences resulting in complex automata, recursion is out of the scope of FSAs. Please have in mind that recursive Petri nets result in infinite occurrence graphs and, as a consequence, checking of properties which are based on the occurrence graph are hardly decidable. Further, most processes in real world applications are limited to iteration and do not need recursion. Therefore, the expressiveness investigated in the following is limited to oWFNs without recursion, since they have comparable expressiveness as aFSA.

III. EQUIVALENCE

Equivalence is a quite rigid comparison operation for service discovery. A detailed discussion and comparison of different equivalence notions is given in [6]. In this paper the equivalence notions of branching bisimilarity and language equivalence are considered since they are important in Petri net and FSA theory respectively.

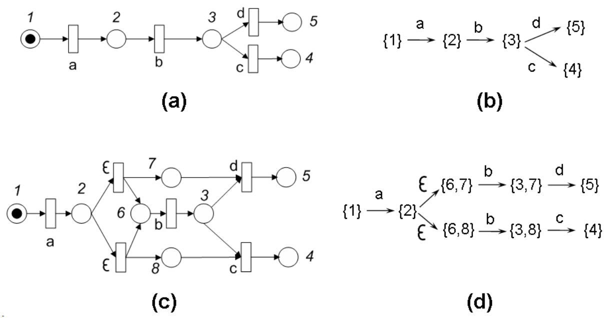

Fig. 1. Branching Bisimilarity: (a) first example (b) occurrence graph of the first example (c) second example (d) occurrence graph of the second example

Branching bisimilarity of two workflows is given, if for all equivalent states of two oWFNs the same options of firing a transition exist, which result in equivalent states again. Figure 1(a) and (c) represent two oWFNs which are not branching bisimilar. This becomes clear when investigating the states of the two oWFNs. The occurrence graphs representing all states and state transitions are depicted in Figure 1(b) and (d), respectively. It can be seen that there exist equivalent states $\{1\}$ in both oWFNs each with a single state transition labeled a resulting in equivalent states $\{2\}$. However, the state $\{3\}$ of the upper oWFN does not have an equivalent state in the lower oWFN, since neither state $\{3,7\}$ nor $\{3,8\}$ support both state transitions labeled c and d respectively.

Reconsidering the workflows in Figure 1(a) and (c) shows that the workflows are language equivalent, i.e., both workflows represent the same language $\{abc, abd\}$ since the ϵ transition represents an empty word and therefore does not contribute to a word. Thus, language equivalence is less restrictive then branching bisimilarity.

FSA theory is based on language equivalence and therefore does not provide a standard operation for checking branching bisimilarity, because there is no way to express that all transitions of a state must be supported by the other workflow. However, this has been the definition of mandatory transitions of the aFSA model used in this paper. Thus, there is a way to represent branching bisimilarity based on existing operations. In particular, branching bisimilarity can be represented as nonempty intersection of two aFSAs, where all outgoing transitions of all states are considered to be mandatory. Applying this concept to the aFSA representations of the workflows in Figure 1(a) and (c) results in the aFSAs depicted in Figure 2(a) and (b) respectively. Calculating the intersection automaton of the two aFSAs in Figure 2(a) and (b) results in the aFSA depicted in Figure 2(c). The state tuple contains the state number of aFSA depicted in Figure 2(a) first and then the state number of aFSA depicted in Figure 2(b). The intersection automaton is empty, because the annotations *c AND d* at states $(3, 4)$ and $(3, 7)$ can not be evaluated to $true$ because there is no outgoing transition labeled d and c respectively. Therefore, the emptiness test determines an empty automaton indicating that the workflows are not branching bisimilar, which fits the expectation.

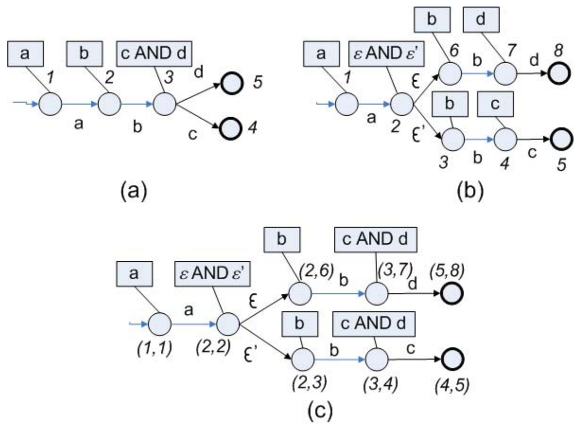

Fig. 2. Branching Bisimilarity with aFSA: (a) aFSA notation of Figure 1(a) (b) aFSA notation of Figure 1(c) (c) intersection aFSA of (a) and (b)

As a consequence both approaches support language equivalence and branching bisimilarity. aFSAs support branching bisimilarity because the requirement of all outgoing transitions being mandatory for the other workflow is the basis of the aFSA extension of standard FSA.

IV. Soundness

A less rigid comparison operation for service discovery than equivalence is soundness. Soundness means that from every reachable state of a workflow a proper final state can be reached. Thus, a successful execution of the workflow is guaranteed. Soundness in its strikt sense additionally requires the coverage of all transition by a sound firing sequence. In this paper, we focus on weak soundness as introduced in [7].

Applying the weak soundness definition to the service discovery scenario means that if two interacting services are sound then there is a guarantee that the interaction can be terminated successfully, i.e., no deadlock can occur. When investigating the interactions of two services/workflows it is essential to understand the assumptions on the underlying communication model. The two basic communication models are synchronous and asynchronous communication.

Synchronous communication ensures that a message m_1 sent before a message m_2 is processed by the recipient in the same order. Synchronous communication consists usually of a request and a corresponding response message, where the exchange is considered to be atomic. Examples of such communication protocols are TCP/IP or HTTP.

Asynchronous communication does not guarantee the message order on the recipient side. In particular, the order in which the recipient processes the messages m_1 and m_2 is up to the recipient. Examples of such communication protocols are Simple Mail Transfer Protocol (SMTP) or message queuing middleware.

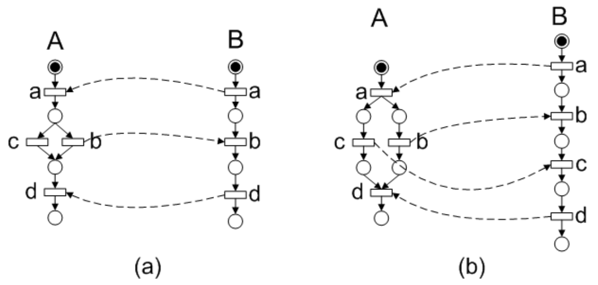

Fig. 3. Soundness Examples (a) XOR example (b) AND example

With regard to SOA, the communication model is not contained in the modeling but in the deployment of services and therefore both options are applicable in SOA and have to be considered in behavior based service discovery. In the following the examples depicted in Figure 3 are used to compare the aFSA and the oWFN approach. The first example represents a simple XOR-split and XOR-join of party A while the second example depicts an AND-split and AND-join of party A. Party B in both cases contains a plain sequence. The examples are given in oWFN notation, while the dotted arcs represent the intended direction of the information exchange. The XOR example is considered to be not sound for both communication models because party A may want to send message c to party B, who is not able to handle this message. The AND example is considered to be sound for the asynchronous and not sound for the synchronous communication model. In the asynchronous case the order in which the messages are processed by party B is considered irrelevant, which means that also a sequential processing is possible. However, in the synchronous case party B can not handle the delivery of message c before message b, which results in an unexpected behavior and therefore does not guarantee a successful interaction.

A. Synchronous Communication

Synchronous communication is the underlying communication model of the aFSA approach as proposed in [2]. To accomplish the behavior based service discovery, the direction of the information exchange is encoded in the transition labels. In particular, a transition label consists of $s\#r\#msg$, where s is the sender, r the recipient, and msg the message name of the exchanged information. Further, each workflow is assigned with a role name, which is used for sender and recipient identification. In particular, two workflows match if they represent different roles and the workflows guarantee successful interaction, which has been called consistency in

[2]. Consistency has been defined as non-empty intersection of the aFSAs, where all messages sent by the own party are mandatory (i.e. annotated as conjunctions) and all messages received by the own party are optional (i.e. annotated as disjunctions).

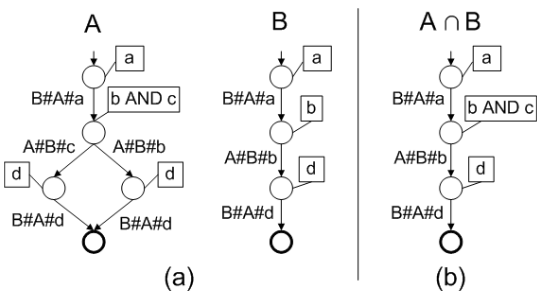

Fig. 4. Synchronous XOR example notated as aFSA: (a) aFSAs of party A and B (b) intersection aFSA of party A and B

Figure 4(a) depicts the aFSA representation of the XOR example depicted in Figure 3(a). The intersection aFSA (see Figure 4(b)) has the same structure as the aFSA of party B but has an annotation of *b AND c* instead of the annotation of b. The evaluation of this different annotation during the emptiness test results in an empty aFSA and therefore not sound aFSAs. This result matches the expectation stated above.

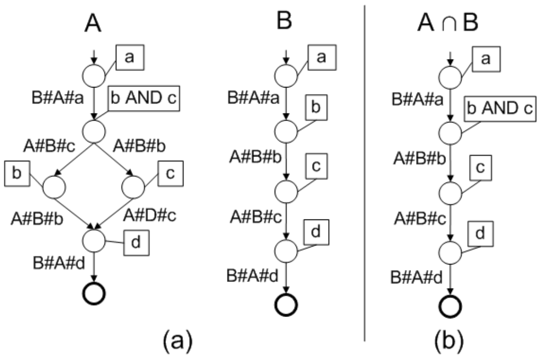

Fig. 5. Synchronous AND example notated as aFSA: (a) aFSAs of party A and B (b) intersection aFSA of party A and B

Figure 5(a) depicts the aFSA representation of the AND example depicted in Figure 3(b). Since aFSAs do not support parallel behavior, all possible execution sequences of the messages b and c have to be enumerated. In the example these are the sequences bc and cb. Calculating the intersection aFSA (see Figure 5(a)) of the two aFSAs results in the same structure as the aFSA of party B but again with a change of annotation b by *b AND c*. The emptiness test performed on the intersection aFSA indicates again not sound aFSAs because there is no transition labeled c leaving this particular state. This matches the expectations stated above.

These two examples illustrate the usability of aFSAs for the synchronous communication model. More formal results including extensions to multi-lateral collaborations can be found in [8].

As briefly described in Section II-A also the WF-Net approach provides an option to handle synchronous communication. In particular, two transitions related with each other via a communication channel are merged to a single transition as depicted in Fig. 6. Similar to the aFSA approach the direction of the messages has to be encoded in the transition labels and it has to take into account the role name of the party.

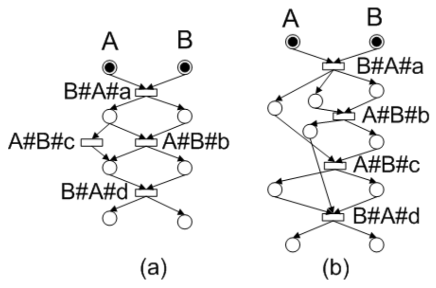

Fig. 6. Synchronous Examples notated as WF-Net with merged transitions (a) XOR example (b) AND example

Fig. 6(a) and (b) depict the merged WF-Net representations of the XOR and the AND example. Both Petri net models are sound since there exists for every reachable state a sequence of transitions resulting in the final state. However, both original examples are considered *not* to be sound. The incorrect conclusion is made, because the modeling of the synchronous communication does not consider the differentiation between mandatory and optional transitions but considers all transitions to be optional. In particular, the lacking treatment of party B for the message $A\#B\#c$ in the XOR example and the sequence $acbd$ in the AND example result in the incorrect decision. In either case mandatory transitions are missing.

Alternatively, an approach for simulating synchronous communication with asynchronous oWFNs is proposed in this paper. The basic idea is that a synchronous communication can be simulated by two subsequent asynchronous communications. For example a message a results in the request message a and the response message a'. As stated in Section II-A asynchronous communication channels are translated into places representing a communication channel. The sending and the receiving transitions are connected with this place, thus the direction of the information exchanged is encoded in the resulting Petri net and therefore does not need to be encoded in the message itself. The corresponding oWFNs derived from the approach proposed above for the XOR and AND example are depicted in Figure 7 and 8. This handling of messages covers mandatory and optional transitions, since every mandatory/sending transition places a token in a communication place which must be consumed to make the oWFN sound. On the other side an optional/receiving transition consumes a token from a communication place and in case the communication place will never contain a token the soundness of the oWFN is not effected.

However, additional synchronization is required. In particular, for the synchronous communication model it has to be ensured that the order of messages sent equals the order of messages received. Therefore, we extend the model with

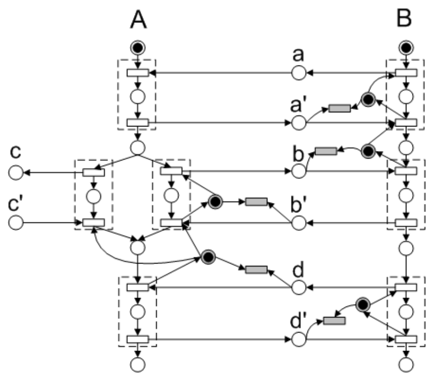

Fig. 7. Synchronous XOR example notated as oWFN with two-way asynchronous communication simulating synchronous communication

error transitions and places to enable these transitions (gray places and transitions in Figure 7 and 8). The error places are initialized with a single token. The error transition can be fired except the transition preceding the receiving transition has been performed. The receiving transition adds a token back to the error place to enable the cleaning up of the error places at the end of the execution of the oWFN (not depicted in the figures). If an error transition fires, the remaining oWFN will never be able to reach the final state. Therefore, the oWFN is sound iff no error transition will be able to fire at any possible state.

For example, in Fig. 7 party B must receive the response message a' directly after sending the request message a. Therefore, the error place is emptied by sending a and filled again after the response a' has been received. Receiving the response a' precedes receiving message b and therefore empties the error place of message b. Receiving message b fills the corresponding error place again.

It turns out that the oWFNs depicted in Figure 7 and 8 are both not sound. The XOR example is not sound, because the communication place c can contain a token, which will never be consumed by party B and therefore the soundness property of reaching the final state from all possible states is not fulfilled. The AND example is not sound, because party A can send message c before message b, which means that the error transition of message c at party B has not been disabled allowing to fire the error transition. This modeling of synchronous communication with oWFNs matches the expectations.

In this proposal it is assumed that only a single process instance is running and that each message name occurs at most once in an execution sequence. Both assumptions can be relaxed by introducing IDs for each message and using color extension (see [9]) of oWFNs to maintain the IDs. Due to the readability and the complexity in this paper this extension remains for future work.

B. Asynchronous Communication

Asynchronous communication means that the order in which messages are sent may differ from the order in which they are processed at the receiving party. As a consequence

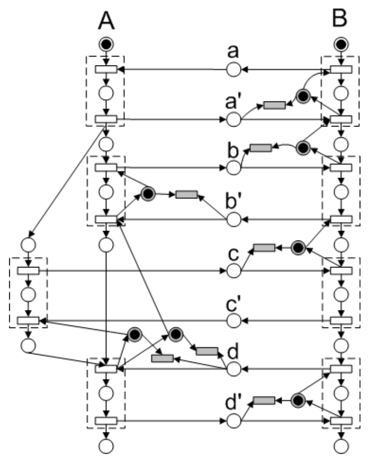

Fig. 8. Synchronous AND example notated as oWFN with two-way asynchronous communication simulating synchronous communication

this leaves more freedom to the interaction between two parties because communication channels function as message buffers to compensate differences in the workflows. Figure 9 depicts the oWFN representation of the examples. The places representing the communication channels are associated with the name of the communication channel. The XOR example (see Fig 9(a)) is not sound, since sending message c results in a token in the place representing communication channel c. This token can never be removed from this place, thus from this state the final state is not reachable and therefore the WF-Net is not sound. The AND example (see Fig 9(b)) is a sound WF-Net. Thus, for both examples the expected results are derived.

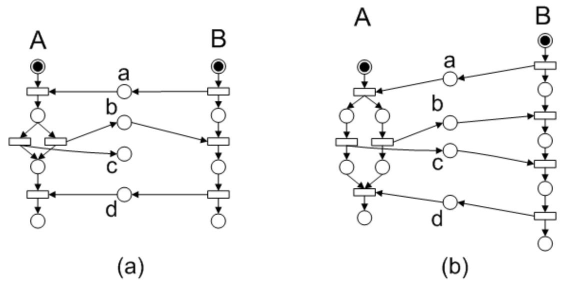

Fig. 9. Asynchronous Examples notated as oWFNs (a) XOR example (b) AND example

The aFSA approach has been designed for synchronous communication. However, it can also be applied to the asynchronous communication case. The underlying idea is that all receiving messages on a path from the start state to a sending transition can be executed in an arbitrary order[1]. Thus, sending messages are synchronization points of an interaction.

Applying this idea to the XOR example results in the aFSAs depicted in Figure 10. The aFSA of party B receives a single message b which is on the path of the sent message d. According to the above stated approach, message b can be received either before or after message a has been sent, but

[1]The construction of an arbitrary order of messages can be accomplished by a standard FSA operation called shuffle product (see e.g. [10]).

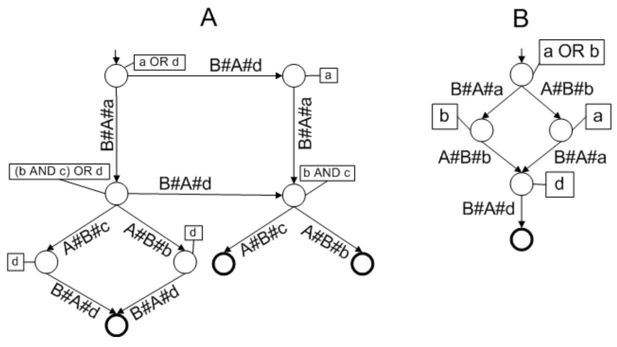

Fig. 10. Asynchronous XOR Example notated as aFSA

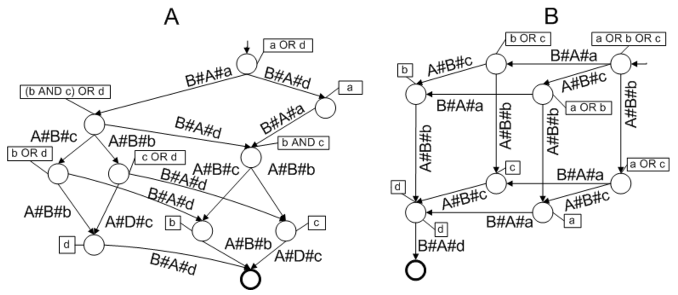

Fig. 12. Asynchronous AND Example notated as aFSA

must be received before message d is sent. Party A has two receiving messages, d being the last message and a being on the path of the sent messages b and c. Thus, message d can be received at any time, while message a has to be received before message b or c is sent. Calculating the intersection aFSA (see Figure 11(a)) of the two constructed aFSAs results in an empty aFSA due to the *b AND c* annotation at party A. Thus, the two aFSAs are not sound.

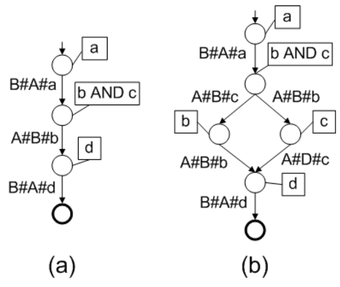

Fig. 11. Asynchronous intersection aFSA: (a) XOR example (b) AND example

Applying the approach to the AND example results in the aFSAs depicted in Figure 12. Party B contains two messages (b and c) which can be received before message d is sent. Party A contains message d which can be received at any time. The intersection aFSA as depicted in Figure 11(b) is now non empty indicating that the two workflows are sound. Different from the synchronous case the two aFSA are now sound because the annotation *b AND c* at party A is now supported by party B due to the freedom to receive message b and c in arbitrary order before sending message d.

The results of the two examples fit the expectations. However, the aFSA representation is complex due to the high number of possible transition sequences and therefore is hardly applicable at more complex examples.

V. Conclusion and Future Work

Soundness and equivalence are two comparison relations which have been applied to behavior based service discovery. Dependent on the underlying communication model different results can be derived. In this paper a Petri Net based approach (open Workflow Nets) and a Finite State Automaton based approach (annotated Finite State Automata) are compared with each other with regard to equivalence notions and soundness for synchronous and asynchronous communication models. In particular, usual notions of equivalence and soundness and a corresponding operationalization based on the two models is presented. It turns out that the notions supported by one model can be represented with the other model as well.

In this paper the operationalization is presented in an informal way and illustrated on two examples only. A more formal presentation could not be provided due to the lack of space. Future work will contain a formalization and proper proofs to show the equivalence of the operationalization. However, the explanations are intuitive and outline the basic ideas of the proofs.

References

[1] W. Aalst, "Interorganizational workflows: An approach based on message sequence charts and petri nets," *Systems Analysis - Modelling - Simulation*, vol. 34, no. 3, pp. 335–367, 1999.

[2] A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold:, "Matchmaking for business processes based on choreographies," *Intl. Journal of Web Services*, vol. 1, no. 4, pp. 14–32, 2004.

[3] W. Aalst and K. Hee, *Workflow Management - Models, Methods, and Systems*. MIT Press, 2002.

[4] P. Massuthe, W. Reisig, and K. Schmidt, "An Operating Guideline Approach to the SOA," *Annals of Mathematics, Computing & Teleinformatics*, vol. 1, no. 3, pp. 35–43, 2005.

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

[6] J. Hidders, M. Dumas, W. Aalst, A. Hofstede, and J. Verelst, "When are two workflows the same?" in *Proceedings of Computing: The 11th Australasian Theory Symposium (CATS'2005)*, ser. Conferences in Research and Practice in Information Technology, vol. 41. Newcastle, Australia: Australian Computer Society, Feb. 2005, pp. 3–11.

[7] A. Martens, "Analyzing Web Service based Business Processes." in *Proceedings of FASE 2005*, ser. LNCS 3442, M. Cerioli, Ed. Edinburgh, Scotland: Springer, Apr. 2005.

[8] A. Wombacher, "Decentralized establishment of consistent, multi-lateral collaborations," Ph.D. dissertation, Technical University Darmstadt, 2005.

[9] K. Jensen, *Coloured Petri Nets*. Springer Verlag, 1992.

[10] O. Matz, A. Miller, A. Podtthoff, W. Thomas, and E. Valkema, "Report on the program AMoRE," Christian-Albrechts Universtaet, Tech. Rep. 9507, 1995.